RFP APPENDIX 1: Reference 7

BAIFA
BAY AREA INFRASTRUCTURE FINANCING AUTHORITY

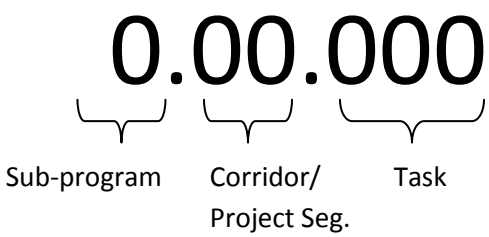
BAIFA Express Lane Network

Toll Collection System

Work Breakdown Structure

Work Breakdown Structure for BAIFA Express Lane Network

The Work Breakdown Structure (WBS) is a numeric taxonomy used through BAIFA's program for project controls including schedules, costs, and document management. The structure of the MTC Express Lanes WBS is based on Caltrans Uniform File System. The WBS assists in organizing and maintaining consistency throughout the project. This reference document provides a portion of existing WBS, of which certain sections may be modified by the Toll System Integrator (TSI), including:



Sub-Program (Single digits 1-6): Sub-programs are the major disciplines within BAIFA's express lanes program, including:

1 Program Management
2 System Management
3 Systems Integration (Lower levels in this sub-program are subject to change by the future System Integrator)
4 Highways Design
5 Highway Construction
6 Operations and Maintenance

Project Segment (Double digits 01-09): The second number indicates a specific corridor or project segment, but 00 is provided for activities that encompass all corridors or a program level effort.

Tasks (digits 000-990, generally in increments of 5): Individual tasks for each segment. Examples are below:

1.00.000	Meetings & Communication
1.00.010	Organization
1.00.050	Contracts & Agreement Administration
1.00.100	Project Management & Controls

Systems Integration is sub-program 3 as provided in the WBS for this section is below.

WBS Codes	Sub-Programs	Packages	Task ID File	Tasks
1	Program Management			
1.00.000			000	Meetings & Communication
1.00.010			010	Organization
1.00.050			050	Contracts & Agreement Administration
1.00.060			060	Financial & General Administration
1.00.100			100	Project Management & Controls
1.01.140			140	Data Collection
1.00.610			610	Reference Documents
1.00.620			620	Press Articles
1.00.630			630	Public Education
1.00.640			640	Engineering/Technical Reviews
1.00.650			650	Special Studies/Analyses
2	Systems Management			
2.00.000			000	Meetings & Communication
2.00.010			010	Organization
2.00.050			050	Contracts & Agreement Administration
2.00.100			100	Project Management & Controls
2.00.140			140	Data Collection
2.00.710			710	Concept of Operations
2.00.720			720	System Requirements & Design
2.00.730			730	Procurement Management
2.00.740			740	System Engineering Management Plan
2.00.750			750	System Design & Development Oversight
2.00.760			760	Testing Oversight
2.00.770			770	Installation Oversight
2.00.780			780	Operations & Maintenance
3	Systems Integration			
3.00.000			000	Meetings & Communication
3.00.010			010	Organization
3.00.050			050	Contract & Agreement Administration
3.00.100			100	Project Management & Controls
3.00.140			140	Data Collection
3.00.810			810	Regulatory Documents
3.00.820			820	Project Initiation Submittals
3.00.830			830	Preliminary Design
3.00.840			840	Detailed Design
3.00.850			850	Factory Acceptance Test (FAT)

WBS Codes	Sub-Programs	Packages	Task ID File	Tasks
3.00.860			860	Installation Planning & Readiness
3.00.870			870	Onsite First Installation Test (OFIT)
3.00.875			875	Onsite First Installation Test (OFIT)
3.00.880			880	Final System Acceptance
3.00.890			890	Operations & Maintenance
4	Highway Design			
4.01		Ph1 PA&ED		
4.01.000			000	Meetings & Communication
4.01.010			010	Organization
4.01.050			050	Contract & Agreement Administration
4.01.100			100	Project Management & Controls
4.01.140			140	Data Collection
4.01.150			150	PID
4.01.155			155	Traffic Forecasting & Operations Analysis
4.01.160			160	Preliminary Engineering and Draft PR
4.01.165			165	Environmental Studies and Draft ED
4.01.170			170	Permits
4.01.175			175	Circulate Environmental Document
4.01.180			180	Project Report and final ED
4.01.185			185	Prepare Base Maps & Plan Sheets
4.01.200			200	Utility Relocation & Coordination
4.02		I-680 PS&E		
4.02.000			000	Meetings & Communication
4.02.050			050	Contract & Agreement Administration
4.02.100			100	Project Management & Controls
4.02.140			140	Data Collection
4.02.190			190	Prepare Structures Site Plans
4.02.200			200	Utility Relocation & Coordination
4.02.205			205	Obtain Permits & Agreements
4.02.220			220	Right of Way
4.02.230			230	Draft PS&E
4.02.240			240	Draft Structures PS&E
4.02.260			260	Prepare Contract Documents
4.02.265			265	Advertise, Award & Approve
4.03		I-880 PS&E		
4.03.000			000	Meetings & Communication
4.03.050			050	Contract & Agreement Administration
4.03.100			100	Project Management & Controls
4.03.140			140	Data Collection
4.03.190			190	Prepare Structures Site Plans
4.03.200			200	Utility Relocation/Coordination

WBS Codes	Sub-Programs	Packages	Task ID File	Tasks
4.03.205			205	Obtain Permits & Agreements
4.03.220			220	Right of Way
4.03.230			230	Draft PS&E
4.03.240			240	Draft Structures PS&E
4.03.260			260	Prepare Contract Documents
4.03.265			265	Advertise, Award & Approve
4.04		Bay Bridge Approach PS&E		
4.04.000			000	Meetings & Communication
4.04.050			050	Contract & Agreement Administration
4.04.100			100	Project Management & Controls
4.04.140			140	Data Collection
4.04.190			190	Prepare Structures Site Plans
4.04.200			200	Utility Relocation/Coordination
4.04.205			205	Obtain Permits & Agreements
4.04.220			220	Right of Way
4.04.230			230	Draft PS&E
4.04.240			240	Draft Structures PS&E
4.04.260			260	Prepare Contract Documents
4.04.265			265	Advertise, Award & Approve
4.05		San Mateo Approach PS&E		
4.05.000			000	Meetings & Communication
4.05.050			050	Contract & Agreement Administration
4.05.100			100	Project Management & Controls
4.05.140			140	Data Collection
4.05.190			190	Prepare Structures Site Plans
4.05.200			200	Utility Relocation/Coordination
4.05.205			205	Obtain Permits & Agreements
4.05.220			220	Right of Way
4.05.230			230	Draft PS&E
4.05.240			240	Draft Structures PS&E
4.05.260			260	Prepare Contract Documents
4.05.265			265	Advertise, Award & Approve
4.06		Dumbarton Approach PS&E		
4.06.000			000	Meetings & Communication
4.06.060			050	Contract & Agreement Administration
4.06.100			100	Project Management & Controls
4.06.140			140	Data Collection

WBS Codes	Sub-Programs	Packages	Task ID File	Tasks
4.06.190			190	Prepare Structures Site Plans
4.06.200			200	Utility Relocation/Coordination
4.06.206			205	Obtain Permits & Agreements
4.06.220			220	Right of Way
4.06.230			230	Draft PS&E
4.06.240			240	Draft Structures PS&E
4.06.260			260	Prepare Contract Documents
4.06.265			265	Advertise, Award & Approve
4.07		I-80 Solano Conversion PS&E		
4.07.000			000	Meetings & Communication
4.07.050			050	Contract & Agreement Administration
4.07.100			100	Project Management & Controls
4.07.140			140	Data Collection
4.07.190			190	Prepare Structures Site Plans
4.07.200			200	Utility Relocation & Coordination
4.07.205			205	Obtain Permits & Agreements
4.07.220			220	Right of Way
4.07.230			230	Draft PS&E
4.07.240			240	Draft Structures PS&E
4.07.260			260	Prepare Contract Documents
4.07.265			265	Advertise, Award & Approve
4.08		I-680 CC North Conv PS&E		
4.08.000			000	Meetings & Communication
4.08.050			050	Contract & Agreement Administration
4.08.100			100	Project Management & Controls
4.08.140			140	Data Collection
4.08.190			190	Prepare Structures Site Plans
4.02.200			200	Utility Relocation & Coordination
4.08.205			205	Obtain Permits & Agreements
4.08.220			220	Right of Way
4.08.230			230	Draft PS&E
4.08.240			240	Draft Structures PS&E
4.08.260			260	Prepare Contract Documents
4.08.265			265	Advertise, Award & Approve
4.09		I-80 Alameda CC Conv PS&E		
4.09.000			000	Meetings & Communication

WBS Codes	Sub-Programs	Packages	Task ID File	Tasks
4.09.050			050	Contract & Agreement Administration
4.09.100			100	Project Management & Controls
4.09.140			140	Data Collection
4.09.190			190	Prepare Structures Site Plans
4.09.200			200	Utility Relocation & Coordination
4.09.205			205	Obtain Permits & Agreements
4.09.220			220	Right of Way
4.09.230			230	Draft PS&E
4.09.240			240	Draft Structures PS&E
4.09.260			260	Prepare Contract Documents
4.09.265			265	Advertise, Award & Approve
5	Highway Construction			
5.00.000			000	Meetings & Communication
5.00.050			050	Contract & Agreement Administration
5.00.100			100	Project Management & Controls
5.00.270			270	Construction Management
5.00.295			295	Accept and Close Contract
				Contractor's Schedule
6	Operations and Maintenance			
6.00.000			000	Meetings & Communication
6.00.050			050	Contract & Agreement Administration
6.00.100			100	Project Management & Controls
6.00.900			900	O&M - Policy and Procedures
6.00.910			910	O&M - Warranty Period
6.00.911			911	O&M - Post-Warranty Period
6.00.920			920	Highway Maintenance
6.00.930			930	Operations Center
6.00.940			940	Enforcement
6.00.950			950	Utilities
6.00.960			960	Performance Monitoring
6.00.990			990	Marketing

RFP APPENDIX 1: Reference 8

BAIFA
BAY AREA INFRASTRUCTURE FINANCING AUTHORITY

BAIFA Express Lane Network

Toll Collection System

System Interface Documentation

A	RCSC Interface Control Documents (ICDs)
B	Bay Area Video Upgrade (BAVU) Cameleon Interface Driver
C	California Freeway Performance Measurement System (PeMS) – XML Interface
D	511 - Open511 Protocol

BAIFA Express Lane Network Toll Collection System

Appendix 1, Reference 8

System Interface Documentation

A: RCSC Interface Control Documents (ICDs)

NOTE: This is to be used for informational purposes only as the interface does not currently support transmission of BAIFA ELN Trip Transactions.

Regional BATA Interface Control Document

Rev. 1.6.13.5

August 16, 2011

23. Tag Status File – HOT

23.1 File type

Variable length, LF delimited

23.2 File name

<PROGRAM_TYPE>_YYYYMMDD_HHMMSS.etc

Example: ht_20030426_100015.etc
Tag status file created at 10:00:15 on 04/26/2003

This file will be zipped (ht_200304261_00015_etc.zip) and contain the following files –
Example: gg_20030426_100001.etc
at_20030426_100002.etc
srat_20041025_030205.tag
tcat_20030426_100006.tag
cvat_20030426_100009.tag
sdat_20030426_100013.tag

23.3 File use

The Tag Status File shall be created by the VECTOR CSC to inform the HOT Host as to the status of each tag associated with an account held by BATA or CTOC customers. This file shall then be used by the HOT Host to generate a tag status file for the HOT lanes.
CSC will generate tag status file for CALTRANS tag range, GGBD tag range separately. Transmission from CSC to HOT host will include 1 CALTRANS, 1 GGBD, 1 each CTOC Tag File zipped as one file.

23.4 File layout

Each field in the header, detail and trailer structure will be separated with Delimiter "," comma.

Table 23-1 Tag Status File – Header Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (7)	"#HEADER"
FILE_TYPE	CHAR (4)	"TAGS"
ACTION_CODE	CHAR (4)	"INIT"
SEQUENCE #	CHAR (6)	Sequence # of the Tag Status File. This number is incremented every day. Values 000000 – 999999 Sequence Number will be unique per agency file. Sequence number will be incremented every time a new file is generated for home tag ranges. For files received from away agency, the sequence number will be as received.
BUSINESS_DATE	CHAR (10)	File creation date, Format MM/DD/YYYY
SOURCE	CHAR (2)	Indicates the file-creating agency. "at" for BATA (CALTRANS AND GGBD) and corresponding CTOC agency names for the CTOC files.
DESTINATION	CHAR (2)	Indicates the destination entity. "ht" for HOT (for HOME tag files) and "at" for CTOC tag files
CREATE_DATE	CHAR (10)	Indicates the file creation date. Format MM/DD/YYYY
CREATE_TIME	CHAR (8)	Indicates the file creation time. Format HH:MM:SS
LINEFEED	CHAR (1)	LF
Header Total	**54**	

Table 23-2 Tag Status File - Detail Structure

Field Name	Type/Size	Description/Valid Values
ETC_TAG_ID	CHAR (8)	Tag Id in HEX Values: 00000000-0FFFE3FF
ACTION_CODE	CHAR (1)	Always "A"
TAG_TYPE	CHAR (1)	Values N – Non-Revenue, V – Valid, I – Invalid
SUBTYPE_1	CHAR (1)	Values N – Default, L – Lost, S – Stolen, B – Low balance, R – Not Used
SUBTYPE_2	CHAR (1)	N – Not Used
SUBTYPE_3	CHAR (1)	N – Not Used
LINEFEED	CHAR (1)	LF
Record Total	**14**	

Table 23-3 Tag Status File – Trailer Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (8)	"#TRAILER"
SEQUENCE #	CHAR (6)	Same as Header
BUSINESS_DATE	CHAR (10)	File creation date, Format MM/DD/YYYY
DETAIL_COUNT	CHAR (8)	Total count of all detail records
LINEFEED	CHAR (1)	LF
Record Total	**33**	

23.5 Processing requirements

1. The VECTOR CSC shall complete the transmission of the comprehensive tag status file to the HOT Host drop-box as defined in Section 2 of the ICD.
2. In the event that an invalid header record is encountered (e.g., character data in a numeric field, etc.), the HOT Host shall reject the file and notify the VECTOR CSC via the Acknowledgement File defined in Section 10 of this document.
3. The RCSC will send one zipped tag status file to HOT. The zipped file will contain 6 separate files (AT; GG; TCA; SNDG; SR91 and CTV). If any of the 6 individual files received are bad, HOT Host will send ACK file to the RCSC with a status of 01. HOT Host will make an attempt to process any of the individual valid files and download to the lanes as per their current processing rules. In the case of a BAD CTOC file, the HOT Host will use their existing mechanism of using the latest CTOC Tag file and ignoring the BAD CTOC file. RCSC will log the problem upon receiving the ACK file (01) from the HOT Host. Upon received notification of an ACK file with a status of 01, the ACS System Admin will log and escalate the issue. They contact the HOT System Admin for detailed information. Once a decision has been reached appropriate action will be taken.
4. In the event that an invalid detail record is encountered (e.g., inappropriate TAG_STATUS, etc.), the HOT Host shall skip the complete file and notify the VECTOR CSC via the Acknowledgement File. Please refer to Appendix C for processing rules on error data in files.
5. The HOT Host shall perform the appropriate sanity checks on the Tag Status File prior to its transmission to the lanes. Such sanity checks should include, but not be limited to:
    a. Unusual growth in the number of tags from previous version
    b. Unusual change in number of tags with a particular tag status
6. One form of validation by the Host could be an upper limit of 10% increase and a lower limit of 2%, as compared to previous file. This check can be lifted on notification from CSC. This can happen if the CSC receives large Tag Inventory. As per the current Business Rules, there is no reason for Tag Status file to decrease in size when compared to previous file. GGBD will perform this check on each individual file (CALTRANS range, GGBD range and on each CTOC agency files).
7. The CHP are currently the only full non-revenue account for HOT

Table 23-4 Valid Tag Status Values for HOT Host

Item #	Tag Status	Account Status	Financial Status	Discount Plan	Regional - CSC GGBD Tag Type	Regional - CSC GGBD Sub Type 1
1	INVENTORY	N/A	N/A	N/A	I	N
2	RETURNED	N/A	N/A	N/A	I	N
3	DAMAGED	N/A	N/A	N/A	I	N
4	RETURNDEF	N/A	N/A	N/A	I	N
5	SHIPVEND	N/A	N/A	N/A	I	N

6	TESTED	N/A	N/A	N/A	I	N
7	EXPIRED	N/A	N/A	N/A	I	N
8	LOST	Active	N/A	N/A	I	L
9	STOLEN	Active	N/A	N/A	I	S
10	ACTIVE	Active	Good Balance	Standard	V	N
11	ACTIVE	Active	Low Balance (Cash/Check)	Standard	V	B
12	ACTIVE	Active	Zero Balance (Cash/Check)	Standard	I	B
13	ACTIVE	Active	Revoked Warning (Cash/Check)	Standard	I	B
14	ACTIVE	Active	Good Balance	Non Revenue	N	N
15	ACTIVE	Active	Low Balance (Cash/Check)	Non Revenue	N	N
16	ACTIVE	Active	Zero Balance (Cash/Check)	Non Revenue	N	N
17	ACTIVE	Active	Revoked Warning (Cash/Check)	Non Revenue	N	N
18	N/A	Closed Pending	N/A	N/A	I	N

Table 23-5 CTOC Tag Status Mapping Values for HOT Host

Item #	CTOC Tag Type	CTOC Sub Type 1	Regional - CSC GGBD Tag Type	Regional - CSC GGBD Sub Type 1
1	N – Non Revenue (Universal to all entities)	N – Not Used	N – Non Revenue	N
2	V – Valid	N – Not Used	V – Valid	N
3	I - Invalid	N – Not Used	I – Invalid	N

23.6 Sample files

CALTRANS Tag File for HOT Lanes

at_20040508_100002.etc

```
#HEADER,TAGS,INIT,000967,05/08/2004,at,gg,05/08/2004,22:45:03
0FE00001,A,V,N,R,R
0FE00006,A,V,N,R,R
0FE00008,A,V,N,R,R
0FE0000A,A,I,L,R,R
.
.
#TRAILER,000967,05/08/2004,00315464
```

SR-91 Tag File for HOT Lanes

srat_20041025_030205.tag

```
#HEADER,TAGS,INIT,000907,05/08/2004,sr,at,05/08/2004,22:45:03
08100000,A,V,N,N,N
08100001,A,V,N,N,N
08100002,A,V,N,N,N
08100003,A,V,N,N,N
.
.
#TRAILER,000907,05/08/2004,00315464
```

24. ETC Transaction File – HOT

24.1 File type

Variable length, LF delimited

24.2 File name

680_YYYYMMDDHHMMSS.hreq

Example: 680_20020928044100.hreq
HOT transactions to VECTOR CSC created at 04:41:00 on 09/28/02

24.3 File use

The Transaction File shall be created by the HOT Host to inform the VECTOR CSC of all toll transactions occurring at HOT lanes. This file shall contain tagged transactions on HOT lanes due to BATA customers or CTOC customers with both valid and invalid statuses.

24.4 File layout

Each field in the header, detail and trailer structure will be separated with Delimiter "," comma.

Figure 24-1 ETC Transaction File – Header Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (7)	"#HEADER"
FILE_TYPE	CHAR (4)	"HREQ"
SEQUENCE #	CHAR (6)	Sequence # of the Transaction File. This unique number is incremented for every file. Values 000000 – 999999
BUSINESS_DATE	CHAR (10)	This field will be populated with the transaction date of the first transaction in the file. Format MM/DD/YYYY
SOURCE	CHAR (2)	Indicates the file-creating agency. "H1" for HOT
DESTINATION	CHAR (2)	Indicates the destination entity. "AT" for BATA
CREATE_DATE	CHAR (10)	Indicates the file creation date. Format MM/DD/YYYY
CREATE_TIME	CHAR (8)	Indicates the file creation time. Format HH:MM:SS
LINEFEED	CHAR (1)	LF

Header Total	**50**	

Figure 24-2 ETC Transaction File – Detail Structure

Field Name	Type/Size	Description/Valid Values
TRANSACTION_NUMBER	CHAR (10)	Unique transaction number for each ETC transaction. Used to identify the transaction in the ETC reconciliation process. Values 0000000000 to 9999999999
TOL_TRX_TYPE	CHAR (1)	Type of transaction. 1 – ETC. This is the default value.
TOL_TAG_ID	CHAR (4)	This field consists of the ETC Internal Tag ID, in accordance with Title-21 specs. Values: 0000-1023
TOL_TAG_FACILITY_ID	CHAR (6)	This field comprises of the Facility code of the Issuing agency. Values: 000000-262143
ENTRY_TOL_PLAZA_ID	CHAR (3)	The entry plaza code of the agency at which the transaction occurred. This information shall be shown on customer statements to indicate the place of occurrence of the transaction. Value = Refer to table 24.7 (other HOT agencies will be given a new Plaza Id)
ENTRY_TOL_LANE_ID	CHAR (2)	The entry lane ID at the plaza where the transaction occurred. The information from this field shall be used on customer statements to indicate the point of occurrence of the transaction. Values = 00 – 99.
ENTRY_TOL_TRX_DATE	CHAR (10)	The date of the occurrence of the transaction at ENTRY_TOL_LANE_ID. Format: MM/DD/YYYY. This toll transaction date information shall be shown on customer statements.
ENTRY_TOL_TRX_TIME	CHAR (8)	The time of the occurrence of the transaction at ENTRY_TOL_LANE_ID. Format: HH:MM:SS. This toll transaction time information shall be shown on customer statements.
EXIT_TOL_PLAZA_ID	CHAR (3)	The exit plaza code of the agency at which the transaction occurred. This information shall be shown on customer statements to indicate the place of occurrence of the transaction. Value = Refer to table 24.7 (other HOT agencies will be given a new Plaza Id)
EXIT_TOL_LANE_ID	CHAR (2)	The exit lane ID at the plaza where the transaction occurred. The information from this field shall be used on customer statements to indicate the point of occurrence of the transaction. Values = 00 – 99.
EXIT_TOL_TRX_DATE	CHAR (10)	The date of the occurrence of the transaction at EXIT_TOL_LANE_ID. Format: MM/DD/YYYY. This toll transaction date information shall be shown on customer statements.
EXIT_TOL_TRX_TIME	CHAR (8)	The time of the occurrence of the transaction at EXIT_TOL_LANE_ID. Format: HH:MM:SS. This toll transaction time information shall be shown on customer statements.
TOL_FARE_ETC_AMT	CHAR (5,2)	The toll due as calculated by the HOT Lane / Host. This is the amount to be posted to the ETC home or away account, posting by Tag. Values: 00000 ($000.00) – 99999 ($999.99)

Field Name	Type/Size	Description/Valid Values
TOL_MSG_FLAG	CHAR (2)	The message buffer status flag. This field indicates whether or not a transaction was buffered. Values: 00-99. 1 – Toll packet transaction. 2 – Buffered tag transaction
TOL_AVC_CLASS	CHAR (2)	The class of the vehicle involved in the transaction. This field shall contain AVC class or as overridden by the collector classification. Values: Default 02
LANE_TX_SEQUENCE_NUMBER	CHAR (8)	The unique vehicle transaction sequence number generated by lane (Lane sequence number). Values:00000000 – 99999999
TOL_TAG_STATUS	CHAR (1)	The status of the tag at the time of the transaction. Values: 0 – 9 0 - Invalid 1 – Good 2 – Lost 3 – Stolen 4 – Low Balance 8 – Non-revenue vehicle (NRV)
TOL_DST_FLAG	CHAR (1)	The daylight savings time. The contents of this field shall be used to govern certain processing rules at the VECTOR CSC This field will always default to asterisk (*)
TOL_TRX_SPEED	CHAR (3)	The transaction speed as reported by the lane. Values 000 – 999
VIOL_NUMBER/ORIG_TRX_NUMBER	CHAR (10)	For ETC transactions this field will contain 0000000000
RESOLV_CODE	CHAR (2)	Default to 00
LINEFEED	CHAR (1)	LF
Detail Record Total	**84**	

Figure 24-3 ETC Transaction File – Trailer Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (8)	"#TRAILER"
SEQUENCE #	CHAR (6)	Same as Header
BUSINESS_DATE	CHAR (10)	File creation date, Format MM/DD/YYYY
DETAIL_COUNT	CHAR (8)	Total count of all detail records
DETAIL_TRANS_AMOUNT	CHAR (10)	Total Amount of the Amount Due field for all the transactions in the file
LINEFEED	CHAR (1)	LF
Trailer Total	**43**	

24.5 Processing requirements

1. The RCSC shall receive and process ETC Transaction Files from the HOT Host multiple times a day at predetermined intervals (viz. every 0.5hrs, 1hr etc to be determined later).
2. There are no violation transactions from the HOT lanes

3. Please refer to Appendix D for all transaction-processing rules.
4. ETC transactions in this file will have a unique transaction number for each record in the file.
5. All transactions coming in this interface will be processed and the resolve code values will be ignored.
6. The RCSC shall perform sanity checks on the ETC Transaction File to look for formatting errors, record count mismatch between header and detail records etc. In the event the file fails on these sanity checks, the VECTOR CSC shall notify the HOT Host of the anomaly by means of the acknowledgment file.
7. If the RCSC determines an error in a detail record, the VECTOR CSC shall reject the transaction record with the error and process the remainder of the transaction file and notify the HOT Host of the error via the acknowledgment file. The ACK file shall have a corresponding error code indicative of the error.
8. The RCSC shall not compute toll amounts for ETC transactions received from the HOT Host. The toll amount calculated at the HOT Host as supplied in the TOL_FARE_ETC_AMT field of the transaction file shall be used to debit the BATA Regional CSC accounts. This shall include transactions due to non-revenue customers also
9. RCSC has the capability of rejecting transaction based on the age of the transaction. VECTOR will set 180 days for all incoming transactions from Away Agency (TCA, SR91 or SNDG) and 365 days for all incoming transactions from Home Agencies (CALTRANS, GGBD, HOT). This value can be changed on BATA direction.
10. The RCSC shall first check its own customer base to see if the transaction can be applied to one of its own accounts before including the transaction in a Transaction File destined for another CTOC agency.
11. TOLL_DST_FLAG is not part of the unique key for toll transactions and there shall not be any duplicate values as a result of asterisks (*). Added to ICD 1.4.1
12. Agency/Facility/Plaza IDs:
    A. CTOC: RCSC sends the entry details (lane, plaza date and time) to CTOC as per CTOC ICD.
    B. Mail House: CSC sends both entry & exit information to the mail house and the mail house prints the entry and exit plazas and entry time detail on the customer statement.
    C. Fastrak Website: The customer accounts statement page will have the entry and exit plazas and entry time information available for I-680 and SR 237 transactions. Entry is entry and exit is exit. The plaza descriptions for the web will be (maximum of 20 characters): SR237/I880 ExpressWB and SR237/I880 ExpressEB.
    D. Vector Online: The exit plaza/lane/date & time and entry plaza/lane/date & time will be listed.
    E. Reports: RCSC reports only show the exit plaza.

24.6 Sample file

680_20110304020010_hreq

```
#HEADER,HREQ,000239,03/04/2011,H1,AT,03/04/2011,02:00:10
0000219959,1,0305,145006,WSH,01,03/03/2011,05:07:07,MIS,01,03/03/2011,05:08:04,00100,01,02,00000000,1
,*,000,0000000000,00
0000219960,1,0103,261404,AND,01,03/03/2011,05:14:47,MIS,01,03/03/2011,05:19:51,00125,01,02,00000000,1
,*,000,0000000000,00
0000219961,1,0132,261882,AND,01,03/03/2011,05:20:18,CAL,01,03/03/2011,05:28:24,00150,01,02,00000000,1
,*,000,0000000000,00
0000219962,1,0667,145001,WSH,01,03/03/2011,05:21:38,MIS,01,03/03/2011,05:22:35,00100,01,02,00000000,1
,*,000,0000000000,00
0000219963,1,0834,261834,AND,01,03/03/2011,05:25:48,MIS,01,03/03/2011,05:30:34,00125,01,02,00000000,1
,*,000,0000000000,00
0000219964,1,0885,261379,WSH,01,03/03/2011,05:29:55,MIS,01,03/03/2011,05:30:56,00100,01,02,00000000,1
,*,000,0000000000,00
0000219965,1,0289,261242,AND,01,03/03/2011,05:34:28,MIS,01,03/03/2011,05:39:19,00150,01,02,00000000,1
,*,000,0000000000,00
0000221358,1,0159,262047,MIS,01,03/03/2011,19:43:26,CAL,01,03/03/2011,19:47:09,00030,01,02,00000000,1
,*,000,0000000000,00
#TRAILER,000239,03/04/2011,00000008,0000000880
```

24.7 Plaza Id & Entry/Exit Mapping for RCSC

Below plaza-id & entry/exit mapping will be used by the RCSC for Vector view, statement processing & exchanging transactions with CTOC agencies.

Entry/Exit Point	Statement				CTOC Plaza ID
	Facility	Plaza	Lane	Facility Description	
Andrade Entry	I-680	AND	1	I-680 South Andrade	5010
Washington Entry	I-680	WSH	1	I-680 South Washington	5011
Mission Entry/Exit	I-680	MIS	1	I-680 South Mission	5012
Calaveras Exit	I-680	CAL	1	I-680 South Calaveras	5013

Valid Trip combinations

Trip	RCSC File Entry Plaza	RCSC File Exit Plaza	CTOC File Entry Plaza	CTOC File Entry Plaza	DPH Entry Plaza	DPH Exit Plaza
Andrade to Mission	AND	MIS	5010	5012	AND	WSH
Andrade to Calaveras	AND	CAL	5010	5013	AND	MIS
Washington to Mission	WSH	MIS	5011	5012	WSH	WSH
Washington to Calaveras	WSH	CAL	5011	5013	WSH	MIS
Mission to Calaveras	MIS	CAL	5012	5013	MIS	MIS

SR 237/I-880

Entry/Exit Point	Statement				CTOC Plaza ID
	Facility	Plaza	Lane	Facility Description	
SR 237/I-880 Express Connector WB	SR 237	CLW	1	SR 237/I-880 Connector WB	5110
SR 237 First WB	SR 237	FSW	1	SR 237 First WB	5111
SR 237 First EB	SR 237	FSE	1	SR 237 First EB	5118

SR 237/I-880 Express Connector EB	SR 237	CLE	1	SR 237/I-880 Connector EB	5119

Valid Trip combinations

Trip	RCSC File Entry Plaza	RCSC File Exit Plaza	CTOC File Entry Plaza	CTOC File Exit Plaza
SR 237/I-880 Express Connector WB	CLW	FSW	5110	5111
SR 237/I-880 Express Connector EB	FSE	CLE	5118	5119

25. ETC Response File – HOT

25.1 File type

Variable length, LF delimited

25.2 File name

680_YYYYMMDDHHMMSS.hres

Example: 680_20020928044100.hres Created at 04:41:00 on 09/28/02
Transaction Reconciliation file from VECTOR CSC to HOT Host

25.3 File use

The VECTOR CSC shall create an ETC Response File back to the HOT Host, for each transaction (.req) files received.

25.4 File layout

Each field in the header, detail and trailer structure will be separated with delimiter "," comma.

Figure 25-1 ETC Response File – Header Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (7)	"#HEADER"
FILE_TYPE	CHAR (4)	"HRES"
SEQUENCE #	CHAR (6)	Sequence # of the original Transaction File. Values 000000 – 999999
BUSINESS_DATE	CHAR (10)	The date send in the .req header record, in the BUINESS_DATE column, will be sent back in this field.
SOURCE	CHAR (2)	Indicates the destination entity. "AT" for BATA
DESTINATION	CHAR (2)	Indicates the file-creating agency. "H1" for HOT
CREATE_DATE	CHAR (10)	Indicates the file creation date. Format MM/DD/YYYY
CREATE_TIME	CHAR (8)	Indicates the file creation time. Format HH:MM:SS
LINEFEED	CHAR (1)	LF
Header Total	**50**	

Figure 25-2 ETC Response File – Detail Structure

Field Name	Type/Size	Description/Valid Values
TRANSACTION_NUMBER	CHAR (10)	Unique transaction number for which this record is response. Values 0000000000 to 9999999999
TOL_TRX_TYPE	CHAR (1)	Response for the type of transaction received by CSC. 1 – ETC
TOL_TAG_ID	CHAR (4)	This field consists of the ETC Internal Tag ID, in accordance with Title-21 specs. Values: 0000-1023
TOL_TAG_FACILITY_ID	CHAR (6)	This field comprises of the Facility code of the Issuing agency. Values: 000000-262143
TOL_POSTED_DATE	CHAR (10)	This is the Date the transaction was processed (Posted or Rejected) on the CSC / Away Agency. Format: MM/DD/YYYY
ENTRY_TOL_PLAZA_ID	CHAR (3)	The plaza code of the agency at which the transaction occurred. This information shall be shown on customer statements to indicate the place of occurrence of the transaction. Value = Refer to table 24.7
ENTRY_TOL_LANE_ID	CHAR (2)	The entry lane ID at the plaza where the transaction occurred. The information from this field shall be used on customer statements to indicate the point of occurrence of the transaction. Values = 00 – 99.
EXIT_TOL_PLAZA_ID	CHAR (3)	The plaza code of the agency at which the transaction occurred. This information shall be shown on customer statements to indicate the place of occurrence of the transaction. Value = Refer to table 24.7
EXIT_TOL_LANE_ID	CHAR (2)	The exit lane ID at the plaza where the transaction occurred. The information from this field shall be used on customer statements to indicate the point of occurrence of the transaction. Values = 00 – 99.
TOL_FARE_POSTED_AMT	CHAR (5,2)	This is the amount posted to the ETC home or away account, posting by Tag. Values: 00000 ($000.00) – 99999 ($999.99)
NON_REVENUE_FLAG	CHAR (2)	This field indicates if the transaction was posted against Non Revenue account. Values: 00 – Default Value 01 – Non Revenue Account
PAYMENT_TYPE	CHAR (1)	A – Toll posted successfully to ETC account. E – An Exception occurred while trying to post this toll.
CSC_REASON_CODE	CHAR (3)	Reason toll was not posted. CSC generates this code from its own internal processing and it is sent to the HOT Plaza Host for reference. Values 000 – 999. A detailed listing of the various reason codes is provided in Appendix B.
BUSINESS_DATE	CHAR (10)	The actual business date of the transaction. This field would identify the revenue date of the transaction. Format: MM/DD/YYYY
CSC_BATCH	CHAR (10)	This will be used to reconcile CSC and HOT Plaza Host revenue numbers. This field will contain the original file id (extern_file_id assigned by the RCSC), to map the file in which this transaction was received at the CSC. The contents of this field shall be left padded with zeros. Values: 0000000000 – 9999999999

Field Name	Type/Size	Description/Valid Values
CSC_ACCT_ID	CHAR (16)	Not currently used. Can be populated with home agency account IDs and static value CTOC agency account IDs. Current default value for all home and away agency accounts: 0000000000000000
LINEFEED	CHAR (1)	LF
Detail Record Total	**84**	

Figure 25-3 ETC Response File – Trailer Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (8)	"#TRAILER"
SEQUENCE #	CHAR (6)	Same as Header
FILE_DATE	CHAR (10)	File creation date, Format MM/DD/YYYY
DETAIL_COUNT	CHAR (8)	Total count of all detail records
LINEFEED	CHAR (1)	LF
Trailer Total	**33**	

25.5 Processing requirements

1. All transactions received at the RCSC, via the ETC Transaction File, shall be sent back to the HOT Host in the reconciliation file.

2. All regular transactions TOL_TRX_TYPE = 1 (ETC) received by the CSC, will be reconciled back with final status code. The reconciliation will be at file level. Example CSC receives 100 transactions in file 123, same 100 transactions will be reconciled back to HOT host in one file, no less than once a day.

3. The RCSC shall perform transaction reconciliation at a detail level. i.e. the reconciliation file shall contain details at the transaction level instead of a reconciliation summary.

4. In order to achieve file-to-file reconciliation with HOT, the RCSC will reconcile all transactions (home & away) with reconciled revenue. A reconciliation file will be sent to HOT once all transactions are reconciled (home & away) for each file.

5. In cases when a transaction cannot be posted at the RCSC, the RCSC shall indicate the reason, the transaction was not posted in the CSC_REASON_CODE field. The possible reason codes and the description are provided in Appendix B.

6. The RCSC shall assign a unique integer value to all incoming transaction files from the HOT Host. This unique identifier shall be sent as part of the reconciliation file to the HOT Host for all transactions posted and reconciled against a particular agency. The unique identifier shall be specified in the CSC_BATCH field of the reconciliation file.

7. The RCSC shall use the toll amount as supplied in the TOL_FARE_CASH_AMT field to process violations. All postable transactions shall use the amount in the TOL_FARE_ETC_AMT field

8. The HOT Host will periodically generate and transmit ETC files to the CSC. HOT will periodically poll the area ACK files are transferred to the Host by the CSC. When an ACK file is received the HOT database will be updated. If the ACK file shows a FAILURE code the HOT Host will regenerate and resend the original file. A failure count will be maintained and after 3 concurrent failures of a single file an email will be sent to the HOT System Operators.

9. A recon file will always be ACKED with a FAILURE code if it is received before the ACK file for the corresponding ETC Transaction File.

10. Any transaction without entry and exit information will be rejected with the appropriate status and not processed at the RCSC.

25.6 Sample File

680_20110304043301.hres

```
#HEADER,HRES,000239,03/04/2011,AT,H1,03/04/2011,04:33:01
0000219959,1,0305,145006,03/04/2011,WSH,01,MIS,01,00100,00,A,001,03/03/2011,0001633478,0000000000000000
0000219960,1,0103,261404,03/04/2011,AND,01,MIS,01,00125,00,A,001,03/03/2011,0001633478,0000000000000000
0000219961,1,0132,261882,03/04/2011,AND,01,CAL,01,00150,00,A,001,03/03/2011,0001633478,0000000000000000
0000219962,1,0667,145001,03/04/2011,WSH,01,MIS,01,00100,00,A,001,03/03/2011,0001633478,0000000000000000
0000219963,1,0834,261834,03/04/2011,AND,01,MIS,01,00125,00,A,001,03/03/2011,0001633478,0000000000000000
0000219964,1,0885,261379,03/04/2011,WSH,01,MIS,01,00100,00,A,001,03/03/2011,0001633478,0000000000000000
0000219965,1,0289,261242,03/04/2011,AND,01,MIS,01,00150,00,A,001,03/03/2011,0001633478,0000000000000000
0000221358,1,0159,262047,03/04/2011,MIS,01,CAL,01,00030,00,A,001,03/03/2011,0001633478,0000000000000000
#TRAILER,000239,03/04/2011,00000008
```

26. ETC Correction Request File – HOT

26.1 File type

Variable length, LF delimited

26.2 File name

<from_agency>_YYYYMMDDHHMMSS.creq

Example: 680_20020928044100.creq
HOT request for correction transactions to VECTOR CSC created at 04:41:00 on 09/28/02

26.3 File use

The correction file shall be created by the HOT Host to request toll posting corrections @ the RCSC. This file shall contain tagged transactions on HOT lanes that were previously reconciled by the RCSC and the HOT agency would be sending a correction request for such transactions.

26.4 File layout

Each field in the header, detail and trailer structure will be separated with Delimiter "," comma.

Figure 26-1 ETC Correction Transaction Request File – Header Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (7)	"#HEADER"
FILE_TYPE	CHAR (4)	"CREQ"
SEQUENCE #	CHAR (6)	Sequence # of the correction request file. This unique number is incremented for every file. Values 000000 – 999999
BUSINESS_DATE	CHAR (10)	This field will be populated with the transaction date of the first transaction in the file. Format MM/DD/YYYY
SOURCE	CHAR (2)	Indicates the file-creating agency. "H1" for HOT

DESTINATION	CHAR (2)	Indicates the destination entity. “AT” for BATA
CREATE_DATE	CHAR (10)	Indicates the file creation date. Format MM/DD/YYYY
CREATE_TIME	CHAR (8)	Indicates the file creation time. Format HH:MM:SS
LINEFEED	CHAR (1)	LF
Header Total	**50**	

Figure 26-2 ETC Correction Transaction Request File – Detail Structure

Field Name	Type/Size	Description/Valid Values
TRANSACTION_NUMBER	CHAR (10)	This must be the same as the original transaction. Values 0000000000 to 9999999999
TOL_TRX_TYPE	CHAR (1)	Type of transaction. 2 – ETC Correction This is the default value.
TOL_CORRECTION_FLAG	CHAR (1)	A – Corrected Amount
TOL_TAG_ID	CHAR (4)	This field consists of the ETC Internal Tag ID, in accordance with Title-21 specs. Values: 0000-1023
TOL_TAG_FACILITY_ID	CHAR (6)	This field comprises of the Facility code of the Issuing agency. Values: 000000-262143
ENTRY_TOL_PLAZA_ID	CHAR (3)	The entry plaza code of the agency at which the transaction occurred. This information shall be shown on customer statements to indicate the place of occurrence of the transaction. Value = Refer to table 24.7 (other HOT agencies will be given a new Plaza Id)
ENTRY_TOL_LANE_ID	CHAR (2)	The entry lane ID at the plaza where the transaction occurred. The information from this field shall be used on customer statements to indicate the point of occurrence of the transaction. Values = 00 – 99.
ENTRY_TOL_TRX_DATE	CHAR (10)	The date of the occurrence of the transaction at ENTRY_TOL_LANE_ID. Format: MM/DD/YYYY. This toll transaction date information shall be shown on customer statements.
ENTRY_TOL_TRX_TIME	CHAR (8)	The time of the occurrence of the transaction at ENTRY_TOL_LANE_ID. Format: HH:MM:SS. This toll transaction time information shall be shown on customer statements.
EXIT_TOL_PLAZA_ID	CHAR (3)	The exit plaza code of the agency at which the transaction occurred. This information shall be shown on customer statements to indicate the place of occurrence of the transaction. Value = Refer to table 24.7 Other HOT agencies will be given a new Plaza Id
EXIT_TOL_LANE_ID	CHAR (2)	The exit lane ID at the plaza where the transaction occurred. The information from this field shall be used on customer statements to indicate the point of occurrence of the transaction. Values = 00 – 99.
EXIT_TOL_TRX_DATE	CHAR (10)	The date of the occurrence of the transaction at EXIT_TOL_LANE_ID. Format: MM/DD/YYYY. This toll transaction date information shall be shown on customer statements.

Field Name	Type/Size	Description/Valid Values
EXIT_TOL_TRX_TIME	CHAR (8)	The time of the occurrence of the transaction at EXIT_TOL_LANE_ID. Format: HH:MM:SS. This toll transaction time information shall be shown on customer statements.
TOL_FARE_ETC_AMT	CHAR (5,2)	Original toll due as calculated by the HOT Lane / Host. This is the amount posted to the ETC home account, posting by Tag. Values: 00000 ($000.00) – 99999 ($999.99)
TOL_FARE_CORR_ETC_AMT	CHAR (5,2)	The new toll due as calculated by the HOT Lane / Host. Values: 00000 ($000.00) – 99999 ($999.99)
TOL_MSG_FLAG	CHAR (2)	The message buffer status flag. This field indicates whether or not a transaction was buffered. Values: 00-99. 01 – Toll packet transaction. 02 – Buffered tag transaction
TOL_AVC_CLASS	CHAR (2)	The class of the vehicle involved in the transaction. This field shall contain AVC class or as overridden by the collector classification. Values: Default 02
LANE_TX_SEQUENCE_NUMBER	CHAR (8)	The unique vehicle transaction sequence number generated by lane (Lane sequence number). Values:00000000 – 99999999
TOL_TAG_STATUS	CHAR (1)	The status of the tag at the time of the transaction. Values: 0 – 9 0 - Invalid 1 – Good 2 – Lost 3 – Stolen 4 – Low Balance 8 – Non-revenue vehicle (NRV)
TOL_DST_FLAG	CHAR (1)	The daylight savings time. The contents of this field shall be used to govern certain processing rules at the VECTOR CSC This field will always default to asterisk (*)
TOL_TRX_SPEED	CHAR (3)	The transaction speed as reported by the lane. Values 000 – 999
VIOL_NUMBER/ORIG_TRX_NUMBER	CHAR (10)	For ETC transactions this field will contain 0000000000
RESOLV_CODE	CHAR (2)	For ETC transactions this field will contain 00
LINEFEED	CHAR (1)	LF
Detail Record Total	**84**	

Figure 26-3 ETC Correction Transaction Request File – Trailer Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (8)	"#TRAILER"
SEQUENCE #	CHAR (6)	Same as Header
BUSINESS_DATE	CHAR (10)	File creation date, Format MM/DD/YYYY
DETAIL_COUNT	CHAR (8)	Total count of all detail records
DETAIL_TRANS_AMOUNT	CHAR (10)	Total Amount of the Amount Due field for all the transactions in the file
LINEFEED	CHAR (1)	LF
Trailer Total	**43**	

26.5 Processing requirements

1. This interface is for toll amount correction/update only.
2. HOT system will send a maximum of 1 correction file per day.
3. HOT system will check for posting status of original transactions prior to sending a correction request for the same transaction.
4. HOT system will send original toll posted amount (for validation purposes) along with the corrected amount.
5. RCSC will perform corrections only on transactions that have originally posted to Fastrak customer accounts.
6. This interface is for BATA/Home customers only. In case any CTOC transactions are included in this file, the RCSC will reject the entire file with an ack file status of 01.
7. There will only be 1 adjustment per 1 posted transaction.
8. The RCSC will create 2 new transactions based on this interface.
    i. Transaction #1 – will be posted to customer account as a reversal amount of the original posted amount.
    ii. Transaction #2 – will be the new amount received from HOT in the correction file.
9. There is no time limit for creation & processing of a correct file

26.6 Sample file

680_20040202222030.creq

```
680_20101210020000_creq

#HEADER,CREQ,000002,12/10/2010,H1,AT,12/10/2010,02:00:00
0000107830,2,A,0816,260805,AND,01,12/08/2010,19:56:06,MIS,01,12/08/2010,20:01:41,00050,00040,01,02,00
000000,1,*,000,0000000000,00
#TRAILER,000002,12/10/2010,00000001,-000000009
```

27. ETC Correction Response File – HOT

27.1 File type

Variable length, LF delimited

27.2 File name

680_YYYYMMDDHHMMSS.cres

Example: 680_20020928044100.cres Created at 04:41:00 on 09/28/02
Correction transaction reconciliation file from RCSC to HOT Host

27.3 File use

The RCSC shall create a correction response file back to the HOT Host, for each transaction (.creq) files received.

27.4 File layout

Each field in the header, detail and trailer structure will be separated with delimiter "," comma.

Figure 27-1 ETC Correction Response File – Header Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (7)	"#HEADER"
FILE_TYPE	CHAR (4)	"CRES"
SEQUENCE #	CHAR (6)	Sequence # of the original correction request File. Values 000000 – 999999
BUSINESS_DATE	CHAR (10)	The date send in the .creq header record, in the BUINESS_DATE column, will be sent back in this field.
SOURCE	CHAR (2)	Indicates the destination entity. "AT" for BATA
DESTINATION	CHAR (2)	Indicates the file-creating agency. "H1" for HOT
CREATE_DATE	CHAR (10)	Indicates the file creation date. Format MM/DD/YYYY
CREATE_TIME	CHAR (8)	Indicates the file creation time. Format HH:MM:SS

LINEFEED	CHAR (1)	LF
Header Total	**50**	

Figure 27-2 ETC Correction Response File – Detail Structure

Field Name	Type/Size	Description/Valid Values
TRANSACTION_NUMBER	CHAR (10)	This must be the same as the original transaction. Values 0000000000 to 9999999999
TOL_TRX_TYPE	CHAR (1)	Type of transaction. 2 – ETC Correction This is the default value.
TOL_TAG_ID	CHAR (4)	This field consists of the ETC Internal Tag ID, in accordance with Title-21 specs. Values: 0000-1023
TOL_TAG_FACILITY_ID	CHAR (6)	This field comprises of the Facility code of the Issuing agency. Values: 000000-262143
TOL_POSTED_DATE	CHAR (10)	This is the Date the correction transaction was processed (Posted or Rejected) on the CSC. Format: MM/DD/YYYY
ENTRY_TOL_PLAZA_ID	CHAR (3)	The plaza code of the agency at which the transaction occurred. This information shall be shown on customer statements to indicate the place of occurrence of the transaction. Value = Refer to table 24.7
ENTRY_TOL_LANE_ID	CHAR (2)	The entry lane ID at the plaza where the transaction occurred. The information from this field shall be used on customer statements to indicate the point of occurrence of the transaction. Values = 00 – 99.
EXIT_TOL_PLAZA_ID	CHAR (3)	The plaza code of the agency at which the transaction occurred. This information shall be shown on customer statements to indicate the place of occurrence of the transaction. Value = Refer to table 24.7
EXIT_TOL_LANE_ID	CHAR (2)	The exit lane ID at the plaza where the transaction occurred. The information from this field shall be used on customer statements to indicate the point of occurrence of the transaction. Values = 00 – 99.
TOL_FARE_POSTED_AMT	CHAR (5,2)	This is the amount posted to the home customer account (posting by Tag) This is the amount received in the TOL_FARE_CORR_ETC_AMT from the creq file Values: 00000 ($000.00) – 99999 ($999.99)
NON_REVENUE_FLAG	CHAR (2)	This field indicates if the transaction was posted against Non Revenue account. Values: 00 – Default Value 01 – Non Revenue Account
PAYMENT_TYPE	CHAR (1)	A – Toll posted successfully to ETC account. E – An Exception occurred while trying to post this toll.
CSC_REASON_CODE	CHAR (3)	Reason toll was not posted. CSC generates this code from its own internal processing and it is sent to the HOT Plaza Host for reference. Values 000 – 999. A detailed listing of the various reason codes is provided in Appendix B.
BUSINESS_DATE	CHAR (10)	The actual business date of the transaction. This field would identify the revenue date of the transaction. Format: MM/DD/YYYY

Field Name	Type/Size	Description/Valid Values
CSC_BATCH	CHAR (10)	This will be used to reconcile CSC and HOT Plaza Host revenue numbers. This field will contain the original file id (extern_file_id assigned by the RCSC), to map the file in which this transaction was received at the CSC. The contents of this field shall be left padded with zeros. Values: 0000000000 – 9999999999
CSC_ACCT_ID	CHAR (16)	Not currently used. Can be populated with home agency account IDs and static value CTOC agency account IDs. Current default value for all home and away agency accounts: 0000000000000000
LINEFEED	CHAR (1)	LF
Detail Record Total	**84**	

Figure 27-3 ETC Correction Trailer File – Detail Structure

Field Name	Type/Size	Description/Valid Values
RECORD_TYPE	CHAR (8)	"#TRAILER"
SEQUENCE #	CHAR (6)	Same as Header
FILE_DATE	CHAR (10)	File creation date, Format MM/DD/YYYY
DETAIL_COUNT	CHAR (8)	Total count of all detail records
LINEFEED	CHAR (1)	LF
Trailer Total	**33**	

27.5 Processing requirements

10. This interface is for toll amount correction/update only.
11. HOT system will send a maximum of 1 correction file per day.
12. HOT system will ensure that at any given point in time, only 1 correction request will be sent for a transaction that is posted to a customers account.
13. HOT system will check for posting status of original transactions prior to sending a correction request for the same transaction.
14. HOT system will send original toll posted amount (for validation purposes) along with the corrected amount.
15. RCSC will perform corrections only on transactions that have originally posted to Fastrak customer accounts.
16. There will only be 1 adjustment per 1 posted transaction.
17. This interface is for BATA/Home customers only. In case any CTOC transactions are included in this file, the RCSC will reject the entire file with an ack file status of 01.
18. The RCSC will create 2 new transactions based on this interface.
    i. Transaction #1 – will be posted to customer account as a reversal amount of the original posted amount.
    ii. Transaction #2 – will be the new amount received from HOT in the correction file.
19. There is no time limit for creation & processing of a correct file.

27.6 Sample file

```
#HEADER,CRES,000002,12/10/2010,AT,H1,12/10/2010,07:58:01
0000107830,2,0816,260805,12/10/2010,AND,01,MIS,01,00040,00,A,001,12/08/2010,0001557351
,0000000000000000
#TRAILER,000002,12/10/2010,00000001
```

Advanced Toll Collection and Accounting System (ATCAS II)

System Design Document

External Interfaces

Version 1.5.1

TransCore, LP

9440 Carroll Park Drive, Suite 150
San Diego, CA 92121
858-736-8200 telephone, 858-736-8353 fax

February 2012

Contract No. C002164, Project Number 1309150, Document Number 135D0119

Revision History

VERSION #	DATE	SECTION #	CHANGE DESCRIPTION
Version 1	January 2010	All	Initial Release
Version 1.1	April 2010	All	Update to support the customers' comments
Version 1.2	August 2010	All	Final System Design Document for customer review
Version 1.3	October 2010	All	Final System Design Document
Version 1.4	February 2011	All	Final System Design Document with Version 1.3 comments applied
Version 1,5.1	February 2012	All	As-Built Release with Version 1.4 comments applied

Table of Contents

10 EXTERNAL INTERFACES 10-1

10.1 REGIONAL CUSTOMER SERVICE CENTER INTERFACE 10-2
10.1.1 Architecture Overview 10-4
10.1.2 Business Rules 10-4
10.1.3 Process Activation 10-6
10.1.4 Interface Logic Flow 10-6
10.1.4.1 Logic Flow Example 10-10
10.1.5 Exchanging Data to and from the RCSC 10-13
10.1.5.1 File Transfer 10-13
10.1.5.2 Directory and File Naming Conventions 10-13
10.1.6 File Layouts and Validation 10-15
10.1.6.1 Tag Status File Received from the RCSC 10-16
10.1.6.2 ETC Transaction File Sent to the RCSC 10-20
10.1.6.3 Violation Transaction File Sent to the RCSC 10-25
10.1.6.4 Transaction Correction File Sent to the RCSC 10-31
10.1.6.5 ETC Response File Received from the RCSC 10-39
10.1.6.6 Violation Response File Received from the RCSC 10-43
10.1.6.7 Correction Response File Received from the RCSC 10-47
10.1.6.8 Acknowledgement File Sent to and Received from the RCSC 10-52
10.1.6.9 Violation Data Files and Images Sent to the RCSC 10-53
10.1.6.10 Incoming File Validation 10-54
10.1.6.11 RCSC Reason Codes 10-56
10.1.7 Business Day Tracking 10-57
10.1.8 Error Processing 10-60
10.1.9 RCSC Interface Database Schema 10-60
10.2 BANK INTERFACE 10-63
10.2.1 Architecture Overview 10-63
10.2.2 Business Rules 10-64
10.2.3 Process Activation 10-64
10.2.4 Interface Logic Flow 10-65
10.2.4.1 Logic Flow Example 10-66
10.2.5 Exchanging Data to and from the Bank 10-67
10.2.5.1 File Transfer 10-67
10.2.5.2 Directory and File Naming Conventions 10-68
10.2.6 File Layouts and Validation 10-69
10.2.6.1 Collector Cash Report Sent to the Bank 10-69
10.2.6.2 Collector Cash Report Received Response from the Bank 10-70
10.2.6.3 Cash Deposit Confirmation Received from the Bank 10-70
10.2.6.4 Cash Deposit Confirmation Received Response to the Bank 10-74
10.2.6.5 Collector Cash Report Sent in an Email to Finance 10-75
10.2.6.6 Cash Deposit Confirmation Sent in an Email to Finance 10-76
10.2.6.7 Incoming File Validation 10-76
10.2.7 Business Day Tracking 10-77
10.2.8 Error Processing 10-78
10.2.9 Bank Interface Database Schema 10-79
10.3 INTEGRATED FINANCIAL AND ADMINISTRATIVE SOLUTION INTERFACE 10-80
10.3.1 Architecture Overview 10-80
10.3.2 Business Rules 10-81
10.3.3 Process Activation 10-81

10.3.4 Interface Logic Flow 10-82
10.3.4.1 Logic Flow Example 10-83
10.3.5 Sending Data to BATA 10-85
10.3.5.1 File Transfer 10-85
10.3.5.2 Directory and File Naming Conventions 10-85
10.3.6 File Layout 10-85
10.3.7 Distribution of Fare Amounts Across IFAS Fund Codes 10-97
10.3.8 Error Processing 10-99
10.3.9 IFAS Interface Database Schema 10-99

List of Figures

FIGURE 10-1: RCSC INTERFACE SYSTEM ARCHITECTURE 10-4
FIGURE 10-2: RCSC INTERFACE FLOW DIAGRAM (PART 1) 10-7
FIGURE 10-3: RCSC INTERFACE FLOW DIAGRAM (PART 2) 10-8
FIGURE 10-4: RCSC INTERFACE FLOW DIAGRAM (PART 3) 10-9
FIGURE 10-5: RCSC INTERFACE PROCESS DATABASE SCHEMA 10-61
FIGURE 10-6: BANK INTERFACE SYSTEM ARCHITECTURE 10-63
FIGURE 10-7: BANK INTERFACE FLOW DIAGRAM 10-65
FIGURE 10-8: SAMPLE COLLECTOR CASH REPORT RECEIVED RESPONSE FROM BANK 10-70
FIGURE 10-9: SAMPLE CASH DEPOSIT CONFIRMATION RECEIVED RESPONSE TO BANK 10-74
FIGURE 10-10: SAMPLE COLLECTOR CASH REPORT EMAIL TO FINANCE 10-75
FIGURE 10-11: SAMPLE CASH DEPOSIT CONFIRMATION EMAIL TO FINANCE 10-76
FIGURE 10-12: BANK INTERFACE PROCESS DATABASE SCHEMA 10-79
FIGURE 10-13: IFAS INTERFACE SYSTEM ARCHITECTURE 10-80
FIGURE 10-14: IFAS INTERFACE FLOW DIAGRAM 10-82
FIGURE 10-15: IFAS INTERFACE PROCESS DATABASE SCHEMA 10-100

PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 10-1: RCSC Interface Files 10-2
Table 10-2: Tag Status Mapping Between RCSC and ATCAS II 10-5
Table 10-3: RCSC Interface Processing 10-10
Table 10-4: RCSC Interface Directory Structure on the FTP Server 10-13
Table 10-5: RCSC Interface Files and Naming Conventions 10-14
Table 10-6: Local Tag Status File Received from the RCSC (Header Record) 10-16
Table 10-7: Away Tag Status File Received from the RCSC (Header Record) 10-16
Table 10-8: Local Tag Status File Received from the RCSC (Detail Record) 10-17
Table 10-9: Away Tag Status File Received from the RCSC (Detail Record) 10-18
Table 10-10: Tag Status File Received from the RCSC (Trailer Record) 10-19
Table 10-11: ETC Transaction File Sent to the RCSC (Header Record) 10-20
Table 10-12: ETC Transaction File Sent to the RCSC (Detail Record) 10-21
Table 10-13: ETC Transaction File Sent to the RCSC (Trailer Record) 10-24
Table 10-14: Violation Transaction File Sent to the RCSC (Header Record) 10-25
Table 10-15: Violation Transaction File Sent to the RCSC (Detail Record) 10-26
Table 10-16: Violation Transaction File Sent to the RCSC (Trailer Record) 10-29
Table 10-17: Transaction Correction File Sent to the RCSC (Header Record) 10-31
Table 10-18: Transaction Correction File Sent to the RCSC (Detail Record) 10-32
Table 10-19: Transaction Correction File Sent to the RCSC (Trailer Record) 10-36
Table 10-20: ETC Response File Received from the RCSC (Header Record) 10-39
Table 10-21: ETC Response File Received from the RCSC (Detail Record) 10-40
Table 10-22: ETC Response File Received from the RCSC (Trailer Record) 10-42
Table 10-23: Violation Response File Received from the RCSC (Header Record) 10-43
Table 10-24: Violation Response File Received from the RCSC (Detail Record) 10-44
Table 10-25: Violation Response File Received from the RCSC (Trailer Record) 10-46
Table 10-26: Correction Response File Received from the RCSC (Header Record) 10-47
Table 10-27: Correction Response File Received from the RCSC (Detail Record) 10-48
Table 10-28: Correction Response File Received from the RCSC (Trailer Record) 10-50
Table 10-29: Acknowledgement File Sent to and Received from the RCSC 10-52
Table 10-30: Violation Data File Sent to the RCSC 10-53
Table 10-31: Tag Status File Validation 10-54
Table 10-32: ETC and Violation Response File Validation 10-54
Table 10-33: Acknowledgement File Validation 10-55
Table 10-34: Violation Data File Validation 10-55
Table 10-35: RCSC Reason Codes 10-56
Table 10-36: tbBusinessDay RCSC Interface Field Values 10-58
Table 10-37: Bank Interface Processing 10-66
Table 10-38: Bank Interface Directory Structure on the sFTP Server 10-68
Table 10-39: Bank Interface Files and Naming Conventions 10-68
Table 10-40: Collector Cash Report Sent to the Bank (Header Record) 10-69
Table 10-41: Collector Cash Report Sent to the Bank (Detail Record) 10-69
Table 10-42: Collector Cash Report Sent to the Bank (Trailer Record) 10-70
Table 10-43: Cash Deposit Confirmation Received from the Bank (Header Record) 10-71
Table 10-44: Cash Deposit Confirmation Received from the Bank (Detail Record) 10-71
Table 10-45: Cash Deposit Confirmation Received from the Bank (Plaza Trailer Record) 10-72
Table 10-46: Cash Deposit Confirmation Received from the Bank (Plaza Separator Record) 10-72
Table 10-47: Cash Deposit Confirmation Received from the Bank (Trailer Record) 10-73

TABLE 10-48: CASH DEPOSIT CONFIRMATION FILE VALIDATION 10-76
TABLE 10-49: TBBUSINESSDAY BANK INTERFACE FIELD VALUES 10-77
TABLE 10-50: IFAS INTERFACE PROCESSING 10-83
TABLE 10-51: IFAS INTERFACE FILE NAMING CONVENTIONS 10-85
TABLE 10-52: IFAS IMPORT FILE 10-87
TABLE 10-53: IFAS FUND CODES 10-94
TABLE 10-54: IFAS PROGRAM NUMBERS 10-95
TABLE 10-55: IFAS PROGRAM IDENTIFIERS 10-95
TABLE 10-56: IFAS OBJECT CODES 10-96
TABLE 10-57: STBIFASFUNDCODE VALUES 10-98
TABLE 10-58: STBFAREDETAIL VALUES 10-98

10 External Interfaces

The external interfaces operate as back end processes primarily written in Transact SQL, using assembly language built in C#, to communicate with non-SQL Server resources such as the file system and external interfaces.

The overall architecture of the back office is based on queue and work flow. Refer to Appendix J Job Scheduling for more details on when and how these jobs are scheduled to run.

The following external interfaces are documented in this chapter:

- **RCSC Interface –** This describes the Regional Customer Service Center (RCSC) External Interface
- **Bank Interface –** This describes the Bank External Interface
- **IFAS Interface –** This describes the Integrated Financial and Administrative Solution (IFAS) External Interface

10.1 Regional Customer Service Center Interface

The Regional Customer Service Center (RCSC) interface supports the exchange of Electronic Toll Collection (ETC) transponder statuses from the RCSC to the ATCAS II Host, and the exchange of ETC toll transactions, toll violation transactions, toll transaction cancellations and corrections, and violation images from the ATCAS II Host and the RCSC. Response files are sent by the RCSC to the ATCAS II Host for the ETC toll transactions and toll violation transactions to indicate the disposition of these transactions.

The transponder status, transaction, and response files are zipped before being transferred through BATA FTP server. The receiving system sends an acknowledgement file through the FTP servers indicating the status of each file received. A violation image data file is created by the Image Server at the plaza for each set of Violation Enforcement System (VES) images that can be associated with a lane transaction. The violation image data file and corresponding images are zipped into a violation package and sent to the Host Image Server (see Chapter 6 Section 6.6.2.4). All new violation packages that can be associated with violation transactions posted to the ATCAS II Host Database since the last transfer are sent to the RCSC through the BATA FTP server.

The following table indicates which files are sent to the ATCAS II Host and which are sent to the RCSC, the file extension identifies the type of file, and how the file is used. For zipped files, the period that precedes the file extension is replaced with an underscore ("_") and a new .zip file extension is added to the file name. The table also indicates which files are zipped.

Table 10-1: RCSC Interface Files

FILE	FROM -> TO	FILE EXT	ZIP'D	FILE USAGE
Tag Status File	RCSC -> ATCAS II	AETC	Yes	Contains the status for each ETC Transponder at the RCSC and other CTOC agencies. The other CTOC agency tags are stored in separate files with file extension ".tag" and a prefix to the file name indicates which away agency tags are included as follows: "srat_" = SR-91 "tcat_" = TCA "sdat_" = SANDAG "cvat_" = SBX These away agency files are included in the same zip file as the local RCSC file (file extension ".AETC".
ETC Transaction File	ATCAS II -> RCSC	AREQ	Yes	Contains ETC transactions with a valid tag status
Violation Transaction File	ATCAS II -> RCSC	AVIO	Yes	Contains violation transactions with or without a transponder
Correction File	ATCAS II -> RCSC	CTRE	Yes	Contains ETC and violation transaction reversals, late postings, and corrections

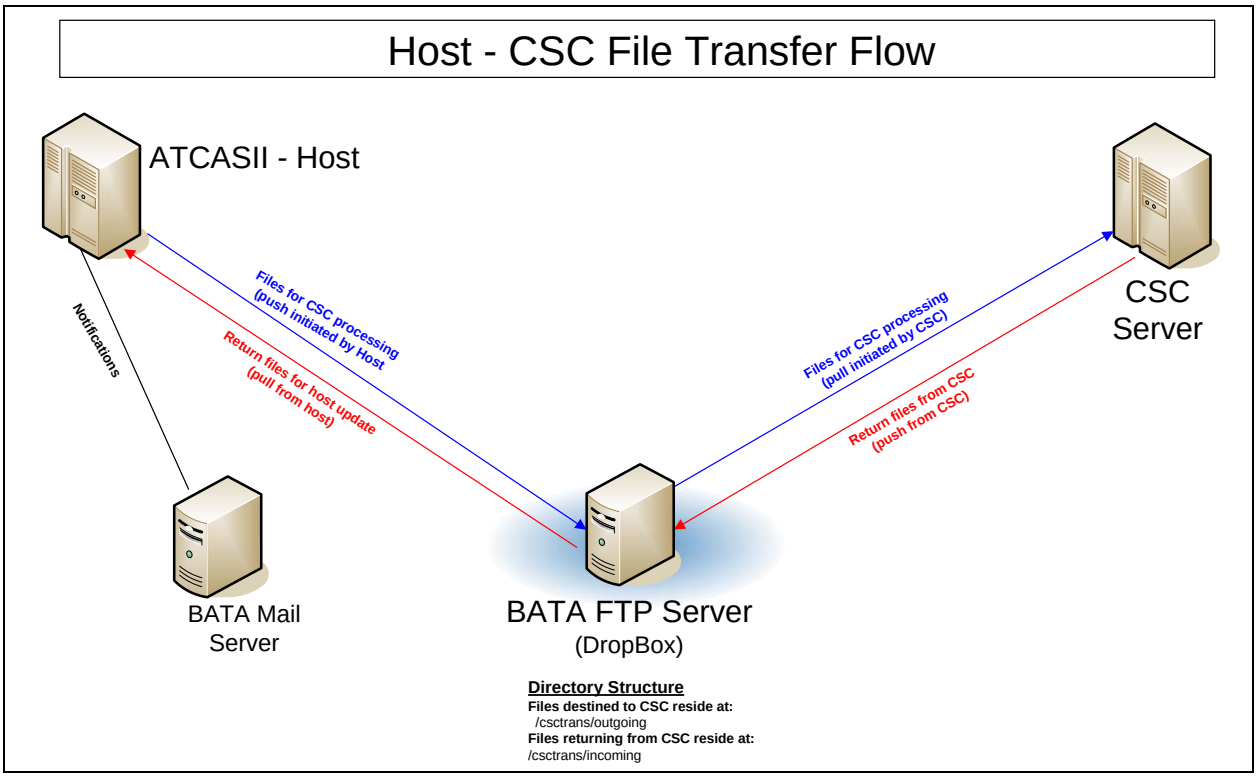
FILE	FROM -> TO	FILE EXT	ZIP'D	FILE USAGE
ETC Response File	RCSC -> ATCAS II	ARES	Yes	Contains the disposition for ETC transactions that were previously sent. Dispositions are posted to an account based on the tag status at the lane or rejected based on an error exception
Violation Response File	RCSC -> ATCAS II	AVRES	Yes	Contains the disposition for violation transactions that were previously sent. Dispositions are posted to an account based on a change in tag status or based on the violation image or rejected based on an error exception
Correction Response File	RCSC -> ATCAS II	RTRE	Yes	Contains the disposition for correction transactions that were previously sent. Dispositions are posted to an account based on the tag status at the lane or rejected based on an error exception
Acknowledgment File	Both Directions	ACK	No	Acknowledges the receipt of a file and indicates whether or not the file was valid
Violation Image Data File and Corresponding Image Files	ATCAS II -> RCSC	VDF and <IMAGE _TYPE>	Yes (as a group)	Zipped file containing the violation image data file and corresponding image files. The violation image data file name indicates the <AGENCY_ID><PLAZA_ID> <LANE_ID><TRX_DATE><TRX_TIME> <VEHICLE_SEQUENCE_NUMBER> so that it can be matched to the actual violation transaction.

10.1.1 Architecture Overview

The RCSC Interface system architecture is shown in Figure 10-1. Files are exchanged between the BATA FTP server and the RCSC via a dedicated WAN connection.

The access information for the FTP server is a configurable parameter in the tbApplParam table.

Figure 10-1: RCSC Interface System Architecture



10.1.2 Business Rules

The following is a list of business rules for the RCSC interface:

- Data transferred between the RCSC interface processes and the RCSC are file-based.
- The RCSC interface provides transaction response files that give a disposition status for each of the ETC and violation transactions passed in the ETC transaction and violation transaction files.
- The RCSC interface process maintains a tbRcscLastTransProcessed table to keep track of the Host database posting date/time for the last ETC/violation transaction, the last business day approval, and the last images and violation data file passed to the RCSC for each plaza and lane. This is so the process knows at what posting date/time it should start picking up new data for the next run.
- Transactions assigned to different business days are not passed in the same transaction file. The file's header record indicates the business day associated with all transactions contained in the file.

- Transaction corrections are approved for the previous business day as part of the business day approval process. The business day field is updated and a record inserted into the tbBusinessDay table. A transaction correction file is created for each business day approval and contains any adjustments done during the corresponding business day. Transactions from earlier business days can be cancelled or corrected in the correction file but the adjustments are applied to the business day during which the adjustments were made. Please see Chapter 8 Host System Software Section 8.10.1 Business Day Complete for more details on the business day approval process.
- The RCSC interface process only sends violation data files and images corresponding to violation transactions that have also been sent.
- The tbTag table is used to maintain the tag status in the local ATCAS II Host database and the individual tag status values are compared to the values in the new tag status file. When the tag status values are different, the tbTag table is updated. A trigger is defined so that whenever a tag status is updated it is inserted into the tbSendLaneUpdate table. The File Creator (FC) task monitors the tbSendLaneUpdate table and sends the tag status updates to all the lane/zone controllers as non-guaranteed messages. A new tag status file with all current tag status values is created by FC and downloaded to all the lane/zone controllers nightly. Table 10-2 shows the mapping between the RCSC tag and account statuses to the ATCAS II tag status and tag attribute values. Please see Chapter 8 Host System Software for more details related to the FC task.

Table 10-2: Tag Status Mapping Between RCSC and ATCAS II

ITEM #	RCSC				ATCAS II	
	TAG STATUS	ACCOUNT STATUS	FINANCIAL STATUS	DISCOUNT PLAN	TAG STATUS	ATTRIBUTE
1	INVENTORY	N/A	N/A	N/A	4 = Invalid	0 = Standard
2	RETURNED	N/A	N/A	N/A	4 = Invalid	0 = Standard
3	DAMAGED	N/A	N/A	N/A	4 = Invalid	0 = Standard
4	RETURNEDEF	N/A	N/A	N/A	4 = Invalid	0 = Standard
5	SHIPVEND	N/A	N/A	N/A	4 = Invalid	0 = Standard
6	TESTED	N/A	N/A	N/A	4 = Invalid	0 = Standard
7	EXPIRED	N/A	N/A	N/A	4 = Invalid	0 = Standard
8	LOST	Active	N/A	N/A	5 = Lost/Stolen	0 = Standard
9	STOLEN	Active	N/A	N/A	5 = Lost/Stolen	0 = Standard
10	ACTIVE	Active	Good Balance	Standard	0 = Valid	0 = Standard
11	ACTIVE	Active	Low Balance	Standard	1 = Low Bal	0 = Standard
12	ACTIVE	Active	Zero Balance	Standard	4 = Invalid	0 = Standard
13	ACTIVE	Active	Pending Revoked	Standard	4 = Invalid	0 = Standard
14	ACTIVE	Active	Good Balance	Non Revenue	0 = Valid	1 = Non Rev

ITEM #	RCSC				ATCAS II	
	TAG STATUS	ACCOUNT STATUS	FINANCIAL STATUS	DISCOUNT PLAN	TAG STATUS	ATTRIBUTE
15	ACTIVE	Active	Low Balance	Non Revenue	0 = Valid	1 = Non Rev
16	ACTIVE	Active	Zero Balance	Non Revenue	0 = Valid	1 = Non Rev
17	ACTIVE	Active	Pending Revoked	Non Revenue	0 = Valid	1 = Non Rev
18	ACTIVE	Active	Good Balance	Standard with Hybrid Tag[1]	0 = Valid	8 = Hybrid
19	ACTIVE	Active	Low Balance	Standard with Hybrid Tag	1 = Low Bal	8 = Hybrid
20	ACTIVE	Active	Zero Balance	Standard with Hybrid Tag	4 = Invalid	8 = Hybrid
21	ACTIVE	Active	Pending Revoked	Standard with Hybrid Tag	4 = Invalid	8 = Hybrid
22	N/A	Closed	N/A	Standard	4 = Invalid	0=Standard
23	N/A	Closed	N/A	Non Revenue	4 = Invalid	1 = Non Rev
24	N/A	Closed	N/A	Standard with Hybrid Tag	4 = Invalid	8 = Hybrid

10.1.3 Process Activation

The following process activation configuration is used:
The RCSC interface process runs once every 3 hours and is configurable (see Appendix J Job Scheduling).

Generating and processing files is initiated by the Windows Scheduler as a .NET application on the External Interface Server running as a virtual machine on the Virtual Host (see Appendix J Job Scheduling).

10.1.4 Interface Logic Flow

The RCSC interface process starts by updating the table used to keep track of the next batch of data to be processed and then performs additional processing if new data is found in any of the following areas:

- New ETC or Violation Transaction data in the Database
- New Business Day Approval and Transaction Corrections in the Database
- New Violation Data File on the Image Server
- New Tag Status File on the BATA FTP Server – pushed there by the RCSC

[1] Hybrid is not a discount plan. It is determined by the RCSC based on the Hybrid attribute. Hybrid is only reported to ATCAS II for the Standard discount plan.

- New Transaction Response Files on the BATA FTP Server – pushed there by the RCSC
- New Acknowledgement File on the BATA FTP Server – pushed there by the RCSC

The following three figures illustrate the flow chart diagrams for the RCSC interface processes. Section 10.1.4.1 Logic Flow Example follows the flow chart diagrams and gives a detailed explanation of the logic flow.

Figure 10-2: RCSC Interface Flow Diagram (Part 1)

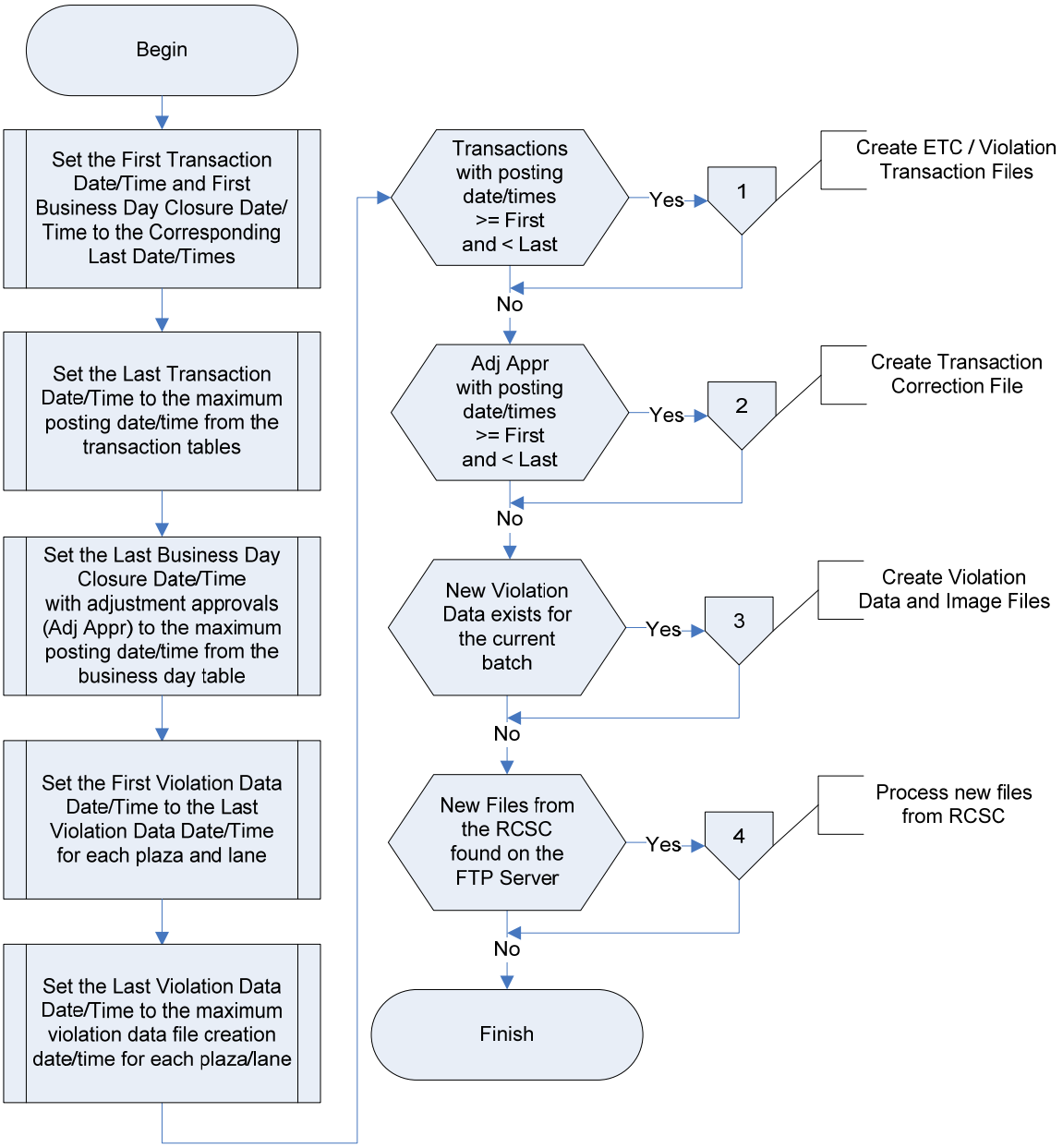


Figure 10-3: RCSC Interface Flow Diagram (Part 2)

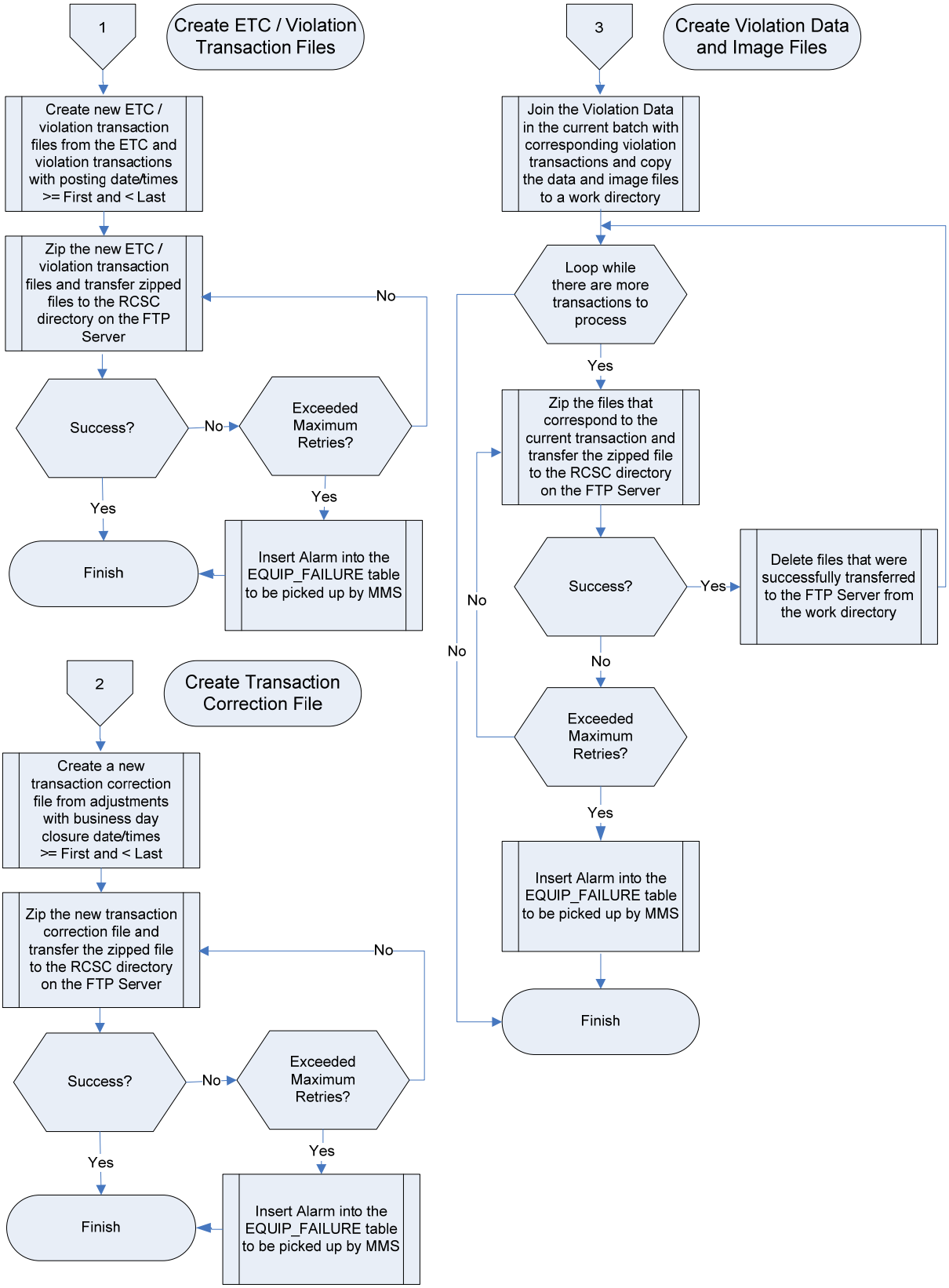
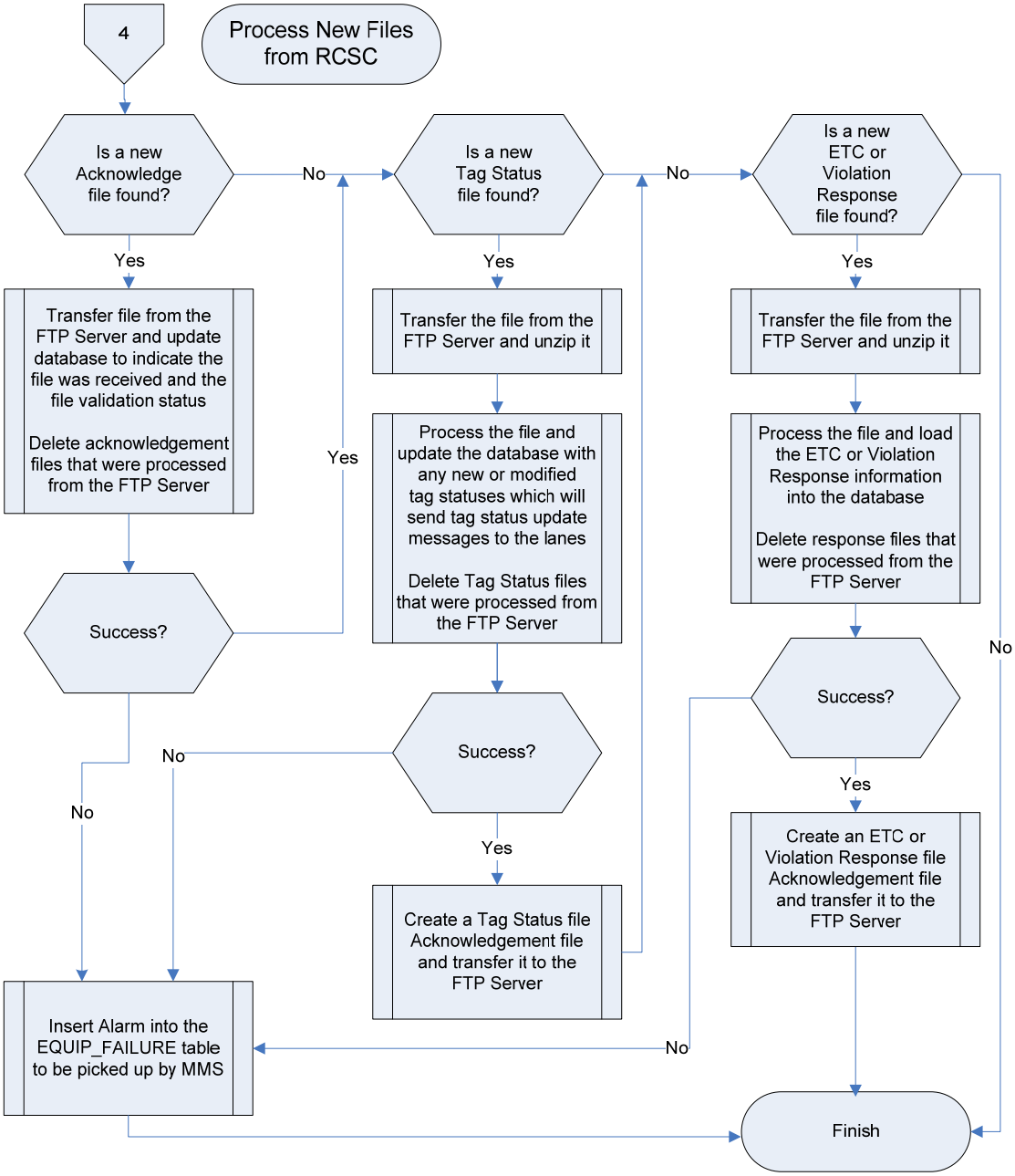


Figure 10-4: RCSC Interface Flow Diagram (Part 3)



10.1.4.1 Logic Flow Example

The RCSC interface process runs on a configurable schedule to generate ETC Transaction and Violation Transaction files to be sent to the RCSC. At the same time the Violation Data and Image Files are sent to the RCSC that correspond to the violations being sent in the Violation Transaction File. The RCSC interface process then checks for File Acknowledgment, AVI Status, and Transaction Response files and processes them accordingly.

The RCSC interface process maintains a tbRcscLastTransProcessed table to keep track of the database posting date time of the ETC/violation transaction, business day closure, and image data transaction that was last picked up for processing from the host database. The next time the process runs, the "LastPostTime" is set to the "NextPostTime" for each type of selection and the "NextPostTime" is set to the maximum posting date time currently available in the database for each type of selection as follows:

- MAX(dtPostingDate) FROM tbFullFormatTrans
- MAX(dtBusinessDayClosed) FROM tbBusinessDay
- MAX(dtPostingDate) FROM tbImageTrans

The RCSC interface then selects data that is greater than the "LastPostTime" and greater than or equal to the "NextPostTime" for the next batch of transactions to be processed.

The RCSC interface process is scheduled to run as indicated in Appendix J Job Scheduling. Using the example where the RCSC interface process is scheduled to run on a 3 hour cycle, it runs at 12:00 midnight, 3:00 am, 6:00 am, 9:00 am, 12:00 noon, 3:00 pm, 6:00 pm, 9:00 pm. When the process runs at midnight, for example, it performs the steps outlined in Table 10-3.

Table 10-3: RCSC Interface Processing

STEP #	RCSC INTERFACE PROCESSING
1	Creates an ETC Transaction File with all ETC transactions in the tbFullFormatTrans table that have the same business day and were posted to the database after the latest ETC transaction previously sent to the RCSC for each plaza and lane. Late posting adjustments in the tbAdjustmentTrans table that have the same business day and were approved after the latest ETC transaction are also included. A separate ETC Transaction file is created for each business day if transactions for more than one business day are selected. Records the new file(s) in the tbRcscSendFile and tbRcscSendTrans tables.
2	Zips each ETC Transaction File and transfers it to the RCSC through the BATA FTP Server.

STEP #	RCSC INTERFACE PROCESSING
3	Creates a Violation Transaction File with all of violation transactions in the tbFullFormatTrans and tbAdjustmentTrans tables that can be joined with violation data in the tbImageTrans table, have the same business day and were posted to the database after the latest violation transaction and violation data previously sent to the RCSC for each plaza and lane. A separate Violation Transaction file is created for each business day if transactions for more than one business day are selected. Violations that can be joined with a tbImageTrans record are stored in the tbRcscWaitForVioImage table. When the violation image data file is uploaded to RCSC, the related record in the tbRcscWaitForVioImage table is removed and the violation reported. If a violation resides in the tbRcscWaitForVioImage table over a configurable length of time (initially set for 4 hours), an alarm is generated as indicated in Section 10.1.8 Error Processing. These transactions can be sent to the RCSC once the missing data becomes available at the ATCAS II host. Any record in the tbRcscWaitForVioImage table older than 60 days is deleted. Variances between the violations reported by the lane and the violations reported to the RCSC are shown as violation variances in the Audit Violation Variances pages described in Chapter 8 Section 8.10.4 and in the Violation Registration to VES and RCSC Acknowledgement Reconciliation report (see Chapter 12 Reports). Records the new file(s) in the tbRcscSendFile and tbRcscSendTrans tables.
4	Zips each Violation Transaction File and transfers it to the RCSC through the BATA FTP Server.
5	Creates a Correction File with approved adjustments in the tbAdjustmentTrans table. Adjustments are selected for the Correction File as follows: The RCSC interface keeps track of the latest business day closure date/time previously sent to the RCSC for each plaza The RCSC interface finds the approved correction and reversal adjustments for a business day that closed after the latest business day closure date/time previously sent to the RCSC for each plaza A separate Correction file is created for each business day selected with approved correction and reversal adjustments and is sent to the RCSC. Adjustments are considered “approved” after being flagged using the Review Adjustment page described in Chapter 8 Section 8.10.5 Adjustment. Adjustments are only sent to the RCSC after the corresponding business day has closed (see Chapter 8 Section 8.10.1 Business Day Complete). For example, consider the case where the business days for Monday and Tuesday were left open. Adjustments were then made and approved on Wednesday for transactions that occurred on both Monday and Tuesday and for the previous Friday (which is closed). After making and approving the adjustments, the Revenue Manager closes the business days for Monday and Tuesday. The next time the RCSC interface process runs, two Correction files are created, one each for Monday and Tuesday. After the business day for Wednesday is closed, a Correction file is created with the adjustments made for the previous Friday because they are assigned a business day of Wednesday. Late posting Adjustments are sent in the ETC Transaction File or Violation Transaction File without any corresponding record in the Correction File (see steps 1 and 3). The new corrections are recorded in the tbRcscSendFile and tbRcscSendCorrections tables.
6	Zips each Correction File and transfers it to the RCSC through the BATA FTP Server.

STEP #	RCSC INTERFACE PROCESSING
7	For each violation sent in the Violation Transaction File, validates the violation data file, zips up the corresponding images and violation data file, and transfers the zipped file to the RCSC through the BATA FTP Server. Records each new file in the tbRcscSendFile table. For details on the validation of the violation data file, refer to Section 10.1.6.10.
8	Processes each Acknowledgment File found on the BATA FTP Server and records it in the tbRcscReceiveFile table. For details on the validation of the Acknowledgment File, refer to Section 10.1.6.10 Incoming File Validation.
9	Deletes the Acknowledgment Files processed from the BATA FTP Server.
10	Processes each Tag Status File found on the FTP Server in the order they were created. There should be only one zip file containing a Tag Status File for the local RCSC and separate Tag Status Files for each of the away CTOC agencies. The process then records the file in the tbRcscReceiveFile and tbRcscReceiveTagStatus tables. The tbTag table is updated with the latest tag statuses. This causes a record to be inserted in the tbSendLaneUpdate table so that the new status is distributed to all the lane/zone controllers. If a new CTOC facility code is received that is not already covered in the stbIOPAgencyFacilityRange table, the RCSC interface process automatically adds the new code to the stbIOPAgencyFacilityRange table and associates it with the appropriate CTOC agency based on the SOURCE value passed in the file's header record. For details on the validation of the Tag Status File, refer to Section 10.1.6.10 Incoming File Validation.
11	Creates an Acknowledge File for each Tag Status File processed and transfers it to the RCSC through the BATA FTP Server.
12	Deletes the Tag Status Files processed from the BATA FTP Server.
13	Processes each ETC Response File and Violation Response File found on the BATA FTP Server and records it in the tbRcscReceiveFile table. ETC and violation response records are inserted into the tbRcscReceiveResponse table for each valid ETC and violation response received. For details on the validation of the ETC Response File and Violation Response File, refer to Section 10.1.6.10 Incoming File Validation.
14	Creates an Acknowledge File for each ETC Response File and Violation Response File processed and transfers it to the RCSC through the BATA FTP Server.
15	Deletes the ETC Response Files and Violation Response Files processed from the BATA FTP Server.
16	Performs the error processing outlined in Section 10.1.8 Error Processing.

10.1.5 Exchanging Data to and from the RCSC

The RCSC interface process runs on a configurable schedule to check for new ETC toll transactions, toll violation transactions, adjustments made for a closed business day, and violation images to be sent to the RCSC via the BATA FTP Server as well as for new incoming files on the BATA FTP Server from the RCSC. This section defines the file transfer, the outgoing and incoming file directory structure and file naming conventions.

10.1.5.1 File Transfer

ATCAS II Host is responsible for picking up files from the BATA FTP Server and dropping off files on the BATA FTP Server. This is the same FTP server used for ATCAS I. There is a dedicated WAN between the BATA and RCSC server and there is a firewall on both sides to prevent intrusions into the respective networks. An FTP user, password, and folder is provided for the FTP server so that the RCSC interface can be configured to access necessary directories on the FTP server.

The file transfer is accomplished using FTP.

Outgoing files are transferred from where they are created in the Outgoing directory on the ATCAS II Host to the csctrans/outgoing directory on the BATA FTP server. It is the responsibility of the RCSC to clean up outgoing files on the FTP server once the transfer is complete.

Incoming files are transferred from the csctrans/incoming directory on the BATA FTP server to the Incoming directory on the ATCAS II Host. Files with a file type of “zip” are extracted from the zip file. The extracted file is then deleted after processed and the zip file is archived in the Archive directory.

Transferred incoming files are deleted from the BATA FTP server and the file type is used to determine how each incoming file is processed. The name of the FTP server and the names of outgoing and incoming directories on the ATCAS II host and on the FTP server are configurable in the ATCAS II Host database.

10.1.5.2 Directory and File Naming Conventions

Below is the directory structures used by the RCSC interface process:

Table 10-4: RCSC Interface Directory Structure on the FTP Server

FTP SERVER	DIRECTORY	DESCRIPTION
BATA FTP Server	/csctrans/incoming	All files received from the RCSC
BATA FTP Server	/csctrans/outgoing	All files sent to the RCSC

The RCSC interface process pushes files to the outgoing drop box folder on the BATA FTP server and pulls files from the incoming drop box folder on the BATA FTP server.

The table below indicates the file naming conventions agreed upon with the RCSC and the initial configuration for the file location. The location of these files is configurable.

Table 10-5: RCSC Interface Files and Naming Conventions

FILE NAME	DESCRIPTION	LOCATION
TAG STATUS FILE		
YYYYMMDDHHMM.aetc, srat_YYYYMMDDHHMM.tag, tcat_YYYYMMDDHHMM.tag, sdat_YYYYMMDDHHMM.tag, cvat_YYYYMMDDHHMM.tag, and YYYYMMDDHHMM_aetc.zip after being zipped	ETC transponder statuses for all RCSC accounts and CTOC away agencies as an incoming file. YYYYMMDDHHMM represents the year, month, day, hour, and minute the file was created.	BATA FTP Server /csctrans/incoming Directory
ETC TRANSACTION FILE		
YYYYMMDDHHMMSS.areq and YYYYMMDDHHMMSS_areq.zip after being zipped	ETC transactions since the last ETC Transaction File generation. YYYYMMDDHHMMSS represents the year, month, day, hour, minute, and second at which the file was created.	BATA FTP Server /csctrans/outgoing Directory
VIOLATION TRANSACTION FILE		
YYYYMMDDHHMMSS.avio and YYYYMMDDHHMMSS_avio.zip after being zipped	Violation transactions since the last Violation Transaction File generation. YYYYMMDDHHMMSS represents the year, month, day, hour, minute, and second at which the file was created.	BATA FTP Server /csctrans/outgoing Directory
TRANSACTION CORRECTION FILE		
YYYYMMDDHHMMSS.ctre and YYYYMMDDHHMMSS_ctre.zip after being zipped	ETC and violation transaction corrections since the last Transaction Correction File. YYYYMMDDHHMMSS represents the year, month, day, hour, minute, and second at which the file was created.	BATA FTP Server /csctrans/outgoing Directory
ETC RESPONSE FILE		
YYYYMMDDHHMMSS.ares and YYYYMMDDHHMMSS_ares.zip after being zipped	ETC response records generated by the RCSC since the last ETC Response File. YYYYMMDDHHMMSS represents the year, month, day, hour, minute, and second at which the file was created.	BATA FTP Server /csctrans/incoming Directory
VIOLATION RESPONSE FILE		
YYYYMMDDHHMMSS.avres and YYYYMMDDHHMMSS_avres.zip after being zipped	Violation response records generated by the RCSC since the last Violation Response File. YYYYMMDDHHMMSS represents the year, month, day, hour, minute, and second at which the file was	BATA FTP Server /csctrans/incoming Directory

FILE NAME	DESCRIPTION	LOCATION
	created.	
CORRECTION RESPONSE FILE		
YYYYMMDDHHMMSS.rtre and YYYYMMDDHHMMSS_rtre.zip after being zipped	Correction response records generated by the RCSC since the last Correction Response File. YYYYMMDDHHMMSS represents the year, month, day, hour, minute, and second at which the file was created.	BATA FTP Server /csctrans/incoming Directory
ACKNOWLEDGEMENT FILE		
{FROM_AGENCY}_{FILE_NAME}_{FILE_TYPE}.ack	File transfer acknowledgement. {FROM_AGENCY} represents the name of the acknowledging agency (i.e. “AT2” or “CSC”), {FILE_NAME} represents the name of the file being acknowledged, and {FILE_TYPE} represents the type of file being acknowledged.	BATA FTP Server /csctrans/incoming and BATA FTP Server /csctrans/outgoing Directories
VIOLATION DATA FILES AND IMAGES		
{AGENCY_ID}00{PLAZA_ID}{LANE_ID}_{DATE}{TIME}{SEQ_NO}.vdf, {AGENCY_ID}{PLAZA_ID}{LANE_ID}{SEQ_NO}{DATE}{IMAGE_NO}.{IMAGE_TYPE}, and {AGENCY_ID}{PLAZA_ID}{LANE_ID}_{DATE}{TIME}{SEQ_NO}_vdf.zip after violation data file and the corresponding images are zipped Refer to Chapter 6 Section 6.6.2.3 Figure 6-17 Example Violation Data Package for an example of this file naming convention: {AGENCY_ID} is 3 characters long in vdf - or 4 characters long in image files {PLAZA_ID} is 2 characters long {LANE_ID} is 2 characters long {DATE} is 8 characters long {TIME} is 8 characters long {SEQ_NO} is 10 characters long in vdf - or 6 characters long in image files {IMAGE_NO} is 2 characters long {IMAGE_TYPE} is 3 characters long	Violation data captured by the LPR with corresponding images. {AGENCY_ID} represents the name of the agency (i.e. “CAL” for vdf and zip and “CALT” for image files), {PLAZA_ID} represents the plaza ID where the transaction occurred, {LANE_ID} represents the lane where the transaction occurred, {DATE}{TIME} represents the year, month, day, hour, minute, second, and tick at which transaction occurred, {SEQ_NO} represents the vehicle sequence number, and {IMAGE_NO} represents the image number for jpg and “91” for bmp files. {IMAGE_TYPE} represents type of the image file (i.e. “bmp” or “jpg”).	BATA FTP Server /csctrans/outgoing Directory

10.1.6 File Layouts and Validation

The files to be sent to and from the RCSC are flat files with fixed length records. The file layouts and validation steps are presented in the sections that follow.

10.1.6.1 Tag Status File Received from the RCSC

The tag status file received from the RCSC at configurable intervals consists of a header record, followed by a detail record for each transponder (tag), then by a trailer record. The record formats are shown in the following tables.

Table 10-6: Local Tag Status File Received from the RCSC (Header Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 7	RECORD_TYPE	7	“#HEADER”
2	8	FILLER_1	1	<comma> separator (“,”)
3	9 – 12	FILE_TYPE	4	”AETC“
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 19	SEQUENCE_NO	6	Unique sequence number for the ETC Tag Status File. Incremented with the creation of each file. (000000 – 999999)
6	20	FILLER_3	1	<comma> separator (“,”)
7	21 – 30	BUSINESS_DATE	10	Business Date (MM/DD/YYYY) – same as File Creation Date
8	31	FILLER_4	1	<comma> separator (“,”)
9	32 – 33	SOURCE	2	“RC” if from the local RCSC
10	34	FILLER_5	1	<comma> separator (“,”)
11	35 – 36	DESTINATION	2	“AT”
12	37	FILLER_6	1	<comma> separator (“,”)
13	38 – 47	CREATE_DATE	10	File Creation Date (MM/DD/YYYY)
14	48	FILLER_7	1	<comma> separator (“,”)
15	49 – 56	CREATE_TIME	8	File Creation Time (HH:MM:SS)
16	57	LINEFEED	1	<line feed>

Table 10-7: Away Tag Status File Received from the RCSC (Header Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 7	RECORD_TYPE	7	“#HEADER”
2	8	FILLER_1	1	<comma> separator (“,”)
3	9 – 12	FILE_TYPE	4	”TAGS“
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 17	ACTION_CODE	4	“INIT”

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
6	18	FILLER_2	1	<comma> separator (“,”)
7	19 – 24	SEQUENCE_NO	6	Unique sequence number for the ETC Tag Status File. Incremented with the creation of each file. (000000 – 999999)
8	25	FILLER_3	1	<comma> separator (“,”)
9	26 – 35	BUSINESS_DATE	10	Business Date (MM/DD/YYYY) – same as File Creation Date
10	36	FILLER_4	1	<comma> separator (“,”)
11	37 – 38	SOURCE	2	“SR” if from SR-91 “TC” if from TCA “SD” if from SANDAG “CV” if from SBX
12	39	FILLER_5	1	<comma> separator (“,”)
13	40 – 41	DESTINATION	2	“AT”
14	42	FILLER_6	1	<comma> separator (“,”)
15	43 – 52	CREATE_DATE	10	File Creation Date (MM/DD/YYYY)
16	53	FILLER_7	1	<comma> separator (“,”)
17	54 – 61	CREATE_TIME	8	File Creation Time (HH:MM:SS)
18	62	LINEFEED	1	<line feed>

Table 10-8: Local Tag Status File Received from the RCSC (Detail Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 4	ETC_TAG_ID	4	CTOC Tag ID (0000 – 1023)
2	5	FILLER_1	1	< comma> separator (“,”)
3	6 – 11	ETC_TAG_FAC_ CODE	6	CTOC Facility Code (000000 – 262143)
4	12	FILLER_2	1	< comma > separator (“,”)
5	13	TOL_TAG_ TYPE	1	Four Bit Tag Type read from the ETC transponder in hexadecimal (0 – F): “0” = CTOC transponder Possible future use for switchable transponders: “4” = HOV2 “8” = HOV3

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
6	14	FILLER_3	1	<comma> separator (“,”)
7	15	ETC_TAG_ STATUS	1	ATCAS II Tag Status (“0” indicates valid, “1” indicates low balance, “4” indicates invalid, and “5” indicates lost/stolen)
8	16	FILLER_4	1	< comma> separator (“,”)
9	17	ETC_TAG_ ATTRIBUTE	1	ATCAS II Tag Attribute (“0” indicates standard, “1” indicates non-revenue, and “8” indicates hybrid)
11	18	LINEFEED	1	<line feed>

Table 10-9: Away Tag Status File Received from the RCSC (Detail Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 8	ETC_TAG_ID	8	Tag Number in Hexadecimal (00000000 – 0FEFF3FF)
2	9	FILLER_1	1	< comma> separator (“,”)
3	10	ACTION_CODE	1	“A”
4	11	FILLER_2	1	< comma > separator (“,”)
5	12	TAG_ TYPE	1	“N” = Non-Revenue “V” = Valid “I” = Invalid
6	13	FILLER_3	1	<comma> separator (“,”)
7	14	SUBTYPE_1	1	“N” indicates valid “L” indicates lost “S” indicates stolen “B” indicates low balance “R” indicates not used
8	15	FILLER_4	1	< comma> separator (“,”)
9	16	SUBTYPE_2	1	“N” indicates not used
10	17	FILLER_5	1	< comma> separator (“,”)
11	18	SUBTYPE_3	1	“N” indicates not used
12	19	LINEFEED	1	<line feed>

Table 10-10: Tag Status File Received from the RCSC (Trailer Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 8	RECORD_TYPE	8	"#TRAILER"
2	9	FILLER_1	1	<comma> separator (",")
3	10 – 15	SEQUENCE_NO	6	Same as Header Record SEQUENCE_NO
4	16	FILLER_2	1	<comma> separator (",")
5	17 – 26	FILE_DATE	10	Same as Header Record CREATE_DATE
6	27	FILLER_3	1	<comma> separator (",")
7	28 – 35	DETAIL_COUNT	8	Count of all tag statues contained in the detail records (00000000 – 99999999)
8	36	LINEFEED	1	<line feed>

10.1.6.1.1 RCSC Processing Requirements

The first comprehensive tag download file is pushed to the drop box folder on the BATA FTP server no later than 04:00. All subsequent comprehensive tag download files are sent in 6 hour intervals (i.e., the first tag file no later than 04:00, then 10:00, 16:00, and 22:00).

In the event that an invalid detail record is encountered (e.g., invalid ETC_TAG_STATUS or ETC_TAG_ATTRIBUTE value), the ATCAS II Host skips the file completely and notifies the RCSC via the Acknowledgement File. See Section 10.1.6.10 Incoming File Validation.

The ETC_TAG_STATUS and ETC_TAG_ATTRIBUTE values are downloaded to the lane/zone controllers at the BATA bridges by the ATCAS II Host via ATCAS II plaza computers so that customers can be notified of the tag's status and charged the correct toll amount based on the tag's attributes. The tag status messages displayed on the Patron Fare Display (PFD) and the amount charged are configurable at the ATCAS II Host and can be modified by authorized BATA employees.

Should the RCSC fail to send a tag download file as required, the previous tag status and attribute values continue to be used and an alarm is issued. See Section 10.1.8

Error Processing.

The ATCAS II Host performs appropriate sanity checks on the Tag Status File prior to its transmission to the lanes; such as checking for an unusual increase or decrease in the number of tags from the previous version.

Section 10.1.6.10 Incoming File Validation provides more details.

10.1.6.2 ETC Transaction File Sent to the RCSC

The ETC transaction file generated at configurable intervals and sent to the RCSC consists of a header record, followed by a detail record for each ETC transaction in the batch, then by a trailer record. ETC transactions are reported for the Business Day equal to the date the ETC transponder was read in the toll zone. The only exception is in cases of buffered and flushed tag reads and other late posting adjustments to revenue which are reported for the Business Day equal to the day the adjustments were made. The record formats are shown in the following tables.

Table 10-11: ETC Transaction File Sent to the RCSC (Header Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 7	RECORD_TYPE	7	“#HEADER”
2	8	FILLER_1	1	<comma> separator (“,”)
3	9 – 12	FILE_TYPE	4	”REQ “
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 19	SEQUENCE_NO	6	Unique sequence number for the ETC Transaction File. Incremented with the creation of each file. (000000 – 999999)
6	20	FILLER_3	1	<comma> separator (“,”)
7	21 – 30	BUSINESS_DATE	10	Business Date – midnight to midnight (MM/DD/YYYY)
8	31	FILLER_4	1	<comma> separator (“,”)
9	32 – 33	SOURCE	2	“AT”
10	34	FILLER_5	1	<comma> separator (“,”)
11	35 – 36	DESTINATION	2	“RC”
12	37	FILLER_6	1	<comma> separator (“,”)
13	38 – 47	CREATE_DATE	10	File Creation Date (MM/DD/YYYY)
14	48	FILLER_7	1	<comma> separator (“,”)
15	49 – 56	CREATE_TIME	8	File Creation Time (HH:MM:SS)
16	57	LINEFEED	1	<line feed>

Table 10-12: ETC Transaction File Sent to the RCSC (Detail Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 10	TRANSACTION_ NUMBER	10	Unique transaction number assigned to the transaction being sent from the iSentTransId identity field in the tbRcscSendTrans table and used to identify the transaction in the ETC response file (0000000000 – 9999999999)
2	11	FILLER_1	1	<comma> separator (“,”)
3	12	TOL_TRX_TYPE	1	Type of transaction. “1” = ETC “3” = Carpool “4” = Non-Revenue “5” = Hybrid “6” = HOV2 “7” = HOV3
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 17	TOL_TAG_ID	4	Tag ID in accordance with the Title-21 specs (0000 – 1023)
6	18	FILLER_3	1	<comma> separator (“,”)
7	19 – 24	TOL_TAG_ FACILITY_ID	6	Facility Code assigned to issuing agency in accordance with the Title-21 specs (000000 – 262143)
8	25	FILLER_4	1	<comma> separator (“,”)
9	26	TOL_TAG_ TYPE	1	Four Bit Tag Type read from the ETC transponder in hexadecimal (0 – F): “0” = CTOC transponder Possible future use for switchable transponders: “4” = HOV2 “8” = HOV3
10	27	FILLER_5	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
11	28 – 30	TOL_PLAZA_ID	3	Plaza ID where the transaction occurred that is part of the unique key needed to identify the transaction. Needs to be converted to the name of the corresponding bridge by the RCSC on the customer statement as follows: “002” = “ANTIOCH” “003” = “RICHMOND” “004” = “BAY BRIDGE” “005” = “SAN MATEO” “006” = “DUMBARTON” “007” = “CARQUINEZ” “008” = “BENICIA”
12	31	FILLER_6	1	<comma> separator (“,”)
13	32 – 33	TOL_LANE_ID	2	Lane number where the transaction occurred that is part of the unique key needed to identify the transaction (00 – 99) and is shown on the customer’s statement
14	34	FILLER_7	1	<comma> separator (“,”)
15	35 – 44	TOL_TRX_ DATE	10	Date portion of the timestamp for the transaction that shows when the transaction occurred on the customer’s statement (MM/DD/YYYY)
16	45	FILLER_8	1	<comma> separator (“,”)
17	46 – 53	TOL_TRX_ TIME	8	Time portion of the timestamp for the transaction that shows when the transaction occurred on the customer’s statement (HH:MM:SS)
18	54	FILLER_9	1	<comma> separator (“,”)
19	55 – 59	TOL_FARE_ETC_ AMT	5	ETC fare amount to charge in number of cents (00000 – 99999). This is the amount passed from the lane as revenuecollected in the vehicle transaction or the revenue amount calculated based on the fare schedule and carpool schedule at the time of the transaction for late posting adjustments.
20	60	FILLER_10	1	<comma> separator (“,”)
21	61 – 65	TOL_FARE_CASH_ AMT	5	Cash fare amount to charge in number of cents (always 00000)
22	66	FILLER_11	1	<comma> separator (“,”)
23	67 – 74	TOL_FARE_ID	8	Fare Definition ID that was used in the toll lookup followed by a 5 digit number of seconds from the beginning of the day when the time of day pricing interval started (00100000 – 99999999)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
24	75	FILLER_12	1	<comma> separator (“,”)
25	76 – 77	TOL_MSG_FLAG	2	Indicates whether or not the transaction was a buffered tag read (“1” = not a buffered tag read, “2” = buffered tag read)
26	78	FILLER_13	1	<comma> separator (“,”)
27	79 – 80	AVC_CLASS_ID	2	Class based on the axles counted by the AVC (“02” = 2 axle vehicle, “03” = 3 axle vehicle, “04” = 4 axle vehicle “05” = 5 axle vehicle, “06” = 6 axle vehicle, “07” = 7 axle vehicle, and “08” = 8 or more axle vehicle)
28	81	FILLER_14	1	<comma> separator (“,”)
29	82 – 89	LANE_TX_ SEQUENCE_ NUMBER	8	Unique vehicle sequence number assigned to the vehicle transaction by the lane/zone controller (VEH_SEQUENCE_NO) which can be combined with the plaza id and lane number to uniquely identify the transaction (00000000 – 99999999)
30	90	FILLER_15	1	<comma> separator (“,”)
31	91	TOL_TAG_STATUS	1	ATCAS II Tag Status at the lane at the time of the tag read (“0” indicates valid and “1” indicates low balance)
32	92	FILLER_16	1	<comma> separator (“,”)
33	93	TOL_DST_FLAG	1	Indicates daylight saving time (“0” = standard time and “1” = daylight savings time)
34	94	FILLER_17	1	<comma> separator (“,”)
35	95 – 97	TOL_TRX_SPEED	3	Transaction speed as reported by the lane (Default 000)
36	98	FILLER_18	1	<comma> separator (“,”)
37	99 – 100	RESOLV_CODE	2	00
38	101	LINEFEED	1	<line feed>

Table 10-13: ETC Transaction File Sent to the RCSC (Trailer Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 8	RECORD_TYPE	8	"#TRAILER"
2	9	FILLER_1	1	<comma> separator (",")
3	10 – 15	SEQUENCE_NO	6	Same as Header Record SEQUENCE_NO
4	16	FILLER_2	1	<comma> separator (",")
5	17 – 26	FILE_DATE	10	Same as Header Record CREATE_DATE
6	27	FILLER_3	1	<comma> separator (",")
7	28 – 35	DETAIL_COUNT	8	Count of all transactions contained in the detail records (00000000 – 99999999)
8	36	FILLER_4	1	<comma> separator (",")
9	37 - 46	DETAIL_TRANS_ AMOUNT	10	Total amount for all records in the file (Sum of TOL_FARE_ETC_AMT in cents)
10	47	LINEFEED	1	<line feed>

10.1.6.2.1 RCSC Processing Requirements

The RCSC receives and processes ETC Transaction Files from the ATCAS II Host multiple times a day at predetermined intervals (i.e., the ATCAS II host is configured to send an ETC Transaction File every 3 hours at 00:00, 03:00, 06:00, 09:00, 12:00, 15:00, 18:00, and 21:00).

ETC transactions in this file have a unique transaction number for each record.

All transactions coming in through this interface are processed and resolve code values are ignored.

RCSC can process multiple transaction type (TOL_TRX_TYPE) values in a single file. All transaction types are processed as ETC transactions but the transaction type value is saved for future summarization data in reports.

The tag type (TOL_TAG_TYPE), fare id (TOL_FARE_ID), and transaction class (AVC_CLASS_ID) values are also saved for future summarization data in reports.

The RCSC ensures, upon processing, that the ETC Transaction File does not contain two or more transactions for the same TOL_TAG_ID/TOL_TAG_AGENCY_ID combination at the same TOL_PLAZA_ID/TOL_LANE_ID within a one (1) minute period. However, this parameter is configurable at the RCSC based on business rule decisions between the RCSC and ATCAS II Host.

The RCSC performs sanity checks on the ETC Transaction File to look for formatting errors, record count mismatches between header and detail records, etc. In the event the file fails, the RCSC notifies the ATCAS II Host of the anomaly by means of the acknowledgment file. See Section 10.1.6.8, the RETURN_CODE value.

If the RCSC uncovers an error in a detail record, the RCSC rejects that record and processes the remainder of the transaction file. ATCAS II Host is notified of the error using corresponding error and reason codes in the ETC response file. See Sections 10.1.6.5 and 10.1.6.11.

The RCSC always charges the ETC toll amount received from the ATCAS II Host. The toll amount calculated at the ATCAS II Host as supplied in the TOL_FARE_ETC_AMT field of the transaction file is used to debit the BATA RCSC accounts. This includes the toll amount due for non-revenue customers (ATCAS II sends 00000 in the TOL_FARE_ETC_AMT field for non-revenue customers).

The RCSC has the capability of rejecting a transaction based on the age. This is a configurable value and set to 365 days for all incoming transactions from the ATCAS II Host. This value can be changed as requested by BATA.

The business date (BUSINESS_DATE) is set to the calendar day in which the transaction occurred at the lane. The only exception is when transactions are sent as the result of an adjustment made at the ATCAS II Host after the business day has been closed. In this case, the business date is set to the calendar day in which the adjustment occurred.

All transactions contained in the ETC transaction file have the same transaction date (TOL_TRX_DATE) value. A business date that does not match the transaction date indicates the transaction is from an adjustment made at the ATCAS II host. The mismatched business date reflects the day the adjustment was made.

10.1.6.3 Violation Transaction File Sent to the RCSC

The violation transaction file generated at configurable intervals and sent to the RCSC consists of a header record, followed by a detail record for each violation transaction in the batch, and then by a trailer record. Violation transactions are reported for the Business Day equal to the date the vehicle exited the toll zone. The record formats are shown in the following tables.

Table 10-14: Violation Transaction File Sent to the RCSC (Header Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 7	RECORD_TYPE	7	“#HEADER”
2	8	FILLER_1	1	<comma> separator (“,”)
3	9 – 12	FILE_TYPE	4	”VIO “
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 19	SEQUENCE_NO	6	Unique sequence number for the Violation Transaction File. Incremented with the creation of each file. (000000 – 999999)
6	20	FILLER_3	1	<comma> separator (“,”)
7	21 – 30	BUSINESS_DATE	10	Business Date – midnight to midnight (MM/DD/YYYY)
8	31	FILLER_4	1	<comma> separator (“,”)
9	32 – 33	SOURCE	2	“AT”
10	34	FILLER_5	1	<comma> separator (“,”)
11	35 – 36	DESTINATION	2	“RC”
12	37	FILLER_6	1	<comma> separator (“,”)
13	38 – 47	CREATE_DATE	10	File Creation Date (MM/DD/YYYY)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
14	48	FILLER_7	1	<comma> separator (“,”)
15	49 – 56	CREATE_TIME	8	File Creation Time (HH:MM:SS)
16	57	LINEFEED	1	<line feed>

Table 10-15: Violation Transaction File Sent to the RCSC (Detail Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 10	TRANSACTION_ NUMBER	10	Unique transaction number assigned to the transaction being sent from the iSentTransId identity field in the tbRcscSendTrans table and used to identify the transaction in the violation response file (0000000000 – 9999999999)
2	11	FILLER_1	1	<comma> separator (“,”)
3	12	TOL_TRX_TYPE	1	Type of transaction. “2” = Violation
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 17	TOL_TAG_ID	4	Tag ID in accordance with the Title-21 specs (0000 – 1023) or <blank> if no tag was read
6	18	FILLER_3	1	<comma> separator (“,”)
7	19 – 24	TOL_TAG_ FACILITY_ID	6	Facility Code assigned to issuing agency in accordance with the Title-21 specs (000000 – 262143) or <blank> if no tag was read
8	25	FILLER_4	1	<comma> separator (“,”)
9	26	TOL_TAG_ TYPE	1	Four Bit Tag Type read from the ETC transponder in hexadecimal (0 – F): “0” = CTOC transponder Possible future use for switchable transponders: “4” = HOV2 “8” = HOV3
10	27	FILLER_5	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
11	28 – 30	TOL_PLAZA_ID	3	Plaza ID where the transaction occurred that is part of the unique key needed to identify the transaction. Needs to be converted to the name of the corresponding bridge by the RCSC on the customer statement as follows: “002” = “ANTIOCH” “003” = “RICHMOND” “004” = “BAY BRIDGE” “005” = “SAN MATEO” “006” = “DUMBARTON” “007” = “CARQUINEZ” “008” = “BENICIA”
12	31	FILLER_6	1	<comma> separator (“,”)
13	32 – 33	TOL_LANE_ID	2	Lane number where the transaction occurred that is part of the unique key needed to identify the transaction (00 – 99) and is shown on the customer statement
14	34	FILLER_7	1	<comma> separator (“,”)
15	35 – 44	TOL_TRX_ DATE	10	Date portion of the timestamp for the transaction that shows when the transaction occurred on the customers statement (MM/DD/YYYY)
16	45	FILLER_8	1	<comma> separator (“,”)
17	46 – 53	TOL_TRX_ TIME	8	Time portion of the timestamp for the transaction that shows when the transaction occurred on the customers statement (HH:MM:SS)
18	54	FILLER_9	1	<comma> separator (“,”)
19	55 – 59	TOL_FARE_ETC_ AMT	5	ETC fare amount to charge in number of cents (00000 – 99999). The ETC fare amount is looked up from the database based on the fare definition that was active at the time of the transaction as indicated in the fare_id value passed from the lane in the vehicle transaction. If the transaction is part of a tour that was in a dedicated carpool mode then the ETC_CARPOOL fare amount is used. This is the amount posted to the ETC account when the violation is converted to an ETC transaction by the RCSC.
20	60	FILLER_10	1	<comma> separator (“,”)
21	61 – 65	TOL_FARE_CASH_ AMT	5	Cash fare amount to charge in number of cents (00000 – 99999). This is the amount passed from the lane as revenue expected in the vehicle transaction.

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
22	66	FILLER_11	1	<comma> separator (“,”)
23	67 – 74	TOL_FARE_ID	8	Fare Definition ID that was used in the toll lookup followed by a 5 digit number of seconds from the beginning of the day when the time of day pricing interval started (00100000 – 99999999)
24	75	FILLER_12	1	<comma> separator (“,”)
25	76 – 77	TOL_MSG_FLAG	2	Indicates whether or not the transaction was a buffered tag read (“1” = not a buffered tag read, “2” = buffered tag read)
26	78	FILLER_13	1	<comma> separator (“,”)
27	79 – 80	AVC_CLASS_ID	2	Class based on the axles counted by the AVC (“02” = 2 axle vehicle, “03” = 3 axle vehicle, “04” = 4 axle vehicle “05” = 5 axle vehicle, “06” = 6 axle vehicle, “07” = 7 axle vehicle, and “08” = 8 or more axle vehicle)
28	81	FILLER_14	1	<comma> separator (“,”)
29	82 – 89	LANE_TX_ SEQUENCE_ NUMBER	8	Unique vehicle sequence number assigned to the vehicle transaction (VEH_SEQUENCE_NO) which can be combined with the plaza id and lane number to uniquely identify the transaction (00000000 – 99999999)
30	90	FILLER_15	1	<comma> separator (“,”)
31	91	TOL_TAG_STATUS	1	ATCAS II Tag Status at the lane at the time of the tag read (“4” indicates invalid and “5” indicates lost/stolen) or <blank> if no tag was read
32	92	FILLER_16	1	<comma> separator (“,”)
33	93	TOL_DST_FLAG	1	Indicates daylight saving time (“0” = standard time and “1” = daylight savings time)
34	94	FILLER_17	1	<comma> separator (“,”)
35	95 – 97	TOL_TRX_SPEED	3	Transaction speed as reported by the lane (Default 000)
36	98	FILLER_18	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
37	99 – 100	RESOLV_CODE	2	02
38	101	LINEFEED	1	<line feed>

Table 10-16: Violation Transaction File Sent to the RCSC (Trailer Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 8	RECORD_TYPE	8	"#TRAILER"
2	9	FILLER_1	1	<comma> separator (",")
3	10 – 15	SEQUENCE_NO	6	Same as Header Record SEQUENCE_NO
4	16	FILLER_2	1	<comma> separator (",")
5	17 – 26	FILE_DATE	10	Same as Header Record CREATE_DATE
6	27	FILLER_3	1	<comma> separator (",")
7	28 – 35	DETAIL_COUNT	8	Count of all transactions contained in the detail records (00000000 – 99999999)
8	36	FILLER_4	1	<comma> separator (",")
9	37 - 46	DETAIL_TRANS_ AMOUNT	10	Total amount for all records in the file (Sum of TOL_FARE_ETC_AMT in cents)
10	47	LINEFEED	1	<line feed>

10.1.6.3.1 RCSC Processing Requirements

The RCSC is capable of receiving and processing Violation Transaction Files from the ATCAS II Host multiple times a day (i.e., the ATCAS II host is configured to send a Violation Transaction File every 3 hours at 00:00, 03:00, 06:00, 09:00, 12:00, 15:00, 18:00, and 21:00).

Violation transactions in this file have a unique transaction number for each record.

The transaction type (TOL_TRX_TYPE), tag type (TOL_TAG_TYPE), fare ID (TOL_FARE_ID), and transaction class (AVC_CLASS_ID) values are saved for future summarization data in reports.

The RCSC performs sanity checks on the Violation Transaction File to look for formatting errors, record count mismatches between header and detail records, etc. In the event the file fails, the RCSC notifies the ATCAS II Host of the anomaly by means of the acknowledgment file. See Section 10.1.6.8, the RETURN_CODE value.

If the RCSC uncovers an error in a detail record, the RCSC rejects the transaction record with the error and processes the remainder of the transaction file. ATCAS II Host is notified of the error using corresponding error and reason codes in the Violation response file. See Sections 10.1.6.6 and 10.1.6.11.

The RCSC uses the toll amount as supplied in the TOL_FARE_CASH_AMT field to process violations. All postable transactions use the amount in the TOL_FARE_ETC_AMT field, while transactions eligible for Notice Escalation use the amount in the TOL_FARE_CASH_AMT field.

The RCSC has the capability of rejecting a transaction based on the age of the transaction. This is a configurable value and is set to 365 days for all incoming transactions from the ATCAS II Host. This value can be changed as requested by BATA.

The business date (BUSINESS_DATE) is set to the calendar day in which the transaction occurred at the lane. The only exception is when transactions are sent as the result of an adjustment made at the ATCAS II host after the business day closed. In this case, the business date is set to the date in which the adjustment occurred.

All transactions contained in the Violation Transaction File have the same transaction date (TOL_TRX_DATE) value. A business date that does not match the transaction date indicates the transaction is from an adjustment made at the ATCAS II host. The mismatched business date reflects the day the adjustment was made.

10.1.6.4 Transaction Correction File Sent to the RCSC

The transaction correction file is generated and sent to the RCSC once following the end of each business day closure when there are corrections or reversals associated with that day. This file consists of a header record followed by a detail record for each transaction adjustment in the batch, and then by a trailer record. Transactions are only sent when there is a change in the ETC fare amount. Refer to Chapter 8 Section 8.10 Financial Management for details related to closing out the business day and Section 8.10.5 Adjustment for details regarding how adjustments are made. The record formats are shown in the following tables.

Table 10-17: Transaction Correction File Sent to the RCSC (Header Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 7	RECORD_TYPE	7	“#HEADER”
2	8	FILLER_1	1	<comma> separator (“,”)
3	9 – 12	FILE_TYPE	4	”CTRE“
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 19	SEQUENCE_NO	6	Unique sequence number for the Transaction Correction File. Incremented with the creation of each file. (000000 – 999999)
6	20	FILLER_3	1	<comma> separator (“,”)
7	21 – 30	BUSINESS_DATE	10	Business Date – midnight to midnight (MM/DD/YYYY)
8	31	FILLER_4	1	<comma> separator (“,”)
9	32 – 33	SOURCE	2	“AT”
10	34	FILLER_5	1	<comma> separator (“,”)
11	35 – 36	DESTINATION	2	“RC”
12	37	FILLER_6	1	<comma> separator (“,”)
13	38 – 47	CREATE_DATE	10	File Creation Date (MM/DD/YYYY)
14	48	FILLER_7	1	<comma> separator (“,”)
15	49 – 56	CREATE_TIME	8	File Creation Time (HH:MM:SS)
16	57	LINEFEED	1	<line feed>

Table 10-18: Transaction Correction File Sent to the RCSC (Detail Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 10	TRANSACTION_ NUMBER	10	Unique transaction number assigned to the transaction being sent from the iSentTransId identity field in the tbRcscSendTrans table used to identify the correction transaction (0000000000 – 9999999999)
2	11	FILLER_1	1	<comma> separator (“,”)
3	12	TOL_TRX_TYPE	1	Type of transaction being corrected. “1” = ETC “2” = Violation “3” = Carpool “4” = Non-Revenue “5” = Hybrid “6” = HOV2 “7” = HOV3
4	13	FILLER_2	1	<comma> separator (“,”)
5	14	TOL_TRX_TYPE_ CORR	1	New corrected type of transaction (when different than TOL_TRX_TYPE). “1” = ETC “2” = Violation “3” = Carpool “4” = Non-Revenue “5” = Hybrid “6” = HOV2 “7” = HOV3
6	15	FILLER_3	1	<comma> separator (“,”)
7	16	TOL_CORRECTION_FL AG	1	Type of correction. “A” = Amount only “T” = Transaction Type, Fare ID, or AVC Class ID correction
8	17	FILLER_4	1	<comma> separator (“,”)
9	18 – 21	TOL_TAG_ID	4	Tag ID in accordance with the Title-21 specs (0000 – 1023) or <blank> if no tag was read
10	22	FILLER_5	1	<comma> separator (“,”)
11	23 – 28	TOL_TAG_ FACILITY_ID	6	Facility Code assigned to issuing agency in accordance with the Title-21 specs (000000 – 262143) or <blank> if no tag was read
12	29	FILLER_6	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
13	30	TOL_TAG_ TYPE	1	Four Bit Tag Type read from the ETC transponder in hexadecimal (0 – F): “0” = CTOC transponder Possible future use for switchable transponders: “4” = HOV2 “8” = HOV3
14	31	FILLER_7	1	<comma> separator (“,”)
15	32 – 34	TOL_PLAZA_ID	3	Plaza ID where the transaction being cancelled or corrected occurred that is part of the unique key needed to identify the transaction “002” = “ANTIOCH” “003” = “RICHMOND” “004” = “BAY BRIDGE” “005” = “SAN MATEO” “006” = “DUMBARTON” “007” = “CARQUINEZ” “008” = “BENICIA”
16	35	FILLER_8	1	<comma> separator (“,”)
17	36 – 37	TOL_LANE_ID	2	Lane number where the transaction being cancelled or corrected occurred that is part of the unique key needed to identify the transaction
18	38	FILLER_9	1	<comma> separator (“,”)
19	39 – 48	TOL_TRX_ DATE	10	Date portion of the timestamp for the transaction that shows when the transaction being cancelled or corrected occurred (MM/DD/YYYY)
20	49	FILLER_10	1	<comma> separator (“,”)
21	50 – 57	TOL_TRX_ TIME	8	Time portion of the timestamp for the transaction that shows when the transaction being cancelled or corrected occurred (HH:MM:SS)
22	58	FILLER_11	1	<comma> separator (“,”)
23	59 – 63	TOL_FARE_ETC_ AMT	5	ETC fare amount to reverse in the transaction being corrected in number of cents (00000 – 99999).
24	64	FILLER_12	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
25	65 – 69	TOL_FARE_CORR_ETC _ AMT	5	New ETC fare amount to charge in number of cents (00000 – 99999). A new ETC fare amount = 00000 indicates the original transaction is being reversed.
26	70	FILLER_13	1	<comma> separator (“,”)
27	71 – 75	TOL_FARE_CASH_ AMT	5	Cash fare amount to reverse in the transaction being corrected in number of cents (always 00000)
28	76	FILLER_14	1	<comma> separator (“,”)
29	77 – 81	TOL_FARE_CORR_CA SH_ AMT	5	New cash fare amount to charge in number of cents (00000 – 99999). A new cash fare amount = 00000 indicates the original transaction is being reversed or is an ETC transaction being corrected.
30	82	FILLER_15	1	<comma> separator (“,”)
31	83 – 90	TOL_FARE_ID	8	Fare Definition ID that was used in the toll lookup charge in the transaction being corrected followed by a 5 digit number of seconds from the beginning of the day when the time of day pricing interval started (00100000 – 99999999)
32	91	FILLER_16	1	<comma> separator (“,”)
33	92 – 99	TOL_FARE_CORR_ID	8	New Fare Definition ID (when different than TOL_FARE_ID) that was used in the toll lookup followed by a 5 digit number of seconds from the beginning of the day when the time of day pricing interval started (00100000 – 99999999)
34	100	FILLER_17	1	<comma> separator (“,”)
35	101 – 102	TOL_MSG_FLAG	2	Indicates whether or not the transaction was a buffered tag read (“1” = not a buffered tag read, “2” = buffered tag read)
36	103	FILLER_18	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
37	104 – 105	AVC_CLASS_ID	2	Class based on the axles counted by the AVC charge in the transaction being corrected (“02” = 2 axle vehicle, “03” = 3 axle vehicle, “04” = 4 axle vehicle “05” = 5 axle vehicle, “06” = 6 axle vehicle, “07” = 7 axle vehicle, and “08” = 8 or more axle vehicle)
38	106	FILLER_19	1	<comma> separator (“,”)
39	107 – 108	AVC_CLASS_ID_ CORR	2	New Class based on the axles counted by the AVC (when different than TOL_CLASS_ID). (“02” = 2 axle vehicle, “03” = 3 axle vehicle, “04” = 4 axle vehicle “05” = 5 axle vehicle, “06” = 6 axle vehicle, “07” = 7 axle vehicle, and “08” = 8 or more axle vehicle)
40	109	FILLER_20	1	<comma> separator (“,”)
41	110 – 117	LANE_TX_ SEQUENCE_ NUMBER	8	Unique vehicle sequence number assigned to the vehicle transaction by the lane/zone controller (VEH_SEQUENCE_NO) which can be combined with the plaza id and lane number to uniquely identify the transaction (00000000 – 99999999)
42	118	FILLER_21	1	<comma> separator (“,”)
43	119	TOL_TAG_STATUS	1	ATCAS II Tag Status at the lane at the time of the tag read (“0” indicates valid and “1” indicates low balance “4” indicates invalid and “5” indicates lost/stolen) or <blank> if no tag was read
44	120	FILLER_22	1	<comma> separator (“,”)
45	121	TOL_DST_FLAG	1	Indicates daylight saving time (“0” = standard time and “1” = daylight savings time)
46	122	FILLER_23	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
47	123 – 125	TOL_TRX_SPEED	3	Transaction speed as reported by the lane (Default 000)
48	126	FILLER_24	1	<comma> separator (“,”)
49	127 – 136	ORIG_TRX_ NUMBER	10	Transaction number assigned to the original transaction being corrected (0000000000 – 9999999999)
50	137	FILLER_25	1	<comma> separator (“,”)
51	138 – 139	RESOLV_CODE	2	00
52	140	LINEFEED	1	<line feed>

Table 10-19: Transaction Correction File Sent to the RCSC (Trailer Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 8	RECORD_TYPE	8	“#TRAILER”
2	9	FILLER_1	1	<comma> separator (“,”)
3	10 – 15	SEQUENCE_NO	6	Same as Header Record SEQUENCE_NO
4	16	FILLER_2	1	<comma> separator (“,”)
5	17 – 26	FILE_DATE	10	Same as Header Record CREATE_DATE
6	27	FILLER_3	1	<comma> separator (“,”)
7	28 – 35	DETAIL_COUNT	8	Count of all cancellations and corrections contained in the detail records (00000000 – 99999999)
8	36	FILLER_4	1	<comma> separator (“,”)
9	37 - 46	DETAIL_TRANS_ AMOUNT	10	Total new amount for all records in the file (Sum of TOL_FARE_CORR_ETC_AMT in cents)
10	47	LINEFEED	1	<line feed>

10.1.6.4.1 RCSC Processing Requirements

This interface is primarily for ETC toll amount (TOL_FARE_CORR_ETC_AMT) corrections. The transaction type (TOL_TRX_TYPE_CORR), fare ID (TOL_FARE_CORR_ID), and transaction class (AVC_CLASS_ID_CORR) can also be modified so that the reason for the change can be saved. Corrections are only sent when there is a change in the ETC fare amount.

The transaction type (TOL_TRX_TYPE_CORR), tag type (TOL_TAG_TYPE), fare ID (TOL_FARE_CORR_ID), and transaction class (AVC_CLASS_ID_CORR) values are saved for future summarization in reports.

ATCAS II Host sends a maximum of one (1) correction file per day. The correction file is generated and sent after the business day is closed at the ATCAS II Host.

Correction transactions in this file have a unique transaction number for each record.

Correction transactions contain the original transaction number (ORIG_TRX_NUMBER).

The ATCAS II host waits for a response that the original transaction was posted before sending the request to correct it.

The ATCAS II host sends the file with a corrected ETC fare amount (TOL_FARE_CORR_ETC_AMT) of 00000 to reverse the original transaction.

Violation transactions (TOL_TRX_TYPE = 2) can only be reversed for the ITOL or VTOL transactions by setting the ETC and cash fare amount (TOL_FARE_CORR_ETC_AMT and TOL_FARE_CORR_CASH_AMT) values to 00000. Modifying the transaction type, fare ID, and transaction class, or setting the fare amount to something other than 00000 causes the RCSC to reject the transaction correction record. The remainder of the transaction file is processed. The ATCAS II Host is notified of the error using corresponding error and reason codes in the Correction response file. See Sections 10.1.6.7 and 10.1.6.11.

All other violation adjustments are recorded in ATCAS II but are not reversed at the RCSC (they need to be manually cancelled out at the RCSC). The auditor should only cancel the ITOL/VTOL violations in ATCAS II after the RCSC has responded that they have posted.

Non-violation transactions (TOL_TRX_TYPE not = 2) cannot be converted to violation transactions since there are no corresponding images. Changing the transaction type to violation (TOL_TRX_TYPE_CORR = 2) causes the RCSC to reject the transaction correction record and continue processing the remainder of the file. The ATCAS II Host is notified of the error using corresponding error and reason codes in the Correction response file. See Sections 10.1.6.7 and 10.1.6.11.

This interface is for BATA/Home customers only. In case any CTOC transactions are included in this file, the RCSC rejects the entire file with an ack file status of 01.

There is only one (1) adjustment per posted transaction. See Chapter 8 Section 8.10.5 regarding how a second adjustment is handled.

The RCSC creates 2 new transactions for each correction transaction.

a. Transaction #1 is posted to the customer account as a reversal for the same amount as the original amount.
b. Transaction #2 is the new amount received from ATCAS II in the correction file.

The RCSC performs sanity checks on the Transaction Correction File to look for formatting errors, record count mismatches between header and detail records, etc. In the event the file fails, the RCSC notifies the ATCAS II Host of the anomaly by means of the acknowledgment file. See Section 10.1.6.8, the RETURN_CODE value.

If the RCSC determines an error in a detail record, the RCSC rejects the transaction record with the error and processes the remainder of the transaction file. The ATCAS II Host is notified of the error using corresponding error and reason codes in the Correction response file. See Sections 10.1.6.7 and 10.1.6.11.

The RCSC always charges the ETC toll amount received from the ATCAS II Host. The toll amount is calculated at the ATCAS II Host and supplied in the TOL_FARE_CORR_ETC_AMT field of the transaction file. This amount is used to debit the BATA RCSC accounts and includes toll amounts due for non-revenue customers (ATCAS II sends 00000 in the TOL_FARE_CORR_ETC_AMT field for non-revenue customers).

The RCSC has the capability of rejecting a transaction based on the age of the transaction. This is a configurable value and is set to 365 days for all incoming transactions from the ATCAS II Host. This value can be changed as requested by BATA.

The business date (BUSINESS_DATE) is set to the calendar day in which the transaction occurred at the lane. A business date that does not match the transaction date indicates the transaction is from an adjustment made at the ATCAS II host. The mismatched business date reflects the day the adjustment was made.

10.1.6.5 ETC Response File Received from the RCSC

The ETC response file is generated by the RCSC after the ETC transaction file is processed and provides the final disposition of the transactions. This file consists of a header record, followed by a detail record for each transaction disposition in the batch, and then by a trailer record. Refer to Chapter 8 Host System Software Section 8.10.1 Business Day Complete and Section 8.10.3 Audit ETC Variances for details on how to reconcile discrepancies in the ETC response file. The record formats for the ETC response file are shown in the following tables.

Table 10-20: ETC Response File Received from the RCSC (Header Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 7	RECORD_TYPE	7	“#HEADER”
2	8	FILLER_1	1	<comma> separator (“,”)
3	9 – 12	FILE_TYPE	4	”RES “
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 19	SEQUENCE_NO	6	Unique sequence number for the ETC Transaction File that corresponds to this response (000000 – 999999)
6	20	FILLER_3	1	<comma> separator (“,”)
7	21 – 30	BUSINESS_DATE	10	Business Date for the ETC Transaction File that corresponds to this response (MM/DD/YYYY)
8	31	FILLER_4	1	<comma> separator (“,”)
9	32 – 33	SOURCE	2	“RC”
10	34	FILLER_5	1	<comma> separator (“,”)
11	35 – 36	DESTINATION	2	“AT”
12	37	FILLER_6	1	<comma> separator (“,”)
13	38 – 47	CREATE_DATE	10	File Creation Date (MM/DD/YYYY)
14	48	FILLER_7	1	<comma> separator (“,”)
15	49 – 56	CREATE_TIME	8	File Creation Time (HH:MM:SS)
16	57	LINEFEED	1	<line feed>

Table 10-21: ETC Response File Received from the RCSC (Detail Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 10	TRANSACTION_ NUMBER	10	Unique transaction number to identify the transaction that corresponds to this response (0000000000 – 9999999999)
2	11	FILLER_1	1	<comma> separator (“,”)
3	12	TOL_TRX_TYPE	1	Type of transaction. “1” = ETC “3” = Carpool “4” = Non-Revenue “5” = Hybrid “6” = HOV2 “7” = HOV3
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 17	TOL_TAG_ID	4	Tag ID in accordance with the Title-21 specs (0000 – 1023)
6	18	FILLER_3	1	<comma> separator (“,”)
7	19 – 24	TOL_TAG_ FACILITY_ID	6	Facility Code assigned to issuing agency in accordance with the Title-21 specs (000000 – 262143)
8	25	FILLER_4	1	<comma> separator (“,”)
9	26	TOL_TAG_ TYPE	1	Four Bit Tag Type read from the ETC transponder in hexadecimal (0 – F): “0” = CTOC transponder Possible future use for switchable transponders: “4” = HOV2 “8” = HOV3
10	27	FILLER_5	1	<comma> separator (“,”)
11	28 – 37	POSTED_DATE	10	Date the transaction was posted to a patron’s account, posted at an away agency, or was rejected (MM/DD/YYYY)
12	38	FILLER_6	1	<comma> separator (“,”)
13	39 – 41	TOL_PLAZA_ID	3	Plaza ID where the transaction occurred that is part of the unique key needed to identify the transaction that corresponds to this response “002” = “ANTIOCH” “003” = “RICHMOND” “004” = “BAY BRIDGE” “005” = “SAN MATEO” “006” = “DUMBARTON” “007” = “CARQUINEZ” “008” = “BENICIA”
14	42	FILLER_7	1	<comma> separator (“,”)
15	43 – 44	TOL_LANE_ID	2	Lane number where the transaction occurred that is part of the unique key needed to identify the transaction that corresponds to this response (00 – 99)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
16	45	FILLER_8	1	<comma> separator (“,”)
17	46 – 50	TOL_FARE_ POSTED_AMT	5	Posted amount in cents (00000 – 99999)
18	51	FILLER_9	1	<comma> separator (“,”)
19	52 – 53	NON_REVENUE_ FLAG	2	Indicates if transaction was posted to a non-revenue account (“00” indicates a revenue account and “01” indicates a non-revenue account)
20	54	FILLER_10	1	<comma> separator (“,”)
21	55	PAYMENT_TYPE	1	Type of payment posted (“A” indicates the transaction was an ETOL posted to an ETC account based on the tag status at the lane, “V” indicates the transaction was posted as a violation based on a failure to pay at the lane, and “E” indicates that the transaction was not posted due to an error exception at the RCSC) (See Table 10-35 below)
22	56	FILLER_11	1	<comma> separator (“,”)
23	57 – 59	CSC_REASON_ CODE	3	Reason toll amount was posted to an account or was rejected by the RCSC (see Section 10.1.6.11 RCSC Reason Codes)
24	60	FILLER_12	1	<comma> separator (“,”)
25	61 – 70	BUSINESS_ DATE	10	The business day assigned to the transaction that corresponds to this response and used to identify the revenue date for the original transaction (MM/DD/YYYY)
26	71	FILLER_13	1	<comma> separator (“,”)
27	72 – 77	CSC_BATCH	6	Sequence number assigned by the RCSC to the ETC Transaction File that contained the transaction that corresponds to this response (000000 – 999999)
28	78	FILLER_14	1	<comma> separator (“,”)
29	79 – 94	CSC_ACCT_NO	16	RCSC account number assigned to the BATA RCSC customer account. (Default for Home accounts 0000000000000000) (For other CTOC agencies the CSC_ACCT_NO are set as follows: 0000000000000098 – SR 91 0000000000000097 – SANDAG 0000000000000096 – TCA 0000000000000095 – CTV 0000000000000094 – SENTRY)
30	95	LINEFEED	1	<line feed>

Table 10-22: ETC Response File Received from the RCSC (Trailer Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 8	RECORD_TYPE	8	"#TRAILER"
2	9	FILLER_1	1	<comma> separator (",")
3	10 – 15	SEQUENCE_NO	6	Same as Header Record SEQUENCE_NO
4	16	FILLER_2	1	<comma> separator (",")
5	17 – 26	FILE_DATE	10	Same as Header Record CREATE_DATE
6	27	FILLER_3	1	<comma> separator (",")
7	28 – 35	DETAIL_COUNT	8	Count of all ETC responses contained in the detail records (00000000 – 99999999)
8	36	LINEFEED	1	<line feed>

10.1.6.5.1 RCSC Processing Requirements

All transactions received at the RCSC, via the ETC Transaction File, are sent back to the ATCAS II Host in the response file.

All transactions received at the RCSC, via the ETC Transaction File, can be reconciled back at the ATCAS II Host with final status code passed in the ETC Response File. The reconciliation takes place at file level. For example, if the RCSC receives 100 transactions in file 123, then the same 100 transactions are reconciled back to the ATCAS II Host in one file.

The RCSC produces a response file within 9 hours of receipt of a transaction file and by 10AM the next day.

The RCSC performs transaction reconciliation at a detail transaction level (i.e., the response file contains details at the transaction level instead of a reconciliation summary).

The RCSC reconciles all Away CTOC Agency transactions with Expected Revenue and does not wait for response files from an Away Agency. However, in the event the transactions are later rejected by the Away agency, the variance between the Expected Revenue and revenue posted by the Away agency is reflected through RCSC Reports.

In cases when a transaction cannot be posted at the RCSC, the RCSC indicates the reason the transaction was not posted in the CSC_REASON_CODE field. The possible reason codes and descriptions are provided in Section 10.1.6.11 RCSC Reason Codes.

The RCSC receives a unique integer value (SEQUENCE_NO) in the header record of an incoming transaction file. This identifier is sent back as part of the header record in the response file in order to associate it to the correct transaction file.

The RCSC receives a unique transaction number (TRANSACTION_NUMBER) in each of the transaction detail record. This identifier is sent back as the transaction number in the response detail record in order to associate it to the correct transaction.

At configurable intervals, the ATCAS II Host checks for response files transferred to the Host by the RCSC. When a new file is received, ATCAS II loads the file and performs certain validations. In all cases the ATCAS II Host generates an acknowledgement file and transmits this back to the RCSC. The

acknowledgment file contains a SUCCESS code (value “00”) if the response file passed validation or a FAILURE code (value “01”) if the file failed validation. See Section 10.1.6.10 Incoming File Validation.

10.1.6.6 Violation Response File Received from the RCSC

The violation response file is generated by the RCSC after the violation transaction file is processed to give the final disposition of the transactions. This file consists of a header record, followed by a detail record for each transaction that was posted to a customer account based on the license plate image (ITOL) or based on a change in the tag status (VTOL) in the batch, and then by a trailer record. The record formats are shown in the following tables.

Table 10-23: Violation Response File Received from the RCSC (Header Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 7	RECORD_TYPE	7	“#HEADER”
2	8	FILLER_1	1	<comma> separator (“,”)
3	9 – 12	FILE_TYPE	4	”VRES“
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 19	SEQUENCE_NO	6	Unique sequence number for the Violation Transaction File that corresponds to this response (000000 – 999999)
6	20	FILLER_3	1	<comma> separator (“,”)
7	21 – 30	BUSINESS_DATE	10	Business Date passed in the Violation Transaction File that corresponds to this response (MM/DD/YYYY)
8	31	FILLER_4	1	<comma> separator (“,”)
9	32 – 33	SOURCE	2	“RC”
10	34	FILLER_5	1	<comma> separator (“,”)
11	35 – 36	DESTINATION	2	“AT”
12	37	FILLER_6	1	<comma> separator (“,”)
13	38 – 47	CREATE_DATE	10	File Creation Date (MM/DD/YYYY)
14	48	FILLER_7	1	<comma> separator (“,”)
15	49 – 56	CREATE_TIME	8	File Creation Time (HH:MM:SS)
16	57	LINEFEED	1	<line feed>

Table 10-24: Violation Response File Received from the RCSC (Detail Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 10	TRANSACTION_ NUMBER	10	Unique transaction number used to identify the transaction that corresponds to this response (0000000000 – 9999999999)
2	11	FILLER_1	1	<comma> separator (“,”)
3	12	TOL_TRX_TYPE	1	Type of transaction. “2” = Violation
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 17	TOL_TAG_ID	4	Tag ID in accordance with the Title-21 specs (0000 – 1023)
6	18	FILLER_3	1	<comma> separator (“,”)
7	19 – 24	TOL_TAG_ FACILITY_ID	6	Facility Code assigned to issuing agency in accordance with the Title-21 specs (000000 – 262143)
8	25	FILLER_4	1	<comma> separator (“,”)
9	26	TOL_TAG_ TYPE	1	Four Bit Tag Type read from the ETC transponder in hexadecimal (0 – F): “0” = CTOC transponder Possible future use for switchable transponders: “4” = HOV2 “8” = HOV3
10	27	FILLER_5	1	<comma> separator (“,”)
11	28 – 37	POSTED_DATE	10	Date the transaction was posted to a patron’s account, posted at an away agency, or was rejected (MM/DD/YYYY)
12	38	FILLER_6	1	<comma> separator (“,”)
13	39 – 41	TOL_PLAZA_ID	3	Plaza ID where the transaction occurred that is part of the unique key needed to identify the transaction that corresponds to this response “002” = “ANTIOCH” “003” = “RICHMOND” “004” = “BAY BRIDGE” “005” = “SAN MATEO” “006” = “DUMBARTON” “007” = “CARQUINEZ” “008” = “BENICIA”
14	42	FILLER_7	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
15	43 – 44	TOL_LANE_ID	2	Lane number where the transaction occurred that is part of the unique key needed to identify the transaction that corresponds to this response (00 – 99)
16	45	FILLER_8	1	<comma> separator (“,”)
17	46 – 50	TOL_FARE_ POSTED_AMT	5	Posted amount in cents (00000 – 99999)
18	51	FILLER_9	1	<comma> separator (“,”)
19	52 – 53	NON_REVENUE_ FLAG	2	Indicates if transaction was posted to a non-revenue account (“00” indicates a revenue account and “01” indicates a non-revenue account) or <blank> if no tag was read
20	54	FILLER_10	1	<comma> separator (“,”)
21	55	PAYMENT_TYPE	1	Type of payment posted (“B” indicates the transaction was a VTOL posted to an ETC account based on a change in the tag status, “I” indicates the transaction was an ITOL posted to an ETC account based on the license plate number extracted from the violation image, “E” indicates that the transaction was not posted due to an error exception at the RCSC) with reason codes between 12 – 31, “F” indicate the final disposition of the violation if other than “B”, “I”, or “E” with final reason codes between 102 – 117) (See Table 10-35)
22	56	FILLER_11	1	<comma> separator (“,”)
23	57 – 59	CSC_REASON_ CODE	3	Reason toll amount was posted to an account or was rejected by the RCSC (see Section 10.1.6.11 RCSC Reason Codes)
24	60	FILLER_12	1	<comma> separator (“,”)
25	61 – 70	BUSINESS_ DATE	10	The business day assigned to the transaction that corresponds to this response and used to identify the revenue date for the transaction (MM/DD/YYYY)
26	71	FILLER_13	1	<comma> separator (“,”)
27	72 – 77	CSC_BATCH	6	Sequence number assigned to the Violation Transaction File that contained the transaction that corresponds to this response (000000 – 999999)
28	78	FILLER_14	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
29	79 – 94	CSC_ACCT_NO	16	RCSC account number assigned to the BATA RCSC customer account. (Default for Home accounts 0000000000000000) (For other CTOC agencies the CSC_ACCT_NO are set as follows: 0000000000000098 – SR 91 0000000000000097 – SANDAG 0000000000000096 – TCA 0000000000000095 – CTV 0000000000000094 – SENTRY)
30	95	LINEFEED	1	<line feed>

Table 10-25: Violation Response File Received from the RCSC (Trailer Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 8	RECORD_TYPE	8	"#TRAILER"
2	9	FILLER_1	1	<comma> separator (",")
3	10 – 15	SEQUENCE_NO	6	Same as Header Record SEQUENCE_NO
4	16	FILLER_2	1	<comma> separator (",")
5	17 – 26	FILE_DATE	10	Same as Header Record CREATE_DATE
6	27	FILLER_3	1	<comma> separator (",")
7	28 – 35	DETAIL_COUNT	8	Count of all ETC responses contained in the detail records (00000000 – 99999999)
8	36	LINEFEED	1	<line feed>

10.1.6.6.1 RCSC Processing Requirements

The RCSC sends back response codes for all violations indicating the final state of the violation, including ITOL, VTOL, rejected, paid, and dismissed violations, as shown in the Reason Codes in Table 10-35. There can be multiple response files for one violation request file. The acknowledgement file is used as an indication that the violation transaction file was valid and that the transactions made it to the RCSC.

The RCSC produces violation response files within 9 hours of receipt of a transaction file for rejected transactions and as it becomes aware of new ITOL, VTOL, paid, and dismissed violation transactions to be reported. Violation response files with ITOL, VTOL, paid, and dismissed violation transactions can be sent many days later than the receipt of corresponding violation transaction files. ITOL and VTOL transactions can no longer be posted after the violation is written off at the RCSC (must be posted within 365 days).

The RCSC performs transaction reconciliation at a detail transaction level (i.e., the response file contains details at the transaction level instead of a reconciliation summary).

The RCSC indicates if the transaction was an ITOL, VTOL, or was rejected in the PAYMENT_TYPE field.

In cases when a transaction is rejected, the RCSC indicates the reason in the CSC_REASON_CODE field. The possible reason codes and descriptions are provided in Section 10.1.6.11 RCSC Reason Codes.

The RCSC receives a unique integer value (SEQUENCE_NO) contained in the header record in the incoming transaction file. This identifier is sent back as part of the header record in the response file so that it can be associated to the correct transaction file.

The RCSC receives a unique transaction number (TRANSACTION_NUMBER) in the transaction detail record. This identifier is sent back as the transaction number in the response detail record so that it can be associated to the correct individual transaction.

At configurable intervals, the ATCAS II Host checks for response files transferred to the Host by the RCSC. When a new file is received, ATCAS II loads the file and performs certain validations. In all cases the ATCAS II Host generates an acknowledgement file and transmits this back to the RCSC. The acknowledgment file contains a SUCCESS code (value “00”) if the response file passed validation or a FAILURE code (value “01”) if the reconciliation file failed validation. See Section 10.1.6.10 Incoming File Validation.

The RCSC attempts to match all violation transactions to the corresponding image files and if unsuccessful, continues to retry the matching periodically. All transactions which are pending image review and in a “waiting for image” status for more than 5 days are researched at the ATCAS II Host.

10.1.6.7 Correction Response File Received from the RCSC

The correction response file is generated by the RCSC after the transaction correction file is processed to give the final disposition of the transactions. This file consists of a header record, followed by a detail record for each transaction disposition in the batch, and then by a trailer record. The record formats for the correction response file are shown in the following tables.

Table 10-26: Correction Response File Received from the RCSC (Header Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 7	RECORD_TYPE	7	“#HEADER”
2	8	FILLER_1	1	<comma> separator (“,”)
3	9 – 12	FILE_TYPE	4	”RTRE“
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 19	SEQUENCE_NO	6	Unique sequence number for the Transaction Correction File that corresponds to this response (000000 – 999999)
6	20	FILLER_3	1	<comma> separator (“,”)
7	21 – 30	BUSINESS_DATE	10	Business Date for the ETC Transaction File that corresponds to this response (MM/DD/YYYY)
8	31	FILLER_4	1	<comma> separator (“,”)
9	32 – 33	SOURCE	2	“RC”

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
10	34	FILLER_5	1	<comma> separator (“,”)
11	35 – 36	DESTINATION	2	“AT”
12	37	FILLER_6	1	<comma> separator (“,”)
13	38 – 47	CREATE_DATE	10	File Creation Date (MM/DD/YYYY)
14	48	FILLER_7	1	<comma> separator (“,”)
15	49 – 56	CREATE_TIME	8	File Creation Time (HH:MM:SS)
16	57	LINEFEED	1	<line feed>

Table 10-27: Correction Response File Received from the RCSC (Detail Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 10	TRANSACTION_ NUMBER	10	Unique transaction number to identify the transaction that corresponds to this response (0000000000 – 9999999999)
2	11	FILLER_1	1	<comma> separator (“,”)
3	12	TOL_TRX_TYPE	1	Type of transaction from the TOL_TRX_TYPE_ CORR contained the transaction that corresponds to this response “1” = ETC “2” = Violation “3” = Carpool “4” = Non-Revenue “5” = Hybrid “6” = HOV2 “7” = HOV3
4	13	FILLER_2	1	<comma> separator (“,”)
5	14 – 17	TOL_TAG_ID	4	Tag ID in accordance with the Title-21 specs (0000 – 1023)
6	18	FILLER_3	1	<comma> separator (“,”)
7	19 – 24	TOL_TAG_ FACILITY_ID	6	Facility Code assigned to issuing agency in accordance with the Title-21 specs (000000 – 262143)
8	25	FILLER_4	1	<comma> separator (“,”)
9	26	TOL_TAG_ TYPE	1	Four Bit Tag Type read from the ETC transponder in hexadecimal (0 – F): “0” = CTOC transponder Possible future use for switchable transponders: “4” = HOV2 “8” = HOV3
10	27	FILLER_5	1	<comma> separator (“,”)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
11	28 – 37	POSTED_DATE	10	Date the transaction correction was posted or was rejected (MM/DD/YYYY)
12	38	FILLER_6	1	<comma> separator (“,”)
13	39 – 41	TOL_PLAZA_ID	3	Plaza ID where the transaction occurred that is part of the unique key needed to identify the transaction that corresponds to this response “002” = “ANTIOCH” “003” = “RICHMOND” “004” = “BAY BRIDGE” “005” = “SAN MATEO” “006” = “DUMBARTON” “007” = “CARQUINEZ” “008” = “BENICIA”
14	42	FILLER_7	1	<comma> separator (“,”)
15	43 – 44	TOL_LANE_ID	2	Lane number where the transaction occurred that is part of the unique key needed to identify the transaction that corresponds to this response (00 – 99)
16	45	FILLER_8	1	<comma> separator (“,”)
17	46 – 50	TOL_FARE_ POSTED_AMT	5	Posted amount from the TOL_FARE_CORR_ETC_AMT contained the transaction that corresponds to this response in cents (00000 – 99999)
18	51	FILLER_9	1	<comma> separator (“,”)
19	52 – 53	NON_REVENUE_ FLAG	2	Indicates if transaction was posted to a non-revenue account (“00” indicates a revenue account and “01” indicates a non-revenue account)
20	54	FILLER_10	1	<comma> separator (“,”)
21	55	PAYMENT_TYPE	1	Type of payment posted (“A” indicates the transaction was an ETOL posted to an ETC account based on the tag status at the lane, “V” indicates the transaction was posted as a violation based on a failure to pay at the lane, and “E” indicates that the transaction was not posted due to an error exception at the RCSC) (See Table 10-35 below)
22	56	FILLER_11	1	<comma> separator (“,”)
23	57 – 59	CSC_REASON_ CODE	3	Reason toll amount was posted to an account or was rejected by the RCSC (see Section 10.1.6.11 RCSC Reason Codes)
24	60	FILLER_12	1	<comma> separator (“,”)
25	61 – 70	BUSINESS_ DATE	10	The business day assigned to the transaction that corresponds to this response and used to identify the revenue date for the original transaction (MM/DD/YYYY)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
26	71	FILLER_13	1	<comma> separator (“,”)
27	72 – 77	CSC_BATCH	6	Sequence number assigned to the Transaction Correction File that contained the transaction that corresponds to this response (000000 – 999999)
28	78	FILLER_14	1	<comma> separator (“,”)
29	79 – 94	CSC_ACCT_NO	16	RCSC account number assigned to the BATA RCSC customer account. (Default for Home accounts 0000000000000000) (For other CTOC agencies the CSC_ACCT_NO are set as follows: 0000000000000098 – SR 91 0000000000000097 – SANDAG 0000000000000096 – TCA 0000000000000095 – CTV 0000000000000094 – SENTRY)
30	95	LINEFEED	1	<line feed>

Table 10-28: Correction Response File Received from the RCSC (Trailer Record)

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 8	RECORD_TYPE	8	“#TRAILER”
2	9	FILLER_1	1	<comma> separator (“,”)
3	10 – 15	SEQUENCE_NO	6	Same as Header Record SEQUENCE_NO
4	16	FILLER_2	1	<comma> separator (“,”)
5	17 – 26	FILE_DATE	10	Same as Header Record CREATE_DATE
6	27	FILLER_3	1	<comma> separator (“,”)
7	28 – 35	DETAIL_COUNT	8	Count of all correction responses contained in the detail records (00000000 – 99999999)
8	36	LINEFEED	1	<line feed>

10.1.6.7.1 RCSC Processing Requirements

All transaction corrections received at the RCSC, via the Transaction Correction File, are sent back to the ATCAS II Host in the response file.

All transactions received at the RCSC, via the Transaction Correction File, can be reconciled back at the ATCAS II Host with final status code that is passed in the Correction Response File. The reconciliation takes place at file level. For example, if the RCSC receives 100 transactions in file 123, then the same 100 transactions are reconciled back to the ATCAS II Host in one file.

The RCSC produces a response file within 9 hours of receipt of transaction correction file and by 10AM the next day.

The RCSC performs transaction reconciliation at a detail transaction level (i.e., the response file contains details at the transaction level instead of a reconciliation summary).

In cases when a transaction cannot be posted at the RCSC, the RCSC indicates the reason the transaction was not posted in the CSC_REASON_CODE field. The possible reason codes and descriptions are provided in Section 10.1.6.11 RCSC Reason Codes.

The RCSC receives a unique integer value (SEQUENCE_NO) contained in the header record in the incoming transaction file. This identifier is sent back as part of the header record in the response file so that it can be associated to the correct transaction file.

The RCSC receives a unique transaction number (TRANSACTION_NUMBER) in each transaction detail record. This identifier is sent back as the transaction number in each response detail record so that it can be associated to the correct transaction.

At configurable intervals, the ATCAS II Host checks for response files transferred to the Host by the RCSC. When a new file is received, ATCAS II loads the file and performs certain validations. In all cases the ATCAS II Host generates an acknowledgement file and transmits this back to the RCSC. The acknowledgment file contains a SUCCESS code (value “00”) if the response file passed validation or a FAILURE code (value “01”) if the response file failed validation. See Section 10.1.6.10 Incoming File Validation.

10.1.6.8 Acknowledgement File Sent to and Received from the RCSC

The acknowledgement file sent to the RCSC in response to incoming files and received from the RCSC in response to outgoing files does not have header or trailer records. The record format is shown in the following table.

Table 10-29: Acknowledgement File Sent to and Received from the RCSC

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 4	FILE_TYPE	4	File type (“ACK “)
2	5 – 7	FROM_AGENCY_ ID	3	Agency that received the file referenced in the ORIG_FILE_ NAME_TYPE field and is sending this acknowledgment (“AT2” for ATCAS II and “CSC” for RCSC)
3	8 – 10	TO_AGENCY_ID	3	Agency that transmitted the file referenced in the ORIG_FILE_ NAME_TYPE field and is receiving this acknowledgment (“AT2” for ATCAS II and “CSC” for RCSC)
4	11 – 60	ORIG_FILE_ NAME_TYPE	50	File name and type of the file being acknowledged (this is the file name and type before the file was zipped separated by a period)
5	61 – 68	FILE_DATE	8	Date the acknowledgment file was created (YYYYMMDD)
6	69 – 74	FILE_TIME	6	Time the acknowledgment file was created (HHMMSS)
7	75 – 76	RETURN_CODE	2	Code indicating the status of the file being acknowledged (“00” indicates that the file use successfully received and verified and “01” indicates that the file was received but contained invalid data)
13	77	LINEFEED	1	<line feed>

10.1.6.8.1 RCSC Processing Requirements

This file contains a single record only. For each file received by ATCAS II or RCSC, an Acknowledgement File is generated and transmitted back to the sender. An Acknowledgement File is generated for every file documented in Section 10.1 with the exception of the Violation Data Files and of other Acknowledgement Files.

The RCSC saves the acknowledgements for use in researching and resolving problems via the ATCAS II RCSC interface.

The ACK file indicates a file was successfully received by the receiving agency. The ACK provides an audit trail for research and can be used as a future key event in the RCSC. The RCSC is not designed to recognize the receipt of an ACK file for the continuation of an operation (i.e. CTOC Invoicing).

All incoming files that fail sanity checks (for example, the header record count does not match the trailer record or the record length in file) are rejected with reject code 01.

Records received in transaction files where data elements are inconsistent with the field definitions in this document (for example, invalid date, invalid plaza_id, or invalid lane id) are acknowledged with a code of 00 if the file passed the sanity check. The individual records have reject codes in the response file going back to the sender.

10.1.6.9 Violation Data Files and Images Sent to the RCSC

The violation data file that sent to the RCSC does not have header or trailer record. This file contains the license plate number and state and associated image processing information performed by the LPR Server. The record format was taken from Section 20 of the VECTOR Regional BATA Interface Control Document Rev. 1.6.8 Final dated April 28, 2006 and is shown in the following table.

Table 10-30: Violation Data File Sent to the RCSC

FIELD	POSITION	NAME	LENGTH	DESCRIPTION / CONTENTS
1	1 – 3	AGENCY_ID	3	Agency where the violation transaction occurred (“CAL”)
2	4 – 7	PLAZA_ID	4	Plaza ID where the violation transaction occurred (2 leading spaces followed by 2 digit plaza ID)
3	8 – 10	LANE_ID	3	Lane number where the violation transaction occurred (1 leading space followed by 2 digit lane number)
4	11 – 18	TRX_DATE	8	Date the violation transaction occurred (YYYYMMDD)
5	19 – 26	TRX_TIME	8	Time the violation transaction occurred (HHMMSSTT where TT is 1/100th of a second)
6	27 – 32	VEHICLE_ SEQUENCE_NUM	6	Unique vehicle sequence number assigned to the violation transaction by the lane/zone controller (000000 – 999999)
7	33 – 35	OCR_READ_ CONFIDENCE	3	Overall license plate recognition confidence from the LPR Server as a percentage (1 leading space followed by 2 digit percentage)
8	36 – 45	PLATE_NUMBER	10	License plate number from the LPR Server (left justified with trailing spaces)
9	46 – 49	PLATE_STATE	4	License plate state from the LPR Server (right justified with leading spaces)
10	50	NUMBER_OF_ IMAGES_TRX	1	Number of violation images associated with the violation transaction from the LPR Server
11	51	IMAGE_INDEX_ NUMBER	1	Image index number of the image that was used by the LPR Server to determine the license plate number and state
12	52 – 61	FILLER	10	<blanks>
13	62	LINEFEED	1	<line feed>

10.1.6.9.1 RCSC Processing Requirements

All images (four color images in JPEG format and one black and white image in BMP format) and the corresponding Violation Data File are zipped into a single file (one per violation).

The PLAZA_ID, LANE_ID, and VEHICLE_SEQUENCE_NUM in the Violation Data File is matched with the violation transaction's TOL_PLAZA_ID, TOL_LANE_ID, and LANE_TX_SEQUENCE_NUMBER in the Violation Transaction File in order to match images with the correct violation.

Images are processed based on the overall image confidence level passed in the Violation Data File.

Violation Data Files are not passed to the RCSC without the corresponding images.

10.1.6.10 Incoming File Validation

Data is validated in the following files; tag status file, ETC and violation response files, acknowledgement file, and violation data file. If any part of the file is determined to be invalid, the entire processing of the file is rolled back. This section describes the steps of validation for incoming files.

Table 10-31: Tag Status File Validation

STEP	DETAILS
Examine the filename	Extract the DateTime from the file name. Verify the DateTime is in the valid format and is greater than the Tag Status File DateTime previously processed.
Read file trailer record	Ensure that the RECORD_COUNT is numeric and matches the number of detail records contained in the file. Ensure that the RECORD_COUNT is not more than 10% greater or more than 2% less than the RECORD_COUNT in the previous tag status file. The percentage values are configurable in the tbGenPurpParms table.
Read and examine all detail records in the file	Ensure all the numeric fields contained in each record contain numeric data within the range of values defined in the record description.

Table 10-32: ETC and Violation Response File Validation

STEP	DETAILS
Examine the filename	Extract the DateTime from the file name. Verify the DateTime is in the valid format and is greater than either the ETC Response DateTime or Violation Response DateTime (as appropriate) previously processed.
Read file header record	Ensure the RECORD_TYPE, FILE_TYPE, SOURCE, and DESTINATION fields contain required values, the CREATE_DATE has a valid date value and the CREATE_TIME has a valid time value Ensure the SEQUENCE_NO and BUSINESS_DATE match the ETC Response or the Violation Response (as appropriate) previously sent.

STEP	DETAILS
Read file footer record	Ensure the RECORD_TYPE contains the required value. Ensure the SEQUENCE_NO and FILE_DATE match the SEQUENCE_NO and CREATE_DATE values in the header. Ensure the RECORD_COUNT is numeric and matches the number of detail records contained in the file.
Read and examine all detail records in the file	Ensure all numeric fields contained in each record contain numeric data within the range of values defined in the record description. Ensure the POSTING_DATE, NON_REVENUE_FLAG, and PAYMENT_TYPE values match the format and values defined in the record description. Ensure the TRANSACTION_NUMBER, TOL_TRX_TYPE, TOL_TAG_ID, ETC_TAG_FACILITY_ID, TOL_PLAZA_ID, TOL_LANE_ID, BUSINESS_DATE, and CSC_BATCH match a transaction in the corresponding ETC or violation transaction file (as appropriate).

Table 10-33: Acknowledgement File Validation

STEP	DETAILS
Examine the filename	Extract the From_Agency, File_Name, and File_Type. Verify From_Agency is “CSC”. Verify File_Name and File_Type match a file previously sent to the RCSC.
Read the file	Ensure there is only a single record in the file. Ensure the FILE_TYPE is “ACK”, the FROM_AGENCY_ID is “CSC”, the TO_AGENCY_ID is “CAL”, and the ORIG_FILE_NAME_TYPE matches the File_Name and File_Type values contained in the filename (i.e. File_Name.File_Type). Ensure the FILE_DATE, FILE_TIME, and RETURN_CODE values match the format and values defined in the record description.

Table 10-34: Violation Data File Validation

STEP	DETAILS
Examine the filename	Extract the AGENCY_ID, PLAZA_ID, LANE_ID, DATE, TIME, and SEQ_NO. Verify AGENCY_ID is “CAL”. Verify PLAZA_ID, LANE_ID, DATE, TIME, and SEQ_NO correspond to a vehicle transaction in the database.
Read the file	Ensure there is only a single record in the file. Ensure all the numeric fields contained in each record contain numeric data. Ensure the AGENCY_ID, PLAZA_ID, LANE_ID, TRX_DATE, TRX_TIME, and VEHICLE_SEQUENCE_NUM match the values contained in the filename. Ensure the OCR_READ_ CONFIDENCE value is a percentage. Ensure there are violation image files corresponding to the number of images indicated in NUMBER_OF_ IMAGES_TRX. Ensure IMAGE_INDEX_ NUMBER is less than or equal to NUMBER_OF_ IMAGES_TRX.

10.1.6.11 RCSC Reason Codes

The following table contains a list of RCSC Reason Codes returned by the RCSC. The Pay Type column shows the PAYMENT_TYPE values passed in the response files from the RCSC for each of the Reason Code values (see Sections 10.1.6.5, 10.1.6.6, and 10.1.6.7).

Table 10-35: RCSC Reason Codes

REASON CODE	STATUS	PAY TYPE	DESCRIPTION
LIST OF CODES APPLICABLE TO HOME AGENCY TRANSACTIONS			
01	TOLL	A	Home Agency toll posted successfully as a normal ETC transaction
02	VTOL	B	Home Agency toll posted successfully as a ETC violation transaction
11	TAGINV	V	Tag Inventory - Tag is currently in the Vector's Inventory status. This status indicates theTag is in the CSC. Any transactions received on a tag are violation transactions and go through the violation processing system
12	TAGLOST	V	Tag Lost
13	TAGSTOLEN	V	Tag Stolen
14	TAGRETURNED	V	Tag in shipping, Returned Defective, Tag Returned
15	TAGDAMAGED	V	Tag Damaged
16	INVTAG	V	Invalid tag
22	DUPL	E	Duplicate transaction – transaction occurred on the same plaza/lane for a given device at the same date/time.
24	INVACC	V	Invalid Account
25	INVACCLSP	V	Invalid Account Closing Pending
26	INVACPEND	V	Invalid Account Pending
27	INVACRVKF	V	Invalid Account Revoked Final
28	INVACCLOS	V	Invalid Account Closed
29	POACHING	E	Poaching transaction – transaction occurred on same tag and date and time within 5 minutes on the same lane.
31	XLANE	E	Cross Lane transaction – transaction occurred on same tag and date and time on the same plaza but a different lane.
51	QINVPLAZA	E	Transaction rejected as invalid due to an invalid plaza
52	QINVDATE	E	Transaction rejected as invalid due to invalid date
53	QINVAGENCY	E	Transaction rejected as invalid due to invalid agency code

REASON CODE	STATUS	PAY TYPE	DESCRIPTION
54	QNONVTRX	V	Unpostable ETC txn - Invalid tag/account status
CSC REASON CODES FOR CTOC AGENCY TRANSACTIONS			
06	POST	A	Transaction posted successfully to a CTOC Agency account due to a tag read at the lanes.
07	PPST	I	Transaction posted successfully to a CTOC Agency account as a pay-by-plate transaction by manual review.
43	TAGB	V	Transaction happened on a tag with a bad status
45	RJDP	E	Transaction rejected as duplicate – CTOC transaction occurred on the same plaza/lane for a given device at the same date/time.
46	OLD1	E	Transaction rejected – Attempt to post the transaction to a closed account after the specified posting limit – 30 days or 60 days (configurable)
48	RINV	E	Transaction rejected as invalid due to invalid detail data – i.e. if the tag in the transaction is out of range etc.
CSC REASON CODES FOR VIOLATION TRANSACTIONS			
09	ITOL	I	Home Agency toll posted successfully to a valid account by manual review of license plate information.
102	VIMGREVRJT	F	Violation transaction rejected after image review
109	VDMVRJT	F	Violation transaction rejected after DMV request
112	VDMVPAY	F	Violation transaction paid at DMV Payment
113	VDMVREL	F	Violation transaction paid at DMV Release
114	VPFULL	F	Violation transaction paid full
116	VCOLLECT	F	Violation transaction sent to collection
117	VDISS	F	Violation transaction dismissed
206	OCRITOLAWAY	I	Toll posted successfully to a valid away agency account using license plate information
209	OCRITOLHOME	I	Toll posted successfully to a valid home agency account using license plate information
999	VNONTRANRJT	E	Violation correction rejected because of non posted transaction

10.1.7 Business Day Tracking

The RCSC interface process sends ETC and Violation transactions to the RCSC in batches at configurable intervals, waits for an acknowledgment and a response file for each batch, and stores the file

transfer information in the tbRCSCSendFile and tbRCSCReceiveFile tables. The RCSC interface uses the tbBusinessDay table to track the status of the RCSC interface process in relation to the business day.

Each batch of transactions sent to the RCSC is sent for a specific business day (see Section 10.1.4.1 Logic Flow Example). As files are sent and acknowledgement and response files received, the RCSC interface updates the tbBusinessDay table for the business day and plaza.

Each time the RCSC interface runs, it builds the transactions to be sent to the RCSC in the tbRCSCSendTrans and tbRCSCSendCorrections tables and then groups them into batches based on business day to be linked to the records inserted into the tbRCSCSendFile table using the iSendFileID field. A file is then created for each record inserted into the tbRCSCSendFile table and sent to the RCSC.

As acknowledgment files are received from the RCSC, a record is inserted in the tbRCSCReceiveFile table and the corresponding tbRCSCSendFile record is updated with the acknowledgment data. As response files are received from the RCSC, a record is inserted in the tbRCSCReceiveFile table. The iSendFileId in the tbRCSCReceiveFile record associates it to the corresponding tbRCSCSendFile record.

The following fields in the tbBusinessDay table are used to track the RCSC interface process in relation to the business day.

tiVioSentStatus – indicates the current status of the violation transfer to the RCSC for a given plaza and business day (0 = not started, 1 = started, 2 = completed for entire business day, 3 = errors encountered)

tiETCSentStatus – indicates the current status of the ETC transaction transfer to the RCSC for a given plaza and business day (0 = not started, 1 = started, 2 = completed for entire business day, 3 = errors encountered)

tiETCPostedStatus – indicates the current status of the ETC transaction transfer to the RCSC for a given plaza and business day (0 = not started, 1 = response file received, 2 = response file received for entire business day, 3 = errors encountered, 4 = response file received for entire business day and not all transactions were posted)

At the end of the RCSC interface process the data inserted is checked and the tbBusinessDay table is updated accordingly. The following table shows conditions causing the RCSC interface to set the indicated values.

Table 10-36: tbBusinessDay RCSC Interface Field Values

Field Value	Conditions That Cause Field to be Set
tiVioSentStatus = 1	Some of the violations for the business day and plaza have been sent and acknowledged.
tiVioSentStatus = 2	All of the violations for the business day and plaza have been sent and acknowledged.
tiVioSentStatus = 3	An error status was received in at least one of the violation transaction file acknowledgments.
tiETCSentStatus = 1	Some of the ETC transactions for the business day and plaza have been sent and acknowledged.
tiETCSentStatus = 2	All of the ETC transactions for the business day and plaza have been sent and acknowledged.
tiETCSentStatus = 3	An error status was received in at least one of the ETC transaction file acknowledgments.
tiETCPostedStatus = 1	Some of the ETC transactions for the business day and plaza have been sent and a response has been received.
tiETCPostedStatus = 2	All of the ETC transactions for the business day and plaza have been sent and a response

Field Value	Conditions That Cause Field to be Set
	has been received and all of the transactions were successfully posted to accounts.
tiETCPostedStatus = 3	A valid ETC response file has not been received and the required time limit for receiving the ETC response file has passed.
tiETCPostedStatus = 4	All of the ETC transactions for the business day and plaza have been sent and a response has been received but not all of the transactions were successfully posted to accounts.

10.1.8 Error Processing

The RCSC interface process may fail if it detects an error due to FTP Server unavailability or transmission data protocol error, or by exceeding the number of transmission retry attempts. In this case the RCSC interface process uses the event code 20010 (External Interface Failure – RCSC FTP Server) and stores a record in the tbEquipFailure table.

Data is also validated for the tag status file, transaction response file, acknowledgement file, and violation data file. When invalid data is found in the tag status, transaction reconciliation, and acknowledgement files, event code 20011 (External Interface Failure – RCSC Invalid File) is generated.

If a tag status file or a transaction response file is not received within a configurable amount of time after received the previous file, event code 20012 (External Interface Failure – RCSC Missing File) is generated. If an acknowledgement file is not received within a configurable amount of time after the transfer of a file to the RCSC, event code 20013 (External Interface Failure – RCSC Missing ACK) is generated. If violation images are not available within a configurable amount of time after receiving a violation transaction, event code 20017 (External Interface Failure – VES Missing Images) is generated.

The MOMS Alarm Loader software processes the event code, generates a work order and notifies the assigned staff on call to resolve the problem.

The detailed trace is stored in the process log file and available to the system administrator and database administrator for further analysis.

The MOMS application reports the failure via the Event History report.

All files sent and received are saved in the archive subfolder. They serve as both an audit of the process, and a source for re-transmittal in the case of an error on either the RCSC or the ATCAS II Host. If the file is not successfully transferred or processed, MOMS interface notifies the maintenance staff. Once the problem area is determined and resolved, the file is re-transmitted to the Outgoing directory and normal processing resumes. An independent cleanup process is scheduled to clean up files from the archive subfolder that are older than two (2) months (configurable).

10.1.9 RCSC Interface Database Schema

The RCSC interface process database schema is shown in Figure 10-5.

BAIFA Express Lane Network Toll Collection System

Appendix 1, Reference 8

System Interface Documentation

B: Bay Area Video Upgrade (BAVU) Cameleon Interface Driver

NOTE: This information is provided for informational purposes and is subject to change.

Driver Application Notes

Host Interface Driver

icx™
360 SURVEILLANCE

Cameleon Driver Application Notes
Host Interface Driver

20APR07

© Copyright 2007
360 Surveillance Inc.
Camera Cameleon is a trademark of 360 Surveillance Inc.

The entire risk of the use or the result of the use of this software and documentation remains with the user. No part of this documentation may be reproduced by any means, electronic or mechanical, for any purpose, without the consent of 360 Surveillance, Inc.

360 Surveillance, Inc.
4218 Commerce Circle
Victoria, B.C. V8Z 6N6
Canada

Phone 1-250-388-7232
Toll free 1-888-242-4291
Fax 1-250-388-7229
E-mail mail1@360surveillance.com
Web www.360surveillance.com

Table of Contents

List of Figures - - -

List of Figures

Section 2 - Before You Begin

Figure 2.1 - Host Interface Driver configuration example 4

Section 4 - Configuration

Figure 4.1 - Create a new database 9
Figure 4.2 - Configure Driver dialog box 10
Figure 4.3 - Add the driver to the server configuration 11
Figure 4.4 - General tab 12
Figure 4.5 - Properties dialog box for a Host Interface 13
Figure 4.6 - Communications tab - Direct serial communications 14
Figure 4.7 - Communications tab - Internet Protocol communications 14
Figure 4.8 - PelcoNet Username/Password dialog box 15
Figure 4.9 - Configure Modems dialog box 16

List of Tables

Section 3 - Serial Protocol

Table 3.1 - Camera Commands 6
Table 3.2 - Switching Commands 6
Table 3.3 - System Commands 7
Table 3.4 - Result codes 7

Section 4 - Configuration

Table 4.1 - Host Interface permission. 16
Table 4.2 - Host Interface read properties 17

SECTION

1

Scope

These Driver Application Notes cover the configuration of the Host Interface Driver.

1.1 Devices Supported

The Host Interface Driver can be used with any third party application that can be programmed to issue PTZ and/or switching commands using the Generic 360 ASCII Protocol to the PC the Host Interface Driver runs on.

For an overview of the Generic 360 ASCII Protocol see Section 3 - *Serial Protocol* (p.5).

SECTION 2

Before You Begin

2.1 Overview

The Host Interface Driver provides a generic interface to access cameras for switching and PTZ control. Using the Generic 360 ASCII Protocol, a third party application can be configured to issue commands to the Host Interface Driver which in turn translates and conveys the commands to the desired devices.

The Host Interface Driver runs as part of the Cameleon Server software. The third party application provides a user interface for controlling the device(s) the Host Interface receives, translates, and sends commands to. The Cameleon Server and Host Interface Driver can operate in the background outside of the realm of users of the third party application.

In order for the Host Interface Driver to control a device, for example a camera, the third party application must be configured to send a command that specifies the device number of the camera the Host Interface will control. For instructions on configuring a third party application to send commands using the Generic 360 ASCII protocol see Section - *Serial Protocol* (p.5). For instructions on defining device numbers refer to the Cameleon Server Manual.

NOTE: The controlled device must be configured in the Cameleon Server and its device driver must be running.

NOTE: Each Host Interface device can control multiple devices though this is only recommended in a single-user environment. If you are configuring a system involving many users controlling many different devices, add as many Host Interfaces to your system as there are controllable devices. Failing to configure a distinct Host Interface for each controlled device will severely limit performance.

2.2 Cabling

Communications between the Host Interface Driver and the third party application can be configured in three ways:

- Direct Serial
- Internet Protocol
- Modem

  NOTE: Modem communications are not recommended.

Communications settings are configured on the Host Interface's Communications tab. See Section 4.2.3 - *Communications Tab* (p.13).

Multiple Host Interfaces can be configured to operate on a single COM port or IP address. This allows the third party application to control multiple devices and multiple Host Interfaces through one direct serial or internet protocol connection.

NOTE: Before configuring multiple devices on a single COM port, refer to Section 2.3 - *A Note on Licensing* (p.3).

2.3 A Note on Licensing

Cameleon licensing for the Host Interface driver is based on the number of distinct ports used by the Host Interface Driver regardless of the number of devices controlled by the driver.

Each port allocated in the Host Interface Driver will subtract five licenses from your configured total. For example, if you have twenty devices configured in the Host Interface Driver spread across two ports you will need ten licenses since each port requires five. Contact Sales at 1-800-242-4291 for more information.

2.4 Preliminary Considerations

Before you configure Host Interfaces in the Cameleon server configuration, configure the third party application's internal settings and the settings of the devices to be controlled. In particular:

- **COMMAND PROTOCOL:** Configure the third party application to send PTZ and switching commands to the Host Interface using the Generic 360 ASCII Protocol. For an overview of the Generic 360 ASCII Protocol see Section - *Serial Protocol* (p.5).
- **COMMUNICATIONS SETTINGS:** If the Host Interface Driver will communicate with third party application via direct serial communications, take note of the baud rate, parity, data bits, stop bits, and flow control settings of the third party application. You will have to enter these settings on the Communications tab when you configure the Host Interface in Cameleon.
- **DEVICE NUMBERING:** Assign devices to be issued commands from the Host Interface Driver distinct device numbers using the Device Number Manager in the Cameleon server. For instructions on assigning device numbers refer to the Cameleon Server Manual. You will need to use these device numbers when configuring the third party application's command protocols.

2.5 System Configuration

In order for the Host Interface Driver to operate correctly in a given system, you must add as many instances of the Driver as there are users of the Driver in the system. The way these instances are configured depends upon the number of cameras the Host Interface Driver controls.

CAMERAS: For each camera you will be using the Host Interface to control and to poll regularly, add a number of instances of the Host Interface Driver all configured to use the communications port (COM or IP address) as the controlled camera.

USERS: The total number of instances of the Host Interface Driver should equal the total number of users in the system. Note that these instances may be configured differently depending on the controlled cameras.

See Section 2.5.1 - *Configuration Example* (p.4) for an example of a system using the Host Interface Driver.

2.5.1 Configuration Example

Figure 2.1 Host Interface Driver configuration example

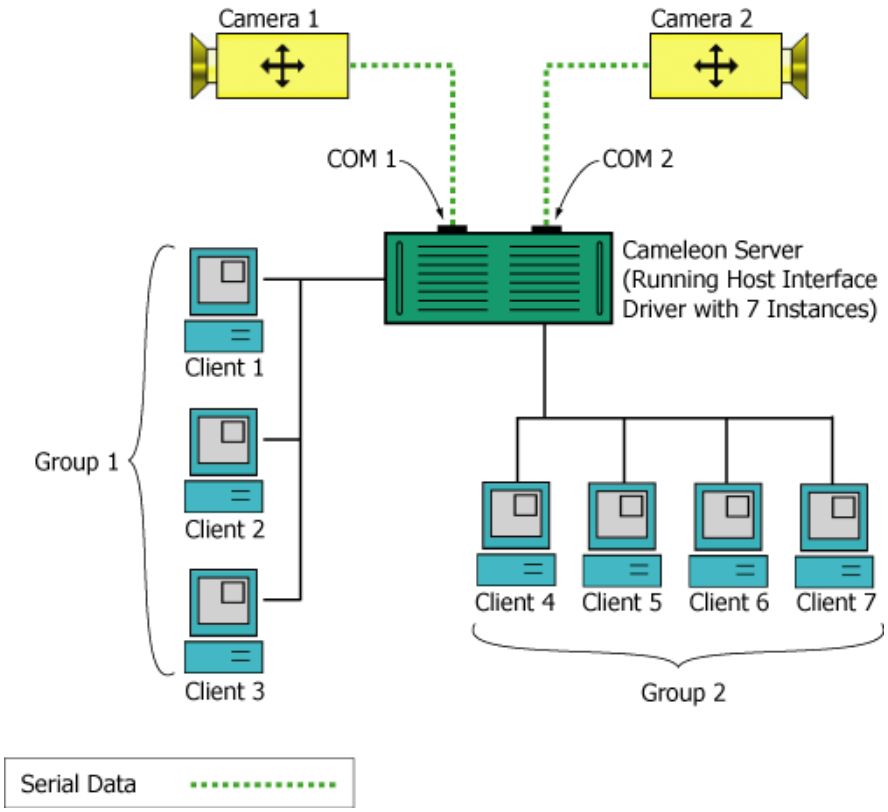


Figure 2.1 outlines a typical Host Interface system configuration. In this system, there are two cameras, Camera 1 and Camera 2. Camera 1 is controlled via COM 1 and Camera 2 is controlled via COM 2. Both cameras are polled regularly.

The system includes two 'Client Groups' (note that this term is only used for explanatory purposes and is not configured in the server application), Group 1 and Group 2. Group 1 has three clients, all of whom will use Camera 1. Group 2 has four clients, all of whom will use Camera 2.

In order for this system configuration to operate correctly, seven instances of the Host Interface Driver are required, reflecting **all** of the clients. Note, however, that these instances will be configured differently for the clients of Group 1 and the users of Group 2.

Host Interface Driver instances 1 through 3 are used by Group 1. Since Group 1 controls Camera 1, these instances are all configured to operate via serial communications and will **all use COM 1, Camera 1's COM port**.

Host Interface Driver instances 4 through 7 are used by Group 2. Since Group 2 controls Camera 2, these instances are configured to operate via serial communications and will **all use COM 2, Camera 2's COM port**.

This configuration uses 10 licenses. This is because the configuration requires the use of 2 COM ports and each COM port requires five licenses. The total number of instances of the Host Interface Driver has no bearing on this figure, only the number of ports used by the Host Interface Driver does.

SECTION

3

Serial Protocol

3.1 Overview

DATA FORMAT: All commands except for the login command are in the following format:

[1-999999] | [Command1] [Command1 Data] | [Command] [Command Data] <CR>

The initial [1-999999] represents the camera number that all subsequent commands apply to until the end of string terminator <CR> is received. Each subsequent to command is applied to the camera specified at the start of the command. No command is processed until the end of string terminator is received. Each command is preceded by "|".

NOTE: If more than one command is sent, any error code corresponds to the first error encountered. Commands following the error are dropped.

NOTE: If more than one camera command of the same type is chained into one command, both commands will be sent to the camera in turn.

MESSAGE SPACING: Every command returns an acknowledgement. The response must be received by the transmitting device prior to sending the next command. If the Host Interface Driver does not respond to a command within 5 seconds, it has failed to interpret or has missed the command, allowing for the next command from the third party application to be sent.

NOTE: Do not send a new message until the previous message's response is received.

Alternately, configure your application to delay each command by a fixed time. The adequate delay will vary depending upon the complexity of your system, the form of communications used, and the system's overall latency. If you experience poor performance, experiment with different fixed command delays.

NOTE: There is not necessarily a one to one correspondence with commands sent to the Host Interface Driver and the commands sent to the devices. For example, if a Pan Right command is sent and is immediately followed by a Pan Left command, the Host Interface Driver will send both commands to the server and on the PTZ camera driver. The PTZ camera driver, however, may just send the Pan Left command if there are any other queued commands waiting to be sent. This is most likely to occur when the Host Interface Driver is configured for a faster communications rate than the PTZ camera driver.

3.2 Generic 360 ASCII Protocol

The Generic 360 ASCII Protocol is comprised of the following types of commands and codes:

- **CAMERA COMMANDS:** See Table 3.1 (p.6).
- **SWITCHING COMMANDS:** See Table 3.2 (p.6).
- **SYSTEM COMMANDS:** See Table 3.3 (p.7).

- **RESULT CODES:** See Table 3.4 (p.7).

Refer to Section 3.2.1 - *Command Examples* (p.7) for examples of commonly used commands.

Table 3.1 Camera Commands

ACTION	COMMAND*	DATA	RESULT	NOTES
Pan Camera	P	[-100 to +100] The speed is an optional parameter. + and - may be used by themselves to indicate direction only. If no speed is provided, the driver will assume a speed of 50% in the case of a variable speed camera. A fixed speed camera will move at the same rate regardless of the speed given. If no argument is provided, the camera will move as though a + was issued.	[CAM#]|<CR>	Positive Pan speed indicates a clockwise movement.
Tilt Camera	T		[CAM#]|<CR>	Positive tilt speed indicates upwards movement
Zoom Camera	Z		[CAM#]|<CR>	Positive zoom speed indicates zoom in.
Focus Camera	F		[CAM#]|<CR>	Positive focus speed indicates focus near.
Open/Close Iris	I		[CAM#]|<CR>	Positive iris speed indicates Iris Open.
Stop All Camera Movement	S		[CAM#]|<CR>	Sending a 0 speed can also be used to stop motion on one axis.
Get Absolute Position	Q	No data	[CAM#]Axxx.xx Exxx.xx Zxxx.xx Fxxx.xx<CR>	Normalized values in degrees.
Goto Absolute Position	A	Optionally [Range of motion]CMD[Speed]: Azimuth: [-180.00 to +180.00]Ä[0-100]S Elevation: [-180.00 to -180.00]E[0-100]S Focus: [0-100]F[0-100]S	[CAM#]|0<CR>	Normalized values in degrees. Any or all of the four options parameters may be specified. Parameters not specified will not be changed at the device level.
Lock/Unlock Camera	L/U	Priority [1-100]	[CAM#]|0<CR>	Cannot be locked at a priority higher than the assigned user priority.
Query Lock State	K		#|L0<CR> (Unlocked) #|L12<CR> (Locked) Where#-Cam#	
Goto Preset Position	\	Preset number [1-99]	[CAM#]|0<CR>	
Set Preset Position	!	Preset number [1-99]	[CAM#]|0<CR>	
Set/Reset Auxiliary	X/Y	Auxiliary number [1-99]	[CAM#]|0<CR>	

*Upper or lowercase accepted

Table 3.2 Switching Commands

ACTION	COMMAND*	DATA	RESULT	NOTES
Switch camera to monitor	M	Monitor number [1-9999]	[CAM#]|<CR>	

*Upper or lowercase accepted

Table 3.3 System Commands

ACTION	COMMAND*	DATA	RESULT	NOTES
Echo Typed characters locally	E	1 (turn on) 0 (turn off)	0|0<CR>	
Login to port (MAY NOT BE FOLLOWED BY OTHER COMMANDS)	*	Username:Password	0|0<CR>	Username and password cannot be defined with ':' character (case sensitive)

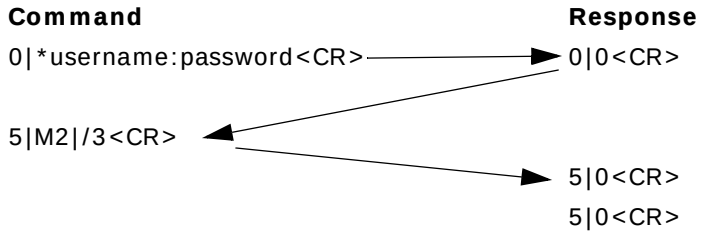
*Upper or lowercase accepted

Table 3.4 Result codes

CODE	DESCRIPTION	RESOLUTION
0<CR>	Command accepted	
1<CR>	Error	General error not covered by specific result code
2<CR>	User not logged in	Login or configure user on server
3<CR>	Username or password incorrect	Retry with correct username and password, case sensitive
11<CR>	Camera does not exist	User may not have permission or camera may not have a device number configured
12<CR>	Camera is locked	Issue unlock command or increase priority level of user
13<CR>	User cannot select camera	User does not permission for this camera
14<CR>	No monitor selected	Cannot move this camera without first switching it to a monitor. Occurs with specific models of switcher, when camera controls are passed onto the switcher, rather than directly to a camera.
15<CR>	Camera failed to respond	Troubleshoot communications with camera
21<CR>	Monitor does not exist	Monitor may not be configured with a device number
22<CR>	User cannot select monitor	User does not have permission for this monitor

3.2.1 Command Examples

1. **LOGIN:** Sample login procedure, switch camera 5 onto monitor 2, and send the camera to a preset.



2. **PAN AT LIMITED SPEED:** Pan camera 2 to the right at a speed of 75%

Command	Response
2|P75<CR>	2|p75<CR>
2|P+75<CR>	2|p+75<CR>

3. **PAN/TILT AT TWO DIFFERENT SPEEDS:** Pan camera 2 to right at a speed of 75% and down at a speed of 40%

   2|P75|T-40<CR>
   2|T-40|P75<CR>
   2|P+75|T-40<CR>

4. **PAN LEFT AT LIMITED SPEED:** Pan camera 2 to the left at a speed of 50%

   2|P-<CR>
   2|P-50<CR>

5. **STOP PANNING:** Stop camera 2 from panning (only when panning)

   2|P0<CR>
   2|S<CR>

6. **STOP PANNING AND TILT:** Stop camera 2 from panning and tilt down at 40%

   2|P0|T-40<CR>

7. **MOVE TO AN ABSOLUTE POSITION:** Move camera 7 to absolute position with azimuth of 121.1 degrees, elevation of 45.6 degrees, and zoom angle of 3.2 degrees at maximum speed

   7|A121.1A100S45.6E100S3.2Z100S<CR>

8. **GET ABSOLUTE POSITION:** Get absolute position after last command complete

Command	**Response**
7|Q<CR>	7|A121.1A45.6Z3.2F0<CR>

   Note that fields not supported will return 0 in the response.

9. **MOVE BY DEGREE:** Move camera 7 10 degrees to the left at a speed of 6%

   7|R-10A6S<CR>

10. **ECHO TYPED CHARACTERS:** Echo typed characters locally. (Camera number is ignored unless followed by other commands that require a camera number)

    0|E1<CR>

11. **CHECK IF CAMERA LOCKED:** Check if camera 2 is locked

Command	**Response**
2|K<CR>	2|12<CR>

    Response code 12 indicates that the camera is locked.

SECTION

4

Configuration

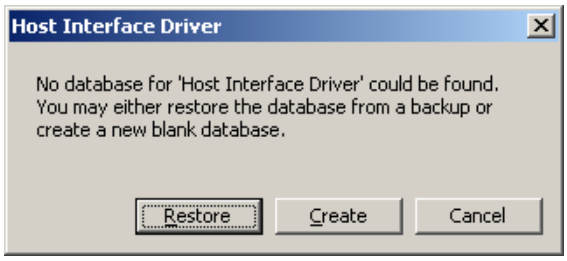
4.1 Overview

This section provides an overview of the steps to configure the Host Interface Driver. Refer to the Cameleon Server Manual for detailed information on each step.

MULTIPLE INSTANCES: Configuring the Host Interface Driver will likely involve creating multiple instances of Host Interface Driver devices. For an overview of how to configure these instances refer to Section 2.5 - *System Configuration* (p.3).

1. **START DRIVER:** Start the device driver. If this is the first time you have configured the Host Interface Driver, click **Create** when asked whether you want to create or restore the database.

Figure 4.1 Create a new database



2. **OPTIONS TAB:**

   (a) **SELECT NETWORK ADAPTER:** If the driver is running on a dual-NIC PC, select the local network adapter from the drop-down list.

Figure 4.2 Configure Driver dialog box

(b) **ENTER PORT:** Enter a unique listening port. No two drivers can share a port. The default listening port for the Host Interface Driver is 2089.

(c) **STARTUP OPTIONS:** You can run the Host Interface Driver application as an executable or as a Windows service. If you install the application as a service, and the PC it runs on is restarted, the Host Interface Driver application will automatically restart. If you run the application as an executable, and the PC it runs on is restarted, the Host Interface Driver application will not automatically restart; you will have to restart the application manually.

Do one of the following:

(i) **REMOVE AS SERVICE:** If the driver is installed as a Windows service and you want to run it as an executable:

-1- Select **Remove the driver as a service** and click **OK**. The Configure Host Interface Driver dialog box will close.

-2- Open the Configure Host Interface Driver dialog box as described in step 1.

-3- Select **Run the driver as an executable**.

(ii) **RUN AS EXECUTABLE:** To run the Host Interface Driver application as an executable, select **Start the driver as an executable**.

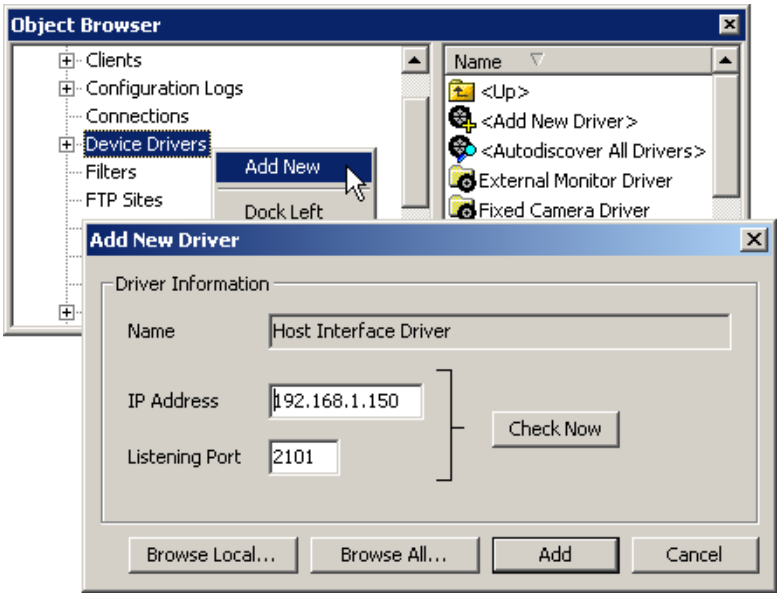
(iii) **INSTALL AS SERVICE:** To install the Host Interface Driver application as a Windows service, select **Install the driver as a service**.

(d) Click **OK** to close the Configure Driver dialog box.

3. **LOG IN:** Log in to server configuration mode. If this is the first time you have logged in, use the predefined administrator account (user name: *administrator*, password: *vis*).

4. **ADD DRIVER:** If you are configuring the Host Interface Driver for the first time, add the driver to the server configuration.

Figure 4.3 Add the driver to the server configuration



5. **ADD HOST INTERFACES:** Add the instances of the Host Interface Driver to the server configuration and configure the settings on the tabs, as described in Section 4.2 - *Host Interface Properties* (p.12). You can add multiple instances of the Driver on the same COM port or IP address. For an overview of how many instances of the driver to add and how to configure those instances refer to Section 2.5 - *System Configuration* (p.3).

   TIP: You can also configure the properties of instances of the Host Interface Driver after you have added the Host Interface to the server configuration by right-clicking the Host Interface in the Object Browser and selecting **Edit** from the pop-up menu.

6. **ASSIGN PERMISSIONS:** Assign users permission to access the Host Interface. See Section 4.3 - *Permissions* (p.16) for more information.

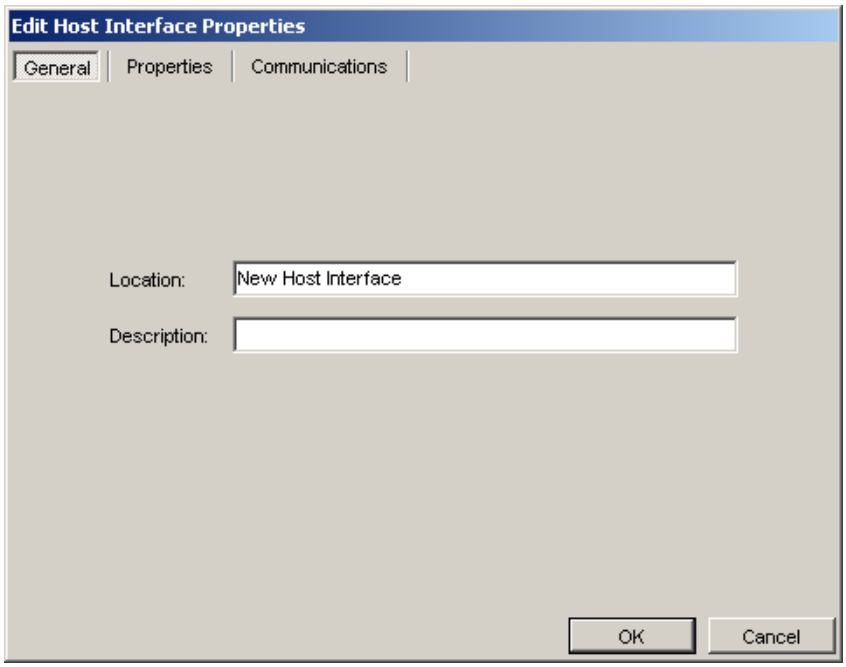
7. **WRITE SCRIPTS:** Write scripts that monitor the Host Interface. See Section 4.4 - *Read Properties* (p.17) for more information.

8. **SAVE CONFIGURATION:** Select **File > Save** to save the server configuration.

4.2 Host Interface Properties

4.2.1 General Tab

Figure 4.4 General tab



The General tab allows you to name and describe the interface.

1. **LOCATION:** Type in the location for the interface. The location acts as a name for the interface and appears in the Object Browser and in the Device List. Enter a location name that helps to locate the port or third party application that will interface with this device.

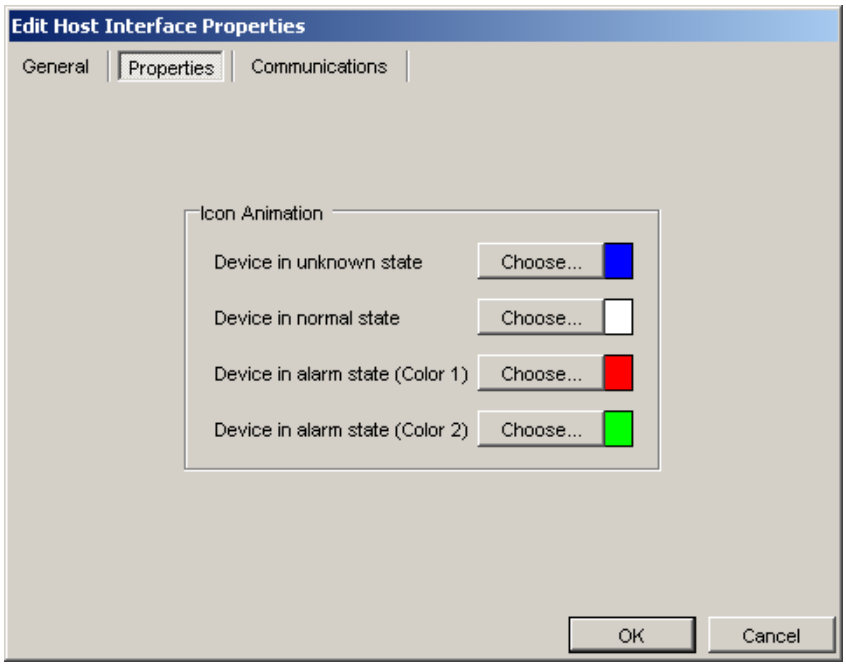
   The location can be at most 50 characters long and can include any letter, digit or special character, with the exception of single and double quotation marks. The location is not case sensitive.

   Cameleon automatically creates a script tag for the device from the location. If the location contains special characters, they will be omitted from the tag. Spaces will be replaced with underscores. Leading digits will be removed. To edit the script tag, edit the device's local settings. Refer to the Cameleon Server Manual for information on configuring a device's local settings.

2. **DESCRIPTION:** Type in a description of the interface, if desired. The description appears in the Object Browser beside the location.

4.2.2 Properties Tab

Figure 4.5 Properties dialog box for a Host Interface



ICON ANIMATION: The icon animation settings allow you to define colors for interface icons to appear under different conditions. Icon animation settings only affect the color of device icons in the Cameleon client; they do not affect the color of device icons in the server configuration.

1. **UNKNOWN STATE:** Click **Choose** to the right of "Device in unknown state" to select a color for the interface's icons when the interface's CommStatus read property has value Offline.

2. **ALARM COLOR 1:** When the interface's Alarm write property has value True, the interface's icons flash alternately between Color 1 and Color 2. Click **Choose** to the right of "Device in alarm state (Color 1)" to select Color 1.

3. **ALARM COLOR 2:** Click **Choose** to the right of "Device in alarm state (Color 2)" to select the second color for flashing device icons.

   NOTE: Host Interface icons will only appear to flash if you choose different colors for Color 1 and Color 2.

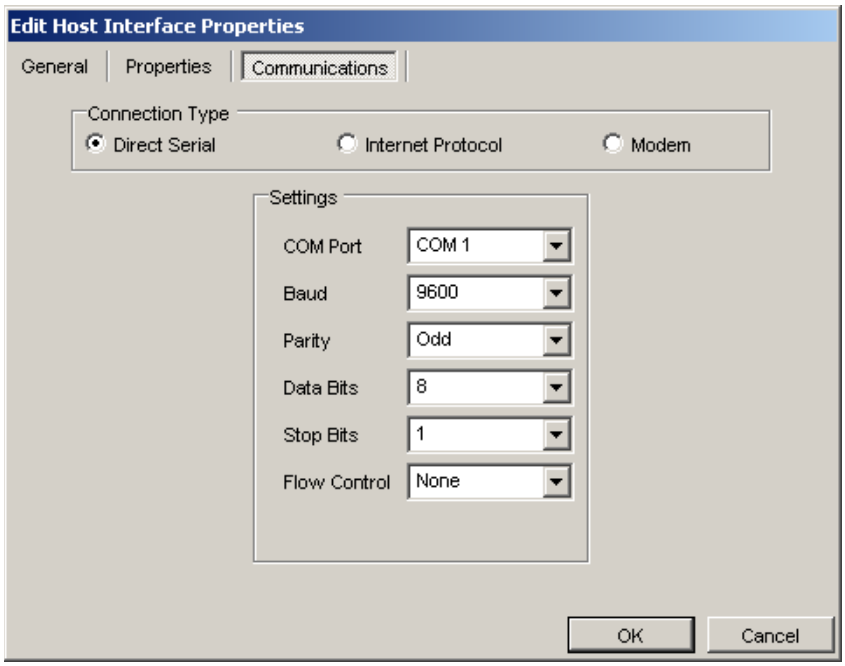
4.2.3 Communications Tab

The Communications tab allows you to configure the communications settings between the Host Interface driver and the third party controller application. Select the form of communications you are using and refer to the instructions for that method.

NOTE: Modem communications are not recommended.

4.2.3.1 Direct Serial Communications

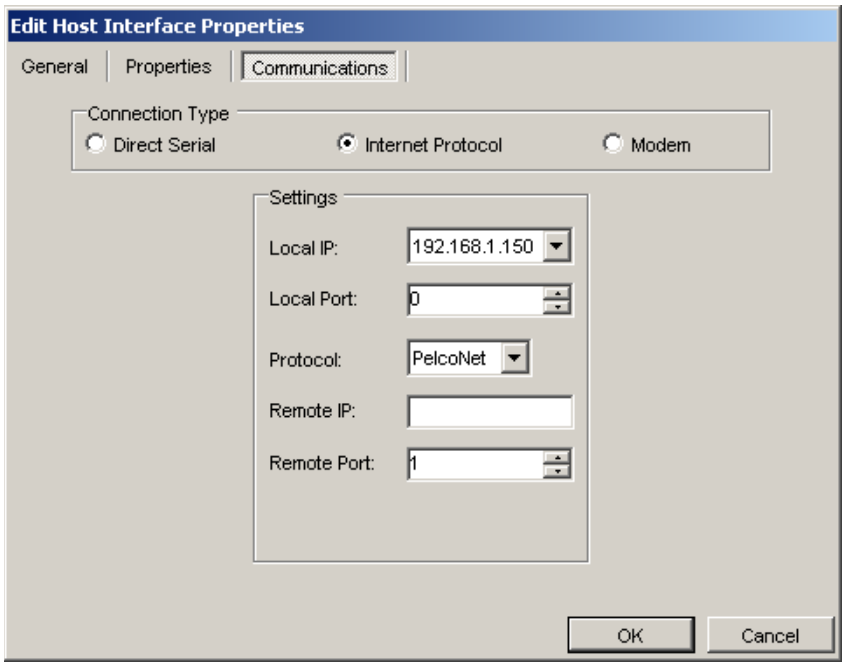
Figure 4.6 Communications tab - Direct serial communications



1. **COM PORT:** Select the COM port the third party application is connected to.
2. **COMMUNICATIONS SETTINGS:** Enter the baud rate, parity, data bits, stop bits, and flow control. These values must match the third party application's internally-configured settings. Refer to the third party documentation for information on configuring the application's internal settings.

4.2.3.2 Internet Protocol Communications

Figure 4.7 Communications tab - Internet Protocol communications



1. **LOCAL IP:** The local IP is the IP address of the computer on which the Host Interface Driver application runs. Select the local IP from the drop-down list.

2. **LOCAL PORT:** The local port is the port the Host Interface Driver uses to receive commands. The default value, 0 (zero), allows the driver to randomly assign an available port. Random port assignment is recommended whenever possible.

   Use a non-zero port in the following circumstances:

   - If your system is secured behind a firewall that restricts incoming connections, enter one of the ports made available through your firewall.
   - If you are running other software on the driver PC, select a non-zero port to prevent port conflicts between the driver application and the other software.

   NOTE: The local port cannot be the same as the listening port configured for any device driver. The listening port is configured when the driver is started.

3. **PROTOCOL, REMOTE IP, REMOTE PORT:** The remote device is the third party application that issues commands to the Host Interface driver. If the third party application issues commands via a PelcoNet device, follow the instructions in (i). If the third party application is connected to some other kind of networked device, follow the instructions in (ii).

   (a) **PELCONET:** Select **PelcoNet** from the Protocol drop-down list. The PelcoNet Username/Password dialog box will open.

   Figure 4.8 PelcoNet Username/Password dialog box

   In the Password box, type in the password that is defined in the PelcoNet unit. If the PelcoNet device is not password protected, leave the Password box blank. Click **OK** to close the PelcoNet Username/Password dialog box.

   Enter the IP address of the PelcoNet device in the Remote IP box.

   In the Remote Port box, select the port on the PelcoNet device that the third party application is connected to. Select **1** if the third party application issues commands to COM1 on the PelcoNet device. Select **2** if the third party application issues commands to COM2.

   NOTE: If your system is secured behind a firewall that restricts outgoing connections, you must open outbound access for port 1756.

   (b) **OTHER:** Select the desired transport protocol from the Protocol drop-down list.

   The remote IP is the IP address of the networked device the third party interface issues commands to through a serial port. Enter the networked device's IP address in the Remote IP box. In the Remote Port box, enter the TCP/UDP port configured in the networked device.

4.2.3.3 Modem Communications

NOTE: Modem communications are not recommended.

4.3 Permissions

Table 4.1 describes the permission for Host Interfaces. Each Host Interface you configure appears in the Permissions tree with this permission.

Figure 4.9 Configure Modems dialog box

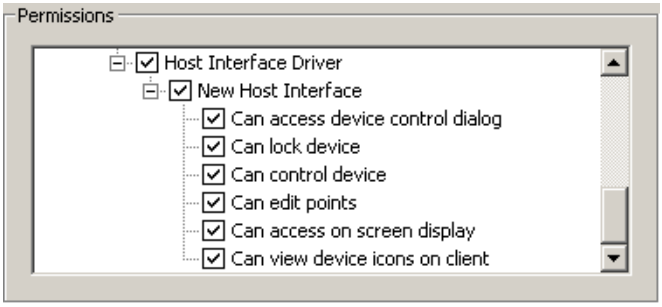


Table 4.1 Host Interface permission

PERMISSION	ENABLES USER TO
Can access device control dialog	*Do not use.*
Can lock device	*Do not use.*
Can control device	*Do not use.*
Can edit points	*Do not use.*
Can access on screen display	*Do not use.*
Can view device icons in client	Allows Host Interface devices to be visible in the device list in the client application. Does not affect Host Interface device icons in the server application.

4.4 Read Properties

Table 4.2 lists the read properties that can be used in scripts and expressions to monitor interfaces. Host Interfaces do not have any write properties.

To use properties in scripts and expressions, properties must be exposed in the device's Local Settings. Properties are exposed by default. Refer to the Cameleon Server Manual for instructions on exposing a device's properties.

TIP Use the global read property System.DriverStatus to test whether the server can connect to all the device drivers that are configured on that server. System.DriverStatus has value Online when the server can connect to every device driver, and value Offline when there are one or more drivers that the server cannot connect to. For example, you might define an alarm that is triggered when the following expression is true:

System.DriverStatus = System.Offline

TIP You can view the current value of a device's read properties on the Cameleon server PC, as follows: Right-click the Cameleon server icon in the Windows system tray and select **Current Status** from the pop-up menu. The Status window will open. Scroll to the device's script tag to view the values of the device's variables. If the device's variables have value Uninitialized, Cameleon is unable to connect to the device driver, probably because the driver is not running. You cannot use the value Uninitialized in scripts or expressions.

Table 4.2 Host Interface read properties

TYPE OF PROPERTY	PROPERTY	VALUE	INDICATES
Read	CommStatus	Offline	The Host Interface driver cannot communicate with the third party application.
		Online	The Host Interface driver can open a port to listen to the third party application.
	Preset	NoPreset	The current view of the camera controlled by the Host Interface driver is not a preset view.
		<preset 1>	The current view of the camera controlled by the Host Interface driver results from going to this user-defined preset. This value is created when the camera's preset is created.
		<preset 2>	The current view of the camera controlled by the Host Interface driver results from going to this user-defined preset. This value is created when the camera's preset is created.
		...	...
		<preset *i*>	The current view of the camera controlled by the Host Interface driver results from going to this user-defined preset. This value is created when the camera's preset is created.
	Pattern	NoPattern	The camera controlled by the Host Interface driver is not currently running a pattern.
		<pattern 1>	The camera controlled by the Host Interface is currently running this user-defined pattern. This value is created when the camera's pattern is created.
		<pattern 2>	The camera controlled by the Host Interface is currently running this user-defined pattern. This value is created when the camera's pattern is created.
		...	...

Table 4.2 Host Interface read properties

TYPE OF PROPERTY	PROPERTY	VALUE	INDICATES
		<pattern *i*>	The camera controlled by the Host Interface is currently running this user-defined pattern. This value is the script tag that is defined when the camera's pattern is created.
	Username		Displays the username the third party application is using to control devices via the Host Interface driver.
	Password		Displays the password associated with the username the third party application is using to control devices via the Host Interface driver.
	Camera		Displays the camera number of the camera the Host Interface driver is currently issuing commands to.
	Monitor		Displays the monitor number of the monitor the Host Interface driver is currently issuing commands to.

BAIFA Express Lane Network Toll Collection System

Appendix 1, Reference 8

System Interface Documentation

C: California Freeway Performance Measurement System (PeMS) – XML Interface

PLACEHOLDER: This document will be included in the FINAL RFP.
General information is available at **http://pems.dot.ca.gov**.

BAIFA Express Lane Network Toll Collection System

Appendix 1, Reference 8

System Interface Documentation

D: 511 – Open511 Protocol

NOTE: The Toll Program protocol appears in draft form and is subject to change.

511 Data Exchange including an Open511 Protocol

Prepared on

May 31, 2013

Version 0.9

METROPOLITAN
TRANSPORTATION
COMMISSION

Table of Contents

1.0 511 Data Exchange including an Open511 Protocol 2
1.1 Why 511 Data Exchange? 2
1.2 Contributions 2
1.3 What is 511 Data Exchange 2
1.4 Locating 511 Data Resources 3
2.0 511 Data Exchange Communication Protocol 4
2.1 Data Exchange Standards, Delivery Mechanism, and Encoding 4
2.2 Open511 6
2.3 511 Data Types/Resources 6
2.4 API: Discovery 7
2.5 API: Jurisdiction 9
3.0 Transit APIs 12
3.1 API: Operator 15
3.2 API: Line 17
3.3 API: Stop 19
3.4 API: Stop Place 22
3.4 API: Pattern 25
3.5 API: Timetable 28
3.6 API: Holidays 32
3.7 API: Announcement 35
3.8 API: Transit Scheduled Departures for a Stop 37
3.9 API: Real-time predictions at a Stop 41
3.10 API: Real-time Vehicle Monitoring 47
3.11 API: Transit Schedule Updates for an agency 53
3.12 API: Transit Addition and Cancellation of Trips by Agency 55
3.13 API: General Announcements 57
4.0Traffic API 60
4.1 API: Event 60
4.2 API: Announcement 70
4.3 API: Area 74
4.4 API: Road 76
4.5 API: Node 77
4.6 API: Segments 80
4.7 API: Segment_Conditons 83
4.8 API: Reports 85

5.0 Toll API 90
5.1 API: Toll_Program 90
5.2 Static_Toll 92
5.3 Dynamic_Toll 94
6.0 Parking API 97
6.1 API: Parking Facility structure 97
7.0 Recommendations 101
7.1 Data Encoding 101
7.2 Bandwidth Optimization and Conditional Requests 101
7.3 Authentication and Encryption Mechanisms 101
7.4 Response Language 102
7.5 Aggregation and Multi-jurisdiction Endpoint 102
7.6 Access Mechanism 102
7.7 API Rate and Throttling 102
7.8 Version control 103
7.9 Next steps? 103
Appendix A: API Response Messages- XML 104
Section 1 – General XML 104
Section 2 – Transit XML 106
Section 3 – Traffic XML 126
Section 4 – Toll XML 131
Section 5 – Parking XML 136
Appendix B: API Response Messages- JSON 137
Section 1: General JSON 137
Section 2– Transit JSON 139
Section 3 – Traffic JSON 151
Section 4 – Toll JSON 156
Section 5 – Parking JSON 159

List of Tables

A.1.1 Example Discovery Response (XML) 104
A.1.2 Example Jurisdiction Response (XML) 104
A.2.1 Example Transit Operator Response (XML.) 106
A.2.2 Example Transit Line Response (XML.) 107
A.2.3 Example Transit Stop Response (XML.) 109
A.2.4 Example Transit Stop Place Response (XML.) 110
A.2.5 Example Transit Pattern Response (XML.) 112
A.2.6 Example Transit Route Response (XML.) 114
A.2.7 Example Transit Day Type Response (XML.) 116
A.2.8 Example Transit Announcement Response (XML.) 117

A.2.9 Example Response (XML) in SIRI ST format 118
A.2.10 Example Transit Real Time Departure Service Delivery Mode Response (XML) in SIRI format 119
A.2.11 Example Real Time Vehicle Monitoring Response (XML) in SIRI format 121
A.2.12 Example Transit Schedule Update Response (XML) in SIRI PT format 123
A.2.13 Example Transit Addition and Cancellation of Trip Response (XML) in SIRI ET format 124
A.2.14 Example Transit General Messaging Service Response (XML) in SIRI GM format 125
A.3.1 Example Traffic Event Structure Response (XML) 126
A.3.2 Example Traffic Announcment Response (XML) 127
A.3.3 Example Traffic Area Response (XML) 127
A.3.4 Traffic Road Example Response (XML) 128
A.3.5 Example Traffic Node Response (XML) 128
A.3.6 Example Traffic Segment Response (XML) 129
A.3.7 Example Traffic Segment Conditions Response (XML) 130
A.4.1 Example Toll Program Response (XML) 131
A.4.2 Example Static Toll Response (XML) 132
A.4.3 Example Dynamic Toll Response (XML) 133
A.5.1 Example Parking Facility Structure Response (XML) 136
B.1.1 Example Discovery Response (JSON) 137
B.1.2 Example Jurisdiction Response (JSON) 137
B.2.1 Example Transit Operator Response (JSON.) 139
B.2.2 Example Transit Line Response (JSON.) 141
B.2.3 Example Transit Stop Response (JSON.) 141
B.2.4 Example Transit Stop Response (JSON.) 141
B.2.5 Example Transit Pattern Response (JSON.) 143
B.2.6 Example Transit Route Response (JSON.) 143
B.2.7 Example Transit Day Type Response (JSON.) 143
B.2.8 Example Transit Announcement Response (JSON.) 143
B.2.9 Example Response (JSON) in SIRI ST format 143
B.2.10 Example Transit Real Time Departure Service Delivery Mode Response (JSON) in SIRI format 145
B.2.11 Example Real Time Vehicle Monitoring Response (JSON) in SIRI format 146
B.2.12 Example Transit Schedule Update Response (JSON) in SIRI PT format 148
B.2.13 Example Transit Addition and Cancellation of Trip Response (JSON) in SIRI ET format 149
B.2.14 Example Transit General Messaging Service Response (JSON) in SIRI GM format 150
B.3.1 Example Traffic Event Structure Response (JSON) 151
B.3.2 Example Traffic Announcment Response (JSON) 152
B.3.3 Example Traffic Area Response (JSON) 152
B.3.4 Traffic Road Example Response (JSON) 153
B.3.5 Example Traffic Node Response (JSON) 154
B.3.6 Example Traffic Segment Response (JSON) 154
B.3.7 Example Traffic Segment Conditions Response (JSON) 155
B.4.1 Example Toll Program Response (JSON) 156
B.4.2 Example Static Toll Response (JSON) 156
B.4.3 Example Dynamic Toll Response (JSON) 157
B.5.1Example Parking Facility Response (JSON) 159

Document History

Description	Version	Date
Working Draft - addressed reorganization comments	0.9	08/28/13

1.0 511 Data Exchange including an Open511 Protocol

Prepared by Metropolitan Transportation Commission (MTC) and OpenNorth
Version 0.9

1.1 Why 511 Data Exchange?

511 systems across North America collect and disseminate traveller information to public for free. Government entities – state and/or regional, provide traffic, transit, bicycling, and ridesharing information, depending on the agency's mandate and jurisdiction, for their respective population through abbreviated phone number, 5-1-1, as well as web and other dissemination channels such as mobile web, smartphone apps, SMS, and large-screen display devices and kiosks. 511 systems host a wealth of traveller information that can be a valuable resource for innovative application development by external parties if the data can be exposed through a data exchange standard. In addition to sharing data with developers, adoption of standard based data exchange would also help share data between a 511 system and other data sources, a transit agency for example, as well as neighbouring 511 systems, facilitating traveller information across neighbouring 511 jurisdictions.

Each 511 system has developed its own mechanism to collect data and disseminate information. An open standard for disseminating data would help 511 data consumers easily access data and develop their products based on a set of known interfaces. Open511 is a newly designed open standard that defines a set of data interfaces in order to facilitate access to 511 data. These interfaces are intended to benefit both internal and external traveller application development.

1.2 Contributions

MTC and OpenNorth developed the initial draft. A group of stakeholders from government and private sectors have contributed by providing valuable feedback.

1.3 What is 511 Data Exchange

Each 511 system collects data and disseminates information relevant for travellers within a given geographical jurisdiction. The 511 data exchange adopts a set of existing standards for data collection and dissemination. In addition, it also includes a new and open data dissemination interface standard, named Open511, targeted for data consumers delivering end-user applications. The following principles guided the development of 511 data exchange and the Open511:

- At one end 511 systems collect data from various public (DOTs, cities/counties, transit agencies, etc.) and private (traffic data vendors) sector data generators. At the other end data is consumed by internal and external applications. 511 data exchange specifications should facilitate both data collection and dissemination.
- A number of traffic and transit data exchange standards exist that are well known in the industry. SIRI, NetEx, GTFS, and GTFS-Realtime for transit data and TMDD for traffic data to name a few. 511 data exchange specifications should adopt a suitable existing standard where it serves the data exchange use case.
- Because some of the existing standards do not lend themselves to wider use due to inherent technical complexity and additional data processing, a simple and open data interface should be supported to facilitate direct consumption.
- 511 data exchange specification should describe:
    - Communication protocol that establishes the relationship between two interacting systems including security, authentication, data access privileges, and system status awareness.

- Protocol for locating 511 data resources; metadata services for geographical coverage (cities, counties), travel modes (traffic, transit, bicycling, ridesharing, walking, parking, air), and transportation infrastructure/service types (highway/arterial/local for traffic, bus/ferry/train for transit).
- Data seeking and delivery mechanisms.
- Data structure and encoding formats.

1.4 Locating 511 Data Resources

511 data disseminators may decide the best way to disseminate data. An online registry for 511 data resources similar to GTFS Data Exchange (http://www.gtfs-data-exchange.com) would make 511 data more widely available. The online registry may provide a service that can be queried to get a list of 511 data resources and saved locally in order to support resource discovery. 511 data disseminators should also provide online information about their own open 511 data resources.

2.0 511 Data Exchange Communication Protocol

Communication for 511 data exchange incorporates a simple, easy to implement protocol.

- Each data disseminator would decide whether or not a license agreement and/or Terms of Use are required by an external consumer to access data. To make data easily accessible such requirements should be as simple and streamlined as possible to encourage wider use of the data.
- If a registration is required to access data, that process may be used to identify data needs by the registrant. Successful registration should result in an authentication key/ID provided to the consumer.
- Data requests should be accompanied with a valid key/ID and would be limited to the granted privileges.
- Communication protocol for 511 data collection would be set up as per the mutual agreement between the two organizations.

2.1 Data Exchange Standards, Delivery Mechanism, and Encoding

511 data exchange involves both bulk and ad-hoc query-based data sharing. Bulk data exchange would provide large volume of data, useful for C2C (center to center) communication whereas app developers are mostly interested in ad-hoc, on demand data in small quantities. Data users who prefer to store data in their own system would choose bulk data transfer. On the other hand, a mobile application would need small data packaged in readily consumable format that the query based APIs would fulfil.

511 data exchange specifications take advantage of existing standards, when feasible. Existing standards that are adopted as 511 standards are travel mode specific. Table 1 and 2 below lists adopted 511 data exchange standards, delivery mechanism, and encoding for transit and traffic data. Travel modes other than traffic and transit will be added to this standard later.

In discussions with agencies and developer community it became obvious that one particular data and communication standard cannot fulfil everyone's needs. For example, agencies producing traffic data prefer standards such as TMDD for data sharing with other agencies. On the other hand, a developer working on a mobile application for traffic information is more comfortable using widespread REST based APIs and JSON/XML data format. Though standards like TMDD are very comprehensive and accepted in the ITS industry they are often seen as heavy and cumbersome by the developer community who are primarily interested in building a simple web/mobile based application. Therefore it was necessary to address this diversity in use cases by adopting TMDD for low level (bulk) data communication between agencies and local data storage and a REST based Open511 API providing data in both XML and JSON formats. 511 data to be collected and disseminated can be broadly classified into two categories:

Configuration data – This category includes data that are mostly static and defines the configuration of a transportation system. For example, routes and stops for a transit service or the roadway network for traffic do not change frequently. Configuration data is very important because of the fact that real-time information is based on the underlying configuration data. Most users would like to download, save, and perform additional processing on configuration data before it is consumed in smaller pieces by their system and applications. Because configuration dataset for a transportation system/service would be large in size, it is efficient to provide this bulk data through transport mechanism such as FTP or data downloadable over HTTP. When it is necessary to provide configuration data in small quantities it will be done through Open511 APIs over Request/Response based communication.

Real-time data – Data in this category generally has a short life time and changes frequently. Data such as transit departure predictions, and incidents and travel times on roadway fall in this category. Whether used in bulk or through Open511 APIs real-time data should be provided through a Request/Response based communication.

Based on the amount of data exchanged and data encoding used, 511 data exchanges have been organized into two groups, bulk and query-based. Bulk data exchanges provide data using data encoding adopted by the given standard. Encoding for query-based Open511 APIs can be JSON, XML, or both. Depending on the use case some users may choose bulk exchanges using standards such as TMDD for both configuration and real-time, some may use only Open511 APIs for configuration and real-time data, others may first download configuration data to provide the foundation for their application and make ad-hoc queries through the Open511 API calls for real-time data. Table 1 and 2 below provide a summary of bulk and query-based data exchanges, respectively.

Table 1 511 Adopted Standards, Delivery Mechanisms and Encodings for Bulk Exchanges

Travel Mode	Data Type	Data Definition[1]	Pattern of Interaction[2]	Data Transport	Data Encoding
Transit	Service configuration and scheduled timetable	Extended GTFS, NetEx Custom[3]	Downloaded	HTTP	Extended GTFS: CSV NetEx: XML
	Predicted and unplanned changes data	SIRI, GTFS-RT	Request/Response	HTTP	SIRI: XML GTFS-RT: Proto Buffer
Traffic	Network configuration data	TMDD V3.0	Request/Response	HTTP	XML
	Real-time speed and incidents data	TMDD V3.0	Request/Response	HTTP	XML

Table 2 Open 511 Data Standards, Delivery Mechanism and Encoding for Query-Based Exchanges

Travel Mode	Data Type	Data Definition[1]	Pattern of Interaction	Data Transport	Data Encoding
Transit	Service configuration and scheduled timetable	NetEx	Request/Response	HTTP	JSON/XML

	Predicted and unplanned changes	SIRI	Request/Response	HTTP	JSON/XML
Traffic	Real-time speed and incidents	Open511	Request/Response	HTTP	JSON/XML
Parking	Service configuration and scheduled timetable	Custom	Request/Response	HTTP	JSON/XML
	Real-time parking information	Custom	Request/Response	HTTP	JSON/XML

1. GTFS – General Transit Feed Specification, GTFS-RT – GTFS for real-time transit information, SIRI – Service Interface for Real-Time Information, TMDD – Traffic Management Data Dictionary
2. All bulk data exchange interactions will be preceded by new data availability notification to registered data subscribers.
3. Each 511 system may adopt a suitable custom data exchange standard that facilitates transit data collection from transit agencies within its jurisdiction. For example, San Francisco Bay Area 511 has developed an XML schema that is used by Bay Area transit agencies to provide data to the 511 system.

2.2 Open511

Open511 aims to create simple, uniform and resource driven APIs that can be easily used by consumers to retrieve data from 511 systems. In order for these APIs to be easily discovered Open511 API providers must create a single entry discovery point that provides links and details for all other resources available from a disseminator/Jurisdiction. Open511 defines a discovery resource that will help consumers locate APIs and other resources. Also to provide a context of the data and other details Open511 recommends defining a Jurisdiction resource which will provide details on the jurisdiction and its coverage.

2.3 511 Data Types/Resources

As shown in the tables above, 511 data exchange adopts existing data standards such as GTFS and TMDD, where those standards are most suitable. Information on adopted standards can be obtained from online resources as listed below.

Transit (service configuration and schedules)

GTFS: https://developers.google.com/transit/gtfs/

NetEx: http://www.kizoom.com/standards/netex/

Transit (real-time information)

GTFS-RT: https://developers.google.com/transit/gtfs-realtime/

SIRI: http://www.kizoom.com/standards/siri/

Traffic

TMDD: http://www.ite.org/standards/tmdd/

In addition, data types for Open511 APIs are defined below.

2.4 API: Discovery

Open511recommends providing an API discovery mechanism/endpoint similar to open311. Jurisdictions/disseminators will setup a service directory that will list all available versions of Open511 APIs that are currently supported by a jurisdiction and their endpoints. The service directory will also provide information on endpoints for multiple versions and stability levels such as production and test.
A specific resource can therefore have more than one endpoint depending upon the different versions and levels of API stability that are being provided. A jurisdiction can choose to provide just the production endpoint for any particular version, though Open511 recommends providing both a test as well as production endpoint.

The structure of an API discovery document consists of

Field	Type	Mandatory / Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
jurisdictions	Collection of **jurisdiction**	*Mandatory*	List all the jurisdictions that are supported by the endpoint. Most of the time, there will be only one occurrence, except for multiple jurisdictions endpoints and aggregators.
-id	String	*Mandatory*	ID of the jurisdiction.
— name	Free text	*Mandatory*	Name of the jurisdiction. The field is simply description and should not be considered as a key.
— URL	Link	*Mandatory*	Link to the jurisdiction resource. In XML, the rel attribute needs the value self. An aggregator should always point the original jurisdiction resource and not copy it locally.
services	Collection of **Service**	*Mandatory*	List all the services supported by the endpoint.
— service_type	Enum	*Mandatory*	The service type value comes from a predefined list of services that are supported by the Open511 API. Value list: - EVENTS - AREAS
-url	Link	*Mandatory*	Link pointing to the resource. In XML, the `rel` attribute needs the value `self`. Even if the current endpoint supports multiple jurisdictions or is an aggregator, there is only one service resource that aggregates the data for all the jurisdictions.
— supported_versions	Collection of versions currently supported	*Optional*	List all the versions supported by the current server.
— - supported_version	Enum	*Optional*	Version identifier. As the specification evolves, version identifiers will be added. For the moment, the only version supported is v0 and is not an official value.

Sample request endpoint for discovery

Request Type	**GET**
Request Endpoint Example	http://api.511.org/Open511

Parameters and Filters

The discovery resource does not support any URL parameter or filter besides the format selection and the language negotiation. The discovery response for XML is shown in Appendix A Section A.1.1. The discovery response for JSON is shown in Appendix B Section B.1.1.

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.

Possible Errors

The numbers represent the HTTP status code returned for each error type:

- 500 - Internal Server Error (System has issues processing your request)

2.5 API: Jurisdiction

The jurisdiction represents any government entity (city, state/province, agency, etc.) that publishes Open511 data. It is a functional concept used to provide metadata such as contact information, etc.

The structure of a Jurisdiction resource consists of

Field	Type	Mandatory / Optional	Description
id	String	*Mandatory*	Unique identifier for the jurisdiction. It's expected to be unique across all Open511 implementations. It must take the form of a hostname within a domain owned by the government entity. The hostname doesn't necessarily need to resolve in DNS; for example, if Roadsville owns roadsville.gov, it's fine to create agency1.roadsville.gov and agency2.roadsville.gov jurisdictions, even if those don't resolve to IP addresses.
name	Free text	*Mandatory*	Name of the jurisdiction.
email	Free text	*Mandatory*	Valid email that can be used to contact the department or the person in charge of the data provided by the Open511 API.
description	Link	*Optional*	URL Link pointing to a human readable resource (web page, pdf file, etc) providing details about the Open511 service and the jurisdiction covered.
Phone	Free text	*Optional*	Phone number to contact the department or person in charge of the data provided by the Open511 API.
description	Free text	*Optional*	Free text that can be used by a client application to provide some information about the Open511 service and the jurisdiction covered.
geography	Link	*Mandatory*	The geography field links an open511 jurisdiction geography resource providing the jurisdiction boundaries.
timezone	timezone	*Optional*	The timezone of the jurisdiction; will be used as the default for events belonging to this jurisdiction. Should always be provided, except in cases where a jurisdiction spans multiple timezones.
license	Link	*Mandatory*	Link to a resource containing the license covering the data provided in the API.
Languages	Collection of **language**	*Optional*	List of languages supported by this endpoint. A language element should be provided for each language supported.
--language	Enum	*Mandatory*	Multiple occurrences supported. The values supported are the same as the language negotiation feature.

Sample request endpoint for jurisdiction

Request Type	GET

Request Endpoint Example	For e.g. http://api.511.org/Open511/sfbayArea

Parameters and Filters

The jurisdiction resource does not support any URL parameter or filter besides the format selection and the language negotiation. The jurisdiction response for XML is shown in Appendix A Section A.1.2. The discovery response for JSON is shown in Appendix B Section B.1.2.

Parameter	Mandatory / Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.

Possible Errors

The numbers represent the HTTP status code returned for each error type:

- 500 - Internal Server Error (System has issues processing your request)

2.6 API: Jurisdiction Geography

The jurisdiction geography represents the boundaries of the jurisdiction.

The structure of a Jurisdiction resource consists of

Field	Type	Mandatory / Optional	Description
geography	GeoSpatial	*Mandatory*	Boundaries of the jurisdiction; a Polygon or MultiPolygon.

Sample request endpoint for jurisdiction

Request Type	GET
Request Endpoint Example	For e.g. http://api.511.org/Open511/sfbayArea/geography

Parameters and Filters

The jurisdiction geography resource does not support any URL parameter or filter besides the format selection. The jurisdiction response for XML is shown in Appendix A Section A.1.2. The discovery response for JSON is shown in Appendix B Section B.1.2.

Parameter	Mandatory / Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.

Possible Errors

The numbers represent the HTTP status code returned for each error type:

- 500 - Internal Server Error (System has issues processing your request)

3.0 Transit APIs

The NeTEx data structures wrapped within the SIRI framework has been adopted for dynamic exchange of Transit service configuration and schedules. Open511 however recommends using HTTP Get method for requests instead of using HTTP Post, as specified by the NeTEx/SIRI standards.

The data communication architecture for San Francisco Bay Area 511 is depicted in Figure 1 below.

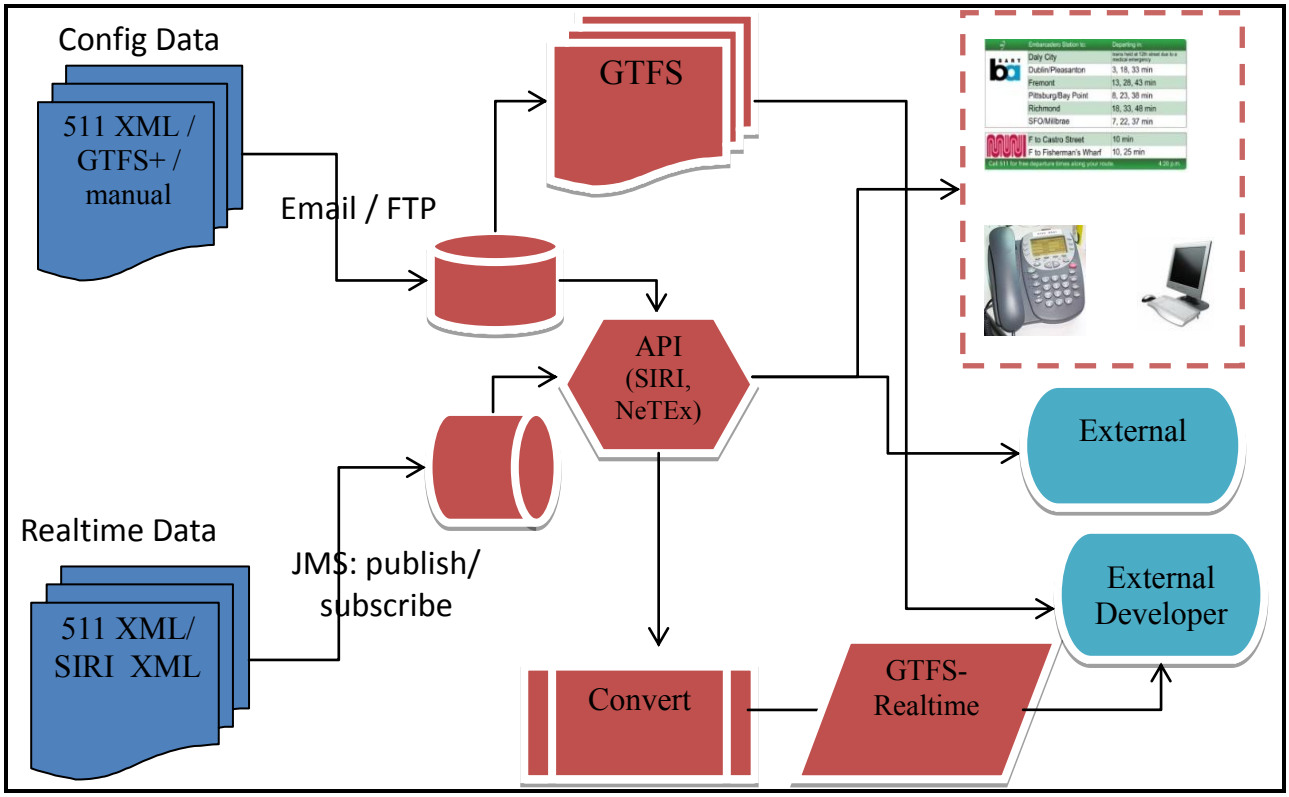


Figure 1 – Data communication architecture for San Francisco Bay Area 511

All NeTEx responses shall be enclosed within the SIRI ServiceDelivery structure as shown below.

Field	Type	Mandatory/ Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Time response was created
DataObjectDelivery	DataObjects Delivery structure	*Mandatory*	Delivery for NeTEx service containing one or more NeTEx data objects
— ResponseTimestamp	DateTime	*Mandatory*	Time individual response element was created
— dataObjects	Collection of NeTEx dataobjects	*Mandatory*	NeTEx Entities of any type

3.1 API: Operator

Operator within a jurisdiction represents a company providing public transport services. Consumers can request a list of all the operators within the jurisdiction or they can use additional filters such as operator code/id to restrict the results as per their needs and use case.

Below is a message structure of dataObjects for Organisations contained within a NeTEx ResourceFrame. Organisations are a collection of the Operator resource.

Field	Type	Mandatory/ Optional	Description
ResourceFrame	NeTEx frame	*Mandatory*	NeTEx container frame for Organizations.
—organisations	Collection of Operators	*Mandatory*	A collection of Operator elements. Can contain multiple operator elements, at least one occurrence is mandatory.

Operator structure

The operator structure is the main element of the organizations collection. It represents a company providing public transport services.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the operator
Name	Free Text	*Optional*	Name of the operator.
ShortName	Free Text	*Optional*	Short name for the operator.
SiriOperatorRef	Free Text	*Optional*	An alternative code that uniquely identifies the operator in real-time systems (AVMS)
Locale	Container	*Optional*	Container for the operator's locale information
TimeZone	Free Text	*Optional*	Timezone Name
—DefaultLanguage	Xsd:Language	*Optional*	Default Language
ContactDetails	Container	*Optional*	Container for operator's contact information
—ContactTelephoneNumber	Free Text	*Optional*	Contact telephone number
—WebSite	Xsd:AnyURI	*Optional*	Website address
PrimaryMode	Enum	*Optional*	Primary transport mode of operator
PrivateCode	Free Text	*Optional*	Agency/operator code used within the jurisdiction
Extensions	Container	*Optional*	Container for extensions to NeTEx
—Monitored	Boolean	*Optional*	Whether agency is real-time enabled or not
—OtherModes	Enum list	*Optional*	List of transport modes other than primary mode.
—Coverage	Container	*Optional*	Coverage area of the operator – can be a polygon or a list of lines
——gml:Polygon	GML structure	*Optional*	GML Polygon representing the coverage
——gml:LineString	GML structure	*Optional*	GML Line representing the coverage. Multiple lines can be provided

Sample request endpoint for operators

Request Type	GET
Request Endpoimt Example	For e.g. http://api.511.org/Open511/operators

Parameters and Filters

Parameters and filters supported with the request are shown in the table below. The transit operator response for XML is shown in Appendix A Section A.2.1. The transit operator response for JSON is shown in Appendix B Section B.2.1.

Parameter	Mandatory/ Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept_language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
Operator_id	*Optional*	The operator_id parameter supports filtering based on a particular operator id/code
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual Operator resource cannot be located). For e.g. http://ap.511.org/Open511/Operators/1345

3.2 API: Line

Lines are routes covered by transit operators within the jurisdiction. Consumers can request list of all the routes within the jurisdiction or they can use additional filters such as operator code, route id to restrict the results as per their needs and use case.

Below is a message structure of dataObjects for lines contained within a NeTEx ServiceFrame. Lines are a collection of the Line (Route) resource.

Field	Type	Mandatory/ Optional	Description
ServiceFrame	NeTEx frame	*Mandatory*	NeTEx container frame for Lines.

—lines	Collection of Lines	*Mandatory*	A collection of Line elements. Can contain multiple line elements, at least one occurrence is mandatory.

Line structure

The line structure is the main element of the Lines collection. It represents a route generally known to the public by a name or a number.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the route.
Name	Free Text	*Mandatory*	Name of the line.
TransportMode	Enum	*Optional*	Mode of transport of line
PublicCode	Free Text	*Optional*	Public identifier of the line
SiriLineRef	Free Text	*Optional*	An alternative code that uniquely identifies the operator in real-time systems (AVMS)
Monitored	Boolean	*Optional*	Indicates if real-time data available for line.
OperatorRef	ID	*Mandatory*	Reference to the operator for the line

Sample request endpoint for lines

Request Type	GET
Request Endpoimt Example	For e.g. http://api.511.org/Open511/lines

Parameters and Filters

Parameters and Filters supported with the request are shown in the table below. The transit line response for XML is shown in Appendix A Section A.2.2. The transit operator response for JSON is shown in Appendix B Section B.2.2.

Parameter	Mandatory/ Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept_language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
Operator_id	*Optional*	The operator_id parameter supports filtering based on a particular operator id/code
Line_id	*Optional*	The line_id parameter supports filtering based on a particular line
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual line resource cannot be located). For e.g. http://api.511.org/Open511/lines/1345

3.3 API: Stop

Stop or ScheduledStopPoint is a location where passengers can board or alight from vehicles. Consumers can request list of all the stops serviced by an agency/operator within the jurisdiction. Stop groupings or StopAreas are also returned when specifically requested using the include_stop_areas parameter.

Below is a message structure of dataObjects for stops contained within a NeTEx ServiceFrame. ScheduledStopPoints are a collection of the ScheduledStopPoint (Stop) resource.

Field	Type	Mandatory/ Optional	Description
ServiceFrame	NeTEx frame	*Mandatory*	NeTEx container frame for ScheduledStopPoints.

— scheduledStopPoints	Collection of ScheduledStop Points	*Mandatory*	A collection of ScheduledStopPoint elements. Can contain multiple ScheduledStopPoint elements, at least one occurrence is mandatory.
— stopAreas	Collection of Stop Areas	*Optional*	A collection of StopArea elements. Stop Areas group stops within an operator or across operators. A hierarchy of stop groups could also be provided. The stopAreas are returned only when specifically requested using the include_stop_areas parameter.

Scheduled Stop Point structure

The ScheduledStopPoint structure is the main element of the ScheduledStopPoints collection. It represents a location where passengers can board or alight from vehicles.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the stop.
Name	Free Text	*Optional*	Name of the stop.
Location	Container	*Optional*	Location of stop
—Longitude	Float	*Optional*	Longitude of stop using WGS84 projection
—Latitude	Float	*Optional*	Latitude of stop using WGS84 projection
StopType	Enum	*Optional*	Indicates type of stop (Bus, Train, Ferry, etc)

Stop Area structure

The StopArea structure is the main element of the stopAreas collection. It represents a grouping of stops within or across multiple operators.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the stop area.
Name	Free Text	*Optional*	Name of the stop group.
Members	Container	*Optional*	Container of stops that belong to the group
— ScheduledStopPointRef	Reference ID	*Optional*	ID of the ScheduledStopPoint (within the 'ref' attribute)
ParentStopAreaRef	Reference ID	*Optional*	Used to build a hierarchy of stop areas. For example, MUNI stops at Embarcadero could be a StopArea 1, Ferry stops at Embarcadero could be

			StopArea 2. Stop Area 3 could be a parent stop area which comprises of all regional transit stops at Embarcadero. Stop Area 3 is then the ParentStopArea for StopArea 1 and 2.

Sample request endpoint for stops

Request Type	GET
Request Endpoimt Example	For e.g. http://api.511.org/Open511/stops

Parameters and Filters

Parameters and Filters supported with the request are shown in the table below. The transit stop response for XML is shown in Appendix A Section A.2.3. The transit stop response for JSON is shown in Appendix B Section B.2.3.

Parameter	Mandatory/ Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept_language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
Operator_id	*Optional*	The operator_id parameter supports filtering based on a particular operator id/code
Line_id	*Optional*	The line_id parameter supports filtering based on a particular line id (route id)
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.
include_stop_areas	*Optional*	When this parameter is set to true, all stop areas (stop groupings) along with the referenced stops (ScheduledStopPoints) are returned. If this parameter is set to true, the Operator_id and Line_id parameters should not be provided as stop groups may span across lines and operators.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual stop resource cannot be located). For e.g. http://api.511.org/Open511/stops/1345

3.4 API: Stop Place

StopPlace is a named place or the physical stop where public transport may be accessed. Consumers can request list of all the stop places by operator code or they can use additional filters such as stop id to restrict the results as per their needs and use case. For a given stop, the physical representation of the stop (StopPlace) and the representation of the stop as a point in the timetable (ScheduledStopPoint) will use the same stop identifier (id).

Below is a message structure of dataObjects for lines contained within a NeTEx ServiceFrame. StopPlaces are a collection of the StopPlace resource.

Field	Type	Mandatory / Optional	Description
SiteFrame	NeTEx frame	*Mandatory*	NeTEx container frame for StopPlaces.
—stopPlaces	Collection of StopPlacecs	*Mandatory*	A collection of stopPlace elements. Can contain multiple stopPlace elements, at least one occurrence is mandatory.

StopPlace structure

The StopPlace structure is the main element of the stopPlaces collection. It represents a physical stop where public transport may be accessed.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the SStopPlace.
Name	Free Text	*Optional*	Name of the StopPlace.
Description	Free Text	*Optional*	Description of StopPlace
Centroid	Container	*Optional*	Center coordinate of the stopPlace
Location	Container	*Optional*	The position of the Point that represents the center of the stopPlace
—Longitude	Float	*Optional*	Longitude of stopPlace using WGS84 projection
—Latitude	Float	*Optional*	Latitude of stopPlace using WGS84 projection
AccessibilityAssessment	Container	*Optional*	The accessibility characteristics of the stopPlace
—MobilityImpairedAccess	Enum	*Mandatory*	Summary indication as to whether the stopPlace is considered accessible or not
—limitations	Container	*Optional*	Accessibility limitations
——AccessibilityLimitation	Container	*Mandatory*	Assessment of the accessibility of the stopPlace
———WheelChairAccess	Enum	*Mandatory*	Whether the stopPlace is wheelchair accessible
alternativeNames	Container	*Optional*	Container for alternative names
—AlternativeName	Container	*Mandatory*	Container for Alternative name
——Name	Free Text	*Mandatory*	Alternative Name
PostalAddress	Container	*Optional*	Postal address of the stopPlace

—AddressLine1	Free Text	*Optional*	First line of address
—Town	Free Text	*Optional*	Town
Url	URI	*Optional*	Web address of stopPLace
OperatorRef	Reference ID	*Optional*	Reference to the operator of the stopPlace (contained in ref attribute of OperatorRef element)
adjacentSites	Container	*Optional*	Reference to adjacent sites such as parking locations
—ParkingRef	Reference ID	*Mandatory*	Reference to parking associated with the stopPlace (contained in ref attribute of ParkingRef element) Multiple ParkingRef elements can be included to associate multiple parking locations to the stopPlace
placeEquipments	Container	*Optional*	Equipments that may be located in the stopPlace
—SanitaryEquipment	Container	*Optional*	Container for a sanitary facility such as a restroom, shower, etc.
——Description	Free Text	*Optional*	Description of the facility
—CycleStorageEquipment	Container	*Optional*	Container for cycle storage equipments
——Description	Free Text	*Optional*	Description of the facility
——CycleStorageType	Enum	*Optional*	Type of storage (e.g. Racks)
——NumberOfSpaces	Integer	*Optional*	Number of storage spaces
—SignEquipment	Container	*Optional*	Sign visible to passengers such as information boards
——Description	Free Text	*Optional*	Description of the sign
—EscalatorEquipment	Container	*Optional*	Escalators in the stopPlace
——Description	Free Text	*Optional*	Description of the escalator
—LiftEquipment	Container	*Optional*	Elevators (Lifts) in the stopPlace
——Description	Free Text	*Optional*	Description of the elevator
—ShelterEquipment	Container	*Optional*	Shelter equipment such as waiting areas
——Description	Free Text	*Optional*	Description of shelter
—SeatingEquipment	Container	*Optional*	Seating equipment such as benches
——Description	Free Text	*Optional*	Description of seating equipment
PublicCode	Free Text	*Optional*	Short public code for passengers to use when uniquely identifying the stop
TransportMode	Enum	*Optional*	Primary mode of transport associated with the stopPlace
StopPlaceType	Enum	*Optional*	Type of stopPlace (for e.g. Rail Station)
quays	Container	*Optional*	A collection of quays
—Quay	Container	*Mandatory*	A place such as platform where passengers have access to Public transport vehicles
——CompassOctant	Enum	*Optional*	Heading of quay relative to street (E/W/N/S/NE/NW/SE/SW)
parkings	Container	*Optional*	A collection of parking locations linked to the stopPlace
—Parking	Container	*Mandatory*	Single parking location
——Name	Free Text	*Optional*	Name of parking location

——Description	Free Text	*Optional*	Description
——Centroid	Container	*Optional*	Container for center location of Parking
———Location	Container	*Optional*	Center point of Parking
————Longitude	Float	*Optional*	Longitude of Parking using WGS84 projection
————Latitude	Float	*Optional*	Latitude of Parking using WGS84 projection
——PostalAddress	Container	*Optional*	Address of Parking
———AddressLine1	Free Text	*Optional*	Address Line 1
———Town	Free Text	*Optional*	Town
——ParkingType	Enum	*Optional*	Parking type (for e.g. Train station parking, Park and Ride)
——TotalCapacity	Integer	*Optional*	Total number of parking places
—— RealTimeOccupancyAvailable	Boolean	*Optional*	Whether real time occupancy data available for the parking location
——parkingAreas	Container	*Optional*	List of Parking areas (Accessible parking, Reserved parking)
———ParkingArea	Container	*Mandatory*	Parking Area
————Description	Free Text	*Optional*	Description of area
————ParkingProperties	Container	*Optional*	Properties of parking area
—————ParkingUserType	Enum	*Optional*	Type of Parking area (for Disabled, Reserved)
—————spaces	Container	*Optional*	Container for parking capacity
————— ParkingCapacity	Container	*Mandatory*	Container for parking capacity
————— NumberOfSpaces	Integer	*Optional*	Number of spaces
—————charges	Container	*Optional*	Parking charges for the parking area
——————tariffBands	Container	*Optional*	Charge bands for parking
————— ParkingTariffChargeBand	Container	*Mandatory*	An area within the parking area for grouping charges (Monthly parking, single day parking, etc.)
————— Description	Free Text	*Optional*	Description of parking charge band
————— MaximumStay	Xsd:Duration	*Optional*	Maximum allowed stay duration for tariff amount
———————Amount	Decimal	*Optional*	Charge for stay

Sample request endpoint for stops

Request Type	GET
Request Endpoint Example	For e.g. http://api.511.org/Open511/stopPlaces

Parameters and Filters supported with the request

Parameter	Mandatory / Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept_language	*Optional.*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
Operator_id	*Optional*	The operator_id parameter supports filtering based on a particular operator id/code
Stop_id	*Optional*	The stop_id parameter supports filtering based on a particular stop id
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The transit stop place response for XML is shown in Appendix A Section A.2.4. The transit stop place response for JSON is shown in Appendix B Section B.2.4.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual StopPlace resource cannot be identified)

3.4 API: Pattern

Pattern is an ordered list of stop points and time points for a Line, it describes a pattern followed by the public transport vehicle. A pattern may pass through the same stoppoint more than once. A Line may consist of more than one pattern.

Below is a message structure of dataObjects for Pattern contained within a NeTEx ServiceFrame.

Field	Type	Mandatory/ Optional	Description
ServiceFrame	NeTEx frame	*Mandatory*	NeTEx container frame for directions and journeyPatterns.
—directions	Collection of Direction	*Optional*	A collection of Direction elements referenced by the patterns within the journeyPatterns collection. Can contain multiple Direction elements, at least one occurrence is mandatory.
—journeyPatterns	Collection of ServiceJourney Pattern	*Mandatory*	A collection of ServiceJourneyPattern elements. Can contain multiple ServiceJourneyPattern elements, at least one occurrence is mandatory.

Direction structure

The Direction structure is the main element of the directions collection. It is a classification for the general orientation of a pattern within a Line.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the Direction.
Name	Free Text	*Optional*	Name of the Direction.

ServiceJourneyPattern structure

The ServiceJourneyPattern structure is the main element of the journeyPatterns collection. It is the journeyPattern for a (passenger carrying) Service.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the ServiceJourneyPattern.
Name	Free Text	*Optional*	Name of the ServiceJourneyPattern.
DirectionRef	ID	*Mandatory*	Reference to the direction
DestinationDisplayView	Container	*Optional*	Container for Pattern Headsign
— FrontText	Free Text	*Optional*	Pattern Headsign (Should contain Pattern Destination information only)
Extensions	Container	*Mandatory*	Container for extensions to NeTEx
— LineRef	Free Text	*Mandatory*	Reference to the Line resource.
pointsInSequence	Container	*Mandatory*	Contains sequence of points in Servicejourneypattern, points may be scheduledstop points or timingpoints.
— TimingPointInJourneyPattern	Container	*Mandatory*	A timing point within the Pattern
— TimingPointInJourneyPattern id (attribute)	Free Text	*Mandatory*	Unique identifier of TimingPointInJourneyPattern
— TimingPointInJourneyPattern order (attribute)	Positive Integer	*Mandatory*	Order of Point within PointsInSequence
——ScheduledStopPointRef ref (attribute)	Free Text	*Mandatory*	Identifier of Schedule Stoppoint corresponding to the timing point

——DestinationDisplayView	Container	*Optional*	If pattern headsign changes at a stop, specify the headsign here
———FrontText	Free Text	*Optional*	Headsign to display at the stop (Pattern Destination information only)
— StopPointInJourneyPattern	Container	*Mandatory*	A stop point within the Pattern
— StopPointInJourneyPattern id (attribute)	Free Text	*Mandatory*	Unique identifier of StopPointInJourneyPattern
— StopPointInJourneyPattern order (attribute)	Positive Integer	*Mandatory*	Order of Point within PointsInSequence
——ScheduledStopPointRef ref (attribute)	Free Text	*Mandatory*	Identifier of Schedule Stoppoint
——DestinationDisplayView	Container	*Optional*	If pattern headsign changes at a stop, specify the headsign here
———FrontText	Free Text	*Optional*	Headsign to display at the stop (Pattern Destination information only)
linksInSequence	Container	*Optional*	Sequence of links (The pattern could be represented as one single link or multiple links in sequence)
— ServiceLinkInJourneyPattern	Container	*Optional*	SeviceLine in a specified order
——projections	Container	*Optional*	Projections of the link
———LinkSequenceProjection	Container	*Optional*	Projection of the link sequence as an ordered series of points
——— gml:LineString	Line string	*Optional*	Series of points representing the link

Sample request endpoint for stops

Request Type	GET
Request Endpoint Example	For e.g. http://api.511.org/Open511/patterns

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept_language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
Pattern_id	*Optional*	The pattern_id parameter supports filtering based on a particular Pattern id

Line_id	*Optional*	The line_id parameter supports filtering based on a particular line id (All patterns for specified line_id will be returned)
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The transit pattern response for XML is shown in Appendix A Section A.2.5. The transit pattern response for JSON is shown in Appendix B Section B.2.5.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual Journey pattern resource cannot be identified)

3.5 API: Timetable

Timetable represents a timetable for a given Line, Direction and DayType. It also contains supporting elements referenced by the timetable such as the Route (ordered list of timepoints for which times are provided), day type (service type) and optionally day assignments (assignment of a daytype to each day within the service period)

Below is a message structure of data Objects for Timetable within a NeTEx CompositeFrame.

Field	Type	Mandatory / Optional	Description
CompositeFrame	NeTEx frame	*Mandatory*	NeTEx container version Frame that groups a set of content version frames to which same validity conditions have been assigned.
— ServiceFrame	NeTEx frame	*Mandatory*	NeTEx container frame for routes which is collection of Route. Route represents an ordered list of timepoint stops for which times are provided in the timetable. Multiple routes could be provided in cases where multiple timetables are returned. Each timetable would reference the appropriate route for the timetable.
— ServiceCalendarFrame	NeTEx frame	*Mandatory*	NeTEx container frame for collection of DayType and DayTypeAssignments. Should contain at least one DayType. DayTypeAssignments are returned only if requested specifically using the input parameter(flag) IncludeDayTypeAssignments
— TimetableFrame	NeTEx frame	*Mandatory*	NeTEx container frame for a timetable. Multiple TimetableFrames can be

			returned, one per timetable. The id attribute of the TimetableFrame should be unique across all timetables.

Service Calendar Frame Structure

To be completed

Route Structure

The Route structure is the main element of the routes collection. At least one Route is mandatory within the routes. Route represents an ordered list of timepoint stops for which times are provided in the timetable.

Field	Type	Mandatory / Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the Route.
Name	Free Text	*Optional*	Name for the Route.
LineRef	ID	*Mandatory*	Reference to the Line, ref attribute contains identifier of the line
DirectionRef	ID	*Mandatory*	Reference to the Direction, ref attribute contains identifier to the Direction
pointsInSequence	Container	*Mandatory*	Container for ordered set of time points making up the Route. It should contain at least 2 PointOnRoute
— PointOnRoute	Free Text	*Mandatory*	It is the reference to the ordered route points of Route, id attribute contains unique identifier for PointOnRoute
—— PointRef	ID	*Mandatory*	It is reference scheduled stop point representing the timepoint, ref attribute contains identifier to the point

DayType Structure

The dayTypes structure contains the collection of DayTypes referenced by the timetables. DayType is a type of day characterized by one or more properties which affect public transport operation.

Field	Type	Mandatory / Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the DayType.
Name	Free Text	*Mandatory*	Name of the DayType.
properties	Container	*Mandatory*	Container for the list of PropertyOfDay. Should contain at least one PropertyOfDay.

—PropertyOfDay	Container	*Mandatory*	A container for DaysOfWeek property.
——PropertyOfDayGroup	Enum	*Mandatory*	It contains DaysOfWeek logically ANDed together

DayType Assignment structure

The dayTypeAssignments structure contains the collection of DayTypeAssignments, which links every operating day within the service period to a daytype. The service period is defined within the Timetable Frame.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the DayTypeAssignment.
Description	Free Text	*Optional*	Description of the DayTypeAssignment
Date	Date	*Mandatory*	Operating Date (within the service period)
DayTypeRef	Reference	*Mandatory*	Reference to a DayType (within the ref attribute).

TimetableFrame structure

TimetableFrame is coherent set of timetable data which consist of vehicle Journeys and blocks to which the same validity condition has been assigned.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the TimetableFrame.
Name	Free Text	*Optional*	Name of the TimetableFrame.
frameValidityConditions	Container	*Mandatory*	Container for the AvailabilityCondition which applies to whole Timetable.
— AvailabilityCondition	Container	*Mandatory*	It is a container for available conditions.
——FromDate	DateTime	*Mandatory*	Start date of Timetable validity period.
——ToDate	DateTime	*Mandatory*	End date of Timetable validity period.
——dayTypes	Container	*Mandatory*	Container for DayType reference. It lists the daytypes referenced by the timetable.
———DayTypeRef	ID	*Mandatory*	It is a reference to DayType, ref attribute has reference value to a DayType
vehicleJourneys	Container	*Mandatory*	Container for collection of ServiceJourney (Trip).
—ServiceJourney	Free Text	*Mandatory*	ServiceJourney is a planned movement of public transport on a DayType. Id attribute has unique identifier for Service Journey

——SiriVehicleJourneyRef	Free Text	*Mandatory*	An alternative code that uniquely identifies the journey. Specifically for use in AVMS systems
——JourneyPatternView	Container	*Mandatory*	It is a container for simplified journey pattern view
——— ServiceJourneyPatternRef	ID	*Mandatory*	Reference to Service Pattern, ref attribute contains identifier for service journey Pattern
——— RouteRef	ID	*Mandatory*	Reference to Route, ref attribute contains identifier for Route
——— DirectionRef	ID	*Mandatory*	Reference to Direction, ref attribute contain identifier for Direction
——calls	Container	*Mandatory*	It is container for complete sequence of stops along the route path.
———call	Container	*Mandatory*	It is a visit to a scheduled stop point as part of a vehicle journey, order attribute contains sequence number within the journey
——— ScheduledStopPointRef	ID	*Mandatory*	Reference to scheduled stop point, ref attribute contains identifier for scheduled stop point
————Arrival	Container	*Mandatory*	Container for arrival time for call
————Time (Arrival)	Time	*Mandatory*	Arrival time for call
————Departure	Container	*Mandatory*	Container for departure time for call
————Time (Departure)	Time	*Mandatory*	Departure time for call

Sample request endpoint for timetable

Request Type	GET
Request Endpoint Example	For e.g. http://api.511.org/Open511/timetable

Parameters and Filters supported with the request

Parameter	Mandatory / Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.

accept_language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
Operator_id	*Mandatory*	The operator_id parameter supports filtering based on a particular operator id/code
Line_id	*Mandatory*	The line_id parameter supports filtering based on a particular line id. All timetables for the line are returned
IncludeDayTypeAssignments	*Optional*	DayTypeAssignments will be included only if this flag is set to true.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The transit route response for XML is shown in Appendix A Section A.2.6. The transit route response for JSON is shown in Appendix B Section B.2.6.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual Holiday resource cannot be identified)

3.6 API: Holidays

Holidays is a collection of holiday services provided by an agency or operator. Since a holiday service could reference a daytype, defintions of all daytypes supported by the operator are also returned.

Below is a message structure of dataObjects for Holiday contained within a NeTEx ServiceCalendarFrame.

Field	Type	Mandatory/ Optional	Description
ServiceCalendarFrame	NeTEx frame	*Mandatory*	NeTEx container frame for Holidays and dayTypes. This frame will hold information for a single service period (validity/signup period). If holidays across multiple signup periods are to be provided, one ServiceCalenderFrame per signup period has to be inluded.
— contentValidityConditions	Collection of AvailabilityCondition	*Mandatory*	A collection of AvailabilityCondition elements. Can contain multiple AvailabilityCondition elements, at least one occurrence is mandatory. It contains the service period and all holiday services within the service period.

— dayTypes	Collection of DayType	*Mandatory*	A collection of all the DayTypes supported by the transit operator. Can contain multiple DayType elements, at least one occurrence is mandatory.

AvailabilityCondition structure

The AvailabilityCondition structure is the main element of the contentValidityConditions collection. It represents either the service period or a holiday service.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	*Mandatory*	Unique identifier of the AvailabilityCondition.
Description	Free Text	*Optional*	Description of the AvailabilityCondition.
FromDate	DateTime	*Mandatory*	Start date of AvailabilityCondition(Service Period/Holiday)
ToDate	DateTime	*Mandatory*	End date of AvailabilityCondition(Service Period/Holiday)
IsAvailable	Boolean	*Mandatory*	Flag to determine if condition makes resource available or not available. Flag is true if Availability condition represents the service period. Flag is false if Availability condition represents a holiday service.
dayTypes	Container	*Optional*	Daytype of alternate service being operated on the holiday.
DayTypeRef	ID	*Mandatory*	ID of the Daytype. In XML, the ref attribute needs to hold the Daytype ID.

DayType structure

The dayTypes structure contains the collection of DayTypes supported by the transit operator.. DayType is a type of day characterized by one or more properties which affect public transport operation.

Field	Type	Mandatory/ Optional	Description
id (Attribute)	Free Text	Mandatory	Unique identifier of the DayType.
Name	Free Text	Optional	Name of the DayType.
properties	Container	Mandatory	Container for the list of PropertyOfDay. Should contain at least one PropertyOfDay.
—PropertyOfDay	Container	Mandatory	A container for DaysOfWeek property
——PropertyOfDayGroup (DaysOfWeek)	Enum	Mandatory	It contains Properties of Day logically ANDed together, as: Days of week Monday to Friday

Sample request endpoint for stops

Request Type	GET For e.g. http://api.511.org/Open511/holidays

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept_language	*Optional.*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
Operator_id	*Mandatory*	The operator_id parameter supports filtering based on a particular operator id/code
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The transit day type response for XML is shown in Appendix A Section A.2.7. The transit day type response for JSON is shown in Appendix B Section B.2.7.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual Holiday resource cannot be identified)

3.7 API: Announcement

Announcement is completely SIRI entity; it is a description of a situation/condition about the public transport. Announcement consists of Situations which is collection of PtSituationElement which contains description of situation/condition, at least one PtSituationElement is mandatory.

Below is a message structure of PtSituationElement for Announcement contained within Situations.

Field	Type	Mandatory/ Optional	Description
CreationTime	DateTime	*Mandatory*	Time of the creation of the situation.
SituationNumber	Integer	*Mandatory*	Unique identifier for the situation.
Source	Container	*Mandatory*	Information about source of information
—SourceType	Enum	*Mandatory*	Nature of source (feed, email, text, etc)
—Name	Free Text	*Optional*	Name of source
ValidityPeriod	Container	*Mandatory*	It is a container for validity period of the situation
—StartTime	DateTime	*Mandatory*	It is inclusive start time of the situation
—EndTime	DateTime	*Optional*	It is inclusive end time stamp for situation. If omitted the situation is interpreted as to be forever.
Priority	Non Negative Integer	*Optional*	An arbitrary rating of the situation priority (1=high).
ScopeType	Enum	*Optional*	Provides the nature of scope, e.g. general, network etc
Summary	Free Text	*Optional*	It is the summary of situation, id absent it is derived from situation Description
Description	Free Text	*Mandatory*	Description of the situation
InfoLinks	Container	*Optional*	Hyperlinks to other resources associated with situation
—InfoLink	Container	*Mandatory*	It is container for the hyperlink associated with situation
——Uri	Link	*Mandatory*	Hyperlink associated with situation
Consequences	Container	*Mandatory*	It is the collection of consequence (SIRI element) which describes effect of the situation on Public Transport system. It has at least one consequence

Consequence structure

The Consequence structure is the main element of the Consequences collection. It contains information about the nature of the effect or disrupt on to the public transport service.

Field	Type	Mandatory/ Optional	Description
Severity	Enum	*Mandatory*	Severity of disruption, it could be different from that of situation

Affects	Free Text	*Optional*	Description about parts of transport network affected by situation.
—Operators	Container	*Mandatory*	Container for collection of affected operators. It has one or more AffectedOperator
——AffectedOperator	Container	*Mandatory*	Container for operators affected by the situation
———OperatorRef	Ref	*Mandatory*	Contains reference to operator affect by situation
———OperatorName	Free Text	*Mandatory*	Public name of the affected operator
——Networks	Container	*Mandatory*	Container for collection of affected Network. It has one or more AffectedOperator
———AffectedNetwork	Container	*Mandatory*	Contains network or Route(s) affected by situation
————AffectedLine	Container	*Mandatory*	Information about the individual lines in the network that are affected. Contains one or more LineRef sub elements
—————LineRef	Ref	*Mandatory*	Contains reference to Line affected by situation
—StopPoints	Container	*Optional*	Container for collection of affected StopPoints. It has one or more AffectedStopPoint
——AffectedStopPoint	Container	*Mandatory*	Container for StopPoints affected by the situation
———StopPointRef	Ref	*Mandatory*	Contains reference to StopPoint affect by situation

Sample request endpoint for stops

Request Type	GET For e.g. http://api.511.org/Open511/transitannoucements/

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept_language	*Optional.*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
Operator_id	*Optional*	The operator_id parameter supports filtering based on a particular operator id/code
Line_id	*Optional*	The line_id parameter supports filtering based on a particular line id
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The transit stop announcement response for XML is shown in Appendix A Section A.2.8. The transit announcement response for JSON is shown in Appendix B Section B.2.8.*Possible Errors*

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual Annoucement resource cannot be identified)

3.8 API: Transit Scheduled Departures for a Stop

SIRI Stop Timetable service provides static/scheduled timetables in the system for a particular stop. Below is a message structure of Transit Scheduled Departures in SIRI ST (Stop Timetable) format which consists of a single **ServiceDelivery** node containing details on scheduled visits to this stop within a departure window.

Field	Type	Mandatory / Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Timestamp of server response.
Status	Enum	*Optional*	Indicates success or failure of request.
			true - success **false** - failure, SIRI error response will be returned
StopTimetableDelivery	Object	*Mandatory*	Contains multiple **TimetabledStopVisit** nodes, one for each visit to the stop within the Departure window.

StopTimetableDelivery structure

Field	Type	Mandatory / Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Timestamp of server response.
SubscriptionRef	Xsd:NMToken	Mandatory	Identifier of service subscription- unique within Service and Subscriber
TimetabledStopVisit	Object	*Mandatory*	A visit to a stop by a vehicle in the production timetable

TimetabledStopVisit structure

This contains details on a single visit to the stop within the Departure window.

Field	Type	Mandatory	Description

		/ Optional	
RecordedAtTime	Date Time	*Mandatory*	Date and time when data was recorded.
MonitoringRef	Free Text	*Mandatory*	Identifier of stop monitoring point that Stop Visit applies.
TargetedVehicleJourney	Object	*Mandatory*	Contains a single **TargetedVehicleJourney** node.

TargetedVehicleJourney structure

This contains details on a single visit to the stop within the Departure window.

Field	Type	Mandatory / Optional	Description
LineRef	Free Text	*Mandatory*	A Line in SIRI is equivalent to a Route in GTFS. Value is RouteCode eg: "917" = "Fremont" for "BART" agency.
DirectionRef	Enum	*Mandatory*	Value could be either INBOUND or OUTBOUND etc.
FramedVehicleJourneyRef	Object	*Optional*	A compound element uniquely identifying the trip the vehicle is serving.
PublishedLineName	Free Text	*Optional*	Value is Route Name eg: "Fremont" for "BART" agency.
OperatorRef	Reference ID	*Optional*	Operator of the journey
OriginRef	Computed Text	*Optional*	The stop ID for the first stop on the trip the vehicle is serving, prefixed by Agency Name and or Route Name to make it unique eg: "BART_11".
OriginName	Free Text	*Optional*	The stop Name for the first stop on the trip the vehicle is serving, prefixed by Agency Name eg: "BART_CIVIC CENTER".
DestinationRef	Computed Text	*Optional*	The stop ID for the last stop on the trip the vehicle is serving, prefixed by Agency Name eg: "BART_99".
DestinationName	Free Text	*Optional*	The stop Name for the last stop on the trip the vehicle is serving, prefixed by Agency Name eg: "BART_16th St-Mission".
TargetedCall	Object	*Optional*	*Contains a single **TargetedCall** node.*

FramedVehicleJourneyRef Structure

Field	Type	Mandatory / Optional	Description
DataFrameRef	Date time	*Mandatory*	The service date for the trip the vehicle is serving.
DatedVehicleJourneyRef	Free Text	*Mandatory*	The trip ID for trip the vehicle is serving, preceded by the agency name or ID to make it unique.

TargetedCall structure

This describes the arrival and departure times for a specific visit.

Field	Type	Mandatory/ Optional	Description
VisitNumber	Numeric	*Mandatory*	For journey patterns that involve repeated visits by a vehicle to a stop, the VisitNumber count is used to distinguish each separate visit.
AimedArrivalTime	DateTime	*Mandatory*	Value is expected arrival time.
AimedDepartureTime	DateTime	*Mandatory*	Value is expected departure time.

Sample request endpoint

Request Type	**GET** For e.g. http://api.511.org/Open511/TransitScheduledDeparturesByStop

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by

		jurisdiction.
LineRef	*Optional*	The RouteCode that uniquely identifies a transit route.
MonitoringRef	*Mandatory*	The StopCode that uniquely identifies a physical stop or platform.
StartTime	*Optional*	The start date parameter allows for requesting departures within a departure window.
EndTime	*Optional*	The end date parameter allows for requesting departures within a departure window.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The example response for XML in SIRI ST format is shown in Appendix A Section A.2.9. The example response for JSON in SIRI ST format is shown in Appendix B Section B.2.9.

3.9 API: Real-time predictions at a Stop

Siri Stop Monitoring service provides current and forthcoming veicles arrivals and depatures at a stop.

Below is a message structure of real-time departures which consists of a single **ServiceDelivery** node containing details on monitored visits to this stop.

Field	Type	Mandatory / Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Timestamp of response from server.
Status	Enum	*Mandatory*	TODO: enum here
StopMonitoringDelivery	Object	*Mandatory*	Contains multiple **MonitoredStopVisit** entries, one per visit to the stop.

StopMonitoringDelivery structure

Field	Type	Mandatory / Optional	Description
MonitoredStopVisit	Object	*Required*	This contains monitored vehicle journey (real-time trip) information.
MonitoredStopVisitCancellation	Object	*Optional*	This contains cancellation information for a trip.
StopLineNotice	Object	*Optional*	This provides notices for lines serving this monitored stop.
StopLineNoticeCancellation	Object	*Optional*	This provides cancellation of previous issued notices for lines serving this monitored stop.

MonitoredStopVisit structure

Field	Type	Mandatory / Optional	Description
RecordedAtTime	DateTime	*Required*	The timestamp of the last real-time update from the particular vehicle.
MonitoringRef	Free Text	*Optional*	Name of the Stop being monitored
MonitoredVehicleJourney	Object	*Optional*	Real-time information about particular vehicles

MonitoredVehicleJourney structure

Field	Type	Mandatory / Optional	Description
OperatorRef	Free Text	*Mandatory*	***For AgencyCode requirement***, eg: "BART". Could be moved under sub-node Extensions because it's NOT part of the SIRI spec.
LineRef	Free Text	*Mandatory*	***For Route Code requirement.*** A Line in SIRI is equivalent to a Route in GTFS. Value could either be RouteCode or RouteName as required. Recommend using RouteCode because "PublishedLineName" is using RouteName. eg: RouteCode "917" = RouteName "Fremont" for BART. Does not identify the Agency, so RouteCode or RouteName would have to be unique to an Agency.
DirectionRef	Defined Text	*Mandatory*	***For Direction requirement.*** "In" = inbound, "Out" = outbound
FramedVehicleJourneyRef	Object	*Mandatory*	A compound element uniquely identifying the trip the vehicle is serving.
PublishedLineName	Free Text	*Mandatory*	***For Route name requirement.***
OriginRef	Computed Text	*Optional*	"The GTFS stop ID for the first stop on the trip the vehicle is serving, prefixed by Agency ID." We don't have an Agency ID, so would use Agency Name eg: "BART_11".
OriginName	Free Text	*Optional*	***For Origin place name requirement.*** "The GTFS stop Name for the first stop on the trip the vehicle is serving, prefixed by Agency ID." We don't have an Agency ID, so would use Agency Name eg: "BART_CIVIC CENTER".
DestinationRef	Computed Text	*Optional*	"The GTFS stop ID for the last stop on the trip the vehicle is serving, prefixed by Agency ID." We don't have an Agency ID, so would use Agency Name eg: "BART_99".
DestinationName	Free Text	*Optional*	***For Destination place name requirement.***

			"The GTFS stop Name for the last stop on the trip the vehicle is serving, prefixed by Agency ID." We don't have an Agency ID, so would use Agency Name eg: "BART_16th St-Mission".
MonitoredCall	Object	*Mandatory*	Call data for the stop
OnwardsCalls	Object	*Optional*	Call data for next stops
PreviousCalls	Object	*Optional*	Call data for previous stops
ProgressStatus	Enum	*Optional*	Status of the current vehicle, On-time, Running early etc.
VehicleLocation	Object	*Optional*	Vehicle location information. (Latitude/Longitude)

FramedVehicleJourneyRef Structure

Field	Type	Mandatory / Optional	Description
DataFrameRef	Date time	*Mandatory*	The service date for the trip the vehicle is serving.
DatedVehicleJourneyRef	Free Text	*Mandatory*	The trip ID for trip the vehicle is serving, preceded by the agency name or ID to make it unique.

Monitored/Onward/Previous Call structure

Field	Type	Mandatory/ Optional	Description
VisitNumber	Numeric	*Mandatory*	For JOURNEY PATTERNs that involve repeated visits by a VEHICLE to a stop, the VisitNumber count is used to distinguish each separate visit.
VehicleLocationAtStop	Object	*Optional*	Vehicle location information at stop. (Latitude/Longitude)
VehicleAtStop	Boolean	*Mandatory*	True if vehicle is at the stop.
AimedArrivalTime	DateTime	*Mandatory*	For Expected arrival time requirement.
ExpectedArrivalTime	DateTime	*Mandatory*	For estimated arrival time requirement.

AimedDepartureTime	DateTime	*Mandatory*	For Expected departure time requirement.
ExpectedDepartureTime	DateTime	*Mandatory*	For estimated departure time requirement.
Distances	Object	*Optional*	Extension to SIRI Call structure to incorporate distance and bearing information of vehicle from the stop.

Distances structure

Field	Type	Mandatory/ Optional	Description
CallDistanceAlongRoute	Numeric	*Optional*	Distance of the stop from the beginning of the trip/route
DistanceFromCall	Numeric	*Optional*	Distance from the vehicle to the stop along the route, in meters
StopsFromCall	Numeric	*Optional*	The number of stops on the vehicle's current trip until the stop in question, starting from 0.
PresentableDistance	Text	*Optional*	Suggested display for the distance of vehicle from the stop.

MonitoredStopVisitCancellation structure

Field	Type	Mandatory/ Optional	Description
RecordedAtTime	DateTime	*Mandatory*	The timestamp of the last real-time update from the particular vehicle.
MonitoringRef	Free Text	*Mandatory*	Name of the Stop being monitored
VisitNumber	Numeric	Mandatory	Cancelled sequence of visit to this stop. For JOURNEY PATTERNs that involve repeated visits by a VEHICLE to a stop, the VisitNumber count is used to distinguish each separate visit.
Reason	Free text	Mandatory	Reason for cancellation of monitoring. For e.g. Vehicle has already arrived at the stop.

StopLineNotice structure

Field	Type	Mandatory/ Optional	Description
RecordedAtTime	DateTime	*Mandatory*	The timestamp of the last real-time update from the particular vehicle.
ItemRef	Free Text	*Mandatory*	Reference to a previously issued notice.
MonitoringRef	Free Text	*Mandatory*	Name of the Stop being monitored
LineRef	Free Text	*Mandatory*	***For Route Code requirement.*** A Line in SIRI is equivalent to a Route in GTFS. Value could either be RouteCode or RouteName as required. Recommend using RouteCode because "PublishedLineName" is using RouteName. eg: RouteCode "917" = RouteName "Fremont" for BART. Does not identify the Agency, so RouteCode or RouteName would have to be unique to an Agency.
DirectionRef	Defined Text	*Mandatory*	***For Direction requirement.*** "In" = inbound, "Out" = outbound
Note	Free Text	*Optional*	Note about the cancellation.

StopLineNoticeCancellation structure

Field	Type	Mandatory/ Optional	Description
RecordedAtTime	DateTime	*Mandatory*	The timestamp of the last real-time update from the particular vehicle.
ItemIdentifier	Free Text	*Mandatory*	Unique identifier for this notice
MonitoringRef	Free Text	*Mandatory*	Name of the Stop being monitored
LineRef	Free Text	*Mandatory*	***For Route Code requirement.*** A Line in SIRI is equivalent to a Route in GTFS. Value could either be RouteCode or RouteName as required. Recommend using RouteCode because "PublishedLineName" is using RouteName.

			eg: RouteCode "917" = RouteName "Fremont" for BART. Does not identify the Agency, so RouteCode or RouteName would have to be unique to an Agency.
DirectionRef	Defined Text	*Mandatory*	***For Direction requirement.*** "In" = inbound, "Out" = outbound
LineNote	Free Text	*Mandatory*	Information about the notice.

Sample request endpoint

Request Type	**GET** For e.g. http://api.511.org/Open511/TransitRealTimeDeparturesAtStop

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
OperatorRef	*Optional*	Agency ID to be monitored.
LineRef	*Optional*	The unique identifier or a transit route. Value could either be RouteCode or RouteName as required. Recommend RouteCode because response has "PublishedLineName" as RouteName.
MonitoringRef	*Mandatory*	The unique identifier of a physical stop or platform. Value could either be StopCode or StopName as required.

DirectionRef	*Optional*	Direction (ID) for the route.
MaximumNumberOfCallsOnwards	*Optional*	Limits the number of OnwardCall elements returned in the query.
MaximumStopVisits	*Optional*	an upper bound on the number of buses to return in the results
MinimumStopVisitsPerLine	*Optional*	a lower bound on the number of buses to return in the results per line/route (assuming that many are available)
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The transit real time departure service delivery mode response for XML is shown in Appendix A Section A.2.10 and for JSON is shown in Appendix B Section B.2.10.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If a resource cannot be located)

3.10 API: Real-time Vehicle Monitoring

Siri Vehicle monitorting service provides information about current location and expected activiites of a particular vehicle. It also provides details for current and subsequent Journey patterns.
Below is a message structure for real-time vehicle/trip monitoring which consists of a single **ServiceDelivery** node containing details on vehicle/trip within an agency that are currently operational and being monitored.

Field	Type	Mandatory/ Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Timestamp of response from server.
Status	Enum	*Mandatory*	Indicates success or failure of request. **true** - success **false** - failure, SIRI error response will be returned
VehicleMonitoringDelivery	Object	*Mandatory*	Contains multiple **VehicleActivity** entries, one per trip, if monitored.

VehicleMonitoringDelivery structure

Field	Type	Mandatory/ Optional	Description
VehicleActivity	Object	*Required*	This contains monitored vehicle journey (real-time trip) information.
VehicleActivityCancellation	Object	*Optional*	This contains cancellation information for a trip.

VehicleActivity structure

Field	Type	Mandatory / Optional	Description
RecordedAtTime	DateTime	*Required*	The timestamp of the last real-time update from the particular vehicle.
ValidUntilTime	DateTime	*Required*	Time until which data is valid.
MonitoredVehicleJourney	Object	*Optional*	Real-time information about particular vehicles

MonitoredVehicleJourney structure

Field	Type	Mandatory/ Optional	Description
OperatorRef	Free Text	*Mandatory*	***For AgencyCode requirement***, eg: "BART". Could be moved under sub-node Extensions because it's NOT part of the SIRI spec.
LineRef	Free Text	*Mandatory*	***For Route Code requirement.*** A Line in SIRI is equivalent to a Route in GTFS. Value could either be RouteCode or RouteName as required. Recommend using RouteCode because "PublishedLineName" is using RouteName. eg: RouteCode "917" = RouteName "Fremont" for BART. Does not identify the Agency, so RouteCode or RouteName would have to be unique to an Agency.
DirectionRef	Defined Text	*Mandatory*	***For Direction requirement.*** "In" = inbound, "Out" = outbound
FramedVehicleJourneyRef	Object	*Mandatory*	A compound element uniquely identifying the trip the vehicle is serving.

PublishedLineName	Free Text	*Mandatory*	***For Route name requirement.***
OriginRef	Computed Text	*Optional*	"The GTFS stop ID for the first stop on the trip the vehicle is serving, prefixed by Agency ID." We don't have an Agency ID, so would use Agency Name eg: "BART_11".
OriginName	Free Text	*Optional*	***For Origin place name requirement.*** "The GTFS stop Name for the first stop on the trip the vehicle is serving, prefixed by Agency ID." We don't have an Agency ID, so would use Agency Name eg: "BART_CIVIC CENTER".
DestinationRef	Computed Text	*Optional*	"The GTFS stop ID for the last stop on the trip the vehicle is serving, prefixed by Agency ID." We don't have an Agency ID, so would use Agency Name eg: "BART_99".
DestinationName	Free Text	*Optional*	***For Destination place name requirement.*** "The GTFS stop Name for the last stop on the trip the vehicle is serving, prefixed by Agency ID." We don't have an Agency ID, so would use Agency Name eg: "BART_16th St-Mission".
MonitoredCall	Object	*Optional*	Call data for the current stop
OnwardsCalls	Object	*Optional*	Call data for next stops
PreviousCalls	Object	*Optional*	Call data for previous stops
ProgressStatus	Enum	*Optional*	Status of the current vehicle, On-time, Running early etc.
VehicleLocation	Object	*Optional*	Vehicle location information. (Latitude/Longitude)

FramedVehicleJourneyRef Structure

Field	Type	Mandatory / Optional	Description
DataFrameRef	Date time	*Mandatory*	The service date for the trip the vehicle is serving.
DatedVehicleJourneyRef	Free Text	*Mandatory*	The trip ID for trip the vehicle is serving, preceded by the agency name or ID to make it unique.

Monitored/Onward/Previous Call structure

Field	Type	Mandatory / Optional	Description
StopName	Free Text	*Mandatory*	Name of the stop
VisitNumber	Numeric	*Mandatory*	For JOURNEY PATTERNs that involve repeated visits by a VEHICLE to a stop, the VisitNumber count is used to distinguish each separate visit.
VehicleLocationAtStop	Object	*Optional*	Vehicle location information at stop. (Latitude/Longitude)
VehicleAtStop	Boolean	*Optional*	True if vehicle is at the stop.
AimedArrivalTime	DateTime	*Optional*	For Expected arrival time requirement.
ExpectedArrivalTime	DateTime	*Optional*	For estimated arrival time requirement.
ActualArrivalTIme	Date Time	*Optional*	Observed arrival time.
AimedDepartureTime	DateTime	*Optional*	For Expected departure time requirement.
ExpectedDepartureTime	DateTime	*Optional*	For estimated departure time requirement.
ActualDepartureTime	Date Time	*Optional*	Observed departure time.
Distances	Object	*Optional*	Extension to SIRI Call structure to incorporate distance and bearing information of vehicle from the stop.

Distances structure

Field	Type	Mandatory/ Optional	Description
CallDistanceAlongRoute	Numeric	*Optional*	Distance of the stop from the beginning of the trip/route
DistanceFromCall	Numeric	*Optional*	Distance from the vehicle to the stop along the route, in meters
StopsFromCall	Numeric	*Optional*	The number of stops on the vehicle's current trip until the stop in question, starting from 0.
PresentableDistance	Text	*Optional*	Suggested display for the distance of vehicle from the stop.

VehicleActivityCancellation structure

Field	Type	Mandatory/ Optional	Description
RecordedAtTime	Date Time	*Mandatory*	The timestamp when data was recorded.
VehicleJourneyRef	Object	Mandatory	A compound element uniquely identifying the trip the vehicle is serving.
LineRef	Free Text	*Mandatory*	***For Route Code requirement.*** A Line in SIRI is equivalent to a Route in GTFS. Value could either be RouteCode or RouteName as required. Recommend using RouteCode because "PublishedLineName" is using RouteName. eg: RouteCode "917" = RouteName "Fremont" for BART. Does not identify the Agency, so RouteCode or RouteName would have to be unique to an Agency.
DirectionRef	Defined Text	*Mandatory*	***For Direction requirement.*** "In" = inbound, "Out" = outbound
Reason	Free Text	Mandatory	Reason for cancellation of this trip. For e.g. Vehicle has completed all its journeys.

Sample request endpoint

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/TransitRealTimeVehicleMonitoring

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
OperatorRef	*Optional*	Agency ID to be monitored.
line_ref	*Optional*	The unique identifier or a transit route. Value could either be RouteCode or RouteName as required. Recommend RouteCode because response has "PublishedLineName" as RouteName.
VehicleRef	*Optional*	The unique identifier of the vehicle to be monitored.
DirectionRef	*Optional*	Direction (ID) for the route.
MaximumNumberOfCallsOnwards	*Optional*	Limits the number of OnwardCall elements returned in the query.
MaximumStopVisits	*Optional*	an upper bound on the number of buses to return in the results
MinimumStopVisitsPerLine	*Optional*	a lower bound on the number of buses to return in the results per line/route (assuming that many are available)
MaximumVehicles	*Optional*	The maximum number of vehicle journeys in a given delivery.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The real time vehicle monitoring response for XML in SIRI format is shown in Appendix A Section A.2.11 and for JSON is shown in Appendix B Section B.2.11.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If a resource cannot be located)

3.11 API: Transit Schedule Updates for an agency

Siri Production Timetable provides information about the expected operation of a transport network for a specified day.

Below is a message structure of Transit Schedule Updates in SIRI PT (Production Timetable) format which consists of a single **ServiceDelivery** node containing details on schedule updates for a specific line and direction by agency.

Field	Type	Mandatory/ Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Timestamp of server response.
Status	Enum	*Mandatory*	Indicates success or failure of request. **true** - success **false** - failure, SIRI error response will be returned
ProductionTimetableDelivery	Object	*Mandatory*	Contains multiple **DatedTimetableVersionFrame** node.

ProductionTimetableDelivery structure

Field	Type	Mandatory/ Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Timestamp of server response.
DatedTimetableVersionFrame	Object	*Mandatory*	A timetable to run on a specific date

DatedTimetableVersionFrame structure

Field	Type	Mandatory/ Optional	Description
RecordedAtTime	DateTime	*Mandatory*	Date and time when data was recorded.
LineRef	Free Text	*Mandatory*	A Line in SIRI is equivalent to a Route in GTFS. Value is RouteCode eg: "917" = "Fremont" for “BART” agency.
DirectionRef	Enum	*Mandatory*	Value is either INBOUND or OUTBOUND
PublishedLineName	Free Text	*Mandatory*	Value is Route Name eg: “Fremont” for “BART” agency.
LineNote	Free Text	*Optional*	Text message describing this change.

DatedVehicleJourney	Object	*Mandatory*	Contains a **DatedVehicleJourney** node.

DatedVehicleJourney structure

Field	Type	Mandatory/ Optional	Description
DatedVehicleJourneyCode	Free Text	*Mandatory*	Identifies the vehicle journey (Tripid).
DatedCalls	Objects	*Mandatory*	May contain multiple **DatedCall** nodes.

DatedCall structure

Field	Type	Mandatory/ Optional	Description
StopPointRef	Numeric	*Mandatory*	The GTFS stop ID for this stop on the trip the vehicle is serving, prefixed by Agency Name e.g.: "BART_11".
AimedArrivalTime	DateTime	*Mandatory*	Value is expected arrival time.
AimedDepartureTime	DateTime	*Mandatory*	Value is expected departure time.
CallNote	Text	*Optional*	Text message describing this change.

Sample request endpoint

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/TransitScheduleUpdatesByAgency

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
OperatorRef	*Mandatory*	The Agency Name that uniquely identifies a transit agency.

Lineref	*Optional*	The unique identifier or a transit route. Value could either be RouteCode or RouteName as required. Recommend RouteCode because response has "PublishedLineName" as RouteName.
DirectionRef	*Optional*	Direction (ID) for the route.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The transit schedule update response for XML is shown in Appendix A Section A.2.12. The transit schedule update response for JSON is shown in Appendix B Section B.2.12.

Possible Errors

To be completed

3.12 API: Transit Addition and Cancellation of Trips by Agency

Siri Estimated Timetable service provides details of the operation of the transport network for a period within the current day, detailing real time deviations from the timetables and control actions affecting the Timetable (cancellations, additional Journeys and Detours).

Below is a message structure of Transit Addition and Cancellation of Trips in SIRI ET (Estimated Timetable) format which consists of a single **ServiceDelivery** node containing details on schedule updates for a specific line and direction by agency.

Field	Type	Mandatory/ Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Timestamp of server response.
Status	Enum	*Mandatory*	Indicates success or failure of request. **true** - success **false** - failure, SIRI error response will be returned
EstimatedTimetableDelivery	Object	*Mandatory*	Contains multiple **EstimatedJourneyVersionFrame** node.

EstimatedJourneyVersionFrame structure

Field	Type	Mandatory/ Optional	Description
RecordedAtTime	DateTime	*Mandatory*	Date and time when data was recorded.
EstimatedVehicleJourney	Object	*Mandatory*	May contain multiple **EstimatedVehicleJourney** nodes, one for each vehicle.

EstimatedVehicleJourney structure

Provides real-time information about a journey along which a vehicle is running.

Field	Type	Mandatory/ Optional	Description
LineRef	Free Text	*Mandatory*	A Line in SIRI is equivalent to a Route in GTFS. Value is RouteCode e.g.: "917" = "Fremont" for “BART” agency.
DirectionRef	Enum	*Mandatory*	Value is either INBOUND or OUTBOUND
DatedVehicleJourneyRef	Free Text	*Mandatory*	Reference to a dated vehicle journey or trip.
Cancellation	Enum	*Optional*	Value is “true” if cancelled.
PublishedLineName	Free Text	*Mandatory*	Value is Route Name e.g.: “Fremont” for “BART” agency.
EstimatedCalls	Objects	*Mandatory*	May contain multiple **EstimatedCall** nodes. Not returned if journey is cancelled.

Estimated Call structure

This describes the times at a stop. A journey must contain at least two calls.

Field	Type	Mandatory/ Optional	Description
StopPointRef	Numeric	*Mandatory*	The GTFS stop ID for this stop on the trip the vehicle is serving, prefixed by Agency Name e.g.: "BART_11".
AimedArrivalTime	DateTime	*Mandatory*	Value is expected arrival time.
AimedDepartureTime	DateTime	*Mandatory*	Value is expected departure time.
CallNote	Text	*Optional*	Text message describing the update.

Sample request endpoint

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/TransitTripAdditionOrUpdatesByAgency

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
OperatorRef	*Mandatory*	The Agency Name that uniquely identifies a transit agency.
Lineref	*Optional*	The unique identifier or a transit route. Value could either be RouteCode or RouteName as required. Recommend RouteCode because response has "PublishedLineName" as RouteName.
DirectionRef	*Optional*	Direction (ID) for the route.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The transit addition and cancellation response for XML is shown in Appendix A Section A.2.13 and for JSON is shown in Appendix B Section B.2.13.

Possible Errors

To be completed

3.13 API: General Announcements

Siri General Messaging Service provides a structured way to exchange arbitrary informative messages between participants, such as travel news, or operational advice.

Below is a message structure of Service Announcements in SIRI GM (General Message) format which consists of a single **ServiceDelivery** node containing details on messages for a specific route by agency.

Field	Type	Mandatory/ Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Timestamp of server response.
Status	Enum	*Mandatory*	Indicates success or failure of request.
			true - success **false** - failure, SIRI error response will be returned

GeneralMessageDelivery	Object	*Mandatory*	May contain multiple **GeneralMessage** nodes.

General MessageDelivery structure

Field	Type	Mandatory/ Optional	Description
ResponseTimestamp	DateTime	*Mandatory*	Date and time when message was recorded.
GeneralMessage	Object	*Optional*	A message from an agency.

General Message structure

Field	Type	Mandatory/ Optional	Description
RecordedAtTime	DateTime	*Mandatory*	Date and time when message was recorded.
InfoMessageIdentifier	String	*Optional*	Unique identifier of this message.
InfoMessageVersion	Int	*Optional*	Version number of this message.
InfoChannelRef	Text	*Optional*	Information channel to which message belongs.
ValidUntilTime	DateTime	*Optional*	Date and time of message expiration. If not provided, message is open-ended.
Content	Free Text	*Mandatory*	Text message.
Extensions	Node	*Optional*	Will contain additional details on Agency, Route and Direction as required.

Sample request endpoint

Request Type	**GET** For e.g. http://api.511.org/Open511/TransitAnnouncementsByAgency

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
Format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't

		support the response in requested language, response could be in default language selected by jurisdiction.
OperatorRef	*Mandatory*	The Agency Name that uniquely identifies a transit agency.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The transit general messaging service response for XML is shown in Appendix A Section A.2.14 and for JSON is shown in Appendix B Section B.2.14.

Possible Errors

To be completed

4.0Traffic API

The core traffic data resources consist of events and roadway segments (Links). Open511provides message structure and API endpoint for accessing these resources. Open511 will also provide metadata/lookup information that can be used by consumers to filter and limit information during requests.

4.1 API: Event

The event resource provides information about various types of events within a jurisdiction. These can be active incident, scheduled construction/roadway work or public events which may or may not have an impact on traffic conditions. Consumers can request list of all the active events or they can use additional filters such as city, roads to restrict the results as per their needs and use case.

Below is a message structure of Events. Events are a collection of the event resource.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
up	Link	*Mandatory*	Link to the parent resource.
events	Collection of **Event**	*Mandatory*	Main content of the events resource, a collection of event elements. Can contain multiple event elements, at least one occurrence is mandatory.
pagination	Pagination structure	*Optional*	The pagination structure provides information about the current position in the pagination and links to other pages.
— offset	Integer	*Optional*	Offset used to access the current page.
— limit	Integer	*Optional*	Number of events listed in the current page.
— next	Link	*Optional*	link pointing to the next page of the events. Relation attribute value is `next`.
— previous	Link	*Optional*	link pointing the previous page of events. Relation attribute value is `previous`.

Event structure

The event structure is the main element of the events collection. Although an event should be considered as independent, it is possible that one major event (mainly construction) could be split across several events.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
jurisdiction	Link	*Mandatory*	Link to the jurisdiction which published the event. In XML, the `rel` attribute needs the value `jurisdiction`.
id	String	*Mandatory*	A globally unique ID for the event, following the format jurisdiction-id/event-id. For example, my.city.gov/8c3f2. The first segment of the event ID is the jurisdiction ID. The second segment is a string ID that must be unique within its jurisdiction. It can contain the characters a-zA-Z0-9_.-. The two segments are separated with a /. The event's self link is of course itself a globally unique ID, and it is a suggested practice to use the event ID in the event URL, but a separate ID field is included for user-friendliness.
status	Enum	*Mandatory*	Status of the event. The status allows a client to determine if the current event should be considered as currently effective. Value list; - **ACTIVE**; The event should be considered as effective - **ARCHIVED**; The event should not be considered as effective
headline	Free text	*Mandatory*	Title of the event. Should be shorter than 500 characters.
event_type	Enum	*Mandatory*	Basic type of event. Value list: **- CONSTRUCTION : planned road work** **- SPECIAL_EVENT: special events (fair, sport event, etc.)** **- INCIDENT : accidents and other unexpected events** **- WEATHER_CONDITION : Weather condition affecting the road** **- ROAD_CONDITION: Status of the road**

			that might affect travelers.
severity	Enum	*Optional*	Severity of the event. Value list: MINOR: the event has very limited impact on traffic. MODERATE: the event will have a visible impact on traffic but should not create significant delay; if there is a delay, it should be small and local. MAJOR: the event will have a significant impact on traffic, probably on a large scale. UNKNOWN: the impact is unknown, for example in the case of an accident that has been recorded without any precise description.
certainty	Enum	*Optional*	Degree of certainty of the event. Should only be used for unplanned events Could be observed/ likely/possible/unknown
creation_date	DateTime	*Mandatory*	Date of creation of the event element.
last_updated	DateTime	*Mandatory*	This field provides the date and time of the last update. The value is the same as the creation date if no change occurred.
related_events	Collection of Links to event resource	*Optional*	This field is used to group events together. In specific situations (for example complex construction projects), several events might be related together. This field should be used to point a related event.
detour	Free text	*Optional*	If some detour information is available, it could be provided in this field as free text.
traveller_message	Enum	*Optional*	In order to simplify analysis by automated tools, this field allows for providing predefined messages with a known semantic. For example resident-only means that the circulation is limited to people living on this street. The exact enum still has to be determined.
geography	Geospatial	*Mandatory*	Geographical information about the event using the GML notation. Valid type of geometry can be points, multipoints, linestring, multilinestrings and polygon.
road	Road segments affected	*Optional*	Road segments affected by the current event. One event can impact several road segments. A road segment is defined by the road the incident is on and from and to the cross roads.

areas	Collection of area elements	*Optional*	Areas affected by the event.
schedule	Schedule structure	*Mandatory*	Complex item to provide time related information.
attachments	Collection of **attachment links**	*Optional*	Collection of attachments provided additional information about the event (PDFs, images, etc.)
-related	Link	*Optional*	Link to an attachment. While only the URL is mandatory, providing the content type, size, and title of the resource (via the type, length, and title attributes in XML) is encouraged.
Created	DateTime (UTC)	*Mandatory*	When the event was initially created.
Updated	DateTime(UTC)	*Mandatory*	When the content of the event last changed. Will be the same as created if no updates have occurred.
event_subtypes	Collection of event_subtype elements	*Optional*	List of values to provide more detail about the type of event.
-event_subtype	Enum	*Optional*	- ACCIDENT - SERIOUS_ACCIDENT - NUMEROUS_ACCIDENTS - STALLED_VEHICLE - SPILL - OBSTRUCTION - MAJOR_HAZARD - DEMONSTRATION - ROAD_CONSTRUCTION - WORK_IN_THE_MEDIAN - NARROW_LANES - TRAFFIC_ALTERNATING_DIRECTIONS - BRIDGE_OPERATIONS - BLASTING - WORK_ON_UNDERGROUND - EMERGENCY_MAINTENANCE - SPORTING_EVENT - MAJOR_EVENT - CONCERT - FESTIVAL - FIREWORKS - PARADE - CROWD

			- SEVERE_WEATHER - HEAVY_SNOW - SNOW - ICE_GLAZE - HEAVY_FROST - ICE_STORM - DAMAGING_HAIL - THUNDERSTORM - HEAVY_DOWNPOUR - TORNADO - HURRICANE - STRONG_WINDS - DENSE_FOG - FREEZING_FOG - ICE_FOG MIST - VISIBILITY_REDUCED - VISIBILITY_BLOCKED - BLOWING_SNOW - BLOWING_DUST - SANDSTORM - INSECT_SWARMS - IMPASSABLE - ALMOST_IMPASSABLE - PASSABLE_WITH_CARE - SURFACE_WATER_HAZARD - HYDROPLANING_DANGER - SLIPPERY - MUD - LOOSE_GRAVEL - OIL_ON_ROADWAY - ICE - BLACK_ICE - WET_ICY_ROAD - SLUSH - FROZEN_SLUSH - SNOW - PACKED_SNOW - PLOWED_SNOW - POWDER_SNOW - DEEP_SNOW
description	Free text	*Optional*	Description of the event. Plain text description of the event, the reason of the event and any other relevant information for travelers.
certainty	Enum	*Optional*	Degree of certainty of the event. Should only be used for unplanned events (e.g incidents, weather conditions and pavement conditions events). Value list to be confirmed. Could be

			observed/likely/possible/unknown. Value list - OBSERVED - LIKELY - POSSIBLE - UNKNOWN
grouped_events	Collection of links	*Optional*	This structure is used to group events together. In specific situations (for example complex construction projects), several events might be related together. This field should be used to point a related event.
-related	Link	*Optional*	Link pointing to another event resource related to the current event.
detour	Free Text	*Optional*	Description of a detour route to avoid this event.

Road_structure

The road data format is used to provide information about the road(s) affected by an event. The structure of road_affected is:

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
road_name	Free Text	*Mandatory*	Name of the road affected by the event. Ideally, road names should be formatted in accordance with national or regional standards, and the same road should be named consistently in different events.
from_	Free text	*Optional*	Approximate start point of the event on the road. It can be an intersection with another street or it can be a mileage indication. This value should not be used to determine the exact start point since it can be an approximation. The geometry field should be considered as the reference for exact location. If no to field is provided, the from can be considered as a "near".
to	Free text	*Optional*	Approximate end point of the event on the road.

			It can be an intersection with another street or it can be a distance indication.
direction	Enum	*Conditional*	Conditional Direction of the road that is affected by the event. If a lane_status is provided, the direction becomes mandatory. In the same situation, if both directions are affected, two occurrences of the road element are needed, one for each direction and with a dedicated lane_status for each direction. Value list: - N - NW - W - SW - S - SE - E - NE - NONE - BOTH
lane_status	bitmap	*Optional*	Bitmap representing the lanes that are closed because of the event. For example `0110` means that the 2 central lanes of a 4-lane road are closed. The lane status is applied to direction provided
impacted_systems	Collection of **ImpactedSystem**	*Optional*	Allows provision of information about other systems that can be affected.
— impacted_system	Enum	*Optional*	Value list of systems: **Road** **SideWalk** **BikeLane** **OnStreet_Parking** **Parking**
restrictions	Collection of **restriction elements**	*Optional*	Some events may come with some restrictions affecting vehicles using the road (speed, weight).

Restriction structure

The schedule defines timelines of an event.

Field	Type	Mandatory/ Optional	Description
restriction_type	Enum	*Mandatory*	Type of restriction that affects vehicles. Value list: SPEED: Limitation of the speed of vehicles. Unit is in kilometers/hour. WIDTH: Width limitation, mainly for trucks. Unit is meters. HEIGHT: Height limitation, mainly for trucks. Unit is meters WEIGHT: Weight limitation for vehicles. Unit is metric tons. AXLE_WEIGHT: Weight limitation per axle for truck. Unit is metric tons.
value	Flaot	*Mandatory*	Value of the limitation. For example a speed limitation of 60km/h will have a value of 60 with a restriction_type set to SPEED

Schedule structure

The schedule defines timelines of an event.

Field	Type	Mandatory/ Optional	Description
timezone	Timezone	*Optional*	Timezone to be used for this event. Timezone abbreviation (e.g EST) or the continent/city timezone code America/Montreal.
start_date	Date	*Mandatory*	Start date of the event.
end_date	Date	*Optional*	End date of the event. If no end date is provided, the event should be considered continuing forever.
days	collection of day elements	*Optional*	Contains a day tag for every day of the week during which the event is active. So, for an event active on Monday and Wednesday, <days><day>1</day><day>3</day></days>. If omitted, the event is assumed to be active on all days of the week between its start and end dates.
-day	Enum	*Optional*	An integer representing a day of the week during which the event is active. The days are Monday to Sunday from 1 to 7.

--times	Collection of time	*Optional*	Provide detail about the specific times of the event. Multiple occurrences of the time structure are allowed so specify multiple timeframes in a day.
specific_dates	Collection of specific_date	*Optional*	Since some days might not follow the regular schedule, it is possible to provide a schedule for individual days (for example vacations).
— specific_date	Custom time format	*Oprional*	Text string containing the affected date, the start hour and the end hour with the follow format: YYYY-MM-DD HH:mm-HH:mm. If there are multiple disjoint periods on that day, more than one start-end period can be included: YYYY-MM-DD HH:mm-HH:mm HH:mm-HH:mm. If the event is not active that day, no time should be provided: YYYY-MM-DD

Time structure

Field	Type	Mandatory/ Optional	Description
time	structure	*Optional*	Provide detail about the specific times of the event.
— start_time	Time	*Optional*	Start time of the event. If the event is spread across several days, the start time is to be applied for each day.
— end_time	Time	*Optional*	Start time of the event. If the event is spread across several days, the start time is to be applied for each day. **end_time** is mandatory if a **start_time** is provided.

Attachment structure

Field	Type	Mandatory/ Optional	Description
media_type	Enum	*Optional*	Type of content that is linked. Must follow the IANA content type. For example `image/jpeg` for an image encoded with JPEG.
Size	number	*Optional*	Size of the content that is linked expressed in kilobytes (kb)
url	Link	*Mandatory*	Atom link pointing the attachment with attribute `related`.

description	Free text	*Optional*	Human readable description of the attachment.

Sample request endpoint for events

Request Type	**GET**
Request Endpoint Examp;e	For e.g. http://api.511.org/Open511/Events

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
start_date	*Optional*	The start date parameter allows for searching events that will be or have been active anytime after the provided value. The start date parameter can be past or future. The parameter should be complete DateTime.
end_date	*Optional*	The end date parameter allows for searching events that will be or have been active any time before the provided parameter. The end date parameter can be past or future. The parameter should include complete DateTime including the timezone.
status	*Optional*	By default the API should only send active events. If the client wants to retrieve archived events, it should provide the status with the value `archived` or a star `*` to retrieve both active and inactive events.
severity	*Optional*	Although it is possible to query events for a given severity, it is also possible to query all events below or equal level 2 with `severity=2-` or above or equal to level 2 `severity=2+`
jurisdiction	*Optional*	Clients should be able to request events for a given

		jurisdiction on multi-jurisdiction endpoints or on aggregators.
geography	*Optional*	Filtering near a point or line string or within a polygon.
area	*Optional*	This allows clients to filter events happening in a city/area.
roads	*Optional*	This allows clients to filter events that are associated to roads (on, from or to) in the affected road segments.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The traffic event structure response for XML is shown in Appendix A Section A.3.1 and for JSON is shown in Appendix B Section B.3.1.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual event resource cannot be located). For e.g.: http://api.511.org/Open511/Events/1345

4.2 API: Announcement

Announcement is special type of event that isn't specific to a location. These may include travel advisories during a bridge closure or emergency messages during a major incident. Announcements can also include notifications by agencies such as start/stop of a new service,

Below is a message structure of Announcements.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
up	Link	*Mandatory*	Link to the parent resource.
Announcements	Collection of **Announcement**	*Mandatory*	Can contain multiple Announcement elements, at least one occurrence is mandatory.
pagination	Pagination structure	*Optional*	The pagination structure provides information about the current position in the pagination and links to other pages.
— offset	Integer	*Optional*	Offset used to access the current page.
— limit	Integer	*Optional*	Number of announcements listed in the current page.

— next	Link	*Optional*	Atom link pointing to the next page of announcements. Relation attribute value is `next`.
— previous	Link	*Optional*	Atom link pointing the previous page of announcements. Relation attribute value is `previous`.

Announcement structure

The announcement is the main element of the announcements collection.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
jurisdiction	Link	*Mandatory*	Link to the jurisdiction which published the announcement. In XML, the `rel` attribute needs the value `jurisdiction`.
status	Enum	*Mandatory*	Status of the announcement. The status allows a client to determine if the current announcement should be considered as currently effective. Value list; **active**; The announcement should be considered as effective **archive**; The announcement should not be considered as effective
headline	Free text	*Mandatory*	Title of the announcement. Should be shorter than 500 characters.
Type	Enum	*Mandatory*	Basic type of announcement. Value list: **Traffic** **Transit** **General**
city	City	*Optional*	City in which the announcement is for (or near).
creation_Date	DateTime	*Mandatory*	Date of creation of the announcement.
last_update	DateTime	*Mandatory*	This field provides the date and time of the last update. The value is the same as the creation date if no change occurred.
related_events	Collection of Links to event resource	*Optional*	This field is used to link an announcement to one of more events.

Sample request endpoint for Announcements

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/Announcements

Parameters and Filters supported with the request

Parameter	Mandatory / Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
start_date	*Optional*	The start date parameter allows for searching announcements that will be or have been active any time after the provided value. The start date parameter can be past or future. The parameter should be complete DateTime.
end_date	*Optional*	The end date parameter allows for searching announcements that will be or have been active any time before the provided parameter. The end date parameter can be past or future. The parameter should include complete DateTime including the timezone.
status	*Optional*	By default the API should only send active announcements. If the client wants to retrieve archived announcements, it should provide the status with the value `archived` or a star `*` to retrieve both active and inactive announcements.
jurisdiction	*Optional*	Clients should be able to request announcements for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
city	*Optional*	This allows clients to filter announcements happening in a city/area.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The traffic announcment response for XML is shown in Appendix A Section A.3.2 and for JSON is shown in Appendix B Section B.3.2.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual announcement resource cannot be located). For e.g.: http://api.511.org/Open511/Announcements/1345

4.3 API: Area

A jurisdiction can provide information about its geographical coverage using an area resource. Area could be type of county, city, neighbourhood etc. Area resource is useful in defining other resources such as roads and segments.

Below is a message structure of Area. Area is a collection of the area resource.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
up	Link	*Mandatory*	Link to the parent resource.
areas	Collection of **area**	*Mandatory*	Main content of the area resource, a collection of area elements. Can contain multiple area elements, at least one occurrence is mandatory.
pagination	Pagination structure	*Optional*	The pagination structure provides information about the current position in the pagination and links to other pages.
— offset	Integer	*Optional*	Offset used to access the current page.
— limit	Integer	*Optional*	Number of events listed in the current page.
— next	Link	*Optional*	link pointing to the next page of the area. Relation attribute value is `next`.
— previous	Link	*Optional*	link pointing the previous page of area. Relation attribute value is `previous`.

Area structure

The area is the main element of the Area collection.

Field	Type	Mandatory/ Optional	Description
name	Free text	*Mandatory*	Name of the area.
id	Integer	*Mandatory*	Unique GeoNames identified for the area
self	Link	*Mandatory*	Link pointing to the RDF representation of the GeoNames place, i.e. http://geonames.org/{id}/about.rdf

Sample request endpoint for area

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/area

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request area for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The traffic area response for XML is shown in Appendix A Section A.3.3 and for JSON is shown in Appendix B Section B.3.3.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)

4.4 API: Road

Roads are list of roadways covered by a jurisdiction. A consumer can use this information to filter events, segments and nodes during a request.

Below is a message structure of Roads.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
up	Link	*Mandatory*	Link to the parent resource.
Roads	Collection of **Road**	*Mandatory*	Can contain multiple Road elements, at least one occurrence is mandatory.
pagination	Pagination structure	*Optional*	The pagination structure provides information about the current position in the pagination and links to other pages.
— offset	Integer	*Optional*	Offset used to access the current page.
— limit	Integer	*Optional*	Number of roads listed in the current page.
— next	Link	*Optional*	link pointing to the next page of roads. Relation attribute value is `next`.
— previous	Link	*Optional*	link pointing the previous page of roads. Relation attribute value is `previous`.

Road structure

The road is the main element of the roads collection.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
jurisdiction	Link	*Mandatory*	Link to the jurisdiction which published the announcement. In XML, the `rel` attribute needs the value `jurisdiction`.
name	Free text	*Mandatory*	Name of the road.
--type	Enum	*Mandatory*	Basic type of road. Value list: **Freeway** **Arterials**
segments	Collection of **segments**	*Optional*	Collection of segments on this road.

Sample request endpoint for Roads

Request Type	**GET** For e.g. http://api.511.org/Open511/Roads

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request cities for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The traffic road for XML is shown in Appendix A Section A.3.4 and for JSON is shown in Appendix B Section B.3.4.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual road resource cannot be located). For e.g.: http://api.511.org/Open511/Roads/I880

4.5 API: Node

Nodes are arbitraty point on roadways that help define start and end location of a roadway segment. These points are typically defined at exits or entrances of freeways and highways or at intersections of streets and arterials.

Nodes are useful in defining traffic segments (see segments definition below).

Below is a message structure of Nodes.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
up	Link	*Mandatory*	Link to the parent resource.
Nodes	Collection of **Node**	*Mandatory*	Can contain multiple Node elements, at least one occurrence is mandatory.
pagination	Pagination structure	*Optional*	The pagination structure provides information about the current position in the pagination and links to other pages.
— offset	Integer	*Optional*	Offset used to access the current page.
— limit	Integer	*Optional*	Number of nodes listed in the current page.
— next	Link	*Optional*	Atom link pointing to the next page of nodes. Relation attribute value is `next`.
— previous	Link	*Optional*	Atom link pointing the previous page of nodes. Relation attribute value is `previous`.

Node structure

The node is the main element of the nodes collection.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
jurisdiction	Link	*Mandatory*	Link to the jurisdiction which published the node. In XML, the `rel` attribute needs the value `jurisdiction`.
name	Free text	*Mandatory*	Name of the node, This should denote name of roads that intersect at this node. For e.g. I680&24
area	area	*Optional*	Area resource to which this node belongs.
roads	Collection of roads	*Optional*	List of roads that intersect at this node.
geography	geospatial	*Mandatory*	Location of the node in WGS84 format.

Sample request endpoint for Nodes

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/Nodes

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request cities for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.
area	*Optional*	Clients should be able to request/filter roads that belong to an area.
road	*Optional*	Clients should be able to request nodes on a particular main road or cross road.

The traffic node response for XML is shown in Appendix A Section A.3.5 and for JSON is shown in Appendix B Section B.3.5.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual node resource cannot be located). For e.g.: http://api.511.org/Open511/Nodes/80&Fremontst

4.6 API: Segments

Traffic and surface conditions along a raoadway are provided on segments. Segments are defined as portion of road between two points (nodes) in one particular direction. The Segments API will provide configuration data for one or many segments covered by a jurisdiction. Segments are often refreered as links in other system.

Below is a message structure of Segment.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
up	Link	*Mandatory*	Link to the parent resource.
Segments	Collection of **Segment**	*Mandatory*	Can contain multiple segment elements, at least one occurrence is mandatory.
pagination	Pagination structure	*Optional*	The pagination structure provides information about the current position in the pagination and links to other pages.
— offset	Integer	*Optional*	Offset used to access the current page.
— limit	Integer	*Optional*	Number of traffic_segments listed in the current page.
— next	Link	*Optional*	link pointing to the next page of traffic_segments. Relation attribute value is `next`.
— previous	Link	*Optional*	link pointing the previous page of traffic_segments. Relation attribute value is `previous`.

Segment structure

The Segment is the main element of the Segments collection.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
jurisdiction	Link	*Mandatory*	Link to the jurisdiction which published the announcement. In XML, the `rel` attribute needs the value `jurisdiction`.
description	text	*Optional*	Descript of this segment, indicating start and end points.
road	Road	*Mandatory*	The roadway on which the segment is located.
speed_limit	Numeric	*Optional*	Speed limit posted for the roadway where this segment is located.
lane_count	Numeric	*Optional*	Total number of lanes on the roadway where this segment is located.
toll	Boolean	*Optional*	Yes, if the segment requires a fee or toll to access.
type	Enum	*Mandatory*	Type of segment freeway, bridge, express lane etc.
start_node	Node	*Mandatory*	Start node for this segment.
end_node	Node	*Mandatory*	End node for this segment.
geography	geospatial	*Mandatory*	Geometry of the segment expressed in gml:linestring format.
restrictions	Collection of **Restriction**	*Optional*	Certain segments may have certain restrictions in terms of weight and height of vehicle allowed, content of vehicle passing this road segment. The restriction structure provides a way to provide the restrictions.
--restriction	Enum	*Optional*	Each restriction element provides a limitation on the segment.The restriction can be of type vehicle, speed, weight, length, width.

Sample request endpoint for Links

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/segments

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request cities for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.
Nodes	*Optional*	Clients should be able to request/filter Links that start or end at a node.
Road	*Optional*	Clients should be able to request links on a particular road.
City	*Optional*	Clients should be able to request links for a particular city.

The traffic segment response for XML is shown in Appendix A Section A.3.6 and for JSON is shown in Appendix B Section B.3.6.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual segment resource cannot be located). For e.g.: http://api.511.org/Open511/segment /22323

4.7 API: Segment_Conditons

Segment_Conditons provides traffic (speed, travel time, incident) and surface condition information along traffic segments.

Below is a message structure of segment_conditions.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
up	Link	*Mandatory*	Link to the parent resource.
segment_Conditions	**Collection of segment_Condition**	*Mandatory*	Can contain multiple segment_Conditions elements, at least one occurrence is mandatory.
pagination	Pagination structure	*Optional*	The pagination structure provides information about the current position in the pagination and links to other pages.
— offset	Integer	*Optional*	Offset used to access the current page.
— limit	Integer	*Optional*	Number of segment_conditions listed in the current page.
— next	Link	*Optional*	link pointing to the next page of segment_conditions. Relation attribute value is `next`.
— previous	Link	*Optional*	link pointing the previous page of segment_conditions. Relation attribute value is `previous`.

Segment_Condition Structure

The segment_condition is the main element of the segment_conditions collection.

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
jurisdiction	Link	*Mandatory*	Link to the jurisdiction which published the announcement. In XML, the `rel` attribute needs the value `jurisdiction`.
current_speed	Numeric	*Mandatory*	Current traffic Speed (miles/hr)
historical_speed	Numeric	*Optional*	Historical traffic speed (miles/hr) for the day and time the speed data requested for.
current_travel_time	Numeric	*Mandatory*	Current travel time for the day and time requested for.
historical_travel_time	Numeric	*Optional*	Historical travel time for the day and time requested for.
Surface_condiitons	Enum	*Optional*	Wet, Icy,slippery etc.
events	Collection of event	*Optional*	List of events that are associated with this segment.

Sample request endpoint for Links

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/Segments_Condition

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request cities for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.
traffic_segments	*Optional*	Clients should be able to request/filter speed and travel time for particular links.
day_time	*Optional*	Client should be able to request speed and travel time data for a specific day and time period. For e.g. Monday at 8:00 am. When this parameter is specified the api will return only historical speed and travel time information. If this parameter is not specified then both current and historical data for day and time identified by current time of the day will be provided.

The traffic segment conditions response for XML is shown in Appendix A Section A.3.7 and for JSON is shown in Appendix B Section B.3.7.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual link resource cannot be located). For e.g.: http://api.511.org/Open511/segment_conditions/22323

4.8 API: Reports

Reports are events that are submitted by citizen and sources that are external to the agency publishing data. Such reports are commonly considered as "crowdsourcing". Support for crowdsourcing is

optional. Crowdsourcing is not available via an aggregator; only the original jurisdiction can receive crowdsourced reports. A report can be posted by a user of the APIs and all reports can be retrieved.

Submit a Report Structure

Method: HTTP POST

Format: x-www-form-urlencoded (not JSON or XML)

The data should be posted to the report endpoint listed in the discovery (e.g. http://api.511.org/Open511/reports/)

Field	Type	Mandatory/ Optional	Description
jurisdiction	Free text	*Mandatory*	Identification of the jurisdiction.
Lat	Float	*Mandatory*	Latitude of the report using WGS84 projection.
Long	Float	*Mandatory*	Longitude of the report using WGS84 projection.
first_name	Free text	*Mandatory*	First name of the person submitting the report.
last_name	Free text	*Mandatory*	Last name of the person submitting the report.
Email	Free text	*Mandatory*	Email of the person submitting the report.
Phone	Free text	*Optional*	Phone number of the person submitting the report.
Device_id	Free text	*Optional*	Unique identifier for the device used to submit the report. Could be the MAC address of network interface or TelephonyManager.getDeviceId() Android device and [UIDevice currentDevice].uniqueIdentifier for iPhone device.
description	Free text	*Mandatory*	Description provided by the person who submitted the report.
event_type	Enum	*Optional*	Type of event following the same enumeration as the event_type in the event resource.
Severity	Enum	*Optional*	Severity of the event as judged by the person who reported it. This field use the same enumeration as the severity of the event resource.
Media	Binary	*Optional*	Inline media data (mainly an image) related to the event.

Parameters and Filters

The jurisdiction resource does not support any URL parameter or filter besides the format selection and the language negotiation.

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request cities for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

Example

```
POST /dev/v2/requests.xml
Host: api.city.gov
Content-Type: application/x-www-form-urlencoded; charset=utf-8

api_key=xyz&jurisdiction =city.gov& lat=37.76524078&long=-122.4212043&
&device_id=tt222111&first_name=john&last_name=smith&phone=111111111&description=Major+accid
ent+on+highway+680&event_type=ACCIDENT&severity=2
```

Retrieve Reports Structure

API client can retrieve all the reports submitted to a jurisdiction with a GET request on the reports resource.

The structure of a report resource consists of:

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
up	Link	*Mandatory*	Link to the parent resource.
Jurisdiction	Free text	*Mandatory*	Identification of the jurisdiction.
Location	Geometry	*Mandatory*	Location of the report.
description	Free text	*Mandatory*	Description provided by the person who submitted the report.
event_type	Enum	*Optional*	Type of event following the same enumeration as the event_type in the event resource.
Severity	Enum	*Optional*	Severity of the event as judged by the person who reported it. This field uses the same enumeration as the severity of the event resource.
Attachment	Attachment structure	*Optional*	Same structure as the attachment structure of the event resource.
Status	Enum	*Mandatory*	Status of the report: active or archived.
Creation_date	DateTime	*Mandatory*	Timestamp when the report was created.
Last_update	DateTime	*Mandatory*	Timestamp of the last update.
Comment	Free text	*Optional*	Comment from the jurisdiction/agency about the report.
Related_link	Link	*Optional*	Link pointing a related resource. As a general rule, the related link should be used to point an official event that would replace the report (e.g. the report is a duplicate or an official event has been created after the report).

Sample request endpoint for jurisdiction

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/reports

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual parking facility resource cannot be located). For e.g.: http://api.511.org/Open511/ParkingFacilities/930

5.0 Toll API

5.1 API: Toll_Program

The toll_program resource defines various toll programs within a jurisdiction. The toll_program resource provides informational data about a toll program.

Toll_Program message structure

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
jurisdiction	Link	*Mandatory*	Link to the jurisdiction under which this toll program exists. In XML, the `rel` attribute needs the value `jurisdiction`.
name	Free text	*Mandatory*	Name of the program (e.g.580 Express Lanes)
toll_authority	Free text	*Optional*	Name of the authority managing the program
toll_type	Enum	*Mandatory*	Pricing type: Value list: **Static** **Dynamic**
facility_type	**Enum**	*Mandatory*	Value List Freeway Bridge
toll_description	**Free Text**	*Optional*	Description text
toll_exemptions	Enum	*Mandatory*	Value list HOV-3+ Emergency vehicle None
vehicle_restrictions	Enum	*Mandatory*	Value list trailers trucks more than two axles
payment_options	Enum	*Mandatory*	Value list FastTrack Cash credit cards – Visa, Master card
toll_sections	Collection of toll_section	*Mandatory*	Defines the begin/start location of the toll. Also provides static and dynamic pricing information.

toll_section message structure

Field	Type	Mandatory/ Optional	Description
roadway	Road	*Mandatory*	Link to roadway resource.
barrier	Enum	*Optional*	Value list restricted entry & exit
from_node	Node	*Mandatory*	Node from where this toll section starts
to_node	Node	*Mandatory*	Node till where this toll section ends.

Sample request endpoint for toll_programs

Request Type	**GET** For e.g. http://api.511.org/Open511/ toll_programs
Request Endpoint Example	For e.g. http://api.511.org/Open511/reports

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request parking facilities for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

The toll program response for XML is shown in Appendix A Section A.4.1 and for JSON is shown in Appendix B Section B.4.1.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual toll_program resource cannot be located). For e.g.: http://api.511.org/Open511/toll_program/680expresslanes

5.2 Static_Toll

The static_toll resource provides pricing information about a toll program that is defined as type "static".

static_toll message structure

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
config	Link	*Mandatory*	Link to toll_program for configuration information
toll_pricing	Pricing element with fee values	*Mandatory*	Defines fees associcated with driving from a start node to multiple end nodes.

toll_pricing message structure

Field	Type	Mandatory/ Optional	Description
toll_from_node	Node	*Mandatory*	Link to start node.
toll_end_nodes	Collection of End node with corresponding fee	*Mandatory*	End_nodes (can have multiple end nodes)

toll_end_node message structure

Field	Type	Mandatory/ Optional	Description
to_node	Node	*Mandatory*	Link to end node
costs	Mutiple cost entries	*Mandatory*	Contains cost entriesbased on vehicle type, passenger_capacity, occupancy, emission and day type.

Toll price message structure

Field	Type	Mandatory/ Optional	Description
price_day_type	Enum	*Mandatory*	Value list M-F Weekdays Midweek Monday Tuesday Wednesday Thrusday Friday

			Saturday Weekends Holidays
occupancy	Enum	*Mandatory*	Value List Carpool (2) No Limit
passenger_capacity	Number	*Optional*	Number of passengers allowed
Vehicle_type	Enum	*Optional*	Value List 2 axle
emission	Enum	*Optional*	Value List Low High
time **-start** **--end**	Time	*Mandatory*	Time for e.g. 0900 to 0700.
price	Cost	*Decimal*	Cost in USD.

Sample request endpoint for toll_programs

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/Static_tolls

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request parking facilities for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.

toll_program	*Optional*	If a toll program is specified, the API will return static toll information for just that program.

The static toll response for XML is shown in Appendix A Section A.4.2 and for JSON is shown in Appendix B Section B.4.2.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual toll_program resource cannot be located). For e.g.: http://api.511.org/Open511/static_tolls /680expresslanes

5.3 Dynamic_Toll

The dynamic_toll resource provides pricing information about a toll program that is defined as type "dynamic".

dynamic_toll message structure

Field	Type	Mandatory/ Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
config	Link	*Mandatory*	Link to toll_program for configuration information
toll_pricing	Pricing element with fee values	*Mandatory*	Defines fee associcated with driving from a start node to multiple end nodes.
last_updated	Date time	*Mandatory*	The date and time data was last updated.

Dynamic toll_pricing message structure

Field	Type	Mandatory/ Optional	Description
toll_from_node	Node	*Mandatory*	Link to start node.
toll_end_nodes	Collection of End node with corresponding fee	*Mandatory*	End_nodes (can have multiple end nodes)
historical_time_slice **-day_type** **-hour** **-minute**	Defines the day and time for historical fee	*Conditional*	Should be provided if data is requested for specific day and time.

Dynamic toll_end_node message structure

Field	Type	Mandatory/ Optional	Description
to_node	Node	*Mandatory*	Link to end node
costs	Mutiple cost entries	*Mandatory*	Contains cost entries based on vehicle type, passenger_capacity, occupancy, emission and day type.

entry message structure

Field	Type	Mandatory/ Optional	Description
occupancy	Enum	*Optional*	Value List Carpool (2) No Limit
passenger_capacity	Number	*Optional*	Number of passengers allowed
Vehicle_type	Enum	*Optional*	Value List 2 axle
emission	Enum	*Optional*	Value List Low High
current_price	Cost	*Decimal*	Current Cost in USD
historical_price	Cost	*Decimal*	Historical Cost in USD

Sample request endpoint for toll_programs

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/dynamic_tolls

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request parking facilities for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Optional*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.
toll_program	*Optional*	If a toll program is specified, the API will return static toll information for just that program.

The dynamic toll response for XML is shown in Appendix A Section A.4.3 and for JSON is shown in Appendix B Section B.4.3.

Possible Errors

Listed below are HTTP status code and message returned for certain common errors:

- 500 - Internal Server Error (System has issues processing your request)
- 401 – Unauthorized (Invalid API key)
- 404 – Not found (If an individual toll_program resource cannot be located). For e.g.: http://api.511.org/Open511/dynamic_tolls /680expresslanes

6.0 Parking API

The core parking data resources consist of facilities and on-street parking locations. Open511provides message structure and API endpoint for accessing these resources. Open511 will also provide metadata/lookup information that can be used by consumers to filter and limit information during requests.

API: Parking Facility structure

The facility is the main element of the facilities collection.

Field	Type	Mandatory / Optional	Description
self	Link	*Mandatory*	Self link to the current resource.
jurisdiction	Link	*Mandatory*	Link to the jurisdiction which published the announcement. In XML, the `rel` attribute needs the value `jurisdiction`.
id	Numeric	*Mandatory*	Unique identifier for this parking facility.
Static	Static	*Mandatory*	Determines the Elements below are static variables.
parkingtype	Enum	*Mandatory*	Basic type of facility. Value list: **Open_lot** **Garage** **Permit** **Contract** **Free** **Pay** **On_street** **Other**
ownership_agency_type	Enum	*Optional*	Ownership agency type of facility: Value list: **Public** **Private** **Owned_by_transit** **Park_and_ride_lot** **Other**
lotName	Free text	*Mandatory*	**Name of the parking facility.**
city	**City** structure	*Mandatory*	City where facility is located.
neighborhood	FreeText	*Mandatory*	Neighborhood where facility is located.
onStreet Info	Free text	*Mandatory*	Street address or range of street numbers where facility is located.
atStreet Info	Free text	*Optional*	The cross streets closest to where the facility is located.
geometry	Geography field	*Mandatory*	Geographical information about the facility using the GML notation. Valid type of geometry

			can be points, multipoints, linestring, multilinestrings and polygon.
spacesTotal	Numeric	*Optional*	Total parking spaces at this facility.
percentFull	Numeric	*Optional*	Total percentage of parking space available. This element is only provided when requesting dynamic data or all.
availability	Numeric	*Optional*	Toal number of available parking space in the facility. This element is only provided when requesting dynamic data or all.
occupied	Numeric	*Optional*	Occupied parking spaces at this facility. This element is only provided when requesting dynamic data or all.
ContactphoneOnSite	Free text	*Optional*	Facility contact details. Element name depends on contact type. Possible Element names are:
			phone
OpertionalHours	Collection of **Optional hour**	*Optional*	List the hours of Operation
prices	Collection of **Entries**	*Optional*	A **price** element should be provided for each available period.

Operational Hours structure

Field	Type	Mandatory / Optional	Description
start	DateTime	*Mandatory*	Facility opening time on a particular day
end	DateTime	*Mandatory*	Facility come time on a particular day of week

Price structure

This defines prices that are in effect for a facility, on a specific day.

Field	Type	Mandatory / Optional	Description
start	DateTime	*Optional*	Time duration from when the price is active
end	DateTime	*Optional*	Time duration when the price is invalid
Entries	Collection of Entry	*Mandatory*	Facilites. Element name depends on spaceType. Possible Element names are: MotorCycle, Cars.
Discounts	Currency	*Optional*	Discount amount per day in decimal format e.g.: $2.50 and reson described in **reason**.

Entry structure

This defines prices that are in effect for a facility on a specific day.

Field	Type	Mandatory / Optional	Description
priceDayType	Enum	*Optional*	Day of week that this price is in effect. Value list: **Sunday** **Monday** **Tuesday** **Wednesday** **Thursday** **Friday** **Saturday** **All** **MondayToFriday** **MondayToFridayExceptHolidays** **Weekend** **MondayToThursday** **Weekly** **Monthly** **Holiday**
spacetype	Enum	*Mandatory*	Type of parking space. Value list: **Assigned** **Auto** **Bicycle** **Compact** **Covered** **Electric** **Handicap** **Monthly** **Motorcycle** **Open_air** **Standard**
time	Numeric	*Mandatory*	Number of minutes per interval.
priceMaximum	Currency	*Mandatory*	Maximum Dollar amount per day in decimal format eg: 2.50 charged every interval, described in **description**.
description	Free text	*Mandatory*	Text description containing time period and interval. "12:00 AM - 9:00 AM Per hour"

Sample request endpoint for Parking Facilities

Request Type	**GET**
Request Endpoint Example	For e.g. http://api.511.org/Open511/Parking/930?all=true

Parameters and Filters supported with the request

Parameter	Mandatory/ Optional	Description
format	*Optional*	The response format (json/xml) desired. If none specified, then default response would be JSON.
accept-language	*Optional*	If multiple languages are supported, this can be used to request data in desired language, If the jurisdiction doesn't support the response in requested language, response could be in default language selected by jurisdiction.
jurisdiction	*Optional*	Clients should be able to request parking facilities for a given jurisdiction on multi-jurisdiction endpoints or on aggregators.
api_key	*Mandatory*	If a jurisdiction/disseminator requires use of API key. An API key registration service will have to be provided by a Jurisdiction.
facility_id	*Optional*	Clients should be able to request details on a single parking facility.
real-time	*Optional*	Clients should be able to request real-time information on a single parking facility. If this parameter is not provided, only static data will be provided.
all	*Optional*	Clients should be able to request all information on a single parking facility. If this parameter is not provided, this is the default mode if real time value is not provided

The parking facility structure response for XML is shown in Appendix A Section A.5.1 and for JSON is shown in Appendix B Section B.5.1.

7.0 Recommendations

7.1 Data Encoding

Open511 defines below listed data encoding and format for data being exchanges.

- Character encoding must be in UTF-8
- DateTime and Date fields will follow the ISO 8601 subset as per the W3C note except when specified otherwise
- When a time zone has to be specified explicitly (in an independent field), it should use a time zone name from the TZ Database
- Geographical data will follow existing standards:
  - In XML, geometrics will use GML notation
  - In JSON, the geometrics will use GeoJSON notation
- Links will use the following rules
  - In XML, links will use a link tag and the IANA link relation list where relevant
  - In JSON, links will be regular attributes with their own names

7.1.1 Response Format: XML and JSON

In order to simplify integration in various contexts, Open511 supports two formats (or representations): XML and JSON. The client can specify the format they want to receive using the following options:

- HTTP Header Accept specifying the type of representation accepted. Currently, the values accepted are application/xml and application/json. The server must answer with the appropriate Content-Type header.
- URL parameter format can also be used (and will override the HTTP header if both are present). Accepted values are xml and json

Each implementation is free to determine its default representation when the client does not specify which format is to be received.

7.2 Bandwidth Optimization and Conditional Requests

Since the API will likely be used by mobile applications, bandwidth optimization is an important criterion. Clients and servers should support the following features:

- Conditional request with ETag (for the server) and If-None-Match (for the client) HTTP headers. See Configure ETags and Wikipedia.
- Compression with HTTP header Accept-Encoding and Content-Encoding is strongly encouraged

7.3 Authentication and Encryption Mechanisms

Authentication and encryption should be managed by the regular HTTP layer. Authentication and encryption are optional for each implementation within the following limits:

- Authentication: The only accepted authentication mechanism is HTTP Basic
- Encryption: The only accepted encryption is would be communication over HTTPS.

These limitations are set to allow for interoperability.

7.4 Response Language

Open511 has the ability to support multiple languages. The negotiation of the language can be done in two ways:

- HTTP Header Accept-Language specifies the language the client would like to receive.
- URL parameter Accept-Language can also be used (and will override the HTTP header if both are present) with the same values as the HTTP header.

In order to use the Accept-Language selector (either HTTP or URL parameter), the client provides a list of supported languages in order of priority. For example, pt, en to get Portuguese before English. The language values accepted are specified in RFC 1766. For simplicity, we recommend using the two-letter language-tag as defined by ISO-639-1

The response will contain information about the language. In XML, the xml:lang will provide the language used for the answer. In JSON, a custom attribute lang will specify the language used, including for event lists in several language.

Mixed language within an event (some fields in one language, some fields in another language) will not be supported. The XML representation should provide all the languages available for all the fields when no Accept-Language is provided (either in HTTP or URL). In this case, each free-text field must contain an xml:lang attribute.

7.5 Aggregation and Multi-jurisdiction Endpoint

The Open511 API is designed to support multi-jurisdiction endpoints (an endpoint that provides data for several jurisdictions) and aggregation of the content (an endpoint that gathers information from other endpoints). The last feature is meant to simplify the work of clients. For example a region can setup an aggregator for all the jurisdictions of the region, or a third party can build a service that serves a province or a country.

For the client, these features should be fairly transparent: endpoints will work the same way as described in the documentation. It is important to know that the jurisdiction links (in discovery and in events) must always point to the original resource (e.g, the original endpoint). This way, the client always has the opportunity to access the data of the original endpoint.

7.6 Access Mechanism

Open511 recommends a token based access mechanism to provide access to these APIs. Disseminators will be required to issue token keys to consumers before they can access these APIs Disseminators can also use token keys issued to consumers for identification and usage tracking. An online token registration tool should be provided.

7.7 API Rate and Throttling

In order to avoid excessive load and usage on system, access to APIs can be restricted by disseminators. When limits are imposed by disseminators and if throttle limits are exceeded by a consumer, appropriate messages and response code should be provided for subsequent requests. For e.g. if the Traffic_Segments_Speed API has a request limit of 60 requests/hour per key and if a consumer requests the API more than 60 times in a hour, the API response would comeback with 429 - Too Many Requests.

7.8 Version control

Open511 recommends versioning of APIs developed by a disseminator. Disseminators are also recommended to maintain backward compatibility with the previous version for a period that provides consumers of the previous version adequate time to react and update their interfaces. A disseminator can adopt any versioning approach as per their needs and experience. Whenever new versions are released, disseminators should proactively notify consumers. Open511 recommends versioning APIs such that the endpoint to a resource remains unchanged. For e.g. a disseminator can choose to allow consumers specify the version using a custom HTTP header and a possible URL parameter.

For e.g. request for http://api.511.org/Events can include a HTTP Accept header such as application/Open511+V1.

Or it can be accessed using http://api.511.org/Events?ver= Open511+V1

To avoid confusion and simplify discovery of these API, Open511 recommends releasing newer versions as one complete set. For e.g. if jurisdiction plans to roll out a new version of events API, it's recommend that all the other APIs should be upgraded to newer versions while maintaining the older version for a certain period of time.

7.9 Next steps?

Open511 specifications may add additional travel modes such as parking, ridesharing, bicycling, etc. and data services for existing modes. Modification and enhancement to this specification will follow a systematic versioning scheme.

Appendix A: API Response Messages- XML

Section 1 – General XML

A.1.1 Example Discovery Response (XML)

```
<Open511  xml:base="http://api.511.org/" xml:lang="en" version="v0">
    <jurisdictions>
        <jurisdiction>
             <id>511.org</id>
            <name>SF Bay Area</name>
            <link rel="self" href="http://api.511.org/Open511/SFBayArea"/>
        </jurisdiction>
    </jurisdictions>
    <services>
        <service>
            <service_type>EVENTS</service_type>
            <service_description>
                Provide information about events impacting the road system
            </service_description>
            <link rel="self" href="http://api.511.org/Open511/events"/>
            <supported_versions>
                <supported_version>1.0</supported_version>
                <supported_version>2.0</supported_version>
            </supported_versions>
        </service>
        <service>
            <service_type>Links</service_type>
            <service_description>
                Provide information about traffic links covered
            </service_description>
            <link rel="self" href="http://api.511.org/Open511/Links"/>
            <supported_versions>
                <supported_version>2.0</supported_version>
            </supported_versions>
        </service>
        <service>
            <service_type>Linkspeed</service_type>
            <service_description>
                Provide speed and travel time for traffic links covered
            </service_description>
            <link rel="self" href="http://api.511.org/Open511/LinkSpeed"/>
            <supported_versions>
                <supported_version>1.0</supported_version>
            </supported_versions>
        </service>
    </services>
</Open511>
```

A.1.2 Example Jurisdiction Response (XML)

```
<Open511 xmlns:gml="http://www.opengis.net/gml" xml:lang="en" version="v0">
        <jurisdiction>
        <!-- all fields not labeled as mandatory are optional -->
        <link rel="self" href="http://api.511.org/Open511/SFBayArea"/>
        <id>511.org</id>
        <!-- mandatory -->
        <name>Bay Area 511</name>
        <!-- mandatory -->
```

```
        <email>nahmed@mtc.ca.gov</email>
        <!-- mandatory -->
        <phone>+1 514-872-0000</phone>
         <description>Official traffic data from SF Bay 511</description>
        <link rel="description" href="http://api.511.org/Open511/presentation.html"/>
        <timezone>US/Western</timezone>
        <link rel="geography" href="/jurisdictions/my.city.gov/geography/"/>
        <link rel="license" href="http://my.city.gov/opendatalicense/"/>
        <languages>

            <language>en</language>
        </languages>
    </jurisdiction>
<link rel="self" href=" http://api.511.org/Open511/sfbayarea" />
  <link rel="up" href="http:///api.511.org/Open511/" />
</Open511>
```

A.1.3 Example Jurisdiction Geography Response (XML)

```
<open511
    xmlns:gml="http://www.opengis.net/gml"
    xml:lang="en"
    xml:base=" http://api.511.org/Open511/SFBayArea”
    version="v0"
>
  <geography>
    <gml:Polygon srsName="EPSG:4326">
      <gml:outerBoundaryIs>
        <gml:LinearRing>
          <gml:coordinates>-73.515580304999958,45.743558682000106 -
73.516801182999984,45.744177048000076 -73.516754262999996,45.745199120000088 -
73.516741493999973,45.745477517000076 -73.51918957800001,45.746719115000111 -
73.519343021999987,45.748003618000077 -73.520130775999974,45.748595939000076 -
73.520120826999971,45.749015964000108 -73.52193461600001,45.750027519000106 -
73.529479503999994,45.754234632000113 -73.527583232999973,45.777799543000107 -
73.515580304999958,45.743558682000106</gml:coordinates>
        </gml:LinearRing>
      </gml:outerBoundaryIs>
    </gml:Polygon>
  </geography>
  <link rel="self" href="/open511/SFBayArea/geography/" />
  <link rel="up" href="/open511/ SFBayArea/" />
</open511>
```

Section 2 – Transit XML

A.2.1 Example Transit Operator Response (XML.)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<siri:Siri xsi:schemaLocation="http://www.siri.org.uk/siri ../../../xsd/NeTEx_siri.xsd"
xmlns:siri="http://www.siri.org.uk/siri" xmlns="http://www.netex.org.uk/netex"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
version="1.0">
  <siri:ServiceDelivery>
    <siri:ResponseTimestamp>2012-12-17T09:30:46-05:00</siri:ResponseTimestamp>
    <DataObjectDelivery>
      <siri:ResponseTimestamp>2012-12-17T09:30:47.0Z</siri:ResponseTimestamp>
      <dataObjects>
        <ResourceFrame id="RF" version="any">
          <organisations>
            <Operator id="SF" version="any">
              <Name>Muni (San Francisco)</Name>
              <ShortName>Muni</ShortName>
              <SiriOperatorRef>SF-MUNI</SiriOperatorRef>
              <Locale>
            <TimeZone>America/Vancouver</TimeZone>
            <DefaultLanguage>en</DefaultLanguage>
           </Locale>
              <ContactDetails>
            <ContactTelephoneNumber>1-415-701-2311</ContactTelephoneNumber>
            <WebSite>http://www.sfmta.com/</WebSite>
           </ContactDetails>
              <PrimaryMode>bus</PrimaryMode>
              <PrivateCode>SF</PrivateCode>
              <Extensions>
```

```
                <Monitored>true</Monitored>
                <OtherModes>tram funicular</OtherModes>
                <Coverage>
                  <gml:Polygon srsName="EPSG:4326">
                    <gml:coordinates>
                      -71.17,47.33 -71.15,47.36 -71.10,47.35 -71.20,47.40
                    </gml:coordinates>
                  </gml:Polygon>
                </Coverage>
              </Extensions>
            </Operator>
            <Operator id="BA" version="any">
              <Name>BART</Name>
              <ShortName>BART</ShortName>
              <SiriOperatorRef>BART</SiriOperatorRef>
              <PrimaryMode>rail</PrimaryMode>
              <PrivateCode>BA</PrivateCode>
              <Extensions>
                <Monitored>true</Monitored>
                <Coverage>
                  <gml:LineString gml:id="l1" srsName="EPSG:4326">
                    <gml:coordinates>45.67, 88.56 55.56,89.44</gml:coordinates>
                 </gml:LineString>
                  <gml:LineString gml:id="l2" srsName="EPSG:4326">
                    <gml:coordinates>56.67, 84.56 65.56,95.44</gml:coordinates>
                 </gml:LineString>
                </Coverage>
              </Extensions>
            </Operator>
          </organisations>
        </ResourceFrame>
      </dataObjects>
    </DataObjectDelivery>
  </siri:ServiceDelivery>
</siri:Siri>
```


A.2.2 Example Transit Line Response (XML.)

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<siri:Siri xsi:schemaLocation="http://www.siri.org.uk/siri ../../../xsd/NeTEx_siri.xsd"
xmlns:siri="http://www.siri.org.uk/siri" xmlns="http://www.netex.org.uk/netex"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:atom="http://www.w3.org/2005/Atom"
version="1.0">
 <siri:ServiceDelivery>
  <siri:ResponseTimestamp>2012-12-17T09:30:46-05:00</siri:ResponseTimestamp>
  <DataObjectDelivery>
   <siri:ResponseTimestamp>2012-12-17T09:30:47.0Z</siri:ResponseTimestamp>
   <dataObjects>
    <ServiceFrame version="any"  id="SF">
     <lines>
      <Line version="any" id="BA:BAY PT/SFIA">
       <Name>Pittsburg/Bay Point to San Francisco International Airport</Name>
       <TransportMode>rail</TransportMode>
       <PublicCode></PublicCode>
       <SiriLineRef>722</SiriLineRef>
       <Monitored>true</Monitored>
       <OperatorRef ref="BA"/>
      </Line>
      <Line version="any" id="BA:DALY/DUBLIN">
       <Name>Daly City To Dublin/Pleasanton</Name>
       <TransportMode>rail</TransportMode>
       <PublicCode></PublicCode>
       <SiriLineRef>920</SiriLineRef>
       <Monitored>true</Monitored>
       <OperatorRef ref="BA"/>
      </Line>
     </lines>
    </ServiceFrame>
   </dataObjects>
  </DataObjectDelivery>
 </siri:ServiceDelivery>
</siri:Siri>
```

A.2.3 Example Transit Stop Response (XML.)


```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<siri:Siri xsi:schemaLocation="http://www.siri.org.uk/siri ../../../xsd/NeTEx_siri.xsd"
xmlns:siri="http://www.siri.org.uk/siri" xmlns="http://www.netex.org.uk/netex"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:atom="http://www.w3.org/2005/Atom" version="1.0">
  <siri:ServiceDelivery>
    <siri:ResponseTimestamp>2012-12-17T09:30:46-05:00</siri:ResponseTimestamp>
    <DataObjectDelivery>
      <siri:ResponseTimestamp>2012-12-17T09:30:47.0Z</siri:ResponseTimestamp>
      <dataObjects>
        <ServiceFrame version="any"  id="SF">
          <scheduledStopPoints>
            <ScheduledStopPoint version="any" id="SF:59921">
              <Name>The Embarcadero &amp; Broadway</Name>
              <Location>
                <Longitude>-122.061515</Longitude>
                <Latitude>37.699237</Latitude>
              </Location>
              <SiriStopPointRef>59921</SiriStopPointRef>
              <StopType>railStation</StopType>
            </ScheduledStopPoint>
            <ScheduledStopPoint version="any" id="SF:58777">
              <Name>The Embarcadero &amp; Green St</Name>
              <Location>
                <Longitude>-122.057358</Longitude>
                <Latitude>37.655923</Latitude>
              </Location>
              <SiriStopPointRef>58777</SiriStopPointRef>
              <StopType>railStation</StopType>
            </ScheduledStopPoint>
            <ScheduledStopPoint version="any" id="SB:98777">
              <Name>San Francisco Ferry Building</Name>
              <Location>
                <Longitude>-122.157358</Longitude>
                <Latitude>37.655923</Latitude>
              </Location>
              <SiriStopPointRef>98777</SiriStopPointRef>
              <StopType>ferryStop</StopType>
            </ScheduledStopPoint>
          </scheduledStopPoints>
          <stopAreas>
        <StopArea version="any" id="StopArea:SF1">
          <Name>MUNI stops at Embarcadero</Name>
          <members>
            <ScheduledStopPointRef version="any" ref="SF:59921"/>
            <ScheduledStopPointRef version="any" ref="SF:58777"/>
          </members>
          <ParentStopAreaRef version="any" ref="StopArea:MTC1"/>
        </StopArea>
        <StopArea version="any" id="StopArea:SB1">
          <Name>SF Bay Ferry stops at Embarcadero</Name>
          <members>
            <ScheduledStopPointRef version="any" ref="SB:98777"/>
          </members>
          <ParentStopAreaRef version="any" ref="StopArea:MTC1"/>
        </StopArea>
        <StopArea version="any" id="StopArea:MTC1">
          <Name>Regional transit stops at Embarcadero</Name>
```

```
        </StopArea>
         </stopAreas>
        </ServiceFrame>
      </dataObjects>
    </DataObjectDelivery>
  </siri:ServiceDelivery>
</siri:Siri>
```


A.2.4 Example Transit Stop Place Response (XML.)


```
<?xml version="1.0" encoding="iso-8859-1"?>
<siri:Siri xsi:schemaLocation="http://www.siri.org.uk/siri ../../../xsd/NeTEx_siri.xsd"
xmlns:siri="http://www.siri.org.uk/siri" xmlns="http://www.netex.org.uk/netex"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:atom="http://www.w3.org/2005/Atom" version="1.0">
  <siri:ServiceDelivery>
    <siri:ResponseTimestamp>2012-12-17T09:30:46-05:00</siri:ResponseTimestamp>
    <DataObjectDelivery>
      <siri:ResponseTimestamp>2012-12-17T09:30:47.0Z</siri:ResponseTimestamp>
      <dataObjects>
        <SiteFrame version="any"  id="SF">
       <stopPlaces>
         <StopPlace version="01" id="BA:12232">
    <Name>BART LAKE MERRIT</Name>
              <Description>800 Madison StreetOakland, CA 94607 (Between Madison St &amp;
Fallon St and 8th &amp; 9th)</Description>
          <Centroid>
                <Location>
                  <Longitude>-122.265668</Longitude>
                  <Latitude>37.797345</Latitude>
                </Location>
    </Centroid>
    <AccessibilityAssessment version="any" id="AccessibilityAssessment:BA:12232">
      <MobilityImpairedAccess>true</MobilityImpairedAccess>
      <limitations>
        <AccessibilityLimitation>
     <WheelchairAccess>true</WheelchairAccess>
   </AccessibilityLimitation>
      </limitations>
    </AccessibilityAssessment>
          <alternativeNames>
     <AlternativeName version="any" id="AlternativeName:BA:12232">          <Name>Lake
Merrit Station</Name>
     </AlternativeName>
          </alternativeNames>
              <PostalAddress version="any" id="PostalAddress:BA:12232">
                <AddressLine1>800 Madison St</AddressLine1>
                <Town>Oakland</Town>
              </PostalAddress>
              <Url></Url>
```

```
            <OperatorRef ref="BA"/>
            <adjacentSites>
              <ParkingRef ref="4234"/>
            </adjacentSites>
            <placeEquipments>
              <SanitaryEquipment version="any" id="123">
                <Description>RestRoom in upper level</Description>
              </SanitaryEquipment>
              <CycleStorageEquipment version="any" id="233">
                <Description>Bike Racks</Description>
                <CycleStorageType>racks</CycleStorageType>
                <NumberOfSpaces>4</NumberOfSpaces>
              </CycleStorageEquipment>
              <CycleStorageEquipment version="any" id="242">
                <Description>Bike Lockers</Description>
                <CycleStorageType>other</CycleStorageType>
                <NumberOfSpaces>10</NumberOfSpaces>
              </CycleStorageEquipment>
              <SignEquipment version="any" id="141">
                <Description>Information Display Board</Description>
              </SignEquipment>
              <EscalatorEquipment version="any" id="335">
                <Description>Escalator 335</Description>
              </EscalatorEquipment>
              <LiftEquipment version="any" id="312">
                <Description>Escalator 312</Description>
              </LiftEquipment>
              <ShelterEquipment version="any" id="12">
                <Description>Waiting area 1</Description>
              </ShelterEquipment>
              <SeatingEquipment version="any" id="4566">
                <Description>Bench near waiting area</Description>
              </SeatingEquipment>
            </placeEquipments>
    <PublicCode>1564</PublicCode>
    <TransportMode>rail</TransportMode>
    <StopPlaceType>railStation</StopPlaceType>
    <quays>
     <Quay version="any" id="543">
      <CompassOctant>W</CompassOctant>
     </Quay>
    </quays>
  </StopPlace>
      </stopPlaces>
         <parkings>
       <Parking version="any" id="4234">
  <Name>Lake Merritt BART Station Parking</Name>
  <Description>On Broadway, between 11th &amp; 14th</Description>
  <Centroid>
   <Location>
    <Longitude>-122.266382</Longitude>
    <Latitude>37.796615</Latitude>
         </Location>
  </Centroid>
            <PostalAddress version="any" id="PostalAddress:4234">
                <AddressLine1>800 Madison St</AddressLine1>
                <Town>Oakland</Town>
              </PostalAddress>
  <ParkingType>trainStationParking</ParkingType>
```

```
        <TotalCapacity>296</TotalCapacity>
            <RealTimeOccupancyAvailable>false</RealTimeOccupancyAvailable>
  <parkingAreas>
   <ParkingArea version="any" id="123">
    <Description>Accessible Parking</Description>
    <ParkingProperties>
           <ParkingUserType>registeredDisabled</ParkingUserType>
     <spaces>
       <ParkingCapacity version="any" id="ParkingCapacity:123">
  <NumberOfSpaces>10</NumberOfSpaces>
       </ParkingCapacity>
      </spaces>
     </ParkingProperties>
    </ParkingArea>
    <ParkingArea version="any" id="124">
     <Description>Reserved Parking</Description>
     <ParkingProperties>
      <ParkingUserType>reservationHolders</ParkingUserType>
      <spaces>
       <ParkingCapacity version="any" id="ParkingCapacity:124">
  <NumberOfSpaces>99</NumberOfSpaces>               </ParkingCapacity>
      </spaces>
            <charges>
              <tariffBands>
                <ParkingTariffChargeBand>
                  <Description>Single Day Reserved Parking</Description>
                  <MaximumStay>P1D</MaximumStay>
                  <Amount>4.50</Amount>
                </ParkingTariffChargeBand>
                <ParkingTariffChargeBand>
                  <Description>Monthly Reserved Parking</Description>
                  <MaximumStay>P1M</MaximumStay>
                  <Amount>100</Amount>
                </ParkingTariffChargeBand>
              </tariffBands>
            </charges>
     </ParkingProperties>
    </ParkingArea>
             </parkingAreas>
   </Parking>
 </parkings>
       </SiteFrame>
      </dataObjects>
    </DataObjectDelivery>
  </siri:ServiceDelivery>
</siri:Siri>
```


A.2.5 Example Transit Pattern Response (XML.)


```
<?xml version="1.0" encoding="iso-8859-1"?>
<siri:Siri xsi:schemaLocation="http://www.siri.org.uk/siri ../../../xsd/NeTEx_siri.xsd"
xmlns:siri="http://www.siri.org.uk/siri" xmlns="http://www.netex.org.uk/netex"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
version="1.0">
  <siri:ServiceDelivery>
    <siri:ResponseTimestamp>2012-12-17T09:30:46-05:00</siri:ResponseTimestamp>
```

```
    <DataObjectDelivery>
      <siri:ResponseTimestamp>2012-12-17T09:30:47.0Z</siri:ResponseTimestamp>
      <dataObjects>
        <ServiceFrame version="any"  id="SF">
       <directions>
        <Direction version="any" id="SC:900:N">
          <Name>North</Name>
        </Direction>
        <Direction version="any" id="SC:900:S">
          <Name>South</Name>
        </Direction>
       </directions>
       <journeyPatterns>
        <ServiceJourneyPattern version="any" id="SC:142402">
          <Name>LRT ALMADEN - OHLO02</Name>
          <DirectionRef version="any" ref="SC:900:N"/>
              <DestinationDisplayView>
                <FrontText>Towards OHLONE/CHYNOWETH STATION</FrontText>
              </DestinationDisplayView>
              <Extensions>
                <LineRef version="any" ref="SC:900"/>
              </Extensions>
          <pointsInSequence>
           <TimingPointInJourneyPattern version="any" order="1" id="SC:142402:1">
            <ScheduledStopPointRef version="any" ref="SC:64729"/>
                  <DestinationDisplayView>
                    <FrontText>To OHLONE/CHYNOWETH STATION</FrontText>
                  </DestinationDisplayView>
                </TimingPointInJourneyPattern>
           <StopPointInJourneyPattern version="any" order="2" id="SC:142402:2">
            <ScheduledStopPointRef version="any" ref="SC:64730"/>
           </StopPointInJourneyPattern>
           <TimingPointInJourneyPattern version="any" order="3" id="SC:142402:3">
            <ScheduledStopPointRef version="any" ref="SC:64733"/>
           </TimingPointInJourneyPattern>
              </pointsInSequence>
              <linksInSequence>
                <ServiceLinkInJourneyPattern>
                  <projections>
                    <LinkSequenceProjection>
                      <gml:LineString gml:id="l1" srsName="EPSG:4326">
                        <gml:coordinates>37.7112,-122.455945 37.70602,-122.46126
37.706613,-122.459196</gml:coordinates>
                      </gml:LineString>
                    </LinkSequenceProjection>
                  </projections>
                </ServiceLinkInJourneyPattern>
              </linksInSequence>
        </ServiceJourneyPattern>
       </journeyPatterns>
        </ServiceFrame>
      </dataObjects>
    </DataObjectDelivery>
  </siri:ServiceDelivery>
</siri:Siri>
```

A.2.6 Example Transit Route Response (XML.)


```xml
<?xml version="1.0" encoding="utf-8"?><siri:Siri xmlns="http://www.netex.org.uk/netex"
xmlns:atom="http://www.w3.org/2005/Atom" xmlns:siri="http://www.siri.org.uk/siri"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../../xsd/NeTEx_siri.xsd">
  <siri:ServiceDelivery>
   <siri:ResponseTimestamp>2012-12-17T09:30:46-05:00</siri:ResponseTimestamp>
   <DataObjectDelivery>
    <siri:ResponseTimestamp>2012-12-17T09:30:47.0Z</siri:ResponseTimestamp>
    <dataObjects>
     <CompositeFrame id="CF" version="1">
      <frames>
       <ServiceFrame id="SF" version="any">
        <routes>
         <Route id="BG:TIBURON:North:Weekday" version="any">
          <Name>Tiburon_North_Weekday</Name>
          <LineRef ref="BG:TIBURON" version="any" />
          <DirectionRef ref="BG:TIBURON:North" version="any" />
          <pointsInSequence>
           <PointOnRoute id="BG:TIBURON:North:Weekday:1" version="any">
            <PointRef ref="BG:4432" version="any" xsi:type="ScheduledStopPointRefStructure" />
           </PointOnRoute>
           <PointOnRoute id="BG:TIBURON:North:Weekday:2" version="any">
            <PointRef ref="BG:4433" version="any" xsi:type="ScheduledStopPointRefStructure" />
           </PointOnRoute>
           <PointOnRoute id="BG:TIBURON:North:Weekday:3" version="any">
            <PointRef ref="BG:4437" version="any" xsi:type="ScheduledStopPointRefStructure" />
           </PointOnRoute>
          </pointsInSequence>
         </Route>
         <Route id="BG:TIBURON:North:Weekend" version="any">
          <Name>Tiburon_North_Weekend</Name>
          <LineRef ref="BG:TIBURON" version="any" />
          <DirectionRef ref="BG:TIBURON:North" version="any" />
          <pointsInSequence>
           <PointOnRoute id="BG:TIBURON:North:Weekend:1" version="any">
            <PointRef ref="BG:4433" version="any" xsi:type="ScheduledStopPointRefStructure" />
           </PointOnRoute>
           <PointOnRoute id="BG:TIBURON:North:Weekend:2" version="any">
            <PointRef ref="BG:4437" version="any" xsi:type="ScheduledStopPointRefStructure" />
           </PointOnRoute>
          </pointsInSequence>
         </Route>
        </routes>
       </ServiceFrame>
       <ServiceCalendarFrame id="SC" version="any">
        <dayTypes>
         <DayType id="BG:Weekday" version="any">
          <Name>Weekday</Name>
          <properties>
           <PropertyOfDay>
            <DaysOfWeek>Monday Tuesday Wednesday Thursday Friday</DaysOfWeek>
           </PropertyOfDay>
          </properties>
         </DayType>
```

```
        <DayType id="BG:Weekend" version="any">
          <Name>Weekend</Name>
          <properties>
            <PropertyOfDay>
              <DaysOfWeek>Saturday Sunday</DaysOfWeek>
            </PropertyOfDay>
          </properties>
        </DayType>
      </dayTypes>
    </ServiceCalendarFrame>
    <TimetableFrame id="BG:TIBURON:North:Weekday" version="any">
      <Name>Tiburon_North_Weekday</Name>
      <frameValidityConditions>
        <AvailabilityCondition id="AC:BG:TIBURON:North:Weekday" version="any">
          <FromDate>2013-02-06T00:00:00Z</FromDate>
          <ToDate>2013-06-06T00:00:00Z</ToDate>
          <dayTypes>
            <DayTypeRef ref="BG:Weekday" version="any" />
          </dayTypes>
        </AvailabilityCondition>
      </frameValidityConditions>
      <vehicleJourneys>
        <ServiceJourney id="BG:11455" version="any">
          <SiriVehicleJourneyRef>11455</SiriVehicleJourneyRef>
          <JourneyPatternView>
            <ServiceJourneyPatternRef ref="BG:112333" version="any" />
            <RouteRef ref="BG:TIBURON:North:Weekday" version="any" />
            <DirectionRef ref="BG:TIBURON:North" version="any" />
          </JourneyPatternView>
          <calls>
            <Call order="1">
              <ScheduledStopPointRef ref="BG:4432" />
              <Arrival>
                <Time>06:05:00</Time>
              </Arrival>
              <Departure>
                <Time>06:05:00</Time>
              </Departure>
            </Call>
            <Call order="2">
              <ScheduledStopPointRef ref="BG:4437" />
              <Arrival>
                <Time>06:30:00</Time>
              </Arrival>
              <Departure>
                <Time>06:30:00</Time>
              </Departure>
            </Call>
          </calls>
        </ServiceJourney>
        <ServiceJourney id="BG:11456" version="any">
          <SiriVehicleJourneyRef>11456</SiriVehicleJourneyRef>
          <JourneyPatternView>
            <ServiceJourneyPatternRef ref="BG:112334" version="any" />
            <RouteRef ref="BG:TIBURON:North:Weekday" version="any" />
            <DirectionRef ref="BG:TIBURON:North" version="any" />
          </JourneyPatternView>
```

```
            <calls>
              <Call order="1">
                <ScheduledStopPointRef ref="BG:4433" />
                <Arrival>
                  <Time>10:10:00</Time>
                </Arrival>
                <Departure>
                  <Time>10:10:00</Time>
                </Departure>
              </Call>
              <Call order="2">
                <ScheduledStopPointRef ref="BG:4437" />
                <Arrival>
                  <Time>10:45:00</Time>
                </Arrival>
                <Departure>
                  <Time>10:45:00</Time>
                </Departure>
              </Call>
            </calls>
          </ServiceJourney>
        </vehicleJourneys>
      </TimetableFrame>
    </frames>
  </CompositeFrame>
 </dataObjects>
</DataObjectDelivery>
</siri:ServiceDelivery>
</siri:Siri>
```


A.2.7 Example Transit Day Type Response (XML.)


```
<?xml version="1.0" encoding="utf-8"?><siri:Siri xmlns="http://www.netex.org.uk/netex"
xmlns:atom="http://www.w3.org/2005/Atom" xmlns:siri="http://www.siri.org.uk/siri"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../../xsd/NeTEx_siri.xsd">
 <siri:ServiceDelivery>
  <siri:ResponseTimestamp>2012-12-17T09:30:46-05:00</siri:ResponseTimestamp>
   <DataObjectDelivery>
    <siri:ResponseTimestamp>2012-12-17T09:30:47.0Z</siri:ResponseTimestamp>
     <dataObjects>
      <ServiceCalendarFrame id="SC" version="any">
        <contentValidityConditions>
          <AvailabilityCondition id="CE:ServiceDays:1" version="any">
            <FromDate>2012-06-06T00:00:00Z</FromDate>
            <ToDate>2012-12-31T00:00:00Z</ToDate>
            <IsAvailable>true</IsAvailable>
          </AvailabilityCondition>
          <AvailabilityCondition id="CE:Holiday:12" version="any">
            <Description>Independence Day</Description>
            <FromDate>2012-07-04T00:00:00</FromDate>
            <ToDate>2012-07-04T24:00:00</ToDate>
            <IsAvailable>false</IsAvailable>
            <dayTypes>
              <DayTypeRef ref="CE:NoService" version="any" />
```

```
          </dayTypes>
        </AvailabilityCondition>
        <AvailabilityCondition id="CE:Holiday:13" version="any">
          <Description>Christmas Day</Description>
          <FromDate>2012-12-25T00:00:00</FromDate>
          <ToDate>2012-12-25T24:00:00</ToDate>
            <IsAvailable>false</IsAvailable>
          <dayTypes>
            <DayTypeRef ref="CE:Weekend" version="any" />
          </dayTypes>
        </AvailabilityCondition>
      </contentValidityConditions>
      <dayTypes>
        <DayType id="CE:Weekday" version="any">
          <Name>Weekday</Name>
          <properties>
            <PropertyOfDay>
              <DaysOfWeek>Monday Tuesday Wednesday Thursday Friday</DaysOfWeek>
            </PropertyOfDay>
          </properties>
        </DayType>
        <DayType id="CE:Weekend" version="any">
          <Name>Weekend</Name>
          <properties>
            <PropertyOfDay>
              <DaysOfWeek>Saturday Sunday</DaysOfWeek>
            </PropertyOfDay>
          </properties>
        </DayType>
        <DayType id="CE:NoService" version="any">
          <Name>No Service Day</Name>
        </DayType>
      </dayTypes>
    </ServiceCalendarFrame>
  </dataObjects>
 </DataObjectDelivery>
</siri:ServiceDelivery>
</siri:Siri>
```


A.2.8 Example Transit Announcement Response (XML.)


```
<?xml version="1.0" encoding="utf-8"?>
<Siri xmlns="http://www.siri.org.uk/siri" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.3" xsi:schemaLocation="http://www.siri.org.uk/siri
http://www.kizoom.com/standards/siri/schema/1.3/siri.xsd">
 <ServiceDelivery>
  <ResponseTimestamp>2013-02-14T16:05:51Z</ResponseTimestamp>
  <SituationExchangeDelivery version="1.3">
   <ResponseTimestamp>2013-02-14T16:05:51Z</ResponseTimestamp>
   <Situations>
    <PtSituationElement>
     <CreationTime>2013-02-14T16:00:01Z</CreationTime>
     <SituationNumber>734</SituationNumber>
     <Source>
      <SourceType>feed</ SourceType >
      <Name>MTC</Name>
     </Source>
     <ValidityPeriod>
```

```
          <StartTime>2013-02-14T16:00:00Z</StartTime>
          <EndTime>2013-02-14T18:00:00Z</EndTime>
        </ValidityPeriod>
        <Priority>1</Priority>
        <ScopeType>route</ScopeType>
        <Summary>Major BART Delay</Summary>
        <Description>On Thursday, February 14, at 4:00pm, BART reports a major delay on the Daly City
Line in the East Bay direction due to an equipment problem on a train.</Description>
        <InfoLinks>
          <InfoLink>
            <Uri>http://www.bart.gov/</Uri>
          </InfoLink>
                </InfoLinks>
        <Consequences>
          <Consequence>
            <Severity>severe</Severity>
            <Affects>
              <Operators>
                <AffectedOperator>
                  <OperatorRef>BA</OperatorRef>
                  <OperatorName>BART</OperatorName>
                </AffectedOperator>
              </Operators>
              <Networks>
                <AffectedNetwork>
                  <AffectedLine>
                    <LineRef>05099</LineRef>
                  </AffectedLine>
                </AffectedNetwork>
              </Networks>
              <StopPoints>
                <AffectedStopPoints>
                  <StopPointRef>198761</OperatorRef>
                  <StopPointRef>198762</OperatorRef>
                  <StopPointRef>198763</OperatorRef>
                  <StopPointRef>198764</OperatorRef>
                </AffectedStopPoints >
              </StopPoints>            </Affects>
          </Consequence>
        </Consequences>
      </PtSituationElement>
    </Situations>
  </SituationExchangeDelivery>
 </ServiceDelivery>
</Siri>
```


A.2.9 Example Response (XML) in SIRI ST format


```
<?xml version="1.0" encoding="UTF-8"?>
<Siri xmlns="http://www.siri.org.uk/siri" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.4" xsi:schemaLocation="http://www.siri.org.uk/siri
http://www.kizoom.com/standards/siri/schema/1.4/siri.xsd">
  <ServiceDelivery>
    <ResponseTimestamp>2013-02-18T09:30:47-08:00</ResponseTimestamp>
    <Status>true</Status>
    <StopTimetableDelivery version="1.4">
```

```
      <ResponseTimestamp>2004-12-17T09:30:47-05:00</ResponseTimestamp>
      <SubscriptionRef>3442</SubscriptionRef>
      <TimetabledStopVisit>
        <RecordedAtTime>2004-12-17T09:25:46-05:00</RecordedAtTime>
        <MonitoringRef>HLTST011</MonitoringRef>
        <TargetedVehicleJourney>
          <LineRef>17</LineRef>
          <DirectionRef>INBOUND</DirectionRef>
          <FramedVehicleJourneyRef>
            <DataFrameRef>2004-12-17</DataFrameRef>
            <DatedVehicleJourneyRef>TRP123214</DatedVehicleJourneyRef>
          </FramedVehicleJourneyRef>
          <PublishedLineName>Fremont</PublishedLineName>
          <OperatorRef>BART</OperatorRef>
          <OriginRef>BART_11</OriginRef>
          <OriginName>BART_CIVIC CENTER</OriginName>
          <DestinationRef>BART_99</DestinationRef>
          <DestinationName>BART_16th St-Mission</DestinationName>
          <TargetedCall>
            <VisitNumber>1</VisitNumber>
            <AimedArrivalTime>2013-02-18T09:30:47-08:00</AimedArrivalTime>
            <AimedDepartureTime>2013-02-18T09:31:47-08:00</AimedDepartureTime>
          </TargetedCall>
        </TargetedVehicleJourney>
      </TimetabledStopVisit>
      <TimetabledStopVisit>
        <RecordedAtTime>2004-12-17T09:25:46-05:00</RecordedAtTime>
        <MonitoringRef>HLTST011</MonitoringRef>
        <TargetedVehicleJourney>
          <LineRef>17</LineRef>
          <DirectionRef>INBOUND</DirectionRef>
          <FramedVehicleJourneyRef>
            <DataFrameRef>2004-12-17</DataFrameRef>
            <DatedVehicleJourneyRef>TRP544514</DatedVehicleJourneyRef>
          </FramedVehicleJourneyRef>
          <PublishedLineName>Fremont</PublishedLineName>
          <OperatorRef>BART</OperatorRef>
          <OriginRef>BART_11</OriginRef>
          <OriginName>BART_CIVIC CENTER</OriginName>
          <DestinationRef>BART_99</DestinationRef>
          <DestinationName>BART_16th St-Mission</DestinationName>
          <TargetedCall>
            <VisitNumber>2</VisitNumber>
            <AimedArrivalTime>2013-02-18T09:45:47-08:00</AimedArrivalTime>
            <AimedDepartureTime>2013-02-18T09:46:47-08:00</AimedDepartureTime>
          </TargetedCall>
        </TargetedVehicleJourney>
>
      </TimetabledStopVisit>
    </StopTimetableDelivery>
  </ServiceDelivery>
</Siri>
```


A.2.10 Example Transit Real Time Departure Service Delivery Mode Response (XML) in SIRI format

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Siri xmlns="http://www.siri.org.uk/siri" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.4" xsi:schemaLocation="http://www.siri.org.uk/siri
http://www.kizoom.com/standards/siri/schema/1.4/siri.xsd">
  <ServiceDelivery>
    <ResponseTimestamp>2004-12-17T09:30:46-05:00</ResponseTimestamp>
    <ProducerRef>BART</ProducerRef>
    <Status>true</Status>
    <StopMonitoringDelivery version="1.4">
      <ResponseTimestamp>2004-12-17T09:30:47-05:00</ResponseTimestamp>
      <Status>true</Status>
      <MonitoredStopVisit>
        <RecordedAtTime>2004-12-17T09:25:46-05:00</RecordedAtTime>
        <MonitoringRef>BART_11</MonitoringRef>
        <MonitoredVehicleJourney>
          <LineRef>17</LineRef>
          <DirectionRef>Out</DirectionRef>
          <FramedVehicleJourneyRef>
            <DataFrameRef>2004-12-17</DataFrameRef>
            <DatedVehicleJourneyRef>Oubound</DatedVehicleJourneyRef>
          </FramedVehicleJourneyRef>
          <PublishedLineName>Fremont</PublishedLineName>
          <OperatorRef>BART</OperatorRef>
          <Monitored>true</Monitored>
          <VehicleLocation>
            <Longitude>180</Longitude>
            <Latitude>90</Latitude>
          </VehicleLocation>
          <ProgressStatus>Service running on time</ProgressStatus>
          <PreviousCalls>
            <PreviousCall>
              <StopPointRef>BART_10</StopPointRef>
              <VisitNumber>2</VisitNumber>
              <StopPointName>BART_DALY CITY</StopPointName>
              <VehicleAtStop>false</VehicleAtStop>
              <AimedDepartureTime>2004-12-17T09:32:43-05:00</AimedDepartureTime>
              <ActualDepartureTime>2004-12-17T09:32:43-05:00</ActualDepartureTime>
            </PreviousCall>
          </PreviousCalls>
          <MonitoredCall>
            <VisitNumber>0014</VisitNumber>
            <VehicleAtStop>false</VehicleAtStop>
            <VehicleLocationAtStop>
              <Longitude>180</Longitude>
              <Latitude>90</Latitude>
            </VehicleLocationAtStop>
            <AimedArrivalTime>2004-12-17T09:40:46-05:00</AimedArrivalTime>
            <ExpectedArrivalTime>2004-12-17T09:40:46-05:00</ExpectedArrivalTime>
            <AimedDepartureTime>2004-12-17T09:42:47-05:00</AimedDepartureTime>
            <ExpectedDepartureTime>2004-12-17T09:40:47-05:00</ExpectedDepartureTime>
          </MonitoredCall>
          <OnwardCalls>
            <OnwardCall>
              <StopPointRef>BART_12</StopPointRef>
              <VisitNumber>4</VisitNumber>
              <StopPointName>BAR_12th St Oakland</StopPointName>
              <VehicleAtStop>false</VehicleAtStop>
```

```
            <AimedArrivalTime>2004-12-17T09:30:56-05:00</AimedArrivalTime>
            <ExpectedArrivalTime>2004-12-17T09:30:56-05:00</ExpectedArrivalTime>
            <AimedDepartureTime>2004-12-17T09:30:57-05:00</AimedDepartureTime>
            <ExpectedDepartureTime>2004-12-17T09:30:57-05:00</ExpectedDepartureTime>
          </OnwardCall>
        </OnwardCalls>
      </MonitoredVehicleJourney>
    </MonitoredStopVisit>
    <MonitoredStopVisitCancellation>
      <RecordedAtTime>2004-12-17T09:30:47-05:00</RecordedAtTime>
      <ItemRef>SED9843214675429</ItemRef>
      <Reason>Arrived</Reason>
    </MonitoredStopVisitCancellation>
    <MonitoredStopVisitCancellation>
      <RecordedAtTime>2004-12-17T09:30:47-05:00</RecordedAtTime>
      <MonitoringRef>BART_11</MonitoringRef>
      <VisitNumber>2</VisitNumber>
      <LineRef>Line123</LineRef>
      <DirectionRef>Out</DirectionRef>
      <VehicleJourneyRef>
        <DataFrameRef>2004-12-17</DataFrameRef>
        <DatedVehicleJourneyRef>0987656</DatedVehicleJourneyRef>
      </VehicleJourneyRef>
      <Reason>Arrived</Reason>
    </MonitoredStopVisitCancellation>
    <StopLineNotice>
      <RecordedAtTime>2004-12-17T09:30:47-05:00</RecordedAtTime>
      <ItemIdentifier>SED9843214675429</ItemIdentifier>
      <MonitoringRef>BART_11</MonitoringRef>
      <LineRef>123</LineRef>
      <DirectionRef>Out</DirectionRef>
      <LineNote>Mechanical Problems on Track</LineNote>
    </StopLineNotice>
    <StopLineNoticeCancellation>
      <RecordedAtTime>2004-12-17T09:30:47-05:00</RecordedAtTime>
      <ItemRef>SED9843214675429</ItemRef>
      <MonitoringRef>BART_11</MonitoringRef>
      <LineRef>123</LineRef>
      <DirectionRef>Out</DirectionRef>
    </StopLineNoticeCancellation>
    <Note>Hello Stop</Note>
  </StopMonitoringDelivery>
</ServiceDelivery>
</Siri>
```


A.2.11 Example Real Time Vehicle Monitoring Response (XML) in SIRI format


```
<?xml version="1.0" encoding="UTF-8"?>
<Siri xmlns="http://www.siri.org.uk/siri" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.4" xsi:schemaLocation="http://www.siri.org.uk/siri
http://www.kizoom.com/standards/siri/schema/1.4/siri.xsd">
  <ServiceDelivery>
    <ResponseTimestamp>2004-12-17T09:30:47-05:00</ResponseTimestamp>
    <ProducerRef>BART</ProducerRef>
    <Status>true</Status>
    <VehicleMonitoringDelivery version="1.4">
        <ResponseTimestamp>2004-12-17T09:30:47-05:00</ResponseTimestamp>
```

```
<VehicleActivity>
  <RecordedAtTime>2004-12-17T09:30:47-05:00</RecordedAtTime>
  <ValidUntilTime>2004-12-17T09:30:47-05:00</ValidUntilTime>
  <MonitoredVehicleJourney>
    <LineRef>17</LineRef>
    <DirectionRef>OUT</DirectionRef>
    <FramedVehicleJourneyRef>
      <DataFrameRef>2004-12-17</DataFrameRef>
      <DatedVehicleJourneyRef>987675</DatedVehicleJourneyRef>
    </FramedVehicleJourneyRef>
    <PublishedLineName>123</PublishedLineName>
    <OperatorRef>BART</OperatorRef>
    <OriginName>SFO</OriginName>
    <Via>
      <PlaceName>16th st</PlaceName>
    </Via>
    <Via>
      <PlaceName>West Oakland</PlaceName>
    </Via>
    <DestinationRef>Fremont</DestinationRef>
    <DestinationName>Fremont</DestinationName>
    <Monitored>true</Monitored>
    <InCongestion>false</InCongestion>
    <VehicleLocation>
      <Longitude>180</Longitude>
      <Latitude>90</Latitude>
    </VehicleLocation>
    <Bearing>123</Bearing>
    <ProgressRate>slowProgress</ProgressRate>
    <Delay>PT2M</Delay>
    <ProgressStatus>On time</ProgressStatus>
    <VehicleRef>VEH987654</VehicleRef>
    <PreviousCalls>
      <PreviousCall>
        <StopPointRef>SFO</StopPointRef>
        <VisitNumber>2</VisitNumber>
        <StopPointName>String</StopPointName>
        <VehicleAtStop>false</VehicleAtStop>
        <AimedDepartureTime>2004-12-17T09:32:43-05:00</AimedDepartureTime>
        <ActualDepartureTime>2004-12-17T09:32:43-05:00</ActualDepartureTime>
      </PreviousCall>
    </PreviousCalls>
    <OnwardCalls>
      <OnwardCall>
        <StopPointRef>80</StopPointRef>
        <VisitNumber>4</VisitNumber>
        <StopPointName>16th Street</StopPointName>
        <VehicleAtStop>false</VehicleAtStop>
        <AimedArrivalTime>2004-12-17T09:30:56-05:00</AimedArrivalTime>
        <ExpectedArrivalTime>2004-12-17T09:30:56-05:00</ExpectedArrivalTime>
        <AimedDepartureTime>2004-12-17T09:30:57-05:00</AimedDepartureTime>
        <ExpectedDepartureTime>2004-12-17T09:30:57-05:00</ExpectedDepartureTime>
      </OnwardCall>
    </OnwardCalls>
  </MonitoredVehicleJourney>
</VehicleActivity>
<VehicleActivity>
```

```
        <RecordedAtTime>2004-12-17T09:30:47-05:00</RecordedAtTime>
        <ValidUntilTime>2004-12-17T09:30:47-05:00</ValidUntilTime>
        <VehicleMonitoringRef>45678</VehicleMonitoringRef>
        <MonitoredVehicleJourney>
          <LineRef>Line123</LineRef>
          <FramedVehicleJourneyRef>
            <DataFrameRef>2004-12-17</DataFrameRef>
            <DatedVehicleJourneyRef>Outbound</DatedVehicleJourneyRef>
          </FramedVehicleJourneyRef>
          <Monitored>true</Monitored>
          <VehicleLocation>
            <Longitude>180</Longitude>
            <Latitude>90</Latitude>
          </VehicleLocation>
          <Delay>PT2M</Delay>
          <VehicleRef>VEH987654</VehicleRef>
          <OnwardCalls>
            <OnwardCall>
              <StopPointRef>HLTST012</StopPointRef>
              <StopPointName>Church</StopPointName>
            </OnwardCall>
          </OnwardCalls>
        </MonitoredVehicleJourney>
      </VehicleActivity>
      <VehicleActivityCancellation>
        <RecordedAtTime>2004-12-17T09:30:47-05:00</RecordedAtTime>
        <VehicleMonitoringRef>9876542</VehicleMonitoringRef>
        <VehicleJourneyRef>
          <DataFrameRef>2001-12-17</DataFrameRef>
          <DatedVehicleJourneyRef>09867</DatedVehicleJourneyRef>
        </VehicleJourneyRef>
        <LineRef>Line123</LineRef>
        <DirectionRef>Out</DirectionRef>
        <Reason>Done for the day</Reason>
      </VehicleActivityCancellation>
    </VehicleMonitoringDelivery>
  </ServiceDelivery>
</Siri>
```


A.2.12 Example Transit Schedule Update Response (XML) in SIRI PT format


```
<?xml version="1.0" encoding="UTF-8"?>
<Siri xmlns="http://www.siri.org.uk/siri" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.4" xsi:schemaLocation="http://www.siri.org.uk/siri
http://www.kizoom.com/standards/siri/schema/1.4/siri.xsd">
  <ServiceDelivery>
    <ResponseTimestamp>2013-02-18T09:30:47-08:00</ResponseTimestamp>
    <Status>true</Status>
    <ProductionTimetableDelivery version="1.4">
      <ResponseTimestamp>2004-12-17T09:30:47-05:00</ResponseTimestamp>
      <ValidUntil>2001-12-17T10:30:47-05:00</ValidUntil>
      <DatedTimetableVersionFrame>
        <RecordedAtTime>2001-12-17T09:30:47-05:00</RecordedAtTime>
        <LineRef>123</LineRef>
        <DirectionRef>Out</DirectionRef>
        <PublishedLineName>String</PublishedLineName>
        <DatedVehicleJourney>
```

```
        <DatedVehicleJourneyCode>DVC0008767</DatedVehicleJourneyCode>
        <DatedCalls>
          <DatedCall>
            <StopPointRef>BART_11</StopPointRef>
            <CallNote>optional message here</CallNote>
            <AimedArrivalTime>2013-02-19T09:55:47-08:00</AimedArrivalTime>
            <AimedDepartureTime>2013-02-19T09:56:47-08:00</AimedDepartureTime>
          </DatedCall>
          <DatedCall>
            <StopPointRef>BART_99</StopPointRef>
            <CallNote>optional message here</CallNote>
            <AimedArrivalTime>2013-02-19T10:15:47-08:00</AimedArrivalTime>
            <AimedDepartureTime>2013-02-19T10:16:47-08:00</AimedDepartureTime>
          </DatedCall>
        </DatedCalls>
      </DatedVehicleJourney>
    </DatedTimetableVersionFrame>
  </ProductionTimetableDelivery>
</ServiceDelivery>
</Siri>
```


A.2.13 Example Transit Addition and Cancellation of Trip Response (XML) in SIRI ET format


```
<?xml version="1.0" encoding="UTF-8"?>
<Siri xmlns="http://www.siri.org.uk/siri" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.4" xsi:schemaLocation="http://www.siri.org.uk/siri
http://www.kizoom.com/standards/siri/schema/1.4/siri.xsd">
  <ServiceDelivery>
    <ResponseTimestamp>2013-02-18T09:30:47-08:00</ResponseTimestamp>
    <Status>true</Status>
    <EstimatedTimetableDelivery version="1.4">
      <ResponseTimestamp>2004-12-17T09:30:47-05:00</ResponseTimestamp>
      <EstimatedJourneyVersionFrame>
        <RecordedAtTime>2013-02-18T09:30:47-08:00</RecordedAtTime>
        <EstimatedVehicleJourney>
          <LineRef>917</LineRef>
          <DirectionRef>INBOUND</DirectionRef>
          <DatedVehicleJourneyRef>00008</DatedVehicleJourneyRef>
          <Cancellation>false</Cancellation>
          <PublishedLineName>Fremont</PublishedLineName>
          <EstimatedCalls>
            <EstimatedCall>
              <StopPointRef>BART_11</StopPointRef>
              <CallNote>optional message here</CallNote>
              <AimedArrivalTime>2013-02-19T09:55:47-08:00</AimedArrivalTime>
              <AimedDepartureTime>2013-02-19T09:56:47-08:00</AimedDepartureTime>
            </EstimatedCall>
            <EstimatedCall>
              <StopPointRef>BART_99</StopPointRef>
              <CallNote>optional message here</CallNote>
              <AimedArrivalTime>2013-02-19T10:15:47-08:00</AimedArrivalTime>
              <AimedDepartureTime>2013-02-19T10:16:47-08:00</AimedDepartureTime>
            </EstimatedCall>
          </EstimatedCalls>
        </EstimatedVehicleJourney>
```

```
        <EstimatedVehicleJourney>
          <LineRef>764</LineRef>
          <DirectionRef>INBOUND</DirectionRef>
          <DatedVehicleJourneyRef>00008</DatedVehicleJourneyRef>
          <Cancellation>true</Cancellation>
          <PublishedLineName>Pittsburgh Bay Point</PublishedLineName>
        </EstimatedVehicleJourney>
      </EstimatedJourneyVersionFrame>
    </EstimatedTimetableDelivery>
  </ServiceDelivery>
</Siri>
```


A.2.14 Example Transit General Messaging Service Response (XML) in SIRI GM format


```
<?xml version="1.0" encoding="UTF-8"?>
<Siri xmlns="http://www.siri.org.uk/siri" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.4" xsi:schemaLocation="http://www.siri.org.uk/siri
http://www.kizoom.com/standards/siri/schema/1.4/siri.xsd">
  <ServiceDelivery>
    <ResponseTimestamp>2013-02-17T09:30:46-08:00</ResponseTimestamp>
    <Status>true</Status>
    <GeneralMessageDelivery version="1.4">
      <ResponseTimestamp>2001-12-17T09:30:47.0Z</ResponseTimestamp>
      <GeneralMessage formatRef="string">
        <RecordedAtTime>2013-02-17T09:30:46-08:00</RecordedAtTime>
        <InfoMessageIdentifier>12345</InfoMessageIdentifier>
        <InfoMessageVersion>2</InfoMessageVersion>
        <InfoChannelRef>WARNINGS</InfoChannelRef>
        <ValidUntilTime>2013-02-18T09:30:46-08:00</ValidUntilTime>
        <Content>some message here</Content>
      </GeneralMessage>
      <GeneralMessage formatRef="string">
        <RecordedAtTime>2013-02-17T09:30:46-08:00</RecordedAtTime>
        <InfoMessageIdentifier>23456</InfoMessageIdentifier>
        <InfoMessageVersion>1</InfoMessageVersion>
        <InfoChannelRef>WARNINGS</InfoChannelRef>
        <ValidUntilTime>2013-02-18T09:30:46-08:00</ValidUntilTime>
        <Content>some message here</Content>
      </GeneralMessage>
    </GeneralMessageDelivery>
  </ServiceDelivery>
</Siri>
```

Section 3 – Traffic XML

A.3.1 Example Traffic Event Structure Response (XML)


```
<Open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.Open511.info/">
    <Events>
        <atom:link rel="self" href="http://api.511.org/Open511/events"/>
        <atom:link rel="up" href="http://api.511.org/Open511"/>
        <roadEvent>
            <status>active</status>
            <atom:link rel="self" href="/events/23948/"/>
            <atom:link rel="jurisdiction" href="http://api.511.org/Open511/SFBayArea/"/>
            <headline>
                Roadwork on I-80 E Eastbound between 4TH St (San Francisco) and I-880 S
            </headline>
            <headline xml:lang="fr">
                Réfection d'urgence d'une conduite d'égout
            </headline>
            <description>
                : Roadwork on I-80 E Eastbound between 4TH St (San Francisco) and I-880
S. Left lane closed until Noon
            </description>
            <description xml:lang="fr">
                La Ville de Montréal vous informe que la Grande avenue sera fermée à la
circulation entre la 1ère et la 4ème avenue pour le mois d'avril pour cause de travaux
d'urgence sur une conduite d'égouts. La circulation locale en direction de l'est sera
permise sur une voie.
            </description>
            <eventType>Incident</eventType>
            <eventSubType>Urgent roadwork</eventSubType>
            <severity>2</severity>
            <certainty>Observed</certainty>
            <creationDate>2012-05-23T20:33:10Z</creationDate>
            <lastUpdate>2012-05-24T10:00:10Z</lastUpdate>
            <detour>Take Sherbrooke travelling east, and Wellington travelling
west.</detour>
            <travelerMessage>Avoid-region.</travelerMessage>
            <geometry>
                <gml:Linestring srsName="EPSG:4326">
                    <gml:coordinates>
                        -71.17,47.33 -71.15,47.36 -71.10,47.35 -71.20,47.40
                    </gml:coordinates>
                </gml:Linestring>
            </geometry>
            <roadaffected>
                <Fromroad>Broadway St</Fromroad>
                <ToRoad>1st avenue</ToRoad>
                <OnRoad>11th st</OnRoad>
                <atom:link rel="Road" href="http://api.511.org/Open511/Roads/1567"/>
                <direction>E</direction>
                <laneStatus>10</laneStatus>
                <impactedSystems>
                    <impactedSystem>Road</impactedSystem>
                    <impactedSystem>On-Street Parking</impactedSystem>
                </impactedSystems>
                <restrictions>
                    <restriction>Local Traffic Only</restriction>
                </restrictions>
```

```
            </roadaffected>
            <schedule>
                <startDate>2012-04-01</startDate>
                <endDate>2012-04-30</endDate>
            </schedule>
            <attachment>
                <startDate>2012-04-01</startDate>
                <endDate>2012-04-30</endDate>
            </attachment>
            <atom:link rel="enclosure" type="application/pdf"
href="http://my.city.gov/trafic/advisory/39473/com.pdf" length="200000"/>
        </roadEvent>
    </Events>
</Open511>
```

A.3.2 Example Traffic Announcment Response (XML)

```
<Open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.Open511.info/">
 <Announcements>
  <atom:link rel="self" href="http://api.511.org/Open511/Announcements"/>
  <atom:link rel="up" href="http://api.511.org/Open511"/>
  <Announcement>
   <status>active</status>
   <atom:link rel="self" href="/announcements/23948/"/>
   <atom:link rel="jurisdiction" href="http://api.511.org/Open511/SFBayArea/"/>
   <headline>
    Bart to run extra trains during world series.
   </headline>
   <description>
    Bart to run extra trains during world series.
   </description>
   <Type>Traffic</Type>
   <creation_date>2012-05-23T20:33:10Z</creation_date>
   <last_update>2012-05-24T10:00:10Z</last_update>
  </Announcement>
 </Announcements>
</Open511>
```

A.3.3 Example Traffic Area Response (XML)

```
<open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.open511.info/">
 <area>
  <atom:link rel="self" href="http://api.511.org/Open511/area"/>
  <atom:link rel="up" href="http://api.511.org/Open511"/>
  <area>
   <atom:link rel="self" href="/area/1234"/>
   <name>San Francisco</name>
   <type>City</type>
   <geometry>
    <gml:Polygon srsName="EPSG:4326">
     <gml:LinearRing>
      <gml:coordinates decimal="." cs="," ts="">119.593002319336,-31.6695003509522
119.595306396484,
                31.6650276184082 119.600944519043,-31.6658897399902
119.603385925293,-31.669527053833
                119.60050201416,-31.6739158630371 119.595664978027,-31.6728610992432
119.593002319336,
                31.6695003509522</gml:coordinates>
```

```
        </gml:LinearRing>
      </gml:Polygon>
    </geometry>
  </area>
  <area>
    <atom:link rel="within" href="/area/1235"/>
    <name>SOMA</name>
    <type>Neighborhood</type>
    <area_within>
      <atom:link rel="within" href="/area/1234"/>
    </area_within>
    <geometry>
      <gml:Polygon srsName="EPSG:4326">
        <gml:LinearRing>
          <gml:coordinates decimal="." cs="," ts="">119.593002319336,-31.6695003509522
119.595306396484,
                    31.6650276184082 119.600944519043,-31.6658897399902
119.603385925293,-31.669527053833
                    119.60050201416,-31.6739158630371 119.595664978027,-31.6728610992432
119.593002319336,
                    31.6695003509522</gml:coordinates>
        </gml:LinearRing>
      </gml:Polygon>
    </geometry>
  </area>
 </area>
</open511>
```


A.3.4 Traffic Road Example Response (XML)


```
<open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.open511.info/">
 <roads>
  <atom:link rel="self" href="http://api.511.org/Open511/roads"/>
  <atom:link rel="up" href="http://api.511.org/Open511"/>
  <road>
    <atom:link rel="self" href="/roads/I880S/"/>
    <atom:link rel="jurisdiction" href="http://api.511.org/Open511/SFBayArea/"/>
    <name>I880S</name>
    <direction>S</direction>
    <type>Freeway</type>
    <segments>
      <atom:link rel="segment" href="http://api.511.org/open511/segment/1567"/>
      <atom:link rel="segment" href="http://api.511.org/open511/segment/1568"/>
    </segments>
  </road>
 </roads>
</open511>
```


A.3.5 Example Traffic Node Response (XML)


```
<open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.open511.info/">
 <Nodes>
  <atom:link rel="self" href="http://api.511.org/Open511/Nodes"/>
  <atom:link rel="up" href="http://api.511.org/Open511"/>
  <Node>
    <atom:link rel="self" href="/Nodes/12345/"/>
    <atom:link rel="jurisdiction" href="http://api.511.org/Open511/SFBayArea/"/>
    <Name>80 E &amp;Fremont St</Name>
```

```
    <roads>
      <atom:link rel="road" href="http://api.511.org/Open511/roads/I880S"/>
      <atom:link rel="road" href="http://api.511.org/Open511/roads/Mowry"/>
    </roads>
    <geometry>
      <gml:point srsName="EPSG:4326">
        <gml:coordinates>-37.819006,-122.267532</gml:coordinates>
      </gml:point>
    </geometry>
  </Node>
  <Node>
    <atom:link rel="self" href="/Nodes/12346/"/>
    <atom:link rel="jurisdiction" href="http://api.511.org/Open511/SFBayArea/"/>
    <Name>680 S Express Lane &amp; Calvera SR84</Name>
    <geometry>
      <gml:point srsName="EPSG:4326">
        <gml:coordinates>-37.819006,-122.267532</gml:coordinates>
      </gml:point>
    </geometry>
  </Node>
 </Nodes>
</open511>
```


A.3.6 Example Traffic Segment Response (XML)


```
<open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.open511.info/">
 <segments>
  <atom:link rel="self" href="http://api.511.org/Open511/segments"/>
  <atom:link rel="up" href="http://api.511.org/Open511"/>
  <segment>
    <atom:link rel="self" href="/segments/12345/"/>
    <atom:link rel="jurisdiction" href="http://api.511.org/segments/SFBayArea/"/>
    <roads>
      <atom:link rel="road" href="http://api.511.org/open511/Roads/80-E"/>
    </roads>
    <description>80-E Between Fremont st and 4th st</description>
    <speed_limt>65</speed_limt>
    <Toll>Yes</Toll>
    <type>
    Freeway
    </type>
    <lane_count>3</lane_count>
    <exit_lane>R</exit_lane>
    <ramp_lanes_count>1</ramp_lanes_count>
    <Start_Node>
      <atom:link rel="node" href="http://api.511.org/open511/Nodes/1234"/>
    </Start_Node>
    <End_Node>
      <atom:link rel="node" href="http://api.511.org/open511/Nodes/1235"/>
    </End_Node>
    <geometry>
      <gml:LineString gml:id="p21" srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
        <gml:coordinates>37.785733,-122.39143 37.776872,-122.403559</gml:coordinates>
      </gml:LineString>
    </geometry>
    <restrictions>
      <vehicle_restriction>2 axels</vehicle_restriction>
    </restrictions>
```

```
  </segment>
 </segments>
</open511>
```

A.3.7 Example Traffic Segment Conditions Response (XML)

```
<open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.open511.info/">
 <segment_conditons>
  <atom:link rel="self" href="http://api.511.org/Open511/segments_conditons"/>
  <atom:link rel="up" href="http://api.511.org/Open511"/>
  <segment_conditon>
   <atom:link rel="self" href="/LinksSpeed/12345/"/>
   <atom:link rel="jurisdiction" href="http://api.511.org/Open511/SFBayArea/"/>
   <current_speed>50
    </current_speed>
   <historical_speed>
     54
    </historical_speed>
   <current_travel_time>
     220
    </current_travel_time>
   <historical_travel_time>
     200
    </historical_travel_time>
   <events>
    <atom:link rel="event" href="events/23958/"/>
    <atom:link rel="event" href="events/23957/"/>
   </events>
   <surface_conditions>
   Wet
   </surface_conditions>
  </segment_conditon>
 </segment_conditons>
</open511>
```

Section 4 – Toll XML

A.4.1 Example Toll Program Response (XML)


```
<open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.open511.info/">
  <toll_programs>
    <atom:link rel="self" href="http://api.511.org/Open511/toll_programs"/>
    <atom:link rel="up" href="http://api.511.org/Open511"/>
    <toll_program>
      <atom:link rel="jurisdiction" href="http://api.511.org/Open511/SFBayArea/"/>
      <atom:link rel="self" href="/tolls/680ExpressLanes"/>
      <!-- Something that identifies a toll-->
      <name>680 Express Lanes</name>
      <toll_authority>Bay Area Toll Authority (BATA)</toll_authority>
      <toll_type>dynamic</toll_type>
      <facility_type>Freeway</facility_type>
      <toll_description>High Occupancy Toll (HOT) lanes priced based on realtime traffic
conditions</toll_description>
      <toll_exemptions>
        <toll_exemption>HOV-3+</toll_exemption>
        <toll_exemption>Emergency vehicle</toll_exemption>
      </toll_exemptions>
      <vehicle_restrictions>
        <vehicle_restriction>trailers</vehicle_restriction>
        <vehicle_restriction>trucks more than two axles</vehicle_restriction>
      </vehicle_restrictions>
      <payment_options>
        <payment_option>FasTrak</payment_option>
      </payment_options>
      <toll_section>
        <roadway>
          <atom:link rel="road" href="http://api.511.org/open511/Roads/680-S"/>
        </roadway>
        <direction>S</direction>
        <barrier>restricted entry &amp; exit</barrier>
        <from_node>
          <atom:link rel="node" href="http://api.511.org/open511/Nodes/1235"/>
        </from_node>
        <to_node>
          <atom:link rel="node" href="http://api.511.org/open511/Nodes/1237"/>
        </to_node>
      </toll_section>
    </toll_program>
    <toll_program>
      <atom:link rel="jurisdiction" href="http://api.511.org/Open511/SFBayArea/"/>
      <toll_authority>Bay Area Toll Authority (BATA)</toll_authority>
      <atom:link rel="self" href="/tolls/BayBridge"/>
      <name>Bay Bridge</name>
      <toll_type>Static</toll_type>
      <facility_type>Freeway</facility_type>
      <toll_description>Bay Bridge toll on I 80 W</toll_description>
      <toll_exemptions>
        <toll_exemption>Emergency vehicle</toll_exemption>
      </toll_exemptions>
      <vehicle_restrictions>
        <vehicle_restriction>none</vehicle_restriction>
```

```
      </vehicle_restrictions>
      <payment_options>
        <payment_option>FasTrak</payment_option>
        <payment_option>Cash</payment_option>
        <payment_option>credit cards – Visa, Master card</payment_option>
        <payment_option>Electronic – pay by plate</payment_option>
      </payment_options>
      <toll_section>
        <roadway>I 80</roadway>
        <atom:link rel="road" href="http://api.511.org/open511/Roads/80-W"/>
        <direction>W</direction>
        <barrier>restricted entry through toll booth</barrier>
        <currency> US dollar</currency>
        <from_node>atom:link rel="node" href="http://api.511.org/open511/Nodes/1435"/>
              </from_node>
        <to_node>
          <atom:link rel="node" href="http://api.511.org/open511/Nodes/1437"/>
        </to_node>
      </toll_section>
    </toll_program>
  </toll_programs>
</open511>
```


A.4.2 Example Static Toll Response (XML)


```
<open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.open511.info/">
  <static_tolls>
    <atom:link rel="self" href="http://api.511.org/Open511/static_tolls"/>
    <atom:link rel="up" href="http://api.511.org/Open511"/>
    <static_toll>
      <atom:link rel="self" href="/statictolls/680ExpressLanes"/>
      <atom:link rel="config" href="/tolls/680ExpressLanes"/>
      <toll_pricing>
        <toll_from_node>
          <atom:link rel="node" href="http://api.511.org/open511/Nodes/1435"/>
        </toll_from_node>
        <toll_end_nodes>
          <toll_end_node>
            <to_node>
              <atom:link rel="node" href="http://api.511.org/open511/Nodes/1437"/>
            </to_node>
            <costs>
              <entries>
                <entry>
                  <price_day_type>M-F</price_day_type>
                  <occupany>Carpool (2)</occupany>
                  <passenger_capacity>2</passenger_capacity>
                  <vehicle_type>2 axle</vehicle_type>
                  <emission>low</emission>
                  <time>
                    <start>0700</start>
                    <!-- Morning Peak-->
                    <end>1000</end>
                  </time>
                  <price>3</price>
                </entry>
                <entry>
```

```
                <price_day_type>Weekend</price_day_type>
                <occupancy>Carpool (2+)</occupancy>
                <passenger_capacity>2+</passenger_capacity>
                <vehicle_type>2 axle</vehicle_type>
                <emission>low</emission>
                <time>
                  <start>0000</start>
                  <!-- All Day-->
                  <end>0000</end>
                </time>
                <price>3</price>
              </entry>
              <entry>
                <price_day_type>Everyday</price_day_type>
                <vehicle_type>3 axle</vehicle_type>
                <time>
                  <start>0000</start>
                  <!-- All Day-->
                  <end>0000</end>
                </time>
                <price>15</price>
              </entry>
            </entries>
          </costs>
        </toll_end_node>
      </toll_end_nodes>
    </toll_pricing>
  </static_toll>
 </static_tolls>
</open511>
```


A.4.3 Example Dynamic Toll Response (XML)


```
<open511 xmlns:atom="http://www.w3.org/2005/Atom" xmlns:gml="http://www.opengis.net/gml"
xml:lang="en" xml:base="http://api.open511.info/">
 <dynamic_tolls>
  <atom:link rel="self" href="http://api.511.org/Open511/dynamictolls"/>
  <atom:link rel="up" href="http://api.511.org/Open511"/>
  <dynamic_toll>
   <last_updated>YYYY-MM-DDTHH:MM:SSZ</last_updated>
   <atom:link rel="self" href="/dynamictolls/680ExpressLanes"/>
   <atom:link rel="config" href="/tolls/680ExpressLanes"/>
   <toll_pricing>
    <historical_time_slice>
     <day_type>M</day_type>
     <hour>09</hour>
     <minute>15</minute>
    </historical_time_slice>
    <toll_from_node>atom:link rel="node" href="http://api.511.org/open511/Nodes/1435"/>
        </toll_from_node>
    <toll_end_nodes>
     <toll_end_node>
      <to_node>
       <atom:link rel="node" href="http://api.511.org/open511/Nodes/1437"/>
      </to_node>
      <costs>
       <entries>
        <entry>
```

```
            <current_price>3</current_price>
            <historical_price>2.5</historical_price>
          </entry>
        </entries>
      </costs>
    </toll_end_node>
    <toll_end_node>
      <to_node>
        <atom:link rel="node" href="http://api.511.org/open511/Nodes/1438"/>
      </to_node>
      <costs>
        <entries>
          <entry>
            <current_price>3.5</current_price>
            <historical_price>2</historical_price>
          </entry>
        </entries>
      </costs>
    </toll_end_node>
  </toll_end_nodes>
 </toll_pricing>
</dynamic_toll>
<dynamic_toll>
 <last_updated>YYYY-MM-DDTHH:MM:SSZ</last_updated>
 <atom:link rel="self" href="/dynamictolls/580ExpressLanes"/>
 <atom:link rel="config" href="/tolls/580ExpressLanes"/>
 <toll_pricing>
  <historical_time_slice>
   <day_type>M</day_type>
   <hour>09</hour>
   <minute>15</minute>
  </historical_time_slice>
  <toll_from_node>atom:link rel="node" href="http://api.511.org/open511/Nodes/1535"/>
      </toll_from_node>
  <toll_end_nodes>
   <toll_end_node>
    <to_node>
     <atom:link rel="node" href="http://api.511.org/open511/Nodes/1536"/>
    </to_node>
    <costs>
     <entries>
      <entry>
       <current_price>3</current_price>
       <historical_price>2.5</historical_price>
      </entry>
     </entries>
    </costs>
   </toll_end_node>
   <toll_end_node>
    <to_node>
     <atom:link rel="node" href="http://api.511.org/open511/Nodes/1537"/>
    </to_node>
    <costs>
     <entries>
      <entry>
       <current_price>3</current_price>
       <historical_price>2.5</historical_price>
      </entry>
     </entries>
```

```
        </costs>
      </toll_end_node>
     </toll_end_nodes>
    </toll_pricing>
   </dynamic_toll>
  </dynamic_tolls>
</open511>
```

Section 5 – Parking XML

A.5.1 Example Parking Facility Structure Response (XML)

TBD

Appendix B: API Response Messages- JSON

Section 1: General JSON

B.1.1 Example Discovery Response (JSON)


```
{
    "content": {
        "services": [
            {
                "service_type": "events",
                "service_description": "Provide information about events impacting the
road system",
                "supported_versions": [
                    "1.0",
                    "2.0"
                ],
                "url": "http://api.511.org/Open511/events"
            },
            {
                "service_type": "Links",
                "service_description": "Provide information about traffic links covered
",
                "supported_versions": [
                    "2.0"
                ],
                "url": "http://api.511.org/Open511/Links"
            },
            {
                "service_type": "Linkspeed",
                "service_description": "Provide speed and travel time for traffic links
covered ",
                "supported_versions": [
                    "1.0"
                ],
                "url": "http://api.511.org/Open511/LinkSpeed"
            }
        ],
        "jurisdictions": [
            {
                "url": "http://api.511.org/Opne511/SFBayArea",
                "name": "SF Bay Area"
            }
        ]
    },
    "meta": {
        "url": "http://api.511.org/",
        "last_updated": "2012-12-05 9:00"
    }
}
```


B.1.2 Example Jurisdiction Response (JSON)


```
{
    "content": {
        "name": "Bay Area 511",
        "description_url": "http://api.511.org/Open511/presentation.html",
        "url": "http://api.511.org/Open511V1/SFBayArea",
        "area": {
```

```
            "type": "Polygon",
            "coordinates": [
                [
                    [
                        -71.170000000000002,
                        47.329999999999998
                    ],
                    [
                        -71.150000000000006,
                        47.359999999999999
                    ],
                    [
                        -71.099999999999994,
                        47.350000000000001
                    ],
                    [
                        -71.200000000000003,
                        47.399999999999999
                    ],
                    [
                        -71.170000000000002,
                        47.329999999999998
                    ]
                ]
            ]
        },
        "languages": [
            "fr",
            "en"
        ],
        "phone": "+1 514-872-0000",
        "email": "nahmed@mtc.ca.gov"
    },
    "meta": {
        "up_url": "http://api.511.org/Open511"
    }
}
```

B.1.3 Example Jurisdiction Geography Response (JSON)

```
{
  "geography" :  {
      "type": "Polygon",
      "coordinates": [
          [ [71.17,47.33], [-71.15,47.36], [-71.10,47.35],
              [-71.20,47.40], [-71.17,47.33] ]
      ]
    },
  "meta": {
      "version": "v0",
      "url": "http://api.511.org/Open511/SFBayArea/geography/",
      "up_url": "http://api.511.org/Open511/SFBayArea "
  }
}
```

Section 2– Transit JSON

B.2.1 Example Transit Operator Response (JSON.)

```
{
  "content": [
    {
      "Id": "SF",
      "Name": "Muni (San Francisco)",
      "ShortName": "Muni",
      "SiriOperatorRef": "SF-MUNI",
      “TimeZone”:”America/Vancouver”,
      “DefaultLanguage”:”en”,
      “ContactTelephoneNumber”:”1-415-701-2311”,
      “WebSite”:” http://www.sfmta.com/”,
      "PrimaryMode": "bus",
      "PrivateCode": "SF",
      "Monitored": true,
      "OtherModes": "tram, funicular",
          "Coverage": {
              "type": "Polygon",
              "coordinates": [
                  [
                      [
                          -71.170000000000002,
                          47.329999999999998
                      ],
                      [
                          -71.150000000000006,
                          47.359999999999999
                      ],
                      [
                          -71.099999999999994,
                          47.350000000000001
                      ],
                      [
                          -71.200000000000003,
                          47.399999999999999
                      ],
                      [
                          -71.170000000000002,
```

```
                        47.329999999999998
                    ]
                ]
            ]
        }
    }
]}
```

B.2.2 Example Transit Line Response (JSON.)


```
{
 "content": [
   {
     "Id": "BA:BAY PT/SFIA",
     "Name": "Pittsburg/Bay Point to San Francisco International Airport",
     "TransportMode": "rail",
     "SiriLineRef": "722",
     "Monitored": "true",
     "OperatorRef": "BA"
   }
 ]
}
```


B.2.3 Example Transit Stop Response (JSON.)


```
   "content": {
     "Id": "AC:59921",
     "Name": "CENTER ST & CIRCLE AV",
     "Location": {
       "Longitude": "-122.061515",
       "Latitude": "37.699237"
     }
   },
   "stopareas": {
     "StopArea": {
       "Id": "StopArea:SF1",
       "Name": "MUNI stops at Embarcadero",
       "ScheduledStopPointRef": [
         "SF:59921",
         "SF:58777"
       ],
       "ParentStopAreaRef": "StopArea:MTC1"
     }
   }
```


B.2.4 Example Transit Stop Response (JSON.)


```
  {
    "content": {
      "Id": "BA:12232",
      "Name": "BART LAKE MERRIT",
      "Description": "800 Madison StreetOakland, CA 94607 (Between Madison St & Fallon St
and 8th & 9th)",
      "Centroid": {
        "Location": {
          "Longitude": "-122.265668",
          "Latitude": "37.797345"
        }
      },
      "AccessibilityAssessment": {
        "MobilityImpairedAccess": "true",
        "limitations": {
          "AccessibilityLimitation": { "WheelchairAccess": "true" }
        }
```

```
      },
      "alternativeNames": {
        "AlternativeName": { "Name": "Lake Merrit Station" }
      },
      "PostalAddress": {
        "AddressLine1": "800 Madison St",
        "Town": "Oakland"
      },
      "OperatorRef": "BA",
      "adjacentSites": { "ParkingRef": "4234" },
      "placeEquipments": {
        "SanitaryEquipment": { "Description": "RestRoom in upper level" },
        "CycleStorageEquipment": [
          {
            "Description": "Bike Racks",
            "CycleStorageType": "racks",
            "NumberOfSpaces": "4"
          },
          {
            "Description": "Bike Lockers",
            "CycleStorageType": "other",
            "NumberOfSpaces": "10"
          }
        ],
        "SignEquipment": { "Description": "Information Display Board" },
        "EscalatorEquipment": { "Description": "Escalator 335" },
        "LiftEquipment": { "Description": "Escalator 312" },
        "ShelterEquipment": { "Description": "Waiting area 1" },
        "SeatingEquipment": { "Description": "Bench near waiting area" }
      },
      "PublicCode": "1564",
      "TransportMode": "rail",
      "StopPlaceType": "railStation",
      "quays": {
        "Quay": { "CompassOctant": "W" }
      },
      "parkings": {
        "Parking": {
          "Id": "4234",
          "Name": "Lake Merritt BART Station Parking",
          "Description": "On Broadway, between 11th & 14th",
          "Centroid": {
            "Location": {
              "Longitude": "-122.266382",
              "Latitude": "37.796615"
            }
          },
          "PostalAddress": {
            "AddressLine1": "800 Madison St",
            "Town": "Oakland"
          },
          "ParkingType": "trainStationParking",
          "TotalCapacity": "296",
          "RealTimeOccupancyAvailable": "false",
          "parkingAreas": {
            "ParkingArea": [
              {
                "Id": "123",
                "Description": "Accessible Parking",
```

```
                "ParkingProperties": {
                  "ParkingUserType": "registeredDisabled",
                  "spaces": {
                    "ParkingCapacity": { "NumberOfSpaces": "10" }
                  }
                }
              },
              {
                "Id": "124",
                "Description": "Reserved Parking",
                "ParkingProperties": {
                  "ParkingUserType": "reservationHolders",
                  "spaces": {
                    "ParkingCapacity": { "NumberOfSpaces": "99" }
                  },
                  "charges": {
                    "tariffBands": {
                      "ParkingTariffChargeBand": [
                        {
                          "Description": "Single Day Reserved Parking",
                          "MaximumStay": "P1D",
                          "Amount": "4.50"
                        },
                        {
                          "Description": "Monthly Reserved Parking",
                          "MaximumStay": "P1M",
                          "Amount": "100"
                        }
                      ]
                    }
                  }
                }
              }
            ]
          }
        }
      }
    }
  }
```


B.2.5 Example Transit Pattern Response (JSON.)

To be added

B.2.6 Example Transit Route Response (JSON.)

To be added

B.2.7 Example Transit Day Type Response (JSON.)

To be added

B.2.8 Example Transit Announcement Response (JSON.)

To be added

B.2.9 Example Response (JSON) in SIRI ST format

```
{
  "Siri": {
    "ServiceDelivery": {
      "ResponseTimestamp": "2013-02-18T09:30:47-08:00",
      "Status": true,
      "StopTimetableDelivery": {
        "version": 1.4,
        "ResponseTimestamp": "2004-12-17T09:30:47-05:00",
        "TimetabledStopVisit": [
          {
            "RecordedAtTime": "2004-12-17T09:25:46-05:00",
            "MonitoringRef": "HLTST011",
            "TargetedVehicleJourney": {
              "LineRef": 17,
              "DirectionRef": "INBOUND",
              "DatedVehicleJourneyRef ": " TRP123214",
              "PublishedLineName": "Fremont",
              "OperatorRef": "BART",              "OriginRef": "BART_11",
              "OriginName": "BART_CIVIC CENTER",
              "DestinationRef": "BART_99",
              "DestinationName": "BART_16th St-Mission",
              "TargetedCall": {
                "VisitNumber": 1,
                "AimedArrivalTime": "2013-02-18T09:30:47-08:00",
                "AimedDepartureTime": "2013-02-18T09:31:47-08:00"
              }
            },
          }, {
            "RecordedAtTime": "2004-12-17T09:25:46-05:00",
            "MonitoringRef": "HLTST011",
            "TargetedVehicleJourney": {
              "LineRef": 17,
              "DirectionRef": "INBOUND",
              "DatedVehicleJourneyRef ": " TRP544514",
              "PublishedLineName": "Fremont",
              "OperatorRef": "BART",
              "OriginRef": "BART_11",
              "OriginName": "BART_CIVIC CENTER",
              "DestinationRef": "BART_99",
              "DestinationName": "BART_16th St-Mission",
              "TargetedCall": {
                "VisitNumber": 2,
                "AimedArrivalTime": "2013-02-18T09:45:47-08:00",
                "AimedDepartureTime": "2013-02-18T09:46:47-08:00"
              }
            },
            }
          }
        ]
      }
    }
  }
}
```

B.2.10 Example Transit Real Time Departure Service Delivery Mode Response (JSON) in SIRI format


```
{
  "Siri": {
    "ServiceDelivery": {
      "ResponseTimestamp": "2004-12-17T09:30:46-05:00",
      "ProducerRef": "BART",
      "Status": true,
      "StopMonitoringDelivery": {
        "version": 1.4,
        "ResponseTimestamp": "2004-12-17T09:30:47-05:00",
        "Status": true,
        "MonitoredStopVisit": {
          "RecordedAtTime": "2004-12-17T09:25:46-05:00",
          "MonitoringRef": "BART_11",
          "MonitoredVehicleJourney": {
            "LineRef": 17,
            "DirectionRef": "Out",
            "FramedVehicleJourneyRef": {
              "DataFrameRef": "2004-12-17",
              "DatedVehicleJourneyRef": "Oubound"
            },
            "PublishedLineName": "Fremont",
            "OperatorRef": "BART",
            "Monitored": true,
            "VehicleLocation": {
              "Longitude": 180,
              "Latitude": 90
            },
            "ProgressStatus": "Service running on time",
            "PreviousCalls": {
              "PreviousCall": {
                "StopPointRef": "BART_10",
                "VisitNumber": 2,
                "StopPointName": "BART_DALY CITY",
                "VehicleAtStop": false,
                "AimedDepartureTime": "2004-12-17T09:32:43-05:00",
                "ActualDepartureTime": "2004-12-17T09:32:43-05:00"
              }
            },
            "MonitoredCall": {
              "VisitNumber": "0014",
              "VehicleAtStop": false,
              "VehicleLocationAtStop": {
                "Longitude": 180,
                "Latitude": 90
              },
              "AimedArrivalTime": "2004-12-17T09:40:46-05:00",
              "ExpectedArrivalTime": "2004-12-17T09:40:46-05:00",
              "AimedDepartureTime": "2004-12-17T09:42:47-05:00",
              "ExpectedDepartureTime": "2004-12-17T09:40:47-05:00"
            },
            "OnwardCalls": {
              "OnwardCall": {
                "StopPointRef": "BART_12",
                "VisitNumber": 4,
                "StopPointName": "BAR_12th St Oakland",
```

```
                "VehicleAtStop": false,
                "AimedArrivalTime": "2004-12-17T09:30:56-05:00",
                "ExpectedArrivalTime": "2004-12-17T09:30:56-05:00",
                "AimedDepartureTime": "2004-12-17T09:30:57-05:00",
                "ExpectedDepartureTime": "2004-12-17T09:30:57-05:00"
              }
            }
          }
        },
        "MonitoredStopVisitCancellation": [
          {
            "RecordedAtTime": "2004-12-17T09:30:47-05:00",
            "ItemRef": "SED9843214675429",
            "Reason": "Arrived"
          }, {
            "RecordedAtTime": "2004-12-17T09:30:47-05:00",
            "MonitoringRef": "BART_11",
            "VisitNumber": 2,
            "LineRef": "Line123",
            "DirectionRef": "Out",
            "VehicleJourneyRef": {
              "DataFrameRef": "2004-12-17",
              "DatedVehicleJourneyRef": "0987656"
            },
            "Reason": "Arrived"
          }
        ],
        "StopLineNotice": {
          "RecordedAtTime": "2004-12-17T09:30:47-05:00",
          "ItemIdentifier": "SED9843214675429",
          "MonitoringRef": "BART_11",
          "LineRef": 123,
          "DirectionRef": "Out",
          "LineNote": "Mechanical Problems on Track"
        },
        "StopLineNoticeCancellation": {
          "RecordedAtTime": "2004-12-17T09:30:47-05:00",
          "ItemRef": "SED9843214675429",
          "MonitoringRef": "BART_11",
          "LineRef": 123,
          "DirectionRef": "Out"
        },
        "Note": "Hello Stop"
      }
    }
  }
}
```


B.2.11 Example Real Time Vehicle Monitoring Response (JSON) in SIRI format


```
{
  "Siri": {
    "ServiceDelivery": {
      "ResponseTimestamp": "2004-12-17T09:30:47-05:00",
      "ProducerRef": "BART",
      "Status": true,
      "VehicleMonitoringDelivery": {
        "version": 1.4,
```

```
    "ResponseTimestamp": "2004-12-17T09:30:47-05:00",
    "VehicleActivity": [
      {
        "RecordedAtTime": "2004-12-17T09:30:47-05:00",
        "ValidUntilTime": "2004-12-17T09:30:47-05:00",
        "MonitoredVehicleJourney": {
          "LineRef": 17,
          "DirectionRef": "OUT",
          "FramedVehicleJourneyRef": {
            "DataFrameRef": "2004-12-17",
            "DatedVehicleJourneyRef": 987675
          },
          "PublishedLineName": 123,
          "OperatorRef": "BART",
          "OriginName": "SFO",
          "Via": [
            {
              "PlaceName": "16th st"
            }, {
              "PlaceName": "West Oakland"
            }
          ],
          "DestinationRef": "Fremont",
          "DestinationName": "Fremont",
          "Monitored": true,
          "InCongestion": false,
          "VehicleLocation": {
            "Longitude": 180,
            "Latitude": 90
          },
          "Bearing": 123,
          "ProgressRate": "slowProgress",
          "Delay": "PT2M",
          "ProgressStatus": "On time",
          "VehicleRef": "VEH987654",
          "PreviousCalls": {
            "PreviousCall": {
              "StopPointRef": "SFO",
              "VisitNumber": 2,
              "StopPointName": "String",
              "VehicleAtStop": false,
              "AimedDepartureTime": "2004-12-17T09:32:43-05:00",
              "ActualDepartureTime": "2004-12-17T09:32:43-05:00"
            }
          },
          "OnwardCalls": {
            "OnwardCall": {
              "StopPointRef": 80,
              "VisitNumber": 4,
              "StopPointName": "16th Street",
              "VehicleAtStop": false,
              "AimedArrivalTime": "2004-12-17T09:30:56-05:00",
              "ExpectedArrivalTime": "2004-12-17T09:30:56-05:00",
              "AimedDepartureTime": "2004-12-17T09:30:57-05:00",
              "ExpectedDepartureTime": "2004-12-17T09:30:57-05:00"
            }
          }
```

```
            }
          }, {
            "RecordedAtTime": "2004-12-17T09:30:47-05:00",
            "ValidUntilTime": "2004-12-17T09:30:47-05:00",
            "VehicleMonitoringRef": 45678,
            "MonitoredVehicleJourney": {
              "LineRef": "Line123",
              "FramedVehicleJourneyRef": {
                "DataFrameRef": "2004-12-17",
                "DatedVehicleJourneyRef": "Outbound"
              },
              "Monitored": true,
              "VehicleLocation": {
                "Longitude": 180,
                "Latitude": 90
              },
              "Delay": "PT2M",
              "VehicleRef": "VEH987654",
              "OnwardCalls": {
                "OnwardCall": {
                  "StopPointRef": "HLTST012",
                  "StopPointName": "Church"
                }
              }
            }
          }
        ],
        "VehicleActivityCancellation": {
          "RecordedAtTime": "2004-12-17T09:30:47-05:00",
          "VehicleMonitoringRef": 9876542,
          "VehicleJourneyRef": {
            "DataFrameRef": "2001-12-17",
            "DatedVehicleJourneyRef": "09867"
          },
          "LineRef": "Line123",
          "DirectionRef": "Out",
          "Reason": "Done for the day"
        }
      }
    }
  }
}
```


B.2.12 Example Transit Schedule Update Response (JSON) in SIRI PT format


```
{
  "Siri": {
    "ServiceDelivery": {
      "ResponseTimestamp": "2013-02-18T09:30:47-08:00",
      "Status": true,
      "ProductionTimetableDelivery": {
        "version": 1.4,
        "ResponseTimestamp": "2004-12-17T09:30:47-05:00",
        "ValidUntil": "2001-12-17T10:30:47-05:00",
        "DatedTimetableVersionFrame": {
          "RecordedAtTime": "2001-12-17T09:30:47-05:00",
          "LineRef": 123,
          "DirectionRef": "Out",
```

```
            "PublishedLineName": "String",
            "DatedVehicleJourney": {
              "DatedVehicleJourneyCode": "DVC0008767",
              "DatedCalls": {
                "DatedCall": [
                  {
                    "StopPointRef": "BART_11",
                    "CallNote": "optional message here",
                    "AimedArrivalTime": "2013-02-19T09:55:47-08:00",
                    "AimedDepartureTime": "2013-02-19T09:56:47-08:00"
                  }, {
                    "StopPointRef": "BART_99",
                    "CallNote": "optional message here",
                    "AimedArrivalTime": "2013-02-19T10:15:47-08:00",
                    "AimedDepartureTime": "2013-02-19T10:16:47-08:00"
                  }
                ]
              }
            }
          }
        }
      }
    }
  }
```


B.2.13 Example Transit Addition and Cancellation of Trip Response (JSON) in SIRI ET format


```
{
  "Siri": {
    "xmlns": "http://www.siri.org.uk/siri",
    "xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
    "version": 1.4,
    "xsi:schemaLocation": "http://www.siri.org.uk/siri
http://www.kizoom.com/standards/siri/schema/1.4/siri.xsd",
    "ServiceDelivery": {
      "ResponseTimestamp": "2013-02-18T09:30:47-08:00",
      "Status": true,
      "EstimatedTimetableDelivery": {
        "version": 1.4,
        "ResponseTimestamp": "2004-12-17T09:30:47-05:00",
        "EstimatedJourneyVersionFrame": {
          "RecordedAtTime": "2013-02-18T09:30:47-08:00",
          "EstimatedVehicleJourney": [
            {
              "LineRef": 917,
              "DirectionRef": "INBOUND",
              "DatedVehicleJourneyRef": "00008",
              "PublishedLineName": "Fremont",
              "EstimatedCalls": {
                "EstimatedCall": [
                  {
                    "StopPointRef": "BART_11",
                    "CallNote": "optional message here",
                    "AimedArrivalTime": "2013-02-19T09:55:47-08:00",
                    "AimedDepartureTime": "2013-02-19T09:56:47-08:00"
                  }, {
```

```
                "StopPointRef": "BART_99",
                "CallNote": "optional message here",
                "AimedArrivalTime": "2013-02-19T10:15:47-08:00",
                "AimedDepartureTime": "2013-02-19T10:16:47-08:00"
              }
            ]
          }
        }, {
          "LineRef": 764,
          "DirectionRef": "INBOUND",
          "DatedVehicleJourneyRef": "00008",
          "Cancellation": true,
          "PublishedLineName": "Pittsburgh Bay Point"
        }
      ]
    }
  }
 }
}
```


B.2.14 Example Transit General Messaging Service Response (JSON) in SIRI GM format


```
{
  "Siri": {
    "ResponseTimestamp": "2013-02-17T09:30:46-08:00",
    "Status": true,
    "GeneralMessageDelivery": {
      "version": 1.4,
      "ResponseTimestamp": "2001-12-17T09:30:47.0Z",
      "GeneralMessage": [
        {
          "formatRef": "string",
          "RecordedAtTime": "2013-02-17T09:30:46-08:00",
          "InfoMessageIdentifier": 12345,
          "InfoMessageVersion": 2,
          "InfoChannelRef": "WARNINGS",
          "ValidUntilTime": "2013-02-18T09:30:46-08:00",
          "Content": "some message here"
        }, {
          "formatRef": "string",
          "RecordedAtTime": "2013-02-17T09:30:46-08:00",
          "InfoMessageIdentifier": 23456,
          "InfoMessageVersion": 1,
          "InfoChannelRef": "WARNINGS",
          "ValidUntilTime": "2013-02-18T09:30:46-08:00",
          "Content": "some message here"
        }
      ]
    }
  }
}
```

Section 3 – Traffic JSON

B.3.1 Example Traffic Event Structure Response (JSON)


```
{
    "content": [
        {
            "status": "active",
            "event_sub_type": "Urgent roadwork",
            "description": ": Roadwork on I-80 E Eastbound between 4TH St (San Francisco)
and I-880 S. Left lane closed until Noon",
            "traveler_message": "Avoid-region.",
            "url": "/events/23948/",
            "severity": "2",
            "certainty": "Observed",
            "schedule": {
                "start_date": "2012-04-01",
                "end_date": "2012-04-30"
            },
            "detour": "Take Sherbrooke travelling east, and Wellington travelling west.",
            "creation_date": "2012-05-23T20:33:10Z",
            "geometry": {
                "type": "LineString",
                "coordinates": [
                    [
                        -71.170000000000002,
                        47.329999999999998
                    ],
                    [
                        -71.150000000000006,
                        47.359999999999999
                    ],
                    [
                        -71.099999999999994,
                        47.350000000000001
                    ],
                    [
                        -71.200000000000003,
                        47.399999999999999
                    ]
                ]
            },
            "headline": "Roadwork on I-80 E Eastbound between 4TH St (San Francisco) and
I-880 S",
            "enclosure_url": "http://my.city.gov/trafic/advisory/39473/com.pdf",
            "attachment": {
                "start_date": "2012-04-01",
                "end_date": "2012-04-30"
            },
            "event_type": "Incident",
            "last_updated": "2012-05-24T10:00:10Z",
            "road_affected": {
                "lane_status": "10",
                "restrictions": [
                    "Local Traffic Only"
                ],
                "direction": "E",
                "Road_url": "http://api.511.org/Open511/Roads/1567",
                "to_road": "1st avenue",
```

```
                "impacted_systems": [
                    "Road",
                    "On-Street Parking"
                ],
                "on_road": "11th st",
                "from_road": "Broadway St"
            },
            "jurisdiction_url": "http://api.511.org/Open511/SFBayArea/"
        }
    ],
    "meta": {
        "url": "http://api.511.org/Open511/events",
        "up_url": "http://api.511.org/Open511"
    }
}
```


B.3.2 Example Traffic Announcment Response (JSON)


```
{
    "content": [
        {
            "status": "active",
            "description": "Bart to run extra trains during world series.",
            "url": "/announcements/23948/",
            "creation_date": "2012-05-23T20:33:10Z",
            "headline": "Bart to run extra trains during world series.",
            "last_update": "2012-05-24T10:00:10Z",
            "Type": "Traffic",
            "jurisdiction_url": "http://api.511.org/Open511/SFBayArea/"
        }
    ],
    "meta": {
        "url": "http://api.511.org/Open511/Announcements",
        "up_url": "http://api.511.org/Open511"
    }
}
```


B.3.3 Example Traffic Area Response (JSON)


```
{
 "content": [
  {
   "url": "/area/1234",
   "name": "San Francisco",
   "type": "City",
   "geometry": {
    "gml:Polygon": {
     "srsName": "EPSG:4326",
     "gml:LinearRing": {
      "gml:coordinates": {
       "decimal": ".",
       "cs": ",",
       "ts": "",
       "Text": "119.593002319336,-31.6695003509522 119.595306396484,\r
31.6650276184082 119.600944519043,-31.6658897399902 119.603385925293,-31.669527053833
\r                    119.60050201416,-31.6739158630371 119.595664978027,-
31.6728610992432 119.593002319336,\r                    31.6695003509522"
      }
     }
    }
```

```
    }
   }, {
    "url": "/area/1235",
    "name": "SOMA",
    "type": "Neighborhood",
    "area_within": {
     "atom:link": {
      "rel": "within",
      "href": "/area/1234"
     }
    },
    "geometry": {
     "gml:Polygon": {
      "srsName": "EPSG:4326",
      "gml:LinearRing": {
       "gml:coordinates": {
        "decimal": ".",
        "cs": ",",
        "ts": "",
        "Text": "119.593002319336,-31.6695003509522 119.595306396484,\r
31.6650276184082 119.600944519043,-31.6658897399902 119.603385925293,-31.669527053833
\r                    119.60050201416,-31.6739158630371 119.595664978027,-
31.6728610992432 119.593002319336,\r                    31.6695003509522"
       }
      }
     }
    }
   }
  ],
  "meta": {
   "url": "http://api.511.org/Open511/Area",
   "up_url": "http://api.511.org/Open511"
  }
}
```


B.3.4 Traffic Road Example Response (JSON)


```
{
 "content": [
 {
  "Direction": "S",
  "jurisdiction_url": "http://api.511.org/Open511/SFBayArea/",
  "url": "/Roads/I880S/",
  "Type": "Freeway",
  "Name": "I880S",
  "segments": [
   {
    "url": "http://api.511.org/open511/segment/1567"
   }, {
    "url": "http://api.511.org/open511/segment/1568"
   }
  ]
 }
 ],
 "meta": {
  "url": "http://api.511.org/Open511/Roads",
  "up_url": "http://api.511.org/Open511"
 }
}
```

B.3.5 Example Traffic Node Response (JSON)


```
{
 "content": [
  {
   "jurisdiction_url": "http://api.511.org/Open511/SFBayArea/",
   "url": "/Roads/12345/",
   "Name": "80 E &amp;Fremont St",
   "roads": [
    {
     "url": "http://api.511.org/Open511/roads/I880S"
    }, {
     "url": "http://api.511.org/Open511/roads/Mowry"
    }
   ],
   "geometry": {
    "gml:point": {
     "srsName": "EPSG:4326",
     "gml:coordinates": "-37.819006,-122.267532"
    }
   }
  }, {
   "jurisdiction_url": "http://api.511.org/Open511/SFBayArea/",
   "url": "/Roads/12346/",
   "Name": "680 S Express Lane & Calvera SR84",
   "geometry": {
    "gml:point": {
     "srsName": "EPSG:4326",
     "gml:coordinates": "-37.819006,-122.267532"
    }
   }
  }
 ],
 "meta": {
  "url": "http://api.511.org/Open511/Nodes",
  "up_url": "http://api.511.org/Open511"
 }
}
```


B.3.6 Example Traffic Segment Response (JSON)


```
{
 "content": [
 {
     "jurisdiction_url": "http://api.511.org/Open511/SFBayArea/",
   "url": "/segments/12345/",
   "description": "80-E Between Fremont st and 4th st",
    "Road":[
     {
      "href": "http://api.511.org/open511/Roads/80-E"
     }
    ],
    "speed_limt": 65,
    "Toll": "Yes",
    "type": "Freeway",
    "lane_count": 3,
    "exit_lane": "R",
    "ramp_lanes_count": 1,
    "Start_Node": {
     "url": "http://api.511.org/open511/Nodes/1234"
```

```
   },
      "End_Node": {
         "url": "http://api.511.org/open511/Nodes/1235"
  },
      "geometry": {
       "gml:LineString": {
        "gml:id": "p21",
        "srsName": "http://www.opengis.net/def/crs/EPSG/0/4326",
        "gml:coordinates": "37.785733,-122.39143 37.776872,-122.403559"
       }
      },
      "restrictions": [{
       "vehicle_restriction": "2 axels"
      }
     ]
     }
],
    "meta": {
   "url": "http://api.511.org/Open511/segments",
   "up_url": "http://api.511.org/Open511"
  }
  }
```


B.3.7 Example Traffic Segment Conditions Response (JSON)


```
{
 "content": [
 {
     "jurisdiction_url": "http://api.511.org/Open511/SFBayArea/",
    "url": "/segments/12345/",
       "current_speed": 50,
       "historical_speed": 54,
       "current_travel_time": 220,
       "historical_travel_time": 200,
       "events": [{

           "url": "events/23958/"
         }, {
           "url": "events/23957/"
         }
        ],
     "surface_conditions": "wet"
    }
    ],
        "meta": {
   "url": "http://api.511.org/Open511/segments",
   "up_url": "http://api.511.org/Open511"
   }
}
```

Section 4 – Toll JSON

B.4.1 Example Toll Program Response (JSON)


```
{
 "content":[ {
  "name": "680 Express Lanes",
  "toll_authority": "Bay Area Toll Authority (BATA)",
  "toll_type": "dynamic",
  "facility_type": "Freeway",
  "toll_description": "High Occupancy Toll (HOT) lanes priced based on realtime traffic
conditions",
  "toll_exemptions": {
   "toll_exemption": [
    "HOV-3+", "Emergency vehicle"
   ]
  },
  "vehicle_restrictions": {
   "vehicle_restriction": [
    "trailers", "trucks more than two axles"
   ]
  },
  "payment_options": {
   "payment_option": "FasTrak"
  },
  "toll_section": {
   "roadway_url": "http://api.511.org/open511/Roads/680-S",
   "direction": "S",
   "barrier": "restricted entry & exit",
   "from_node_url": "http://api.511.org/open511/Nodes/1235",
   "to_node_url": "http://api.511.org/open511/Nodes/1237"
  },
  "jurisdiction_url": "http://api.511.org/Open511/SFBayArea/"
 }],
 "meta": {
  "url": "http://api.511.org/Open511/toll_programs",
  "up_url": "http://api.511.org/Open511"
 }
}
```


B.4.2 Example Static Toll Response (JSON)


```
{
 "content":[ {
  "config_url": "/tolls/680ExpressLanes",
  "toll_pricing": {
   "toll_from_node_url": "http://api.511.org/open511/Nodes/1435",
   "toll_end_nodes": [{
      "to_node_url": "http://api.511.org/open511/Nodes/1437",
    "costs": {
     "entries": {
      "entry": [
       {
        "price_day_type": "M-F",
        "occupany": "Carpool (2)",
        "passenger_capacity": 2,
        "vehicle_type": "2 axle",
        "emission": "low",
        "time": {
```

```
              "start": "0700",
              "Comment": " Morning Peak",
              "end": 1000
            },
            "price": 3
          }, {
            "price_day_type": "Weekend",
            "occupancy": "Carpool (2+)",
            "passenger_capacity": "2+",
            "vehicle_type": "2 axle",
            "emission": "low",
            "time": {
              "start": "0000",
              "Comment": " All Day",
              "end": "0000"
            },
            "price": 3
          }, {
            "price_day_type": "Everyday",
            "vehicle_type": "3 axle",
            "time": {
              "start": "0000",
              "Comment": " All Day",
              "end": "0000"
            },
            "price": 15
          }
        ]
      }
    }
    }
  ]
 }
}],
"meta": {
  "url": "http://api.511.org/Open511/static_tolls",
  "up_url": "http://api.511.org/Open511"
}
}
```


B.4.3 Example Dynamic Toll Response (JSON)


```
{
 "content":[ {
  "config_url": "/tolls/680ExpressLanes",
  "dynamic_toll": {
   "last_updated": "YYYY-MM-DDTHH:MM:SSZ",
   "toll_pricing": {
    "historical_time_slice": {
     "day_type": "M",
     "hour": "09",
     "minute": 15
    },
    "toll_from_node_url": "http://api.511.org/open511/Nodes/1435",
    "toll_end_nodes": [
      {
       "to_node_url": "http://api.511.org/open511/Nodes/1437",
       "costs": {
        "entries": {
```

```
            "entry": {
              "current_price": 3,
              "historical_price": 2.5
            }
          }
        }
      }, {
        "to_node_url:": "http://api.511.org/open511/Nodes/1438",
        "costs": {
          "entries": {
            "entry": {
              "current_price": 3.5,
              "historical_price": 2
            }
          }
        }
      }
    ]
  }
 }
}]
,
"meta": {
  "url": "http://api.511.org/Open511/static_tolls",
  "up_url": "http://api.511.org/Open511"
}
}
```

Section 5 – Parking JSON

B.5.1 Example Parking Facility Response (JSON)

TBD