

TransSuite

TRANSUITE TCS

CONNECTED VEHICLE DATA PORT INTERFACE

Version 4.0

Prepared by:

**TransCore ITS, LLC
192 Technology Parkway, Suite 500
Norcross, Georgia 30092
(770) 447-6831 phone
(770) 449-6268 fax**

PROPRIETARY RIGHTS NOTICE: All rights reserved. This material contains the valuable properties and trade secrets of TransCore ITS, LLC, embodying substantial creative efforts and confidential information, ideas, and expressions, no part of which may be reproduced or transmitted in any form or by any means, including electronic, mechanical, or otherwise, including photocopying and recording, or in connection with any information storage or retrieval system without the permission in writing from TransCore ITS, LLC.

©2019. TransCore, LP. All rights reserved. TRANSCORE and TRANSUITE are registered trademarks and are used under license. All other trademarks are the property of their respective owners. Contents are subject to change.

Table of Contents

1. INTRODUCTION.....	1
2. OVERVIEW.....	1
3. WADL DEFINITIONS.....	1
4. OPERATION CALLS.....	1
4.1 INTERSECTION INVENTORY	1
4.2 START SESSION.....	2
4.3 SET SUBSCRIPTIONS	2
4.4 GET REAL-TIME STATUS	3
4.5 STOP SESSION	10
5. INTERFACE TEST CLIENT.....	11
5.1 OVERVIEW	11
5.2 REQUIRED FILES	11
5.3 CONFIGURATION SETTINGS	11
5.4 ENABLING CVDP FROM TCS	12
5.5 USAGE	14

Document Revision History

Version #	Date	By	Purpose
1.0	09/07/2017		Initial Release
2.0	10/23/2017		Second Release
3.0	02/18/2019		Third Release
4.0	07/08/2019		Fourth Release
4.1	07/19/2019	Keith Patton	Added codes for preempt active/inactive
4.2	12/11/2019	James Martin	Added codes for Intelight and ASC/3 preempts

1. INTRODUCTION

The TransSuite TCS Connected Vehicle Data Port (CVDP) is a REST service that provides data about intersection inventory and real-time status updates. This document describes the interface and how client applications can interact with the interface.

2. OVERVIEW

Client applications may call the CVDP REST service to obtain an inventory of intersections. These intersections are available for clients to subscribe to real-time status updates. Client applications may obtain a session identifier from the REST service and use it in subsequent calls to set intersection status subscriptions and to obtain status updates. Finally, a client application may choose to stop a session when it is no longer interested in receiving updates, or as part of a graceful shutdown. Sessions that have been inactive for at least one minute will automatically be timed-out by the server, with the same effect as stopping a session.

Multiple sessions can be created and are supported by the server. This allows for multiple client applications to connect and retrieve data.

3. WADL DEFINITIONS

Two Web Application Description Language (WADL) files have been generated to describe the interface.

1. **CVDP.simplified.wadl.xml** – A simplified version of the interface with user and core resources only.
2. **CVDP.full.wadl.xml** – The full WADL including extended resources.

4. OPERATION CALLS

As seen in either WADL file, there are six operations that can be called on the REST interface.

4.1 INTERSECTION INVENTORY

Path: /IntersectionInventory

Method: GET

Produces: text/plain

Parameters: None

This will return the inventory of all intersections on the TCS system that have been configured to have their data shared. The response is in CSV format, where the first line is the header. The columns are as follows:

- **Intersection ID** – The numerical identifier of the intersection. This is the ID that is displayed in TransSuite TCS, and is unique to each intersection.
- **Description** – A free-form text description of the intersection, as entered by the user.
- **Main Street** – The main street of the intersection.
- **Cross Street** – The cross street of the intersection.
- **Latitude** – The latitude portion of the location of the intersection. This field is in micro degrees. This value will be zero if the intersection hasn't been plotted on the map.
- **Longitude** – The longitude portion of the location of the intersection. This field is in micro degrees. This value will be zero if the intersection hasn't been plotted on the map.
- **MaxPhaseTimingSets** – The maximum number of phase timing sets the controller supports. For NTCIP controllers, this is the max number of phase timing plans (not to be confused with coordinated patterns). For other controller types, this is the number of phase timing banks.

Example Response:

"Intersection ID","Description","Main Street","Cross Street","Latitude","Longitude","MaxPhaseTimingSets"
 "20101","Test Intersection","Spalding Drive","Technology Parkway Northwest","33968191","-84217253","1"

4.2 START SESSION

Path: /StartSession

Method: GET

Produces: text/plain

Parameters: None

This will create a new client session in the server and return a new session identifier to the client. The session identifier should be used in calls to set subscriptions, get real-time status updates, and when stopping a session.

4.3 SET SUBSCRIPTIONS

Path: /SetSubscriptions

Method: GET

Produces: text/plain

Parameters:

1. **sessionId** - The session identifier obtained from a previous call to "Start Session".
2. **intersectionIds** - A list of zero or more intersection identifiers. These should match up with the "Intersection ID" column from the "Intersection Inventory" response. Alternatively, this can be set to asterisk (*) if the client wishes to subscribe to all intersections.

Possible Responses:

- **"OK"** – Indicates that the command was accepted.

- **“Invalid Session ID”** – Indicates that the session identifier provided by the client is invalid. This could happen if the client is not using the session ID provided by the “Start Session” call, if the client already stopped the session, if the server timed-out the client session due to inactivity, or if the server restarted.
- **<Anything else>** - Indicates some other error has occurred.

This will configure the CVDP server to accumulate real-time status updates for the specified intersections for the client using the specified session identifier. These updates can be obtained by the client by calling “Get Real-Time Status” or “Get Real-Time Status v2”.

4.4 GET REAL-TIME STATUS

Path: /GetRealTimeStatus

Method: GET

Produces: application/octet-stream

Parameters: **sessionId** - The session identifier obtained from a previous call to “Start Session”.

This will query the CVDP server for any real-time status updates that have accumulated since the last call to this operation. If this is the first call since initially setting the subscriptions, it will include status updates since the subscriptions were made.

The CVDP server will reply with new events that have occurred since their last call. The data format will be binary:

- 1 byte indicating status
 - a) 0 = OK
 - b) 1 = Invalid session ID
 - c) 2 = Other error
- 4 bytes indicating number of second-by-second updates that follow. This is a 32-bit integer in big endian format. This value will be zero if the first byte indicating status was non-zero (indicating invalid session ID or other error).
- (Thus begins the second-by-second updates)
- 4 bytes of timestamp in UTC seconds. This is a 32-bit integer in big endian format.
- 4 bytes indicating the number of records that follow. This is a 32-bit integer in big endian format.
- (Thus begins the events that occurred this second)
- Each record contains:
 - a) 4 bytes of intersection ID. This is a 32-bit integer in big endian format.
 - b) 1 byte of event code
 - c) 1 byte of event parameter

The event code and event parameter are derived from the “Indiana Traffic Signal Hi Resolution Data Logger Enumerations” and can be interpreted as such. The subset of event codes returned by this entry point is listed below. Note that codes 132 through 149 will only be returned for intersections running in adaptive mode.

Event Code	Event Descriptor	Parameter
1	Phase Begin Green	Phase # (1-40)
7	Phase Green Termination	Phase # (1-40)
8	Phase Begin Yellow Clearance	Phase # (1-40)
9	Phase End Yellow Clearance	Phase # (1-40)
21	Pedestrian Begin Walk	Phase # (1-40)
22	Pedestrian Begin Clearance	Phase # (1-40)
23	Pedestrian Begin Solid Don't Walk	Phase # (1-40)
32	FYA Begin Permissive	Phase # (1-16)
33	FYA End Permissive	Phase # (1-16)
43	Phase Call Registered	Phase # (1-40)
44	Phase Call Dropped	Phase # (1-40)
45	Pedestrian Call Registered	Phase # (1-40)
67	Pedestrian Overlap Begin Walk	Overlap # (as number A-1, B-2, etc.)
81	Detector Off	DET Channel # (1-128)
82	Detector On	DET Channel # (1-128)
102	Preempt (Call) Input On	Preempt # (1-32)
104	Preempt (Call) Input Off	Preempt # (1-32)
105	Preempt Entry Started	Preempt # (1-32)
106	Preempt Begin Track Clearance	Preempt # (1-32)
107	Preempt Begin Dwell Service	Preempt # (1-32)
108	Preempt Link Input Active	Preempt # (1-32)
110	Preempt Max Presence Exceeded	Preempt # (1-32)

111	Preempt Begin Exit Interval	Preempt # (1-32)
112	TSP Check In	TSP # (1-8)
115	TSP Check Out	TSP # (1-8)
131	Coord Pattern Change	Pattern # (0-255)
132	Cycle Length Change	Seconds (0-255)
133	Offset Length Change	Seconds (0-255)
134	Split 1 Change	New Split Time in Seconds (0-255)
135	Split 2 Change	New Split Time in Seconds (0-255)
136	Split 3 Change	New Split Time in Seconds (0-255)
137	Split 4 Change	New Split Time in Seconds (0-255)
138	Split 5 Change	New Split Time in Seconds (0-255)
139	Split 6 Change	New Split Time in Seconds (0-255)
140	Split 7 Change	New Split Time in Seconds (0-255)
141	Split 8 Change	New Split Time in Seconds (0-255)
142	Split 9 Change	New Split Time in Seconds (0-255)
143	Split 10 Change	New Split Time in Seconds (0-255)
144	Split 11 Change	New Split Time in Seconds (0-255)
145	Split 12 Change	New Split Time in Seconds (0-255)
146	Split 13 Change	New Split Time in Seconds (0-255)
147	Split 14 Change	New Split Time in Seconds (0-255)
148	Split 15 Change	New Split Time in Seconds (0-255)
149	Split 16 Change	New Split Time in Seconds (0-255)
150	Coord cycle state change	Parameter (0-6) defined as: 0 = Free 1 = In Step 2 = Transition - Add

		3 = Transition - Subtract 4 = Transition - Dwell 5 = Local Zero 6 = Begin Pickup
174	Unit Alarm Status 1 Change	unitAlarmStatus1
175	Alarm Group State Change	Bit 0: Alarm 1 Bit 1: Alarm 2 Bit 2: Alarm 3 Bit 3: Alarm 4
176	Special Function Output On	Special Function # (1-16)
177	Special Function Output Off	Special Function # (1-16)
178	Manual Control Enable	off/on # (0,1)
180	Stop Time Input	off/on # (0,1)
184	Power Restored	True (1)

4.5 GET REAL-TIME STATUS V2

Path: /GetRealTimeStatusV2

Method: GET

Produces: application/octet-stream

Parameters: **sessionId** - The session identifier obtained from a previous call to “Start Session”.

This will query the CVDP server for any real-time status updates that have accumulated since the last call to this operation. If this is the first call since initially setting the subscriptions, it will include status updates since the subscriptions were made. This alternative to “/GetRealTimeStatus” uses a 2-byte event code to allow an extended set of events.

The CVDP server will reply with new events that have occurred since their last call. The data format will be binary:

- 1 byte indicating status
 - a) 0 = OK
 - b) 1 = Invalid session ID
 - c) 2 = Other error
 - 4 bytes indicating number of second-by-second updates that follow. This is a 32-bit integer in big endian format. This value will be zero if the first byte indicating status was non-zero (indicating invalid session ID or other error).
- (Thus begins the second-by-second updates)

- 4 bytes of timestamp in UTC seconds. This is a 32-bit integer in big endian format.
- 4 bytes indicating the number of records that follow. This is a 32-bit integer in big endian format.
(Thus begins the events that occurred this second)
- Each record contains:
 - a) 4 bytes of intersection ID. This is a 32-bit integer in big endian format.
 - b) 2 bytes of event code in big endian format.
 - c) 1 byte of event parameter

The event code and event parameter are derived from the “Indiana Traffic Signal Hi Resolution Data Logger Enumerations” along with a new set of event codes, starting at 10000. The set of event codes returned by this entry point is listed below. Note that codes 132 through 149 will only be returned for intersections running in adaptive mode.

Event Code	Event Descriptor	Parameter
1	Phase Begin Green	Phase # (1-40)
7	Phase Green Termination	Phase # (1-40)
8	Phase Begin Yellow Clearance	Phase # (1-40)
9	Phase End Yellow Clearance	Phase # (1-40)
21	Pedestrian Begin Walk	Phase # (1-40)
22	Pedestrian Begin Clearance	Phase # (1-40)
23	Pedestrian Begin Solid Don't Walk	Phase # (1-40)
32	FYA Begin Permissive	Phase # (1-16)
33	FYA End Permissive	Phase # (1-16)
43	Phase Call Registered	Phase # (1-40)
44	Phase Call Dropped	Phase # (1-40)
45	Pedestrian Call Registered	Phase # (1-40)
67	Pedestrian Overlap Begin Walk	Overlap # (as number A-1, B-2, etc.)
81	Detector Off	DET Channel # (1-128)
82	Detector On	DET Channel # (1-128)
102	Preempt (Call) Input On	Preempt # (1-32)

104	Preempt (Call) Input Off	Preempt # (1-32)
105	Preempt Entry Started	Preempt # (1-32)
106	Preempt Begin Track Clearance	Preempt # (1-32)
107	Preempt Begin Dwell Service	Preempt # (1-32)
108	Preempt Link Input Active	Preempt # (1-32)
110	Preempt Max Presence Exceeded	Preempt # (1-32)
111	Preempt Begin Exit Interval	Preempt # (1-32)
112	TSP Check In	TSP # (1-8)
115	TSP Check Out	TSP # (1-8)
131	Coord Pattern Change	Pattern # (0-255)
132	Cycle Length Change	Seconds (0-255)
133	Offset Length Change	Seconds (0-255)
134	Split 1 Change	New Split Time in Seconds (0-255)
135	Split 2 Change	New Split Time in Seconds (0-255)
136	Split 3 Change	New Split Time in Seconds (0-255)
137	Split 4 Change	New Split Time in Seconds (0-255)
138	Split 5 Change	New Split Time in Seconds (0-255)
139	Split 6 Change	New Split Time in Seconds (0-255)
140	Split 7 Change	New Split Time in Seconds (0-255)
141	Split 8 Change	New Split Time in Seconds (0-255)
142	Split 9 Change	New Split Time in Seconds (0-255)
143	Split 10 Change	New Split Time in Seconds (0-255)
144	Split 11 Change	New Split Time in Seconds (0-255)
145	Split 12 Change	New Split Time in Seconds (0-255)

146	Split 13 Change	New Split Time in Seconds (0-255)
147	Split 14 Change	New Split Time in Seconds (0-255)
148	Split 15 Change	New Split Time in Seconds (0-255)
149	Split 16 Change	New Split Time in Seconds (0-255)
150	Coord cycle state change	Parameter (0-6) defined as: 0 = Free 1 = In Step 2 = Transition - Add 3 = Transition - Subtract 4 = Transition - Dwell 5 = Local Zero 6 = Begin Pickup
174	Unit Alarm Status 1 Change	unitAlarmStatus1
175	Alarm Group State Change	Bit 0: Alarm 1 Bit 1: Alarm 2 Bit 2: Alarm 3 Bit 3: Alarm 4
176	Special Function Output On	Special Function # (1-16)
177	Special Function Output Off	Special Function # (1-16)
178	Manual Control Enable	off/on # (0,1)
180	Stop Time Input	off/on # (0,1)
184	Power Restored	True (1)
10000	D4 Transit Phase Remote Check In	TSP #(1-8)
10001	D4 Transit Phase Remote Check Out	TSP #(1-8)
10002	D4 Transit Phase No Call	TSP #(1-8)
10003	D4 Transit Phase No Priority Reservice Inhibit	TSP #(1-8)
10004	D4 Transit Phase Priority Pending	TSP #(1-8)
10005	D4 Transit Phase Defer Next Cycle	TSP #(1-8)
10006	D4 Transit Phase No Adjust	TSP #(1-8)

10007	D4 Transit Phase Recovery	TSP #(1-8)
10008	TransSuite Intersection Status	1 = Comm Failed 2 = Coordinated 3 = Free 4 = Failed 5 = Preempt 6 = Transition 7 = Flash 8 = Offline 9 = Manual Override 10 = Unknown 11 = TSP Active
10009	Preempt active	Preempt # (1-32)
10010	Preempt inactive	Preempt # (1-32)
10011	Intelight preempt track service 2	Preempt # (1-16)
10012	Intelight preempt dwell cycle	Preempt # (1-16)
10013	ASC/3 preempt track clearance	Preempt # (1-10)
10014	ASC/3 preempt cycling	Preempt # (1-10)
10015	ASC/3 preempt cycling delay	Preempt # (1-10)
10016	ASC/3 preempt advancing to flash	Preempt # (1-10)
10017	ASC/3 preempt flash	Preempt # (1-10)
10018	ASC/3 preempt flash delay	Preempt # (1-10)

4.6 STOP SESSION

Path: /StopSession

Method: GET

Produces: text/plain

Parameters: `sessionId` - The session identifier obtained from a previous call to "Start Session".

Possible Responses:

- **"OK"** – Indicates that the command was accepted.

- **“Invalid Session ID”** – Indicates that the session identifier provided by the client is invalid. This could happen if the client is not using the session ID provided by the “Start Session” call, if the client already stopped the session, if the server timed-out the client session due to inactivity, or if the server restarted.
- **<Anything else>** - Indicates some other error has occurred.

This will gracefully stop the client session in the CVDP server. Subscriptions and cached updates will be cleared. The client application should call this when terminating a session. Sessions that have been inactive for at least one minute will automatically be timed-out by the server, with the same effect as stopping a session.

5. INTERFACE TEST CLIENT

5.1 OVERVIEW

This section describes how to use the test client to verify connectivity to the CVDP server and test basic operations. It is not intended to validate encryption certificates. The test client is written in Java, and comes with Java 11. The test client comes already compiled, so there is no need to compile the source code. The source code is provided as a basic programming example.

5.2 REQUIRED FILES

The following table lists the files required to run the test client. These should all be unpacked in a folder together.

File/Folder Name	Description
config.properties	Configuration file for the test client. It includes important settings that should be reviewed/revised before starting the test.
TcsThinClient.jar	Contains compiled code for the test client.
TestCVDP.bat	Batch file to run the test client.
RestTestClient.java	Sample source code.
jersey	Sub-folder containing jars for assorted Apache Jersey libraries and dependencies.
jaf	Sub-folder containing files for the Java Activation Framework library.

5.3 CONFIGURATION SETTINGS

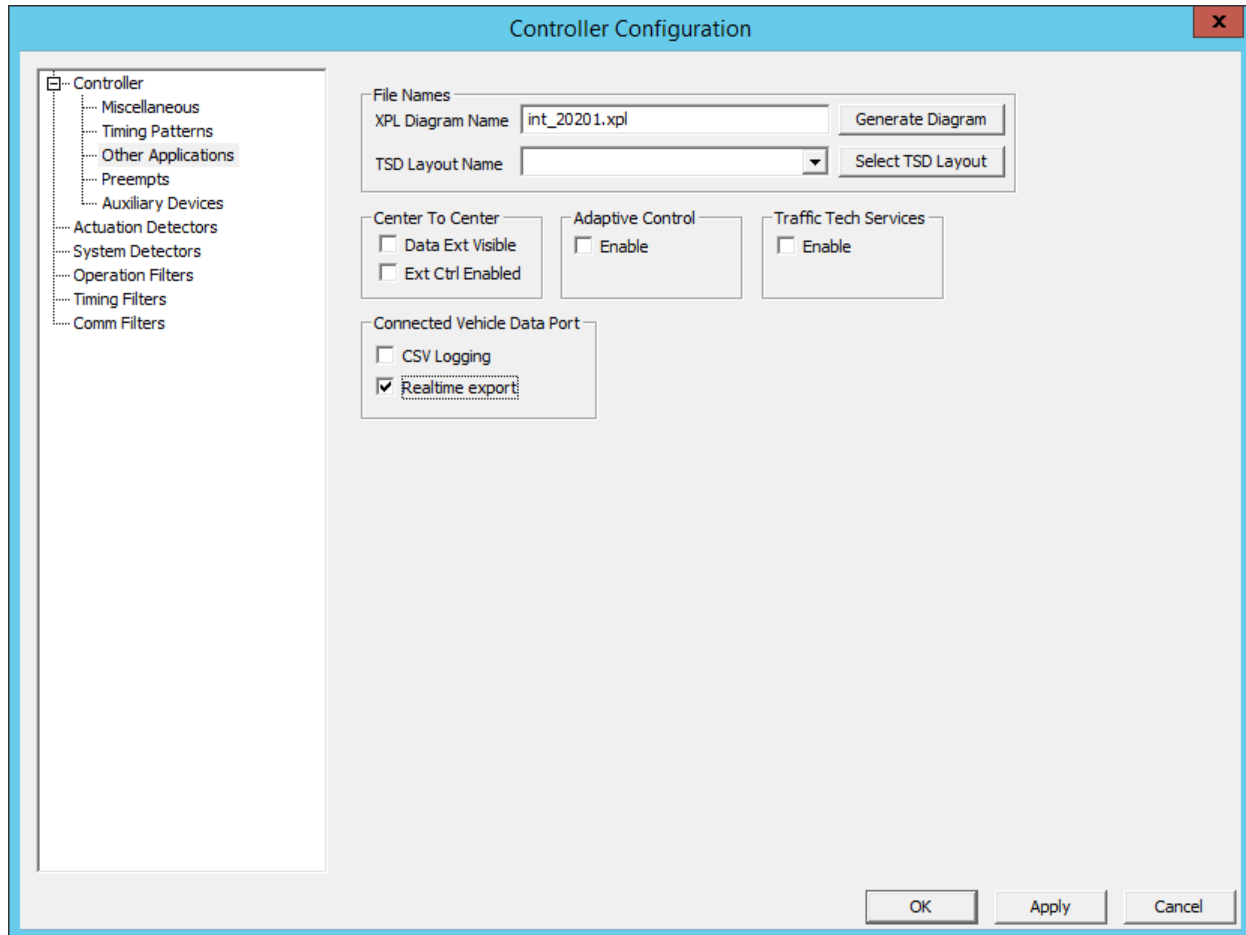
The **config.properties** file contains the configuration settings. These should be reviewed and revised before starting the test. The settings are as follows:

Property Name	Description
URL	This is the URL to access the CVDP REST interface.
status.test.duration	This is the duration of the real-time status test. The units are in seconds. The test client will poll the server once per second, up to this setting's value, and print the server responses to the screen.
status.version	This controls which version of the status messages to poll for. 1 = Use GetRealTimeStatus 2 = Use GetRealTimeStatusV2
subscription.intersection.ids	This is a comma-separated list of intersection IDs. The test client will use these values as the list of intersection identifiers when asking the CVDP server to set subscriptions.

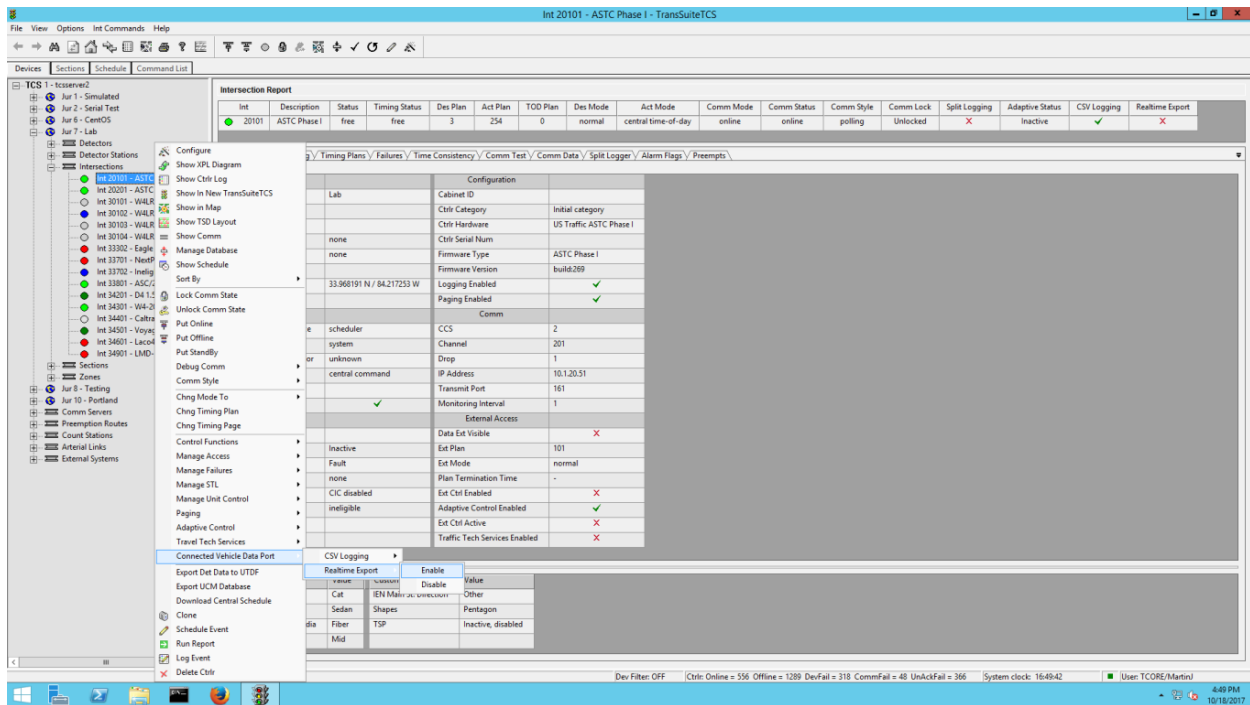
5.4 ENABLING CVDP FROM TCS

There are two ways to enable the CVDP from TransSuite TCS:

1 – Go into the intersection configuration screen, go to the “Other Applications” section, and check the “Realtime export” checkbox. Here is an example:

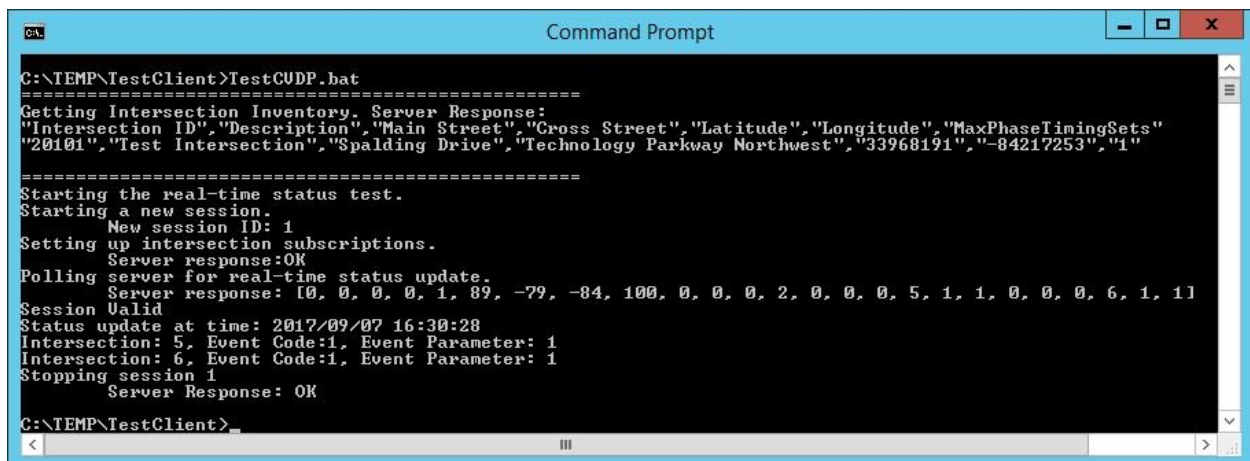


2 – From the intersection context menu. Right-click the intersection and choose *Connected Vehicle Data Port->Realtime Export->Enable*.



5.5 USAGE

Open a command prompt and change into the directory where the test client files are located. Start the test client by running the **TestCUDP.bat** script. An example of the output follows:



The test client starts by obtaining the intersection inventory from the server and printing the response. In this example, we see that intersection 20101 is the only intersection, with a main street of “Spalding Drive”, a cross street of “Technology Parkway Northwest”, latitude of value 33968191, longitude value of -84217253, and only one phase timing set is supported by the controller.

After the inventory, the real-time status test begins. The test client begins by starting a new session with the server. It obtains a session ID of “9”, which it will use in subsequent calls. It sets up subscriptions in the server, again using the session ID it obtained earlier. The intersections it subscribes to are listed in

the **config.properties** file. It proceeds to poll the server for a real-time status update for the intersections it has subscribed to. In this example, it received one status update. The length of this status polling test is configured in **config.properties**, in units of seconds. In this case, only one second was used. The server response displayed shows the raw byte array that includes the event data. The test client then parses the response and prints the meaning of the status byte ("Session Valid"), the timestamp for the set of events, followed by the event data. The test client proceeds to stop the session using the session ID it had obtained earlier, and then terminates.