Real-Time Adaptive Ramp Metering: Phase I – MILOS Proof of Concept

(Multi-Objective, Integrated, Large-Scale, Optimized System)

Final Report 595

Prepared by:
Larry Head
Pitu B. Mirchandani
ATLAS Research Center
Systems & Industrial Engineering Department
The University of Arizona
Tucson, Arizona 85721

December 2006

Prepared for:
Arizona Department of Transportation
206 S. 17th Avenue
Phoenix Arizona 85007
in cooperation with
U.S. Department of Transportation
Federal Highway Administration

This ATRC report is available on the Arizona Department of Transportation's Internet site.

Technical Report Documentation Page

1. Report No. ADOT-AZ-06-595	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle **Real-Time Adaptive Ramp Metering – Phase I:** **MILOS Proof of Concept** **(Multi-Objective, Integrated, Large-Scale, Optimized System)**		5. Report Date December 2006 6. Performing Organization Code
7. Authors Larry Head, Ph.D., Co-Principal Investigator Pitu Mirchandani, Co-Principal Investigator		8. Performing Organization Report No.
9. Performing Organization Name and Address ATLAS Research Center Department of Systems & Industrial Engineering University of Arizona PO Box 210020 Tucson, AZ 85721		10. Work Unit No. 11. Contract or Grant No. JPA 05 015T - R058516P - FY04 SPR-PL-1(65) -595
12. Sponsoring Agency Name and Address Arizona Department Of Transportation 206 S. 17th Avenue, Phoenix, Arizona 85007 ADOT Project Manager: Stephen R. Owen, P.E.		13. Type of Report & Period Covered FINAL REPORT December 2004-June 2006 14. Sponsoring Agency Code
15. Supplementary Notes Prepared in cooperation with the U.S. Department of Transportation, Federal Highway Administration		

16. Abstract

Over the last several years, researchers at the University of Arizona's ATLAS Center have developed an adaptive ramp metering system referred to as MILOS (Multi-Objective, Integrated, Large-Scale, Optimized System). The goal of this project is a "Phase I" implementation of the system for a segment of the I-10 corridor in Phoenix, as part of the Arizona Department of Transportation's (ADOT) Freeway Management System (FMS).

In this Phase I effort the objective was to implement software/hardware components so that detector data from the freeway are provided to MILOS and, in turn, MILOS provides ramp-metering rates, which are subsequently set at the ramp controllers through the ADOT Traffic Management System NTCIP interface. A proof-of-concept demonstration was successful. The integrated system ran successfully for over one hour. It was concluded that the system accomplished the goal of closing the loop between FMS mainline data collection, the MILOS algorithms, and the new NTCIP ramp controllers connected to the ADOT Traffic Management System.

A "Phase II" effort is planned that will identify performance capabilities of MILOS and the communication and detection issues that need to be addressed for wide-scale deployment.

17. Key Words Adaptive Ramp Metering, MILOS, Freeway Management System, Traffic Management System, 2079 Controller		18. Distribution Statement Document is available to the U.S. Public through the National Technical Information Service, Springfield, Virginia, 22161		23. Registrant's Seal Not Applicable
19. Security Classification Unclassified	20. Security Classification Unclassified	21. No. of Pages 70	22. Price	

SI (MODERN METRIC) CONVERSION FACTORS*

APPROXIMATE CONVERSIONS TO SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
	LENGTH			
in	inches	25.4	millimeters	mm
ft	feet	0.305	meters	m
yd	yards	0.914	meters	m
mi	miles	1.61	kilometers	km
	AREA			
in^2	square inches	645.2	square millimeters	mm^2
ft^2	square feet	0.093	square meters	m^2
yd^2	square yards	0.836	square meters	m^2
ac	acres	0.405	hectares	ha
mi^2	square miles	2.59	square kilometers	km^2
	VOLUME			
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft^3	cubic feet	0.028	cubic meters	m^3
yd^3	cubic yards	0.765	cubic meters	m^3
	NOTE: Volumes greater than 1000L shall be shown in m^3.			
	MASS			
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000lb)	0.907	megagrams (or "metric ton")	mg (or "t")
	TEMPERATURE (exact)			
°F	Fahrenheit temperature	5(F-32)/9 or (F-32)/1.8	Celsius temperature	°C
	ILLUMINATION			
fc	foot-candles	10.76	lux	lx
fl	foot-Lamberts	3.426	candela/m^2	cd/m^2
	FORCE AND PRESSURE OR STRESS			
lbf	poundforce	4.45	Newtons	N
lbf/in^2	poundforce per square inch	6.89	kilopascals	KPa

APPROXIMATE CONVERSIONS FROM SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
	LENGTH			
mm	millimeters	0.039	inches	in
m	meters	3.28	feet	ft
m	meters	1.09	yards	yd
km	kilometers	0.621	miles	mi
	AREA			
mm^2	square millimeters	0.0016	square inches	in^2
m^2	square meters	10.764	square feet	ft^2
m^2	square meters	1.195	square yards	yd^2
ha	hectares	2.47	acres	ac
km^2	square kilometers	0.386	square miles	mi^2
	VOLUME			
mL	milliliters	0.034	fluid ounces	fl oz
L	liters	0.264	gallons	gal
m^3	cubic meters	35.315	cubic feet	ft^3
m^3	cubic meters	1.308	cubic yards	yd^3
	MASS			
g	grams	0.035	ounces	oz
kg	kilograms	2.205	pounds	lb
mg	megagrams (or "metric ton")	1.102	short tons (2000lb)	T
	TEMPERATURE (exact)			
°C	Celsius temperature	1.8C + 32	Fahrenheit temperature	°F
	ILLUMINATION			
lx	lux	0.0929	foot-candles	fc
cd/m^2	candela/m^2	0.2919	foot-Lamberts	fl
	FORCE AND PRESSURE OR STRESS			
N	Newtons	0.225	poundforce	lbf
kPa	kilopascals	0.145	poundforce per square inch	lbf/in^2

SI is the symbol for the International System of Units. Appropriate rounding should be made to comply with Section 4 of ASTM E380

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND TERMS

ADOT	Arizona Department of Transportation
ATLAS	Advanced Transportation and Logistics: Algorithms and Systems
ATRC	Arizona Transportation Research Center
C2C	Center-to-Center protocol within NTCIP
CPLEX	Optimization Software from ILOG, Inc.
Det	Detector; detection
FMS	Freeway Management System
Ft	Feet
GUI	Graphical User Interface
i2	Traffic Management System developed by Siemens ITS
IP	Internet Protocol
IPS	Incident Processing System
ITMS	Integrated Traffic Management System
ITS	Intelligent Transportation Systems
MAG	Maricopa Association of Governments
MATLAB	Programming software from The MathWorks, Inc.
MDS	MILOS Data Server
MILOS	Multi-Objective, Integrated, Large-Scale, Optimized System
NTCIP	National Transportation Communications for ITS Protocol
O/D Matrix	Origin-Destination Matrix
PC	Personal Computer
PC-RT	Predictive-Cooperative, Real-Time Algorithm
RADS	Regional Archived Data Server
RHODES	Real-time, Hierarchical, Optimized, Distributed, Effective System
RMS	Ramp Metering System
Sec	Seconds
Seg	Segment
SOAP	Simple Object Access Protocol
TAC	Technical Advisory Committee
TI	Traffic Interchange
UA (or, UofA)	The University of Arizona
VOS	Volume, Occupancy, Speed
VPHPL	Units of "vehicles per hour per lane"

EXECUTIVE SUMMARY

Congested urban freeways result in lost time, driver frustration, higher public and business costs, wasted fuel, decreased productivity, increased accidents, and severe degradation of regional air quality. One successful approach to alleviate some of these congestion effects is to spread the vehicle loading on the freeways through ramp metering, which controls the arrival rates of new vehicles on to the freeway.

Researchers from The University of Arizona's ATLAS (*Advanced Transportation and Logistics: Algorithms and Systems*) Research Center, with support from the Arizona Transportation Research Center (ATRC) of the Arizona Department of Transportation (ADOT), have developed an Adaptive Ramp Metering System, referred to as MILOS (*Multi-Objective, Integrated, Large-Scale, Optimized System*).

MILOS considers the demand on a freeway segment from upstream mainline flow and from on-ramps in the segment, and sets rates adaptively to minimize total delays of the vehicles using this corridor. Earlier simulation based studies have shown that MILOS has the potential to be highly effective in decreasing delays and making freeways flow smoother. In addition, MILOS recovers quickly and smoothly from oversaturated conditions.

ADOT / ATRC will be implementing and testing MILOS on a segment of Interstate 10 in south Phoenix and Tempe, through a two-phase approach. This report documents the work performed on the *SPR 595 MILOS Adaptive Ramp Metering Implementation – Phase I* research project for ADOT. The scope of this Phase I project was to demonstrate the integration of the existing ADOT Freeway Management Systems (FMS), the new ADOT traffic management system (the *i2* Traffic Management System developed by Siemens ITS), and new NTCIP (National Transportation Communications for ITS Protocol)-based ramp metering firmware such that any ramp rate, especially the optimum ramp rate as determined by MILOS, can be downloaded to the ramp meters using the upgraded software/hardware and the communication infrastructure available to the ADOT FMS and the *i2* Traffic Management System.

The scope of the upcoming Phase II effort is to demonstrate and evaluate MILOS' capability to implement adaptive ramp metering rates to positively affect traffic flow performance on the selected freeway corridor, which consists of the following locations on Interstate 10 in southeast Phoenix and Tempe (in the oval overlay in Figure 1, p. 7).

- Broadway Road & I-10 Eastbound
- Baseline Road & I-10 Eastbound
- Elliot Road & I-10 Eastbound
- Warner Road & I-10 Eastbound
- Ray Road & I-10 Westbound
- Warner Road & I-10 Westbound
- Elliot Road & I-10 Westbound

The project required the design, integration and implementation of the interfaces among field ramp-metering controllers, the MILOS software, and ADOT's FMS and *i2* Traffic Management System. The MILOS architecture, algorithms and software were developed in earlier projects, partially supported by ADOT / ATRC.

Figure 2 (see p. 7) illustrates the primary components of the integrated system. MILOS (residing in the server referred to as *phx-MILOS*) receives mainline and ramp detector data from the ADOT Freeway Management System using a binary data stream. The *i2* Traffic Management System and the FMS communicate to the 179 and 2070 controllers using separate channels on the ADOT fiber communications system.

The integrated system and proof-of-concept demonstration was conducted at the ADOT Traffic Operations Center on March 31, 2006, using an *i2* client Graphical User Interface (GUI) to view the status of each ramp controller. In addition, two windows were used to view the MILOS software and the *i2* Center-to-Center (C2C) component. One window showed that MILOS was running and that data was received every 20 seconds from the FMS / IPS (Incident Processing System) server. The *i2* C2C window showed that the desired rates were being sent as the commanded rates to the *i2* interface from MILOS. Finally, the desired rates appeared as the "Active Metering Rates" when implemented on the ramp controller.

The demonstration was considered a success by the ATLAS researchers and the project's Technical Advisory Committee (TAC).

The major outcomes of this research project were:

1. The re-engineering of the MILOS software to communicate with other systems using interfaces for getting streaming detector data (input) from the FMS and sending commands (output) to ramp-metering controllers (to set metering rates).
2. The software interfaces between the different components in the integrated system to operate MILOS: the FMS detector system, MILOS algorithms, the NTCIP compliant ADOT *i2* Traffic Management System, and the ramp meter controllers.
3. Successful demonstration of the integrated system.
4. The identification of issues related to the performance of MILOS due to the partial availability of detector data – which will be addressed in Phase II effort.

The ATLAS Research Center, of the Systems and Industrial Engineering Department at the University of Arizona, is the lead entity that conducted this Phase I research program.

Key partners who contributed towards the writing, software development and/or data gathering include graduate research assistants in the ATLAS Center; Siemens ITS, who coordinated the installation and upgrade of the field ramp meter controllers to new 2070 controllers running NTCIP-compliant firmware; OZ Engineering, who provided the FMS/IPS interface, and ADOT staff from the Information Technology / Transportation Technology Group.

Stakeholders from ADOT and local agencies and in the Phoenix area served on or advised the research project's Technical Advisory Committee; their continual active participation, technical input and support resulted in the project's success.

The following entities were represented on the TAC:

- Transportion Technology Group, ADOT
- Traffic Operations Center, ADOT
- Phoenix Maintenance District, ADOT
- Traffic Engineering Group, ADOT
- Transportation Division, City of Tempe
- Traffic Operations/Signals, City of Phoenix
- Signal Systems, City of Chandler
- Maricopa Association of Governements (MAG)
- Transportation Planning, Maricopa County Department of Transportation
- Federal Highway Administration
- Arizona Transportation Research Center, ADOT

1. INTRODUCTION

1.1 BACKGROUND

Congested urban freeways result in lost time, driver frustration, higher public and business costs, wasted fuel, decreased productivity, increased accidents, and severe degradation of regional air quality. One successful approach to spread the vehicle loading on the freeways and make traffic movement more efficient is ramp metering. Ramp meters control the arrival of new vehicles into the traffic flow on the freeway, allowing for smoother merges into the traffic stream in the right-hand lanes.

The use of ramp metering has been shown to be effective at improving mainline freeway flow, but the impact in the travel corridor, which includes ramps, frontage roads, and adjacent arterials, can be significant. The use of the queue override features is intended to address this concern, but sometimes results in higher than desired ramp metering rates and subsequent breakdown of mainline freeway flow.

Historically, ADOT ramps operate in time-of-day, fixed rate metering mode and, where appropriate, with alternating lanes. Some ramp locations operate a local traffic responsive algorithm that adjusts the ramp-metering rate based on local mainline conditions. All ramps utilize queue spillback detection to override the ramp-metering rate when a queue significantly extends upstream on the ramp.

The MILOS adaptive ramp metering system was developed as part of an ADOT-ATRC research effort in response to the need to develop intelligent strategies to manage congestion in the ADOT freeway system. A key contribution of the research effort was the development of algorithms that consider ramp queues as well as freeway conditions. MILOS [Gettman 1998, Gettman et al., 1999] was the first system to consider the ramp queues as an integral part of the adaptive ramp metering strategy. MILOS' primary objective is to improve freeway flow conditions in a corridor and to recover quickly and smoothly from oversaturated conditions.

1.2 PROJECT HISTORY

In the mid-1990's, researchers at the University of Arizona (UA) studied the various strategies available for setting ramp metering rates, and came to the conclusions that a key consideration missing in proposed state-of-the-art ramp-metering strategies was the limitations of the queue outputs and queue storages at the various on-ramps to a freeway segment. Ramp-metering strategies tested and/or proposed in the US and elsewhere, primarily in Europe, considered only the Origin-Destination demands and the flow on the freeway. Thus these strategies resulted in uneven and sometimes quite large queues on the on-ramps.

Hence, UA developed a ramp-metering approach that considers the trade-offs among the various queues on the ramps as well as flow on the freeway based multi-objective optimization, which was referred to as MILOS (Multi-Objective, Integrated, Large-Scale, Optimized System). This culminated in a PhD dissertation by Gettman [1998]. Simulation-based results showed that MILOS (1) decreased congestion, (2) increased throughput, and (3) recovered faster after incidents or major congestion spikes.

A subsequent ATRC project [Ciarallo and Mirchandani, 2002] investigated the integration of MILOS within the ADOT Freeway Management System. This project identified the need for MILOS to obtain detector data from the FMS and the ramps, and to send ramp metering decisions to the ramp metering controllers. Although in that project MILOS was able to get some emulated detector data from FMS, sending ramp-metering decisions to ramp controllers was not possible at that time because of the existing Type 179 controllers. The issue was primarily a technical challenge based on the existing communications protocol.

In the new technical environment of ADOT's Freeway Management System, with the availability of 2070 controllers to set the ramp metering rates, the introduction of the Siemens *i2* Traffic Management System, and with OZ Engineering, LLC, support at the FMS, it became possible to again try the field operational test of MILOS.

A two-phase program was developed for this field test. Phase I is to develop and integrate the interfaces among MILOS, FMS, *i2*, and 2070 controllers, and to demonstrate that an external algorithm based on detector data and current ramp status, such as MILOS, can set ramp metering rates at the on-ramps. This report documents ATRC research project SPR 595, the Phase I effort.

1.3 PROJECT CONCEPT

ADOT contracted in 2005 with Siemens ITS to deploy 2070 controllers with NTCIP- compliant Ramp Metering Software at the following locations on Interstate 10 (I-10) in southeast Phoenix (*see Figure 1 on the following page*):

- Broadway Road & I-10 Eastbound
- Baseline Road & I-10 Eastbound
- Elliot Road & I-10 Eastbound
- Warner Road & I-10 Eastbound
- Ray Road & I-10 Westbound
- Warner Road & I-10 Westbound
- Elliot Road & I-10 Westbound

Siemens has developed communications to these 2070 controllers for ramp metering with the ADOT fiber system using one serial communications channel with 7 drops through one of the ADOT nodes. The integration with the ADOT *i2* Traffic Management System allows ramp status to be shown and rates to be controlled from the Traffic Operations Center (TOC). Ramp status in this case is: on-line, off-line, metering mode, metering rate, etc. Communications with the ramp meters uses the NTCIP protocol that allows any ramp metering decision to be downloaded (set) to the controller.

Figure 2 shows the primary components of the Adaptive Ramp Metering system. MILOS (residing in a server referred to as *phx-MILOS*) receives mainline and ramp detector data from the ADOT FMS using the binary data stream. *i2* and the FMS communicate to the 179 and 2070 controllers using the ADOT fiber communications system.

The ADOT–MILOS project is intended to demonstrate the operation of MILOS, which includes MILOS' ability to: (1) collect the necessary detector data from the FMS, (2) use this data to determine the appropriate ramp metering rates, and (3) implement these rates at each metering location.

Figure 1: The I-10 Test Corridor in Southeast Phoenix
(corridor indicated by oval)

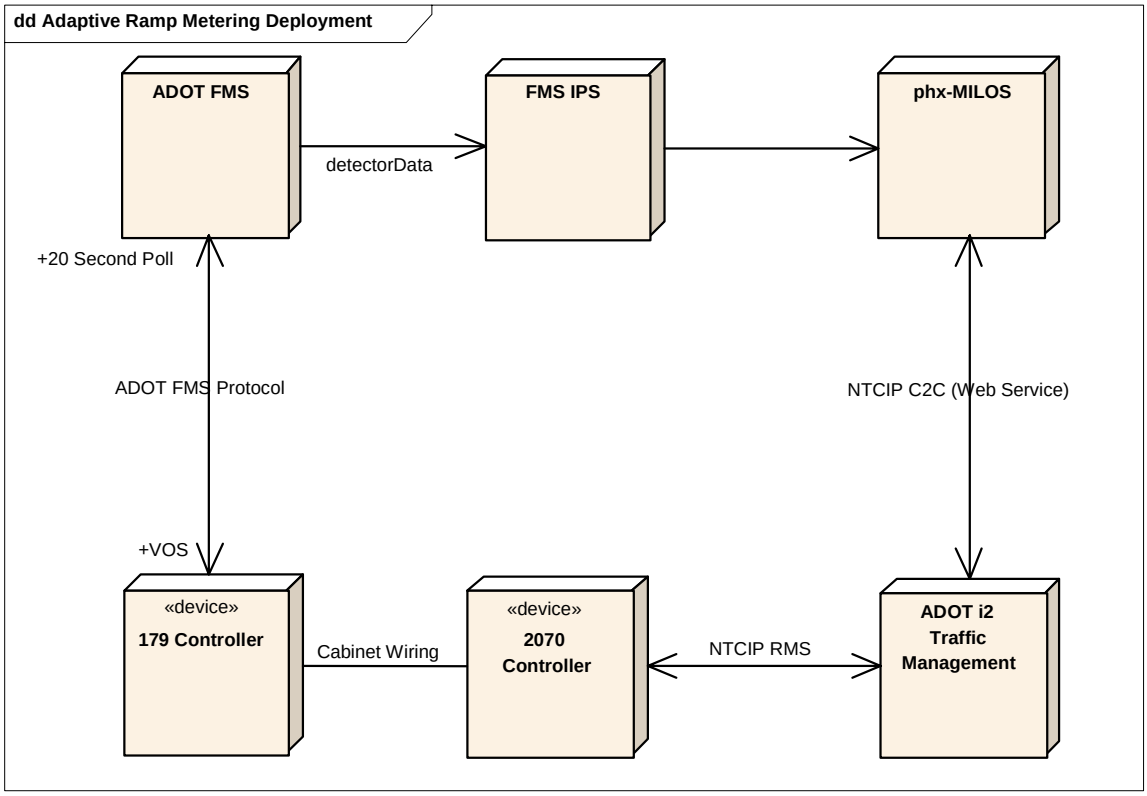


Figure 2: Primary Components of the Adaptive Ramp Metering System
(See also Acronyms section)

The ADOT–MILOS project is intended to demonstrate the operation of MILOS, which includes MILOS' ability to: (1) collect the necessary detector data from the FMS, (2) use this data to determine the appropriate ramp metering rates, and (3) implement these rates at each metering location.

This completed Phase I study is the first of a two-phase effort of the project; it achieved its key goal to develop and integrate the interfaces among MILOS, FMS, *i2*, and the ramp-metering controllers. The future Phase II is to actually implement MILOS in the field and to evaluate its goal to satisfy ADOT's objective to provide effective wide-area traffic-adaptive ramp metering.

1.4 PROJECT SCOPE

This report documents the work performed in 2005-06 by the University of Arizona on the *MILOS Adaptive Ramp Metering Implementation Project – Phase I (SPR 595).* The Arizona Department of Transportation funded this effort through the Arizona Transportation Research Center. The project required the design and implementation of interfaces among field ramp-metering controllers, the ADOT FMS, and MILOS software.

The scope of this project was to demonstrate the integration of the existing ADOT FMS, the new ADOT Traffic Management System (the *i2* system procured from Siemens ITS), and new NTCIP-based ramp metering firmware such that any ramp rate, specifically the optimum ramp rate as determined by MILOS, can be downloaded to the ramps using the upgraded software and hardware, and the communication infrastructure available to the FMS. The MILOS architecture and algorithms were addressed with ADOT support in earlier phases of the RHODES-ITMS research program (Real-time, Hierarchical, Optimized, Distributed, Effective System - Integrated Traffic Management System).

1.5 PROJECT TEAM AND STAKEHOLDERS

This report was written primarily by the co-principal investigators, Larry Head and Pitu B. Mirchandani, both of the ATLAS Research Center, of the Systems and Industrial Engineering Department at the University of Arizona.
Several other individuals also contributed towards the writing, software development, and/or data gathering. In particular, the efforts of the following individuals are acknowledged: Feng Huang, a graduate research assistant in the ATLAS Center; Jim Holmes of Siemens ITS, who coordinated the installation and upgrade of the field ramp meter controllers to modern 2070 controllers running NTCIP-compliant firmware; Tomas Guerra of OZ Engineering LLC., who provided the FMS / IPS interface, and several ADOT personnel who provided technical assistance, specifically Darrell Bingham, Andy Murray and Brian Quinn from the Information Technology / Transportation Technology Group. Siemens ITS provided significant cost share on this project, and the ATLAS Center acknowledges the cooperation of Douglas Gettman and Alan Clelland of Siemens in facilitating this effort.

Stakeholders from ADOT, local agencies and firms in the Phoenix area attended Technical Advisory Committee meetings or served on the TAC; their continual active

participation, technical input and support resulted in the Project's success. The following individuals served on the TAC or attended TAC meetings at various times:

Manny Agah	Traffic Operations Center, ADOT
Darrell Bingham	Traffic Operations Center, ADOT
Jim Decker	Transportation Division, City of Tempe
Ron Doubek	Traffic Operations/Signals, City of Phoenix
Rados Gluscevic	Traffic Engineering Group, ADOT
Kiran Guntupalli	Intelligent Transportaion Systems, Maricopa Association of Governments (MAG)
Alan Hansen	Federal Highway Administration
Glenn Jonas	Traffic Operations Center, ADOT
Sarath Joshua	Intelligent Transportaion Systems, MAG
Leo Luo	Intelligent Transportaion Systems, MAG
Ben McCawley	Signal Systems, City of Chandler
Charles McClatchey	Phoenix Maintenance District, ADOT
Joe McGuirk	Phoenix Maintenance District, ADOT
Steve Owen	Arizona Transportation Research Center, ADOT
Tom Parlante	Traffic Engineering Group, ADOT
Jerry Pfeifer	Traffic Operations Center, ADOT
Manuel E. Sanchez	Federal Highway Administration
Brian Scifers	Signal Systems, City of Chandler
Sonny Sollano	Phoenix Maintenance District, ADOT
Lydia Warnick	Traffic Operations Center, ADOT
Christine Warren	Transportation Division, City of Tempe
Tim Wolfe	Transportion Technolgy Group, ADOT
David Wolfson	Transportation Planning, Maricopa County Department of Transportation

2. SYSTEM ARCHITECTURE

The integration of the MILOS adaptive ramp metering system components is described using the *Unified Process "4+1"* view of system architecture (see Figure 3). The Unified Process is driven by the *Use Case* View. *Use Cases* (see definition below) describe the functions that the system should be capable of performing. The *Design View* describes the structural entities within the system that must perform the desired functions. The Design View will not be described in detail in this report since most of the functions are accomplished by interacting system components and not by new system entities.

The *Dynamic View* is used to describe what entities (classes) are active and operating in parallel. For example, the process that listens for the 20-second updates of detector data from the ADOT FMS IPS runs in parallel to the MILOS algorithms. This view is important to clarify the timing between the different components. The *Implementation View* will be used to describe the different software components (processes – servers, utility functions, MATLAB (programming software from The MathWorks Inc.), etc. that are used to implement the integration and logic of the MILOS adaptive ramp metering system. Finally, the *Deployment View* shows how each of the components defined in the Implementation View are deployed on different computers (or traffic controllers) when the system is running. Figure 2 (page 7) showed the detailed Deployment View of this 4+1 system architecture.

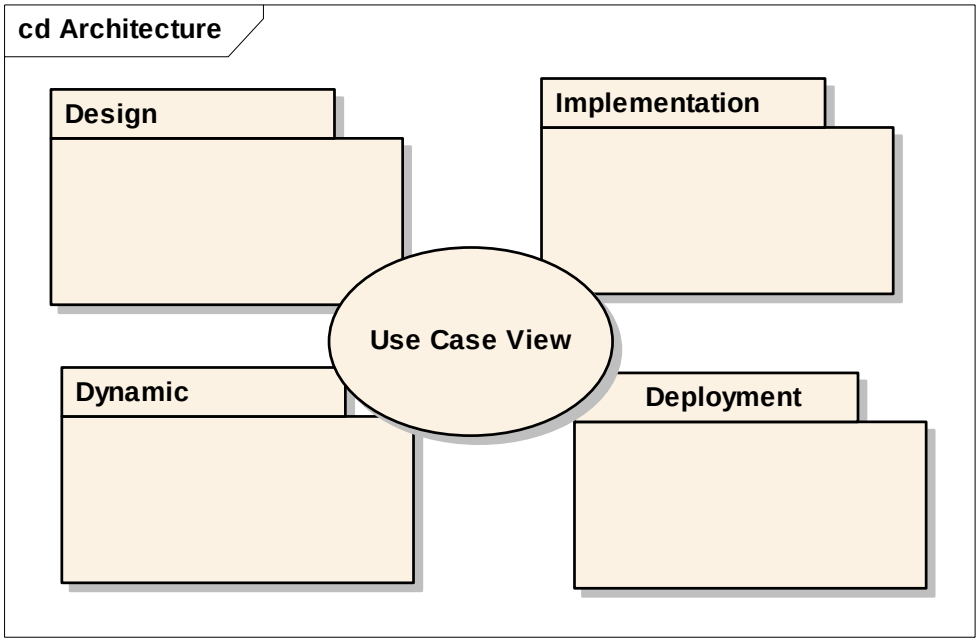


Figure 3: The "4+1" View of a System Architecture

(See also Acronyms section)

2.1 MILOS ADAPTIVE RAMP METERING USE CASES

Definition: **Use Case**: A use case is a term used in the software industry to describe a technical specification of a software system function. It is a step-by-step specification of how an outside entity (such as a user or another system) should interact with the system being specified to accomplish some desired task. Use cases include a primary flow (the main sequence), alternative flows (for exception handling and other alternative methods for accomplishing the task), preconditions (what must be true for this use case to function), post conditions (what might be changed after the use case is executed), and supplementary requirements (other requirements, such as, technology requirements).

Definition: **Scenario**: A scenario is a term used in the software industry to specify a particular instance of a use case where the parameters and preconditions are exactly specified to represent some real application. For example, in a use case that specifies how MILOS will interface to the ADOT FMS, the Internet Protocol (IP) address, port number, and other parameters constitute a scenario. In addition, for this research project, the specification of a scenario includes the use of hypothetical or historical volume data from the freeway and ramp segments for the study corridor.

***Definition*: Actors**. Actors are outside entities that the software interacts with to accomplish the desired function in a use case. The actors involved in the integration of MILOS and the ADOT FMS include the FMS IPS Server which provides freeway data; the ADOT *i2* Traffic Management System which sets the desired metering rate on the Ramp Controllers; and the Operator who starts/stops the MILOS software components.

The sequential use cases included for the ADOT MILOS implementation include:

1. Start MILOS Data Server
2. Start Center-to-Center (C2C) Server
3. Read FMS Detector Data
4. Run MILOS Algorithms
5. Compute Optimal Ramp Metering Rates
6. Set Ramp Metering Rates

These are the high-level use cases that describe how the integrated system functions. Additional use cases, such as specifying the IP address of the FMS IPS Server, selecting detectors from the FMS IPS data block, etc., are not described in this report. These use cases are addressed through the software functions covered by the high-level use cases.

The Use Case Diagram (Figure 4) shows all of the high-level use cases and the associated actors. The detailed specifications for each use case are presented in Appendix A.

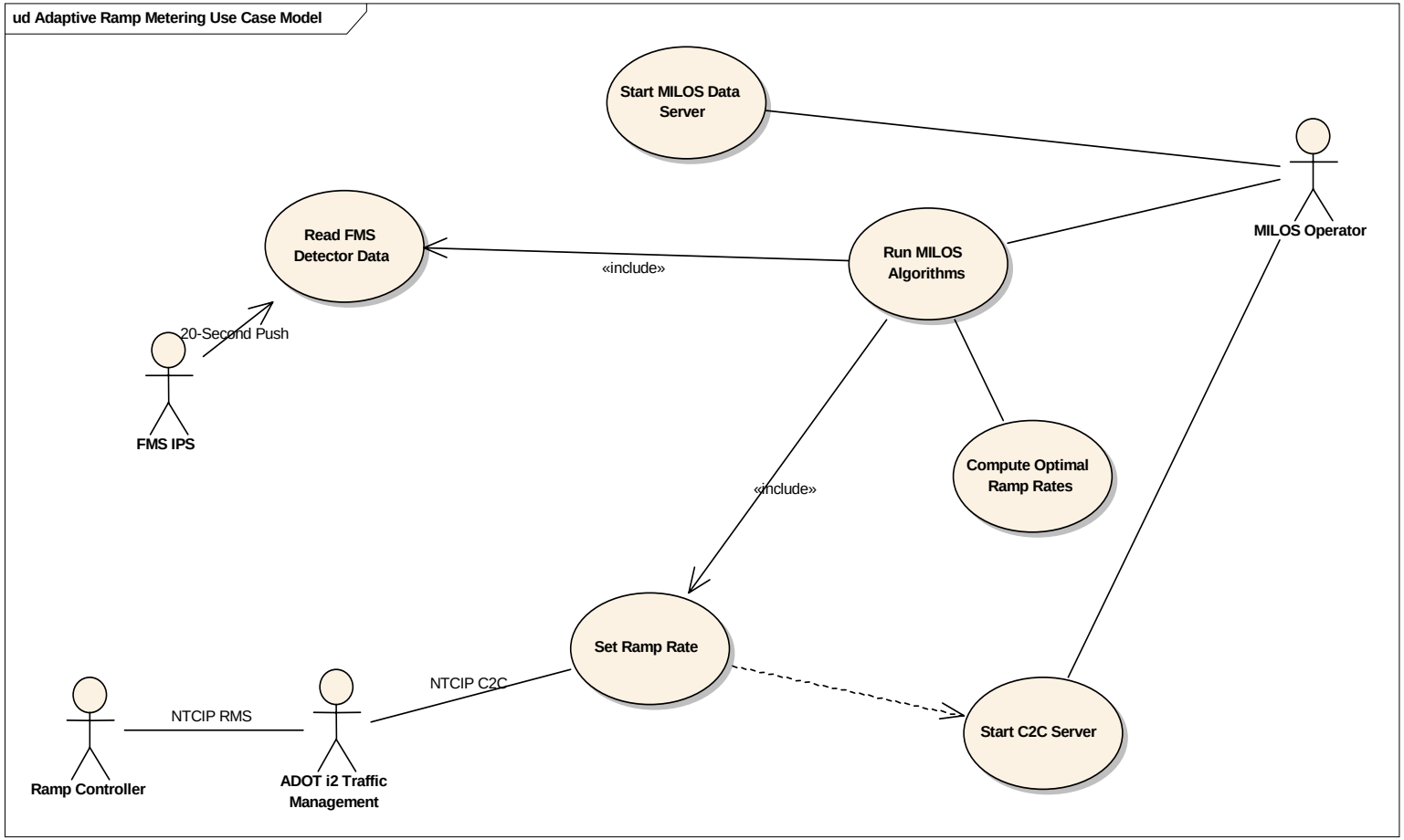


Figure 4: MILOS Adaptive Ramp Metering Use Case Diagram

(See also Acronyms section)

2.2 MILOS SOFTWARE INTEGRATION ARCHITECTURE

Figure 5 shows the software components that were implemented to allow the MILOS algorithms to realize the use cases described in the previous section.

The MILOS Data Server is a program that runs as a background process in a UNIX environment. It listens for the connection for the ADOT FMS IPS Server to establish a client connection and transfer the FMS Detector Data Message (defined in Appendix 7.2; see also Figure 6). It must be started by the MILOS Operator. The MILOS Data Server is responsible for aggregating the 20-second data into 1-minute data for use in the MILOS algorithms.

The MILOS algorithms run within the MATLAB environment and call the CPLEX optimization software (from ILOG, Inc.) to solve the ramp meter rate optimization problem. The MILOS algorithms call a special function that can read data from the MILOS Data Server and call a Utility program, referred to as C2C Set Rate, which translates the optimized ramp metering rates into NTCIP C2C messages for communication via the ADOT Traffic Management System (*i2*) to the ramp controllers in the field.

The MILOS Data Server program was developed in the C Programming Language. The C2C Set Rate utility was developed in Java using the Java Web Services standards (as specified by Siemens ITS).

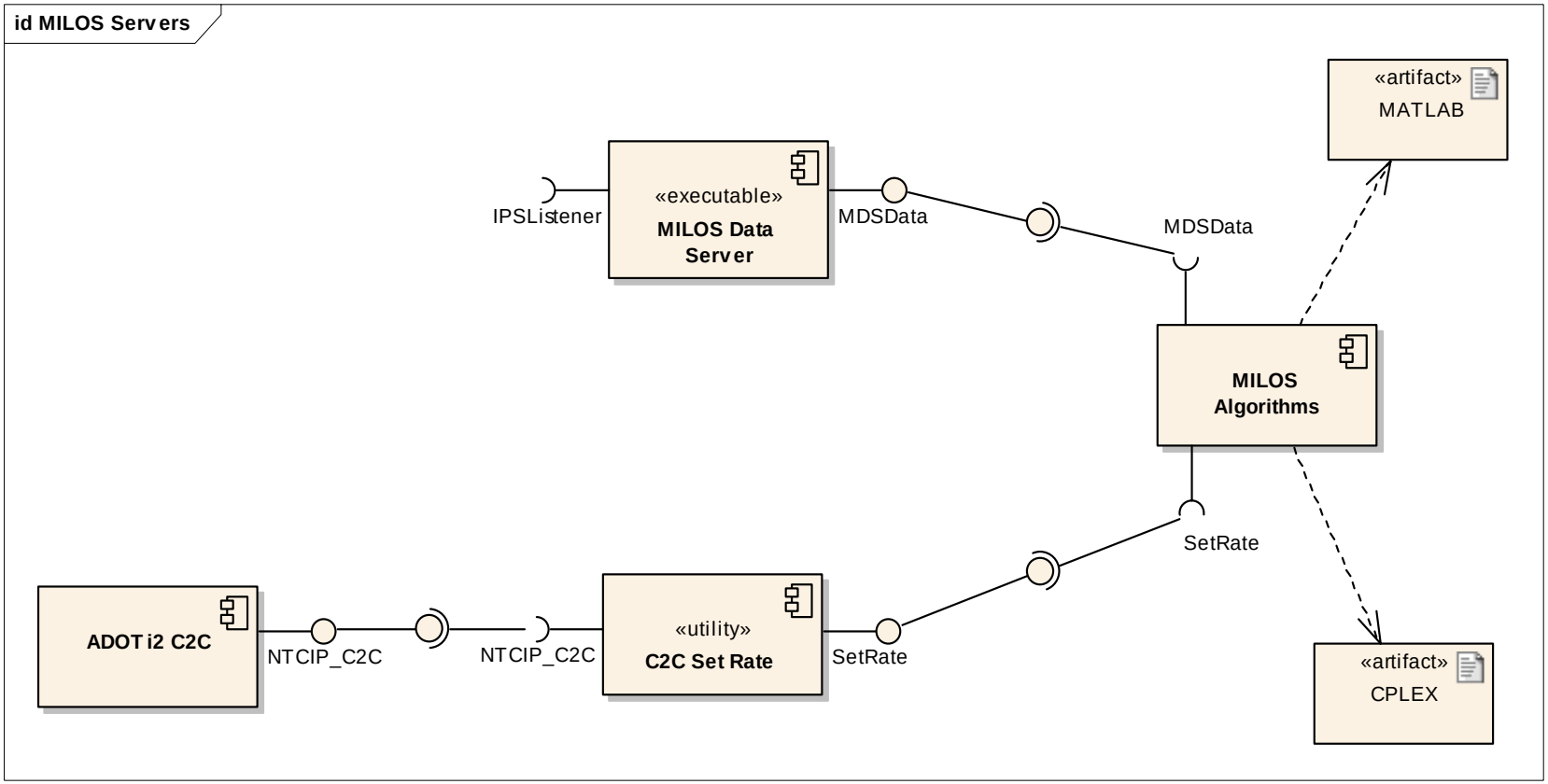


Figure 5: Implementation Diagram: MILOS and Supporting Services & Utilities

(See also Acronyms section)

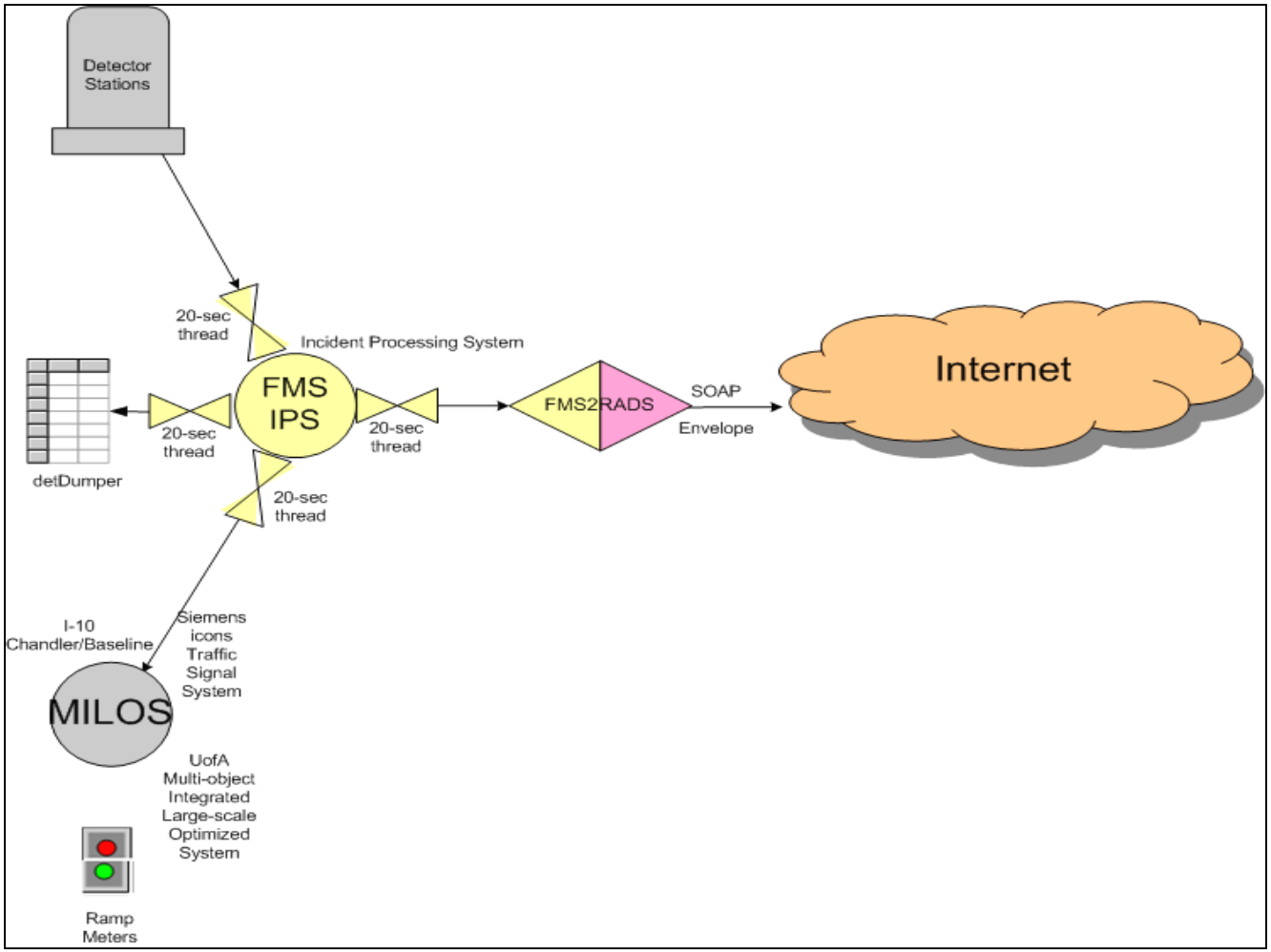


Figure 6: Message Passing from/to FMS IPS

(See also Acronyms section)

(*from document provided by Oz Engineering LLC; see Appendix 7.2*)

3. CONFIGURATION OF MILOS

The MILOS system for deployment in the Phoenix corridor was configured in two parts: the eastbound section of I-10 from Broadway Road to Warner Road; and, the westbound section of I-10 from Warner to Baseline Road. This section summarizes the configuration data that was used in the Phase I Proof-of-Concept effort. Future efforts may require modification of this configuration.

3.1 EASTBOUND I-10

Figure 7 and Figure 8 show the eastbound section of the I-10 study corridor. The corridor is divided into seven sections, and each section is further divided into segments. These divisions form the basis of the dynamic control models used in the MILOS optimization algorithms. Table 1 summarizes the data used to describe the eastbound section of the I-10 study corridor. This data includes the length of each section, the number of lanes, the number of lanes on the on-ramp, and the free flow speed.

Most of the sectional segments are associated with mainline system detectors to allow measurements to be used to estimate flow and density – which are the state variables of the dynamic control models. Some segments do not have detection, and other segments have detectors that are either not working or have been decommissioned.

Table 1. Eastbound I-10 Network Description

Section Number	1	2	3	4	5	6	7
Section Length (feet)	5,085	4,485	2,015	7,250	3,465	5,635	4,500
Number of Lanes	5	4	5	4	4	4	4
Lanes per On-Ramp	2	0	2	2	0	2	2
Free Flow Speeds	55	55	55	55	55	65	65

3.2 WESTBOUND I-10

Figure 9 and Table 2 provide the corresponding information for the westbound segments of Interstate 10. Also, as shown in Figure 9, data for some decommissioned and non-decommissioned westbound detectors are needed for MILOS, as configured currently.

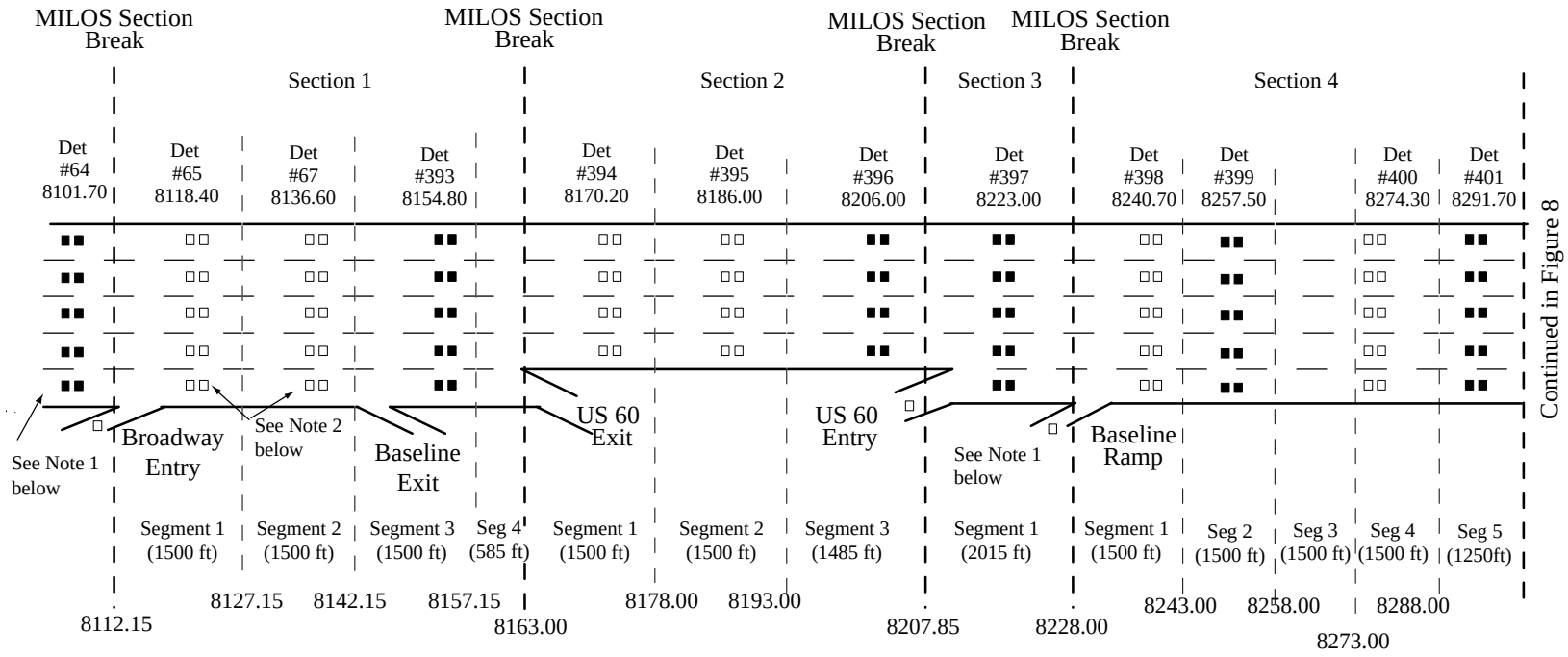


Figure 7: Eastbound I-10 Study Corridor (North Portion) Detection and Status, Sections 1-4

(See also Acronyms section)

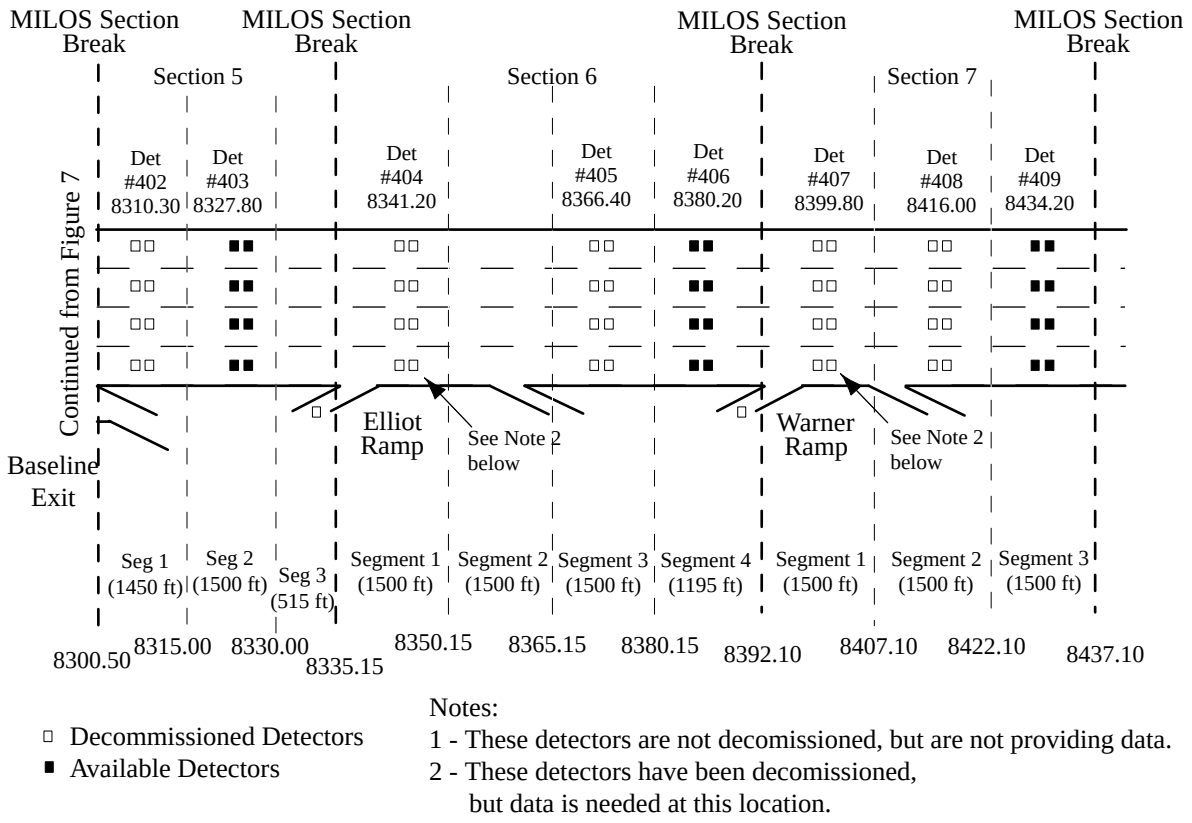


Figure 8: Eastbound I-10 Study Corridor (South Portion) Detection and Status, Sections 5-7

(See also Acronyms section)

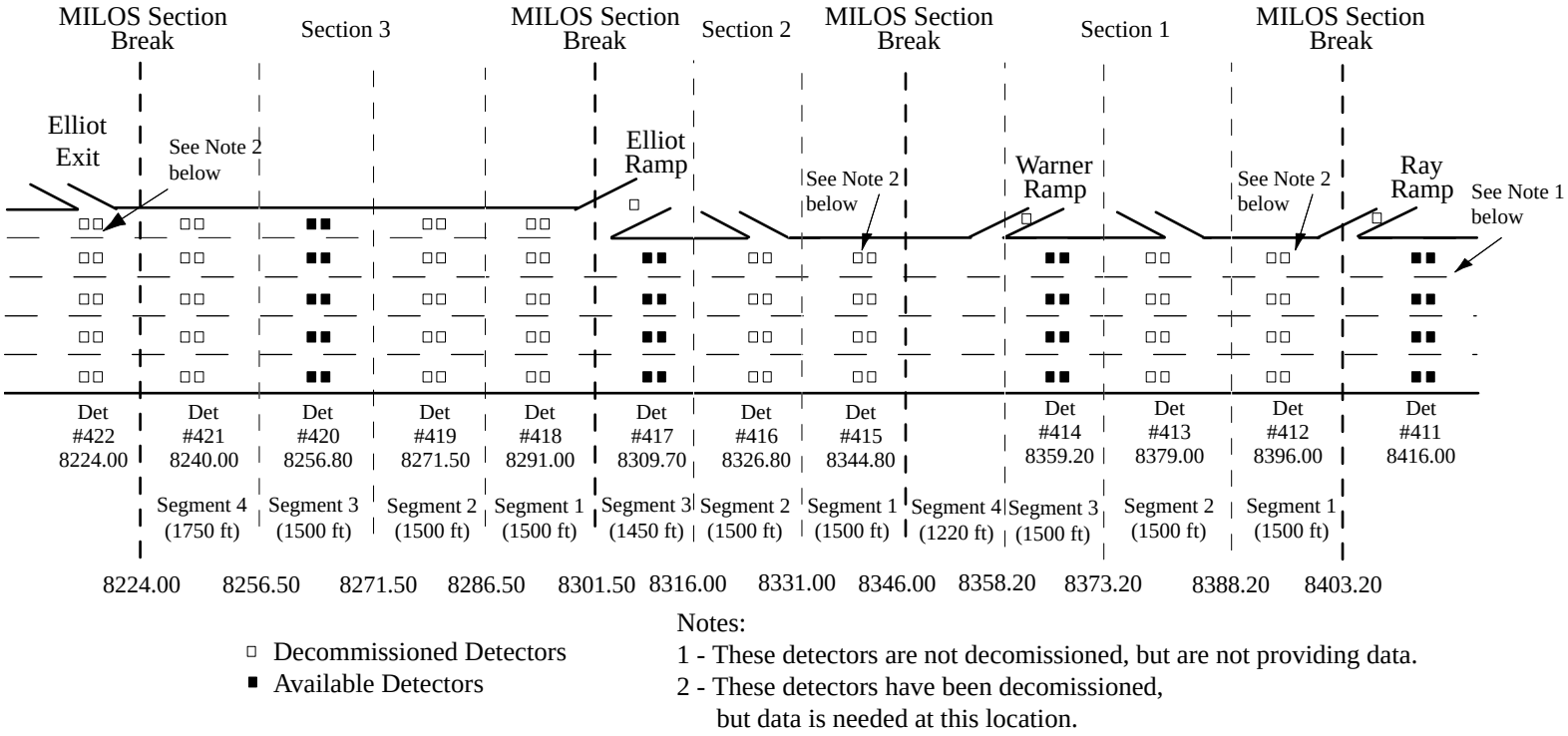


Figure 9: Westbound I-10 Study Corridor Detection and Status, Sections 1-3

(See also Acronyms section)

Table 2. Westbound I-10 Network Description

Section Number	1	2	3
Section Length (feet)	5,720	4,450	6,250
Number of Lanes	4	4	5
Lanes per On-Ramp	2	2	2
Free Flow Speeds	65	65	55

The list below identifies several Phase I detection data issues that will need to be addressed in Phase II for this project's current MILOS system to be fully successful:

1. In Figure 7, at the entrance to the corridor network for eastbound traffic, Detector Set #64 is not providing data even though it is a detector station that has not been decommissioned. This data is vital to the operation of the adaptive control algorithms. (See discussion in Section 5 on using additional upstream detection at this location).
2. Data is needed from either Detector Station #65 or #67 (Figure 7). This site is between the on-ramp at Broadway and the off-ramp at Baseline. Without data from this section it will be difficult to estimate the density and speed accurately, as is needed for the current optimization algorithms.
3. Data is also needed from Detector Stations #404 and #407 (Figure 8) for the same reasons as in Item 2 above.
4. In Figure 9, for westbound traffic, Detector Station #411 does not provide data even though it is designated as operational.
5. Also, as for Items 2 and 3, data from Detector Stations #412, #415, and #422 are needed for the current MILOS algorithms.

These issues on this unavailability of detector data will be thoroughly investigated in Phase II to examine which detector stations "must be operational," and which detectors are not essential for effective adaptive ramp metering using the current MILOS program. Further collaboration on what practical options can best be applied to resolve the issues will be a high initial priority.

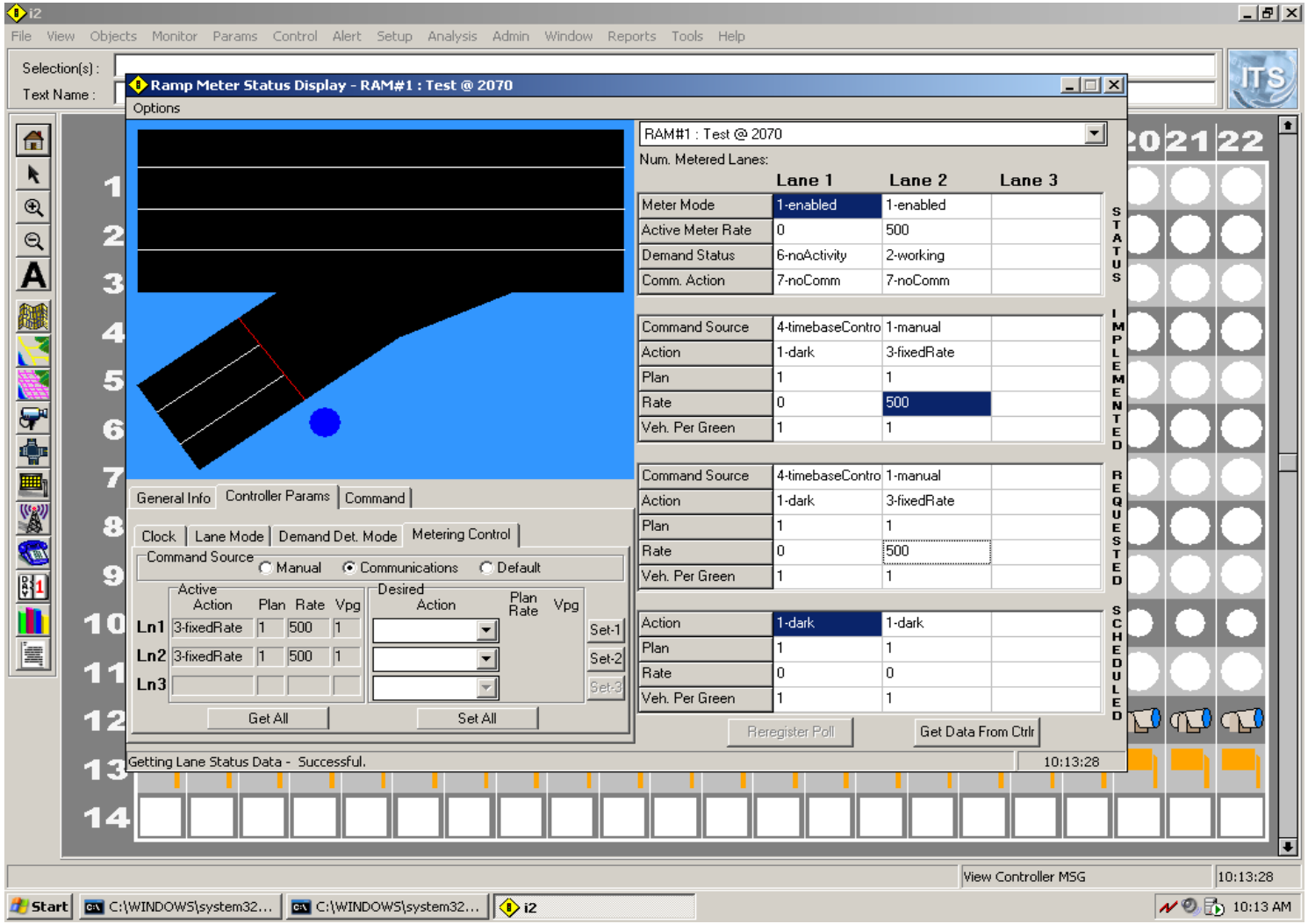


Figure 10: The *i2* User Interface showing Ramp Controller Status

4. PROOF-OF-CONCEPT DEMONSTRATION

The primary purpose of this Phase I research project was to demonstrate the completed integration of MILOS with the FMS freeway data collection and incident processing systems (FMS IPS) and the new NTCIP-based ramp meter controllers (via the ADOT *i2* Traffic Management System). Prior to the final demonstration for the TAC members, each ramp meter cabinet was visited and the metering output to the signal heads was disabled. This prevented the 2070 controllers from displaying metering indications to the public during the test (during non-metering hours of operation).

The integrated system and proof-of-concept demonstration was conducted at the ADOT Traffic Operations Center on March 31, 2006, with an ADOT laptop computer connected to the *i2* server. An *i2* client interface (GUI) was used to view the status of the ramp controller (see Figure 10). This allows monitoring of the Active Metering Rate and the Commanded Metering Rate. In addition, two windows were used to view the MILOS software and the *i2* Center-to-Center (C2C) component. A telnet window showed that MILOS was running and that data was received every 20 seconds from the FMS/IPS server, which was aggregated into 1-minute data since the current version of MILOS uses 1-minute intervals. It then showed the different MILOS algorithms running and the resulting desired ramp metering rates. The *i2* C2C window showed the desired rates being sent as the Commanded Rates to the *i2* interface from MILOS. Finally, the desired rates appeared as the Active Metering Rates when implemented on the ramp controller. See Figure 2 for data/command flows in the system.

The demonstration ran successfully for over one hour. It was concluded that the system accomplished the goal of closing the loop between FMS mainline data collection, the MILOS algorithms, and the new NTCIP ramp controllers connected to the ADOT Traffic Management System.

During the demonstration it was observed that the first ramp on eastbound I-10, at Broadway Road, was metered at a low rate and most of the others were at rates higher than the maximum rate configured on the controller. The low rate at the Broadway ramp was likely due to missing upstream freeway data. The high rates at the other ramps were consistent with the time-of-day (no metering required) operation. In the case where a rate was higher than the maximum rate configured on the controller, the controller defaulted to the maximum allowed metering rate. This was discussed as a feature desired by the Transportation Technology Group since they will have the final approval of the range of allowable metering rates.

The issue of available and missing detector data was a major concern to both the research team as well as the Technical Advisory Committee. It was decided that the study into which detectors are essential, and which may be unavailable or made unavailable without serious degradation of MILOS, will be listed as a separate explicit task in the upcoming Phase II effort for the implementation of MILOS.

5. LESSONS LEARNED AND CONCLUSIONS

The objective of this study was to conduct a successful proof-of-concept demonstration of an integrated adaptive ramp metering system for ADOT, showing the data and information flows among FMS detector stations, the new NTCIP based ramp meter controllers, the new *i2* Traffic Management system and the MILOS prototype system. This was necessary since previous efforts using the legacy ADOT FMS system were not successful. This demonstration was successful, which means it is highly likely that future efforts to deploy an adaptive ramp meter systems, such as MILOS, will be successful.

This effort was "Phase I" in the implementation of MILOS Adaptive Ramp Metering System. A "Phase II" effort has been approved and is slated to begin in early 2007. The goal of the Phase II project will be to demonstrate and evaluate MILOS' capability to implement adaptive ramp metering rates that positively affect traffic flow performance on a freeway corridor.

Phase II will identify and evaluate performance capabilities, deficiencies, and issues that need to be addressed for wide-scale deployment of MILOS. Briefly, in Phase II, the ATLAS team will: (1) simulate the study corridor, (2) study detection needs, including the identification of "must have detection" and the development of algorithmic methods to mitigate the degradation of performance due to missing or decommissioned detectors (*see discussion in last section*), (3) develop software to communicate local on-ramp detection to MILOS via *i2*, (4) integrate the complete MILOS system for the test site, and (5) perform simulation testing, bench testing and field testing of the installed MILOS software and associated software/hardware systems.

There were several key lessons learned in this research project, as are listed below.

1. The use of the NTCIP standards was an enabling factor in this success. The ability for MILOS to read field data and communicate with the Traffic Management System and ramp Controllers was the missing link in the previous efforts. Making the connection between the existing FMS detector data (via the FMS IPS Server), an adaptive ramp metering algorithm, and setting the desired rates on the ramp controllers is crucial in the implementation of a system like MILOS.

2. Prior to the start of this project, the ATLAS Team assumed that detector density (e.g. every 1/3 mile and every 1 mile) in the field test segment will remain the same during the MILOS field test. However, several detector sites were decommissioned during the term of the project. The economic consideration of decommissioning is understandable. However, in locations where significant changes in flow occur, such as between the I-10 on-ramp at Broadway and the off-ramp at Baseline, technical considerations might warrant retention of key detectors. Phase II efforts of this project will include the investigation of methods to estimate flow in areas where it is needed for MILOS but where the detectors have been decommissioned or were never available.

3. The dependence of MILOS on specific mainline entry detection, such as the mainline detection just upstream of Broadway (Detector Station #64) for eastbound traffic, should be addressed by considering additional upstream detection that may be available at the FMS, and propagating the measured flows downstream. This will reduce the dependence of adaptive control algorithms, such as MILOS, on specific detectors, as well as provide additional prediction time horizon for making control decisions.

4. Phase II efforts should carefully consider all on-line inputs available to MILOS, including inputs from on-ramp and off-ramp detectors, time-of-day historical volumes, and, if available, detector and signal data from the arterials at interchanges to estimate and predict short-term traffic demands, all of which should make MILOS even more responsive.

5. It should be remarked that implementing and testing of a new prototype system is a "team" effort that links, in this case, ATLAS, ADOT, Siemens, the ADOT / FMS subcontractors, and ATRC. A delay by any one member of the "team" weakens the link and puts the project schedule and scope at jeopardy. Thus it is essential that in Phase II of the effort, the ADOT leadership and ATLAS leadership work together to make sure that the project tasks are done right and done in time, and that the delays are minimized so that the project is successful in scope and schedule.

6. REFERENCES

Ciarallo, F.W., and Mirchandani, P.B., (2002) *RHODES-ITMS-MILOS: Ramp Metering System Test*, Final Report 481, Arizona Department of Transportation, Phoenix, AZ.

Gettman, D., Head, K.L., and Mirchandani, P.B., (1999) *RHODES-ITMS Corridor Control Project*, Final Report 462. Arizona Department of Transportation, Phoenix, AZ.

Gettman, D., (1998) *A Multiobjective Integrated Large-Scale Optimized Ramp Metering Control System for Freeway/Surface-Street Traffic Management*, Ph.D. Dissertation, University of Arizona, Department of Systems and Industrial Engineering.

7. APPENDICES

Appendix 7.1 USE CASE SPECIFICATIONS

Appendix 7.2 FMS DETECTOR DUMPER PROTOCOL DESCRIPTION

Appendix 7.3 SIEMENS ITS *I2* CENTER-TO-CENTER SPECIFICATIONS

Appendix 7.1 USE CASE SPECIFICATIONS

Name: Start MILOS Data Server (Case 1.0)

Brief Description:

The MILOS Operator will start a service that listens to a specified port for the FMS IPS Server to send a message containing the 20-second FMS Freeway Data. Primary Flow:

1. The use case starts when the operator starts the fms Interface Program (named c2c and located in the /usr/local/milos/c2c directory)
2. The server will listen to port 2500 for a connection from the FMS IPS client
3. When a connection is established
    a. Read the message from the socket buffer
    b. Close the socket
4. Parse the message
5. Check the message CRC for errors
6. Separate the detector data into the east and west bound network data
7. Wait for a new connection (step 2).

Alternative Flow(s):

3. (replace step 3a) if there is no message in the socket buffer, record the event to the fms.log file.
4. (replace step 4) if there was no message, skip parsing the message, go to step 7.
5. (replace step 5) if there is an error in the message CRC, record the error in the fms.log file, discard the data.

Preconditions

The MILOS Data Server must be configured to list port 2500 for messages from the FMS IPS Server.

Post condition

A stream of packets containing the FMS detector data will be available for reading by the MILOS algorithms.

Supplementary Requirements

1. The IP address of the client (ADOT IPS Server) must be validated before the message is read by the MILOS Data Server.

Auxiliary Information

Ramp detector numbers can be found in the following spreadsheet:
2005-0105-ninth-draft-decommission-eligible1.xls

Name: Start C2C Server (Case 2.0)

Brief Description:

The MILOS Operator will start a service that listens to a specified port for the C2C Set Rate utility to send a message containing the desired ramp metering rate(s) for one or more ramp controllers. The C2C Server is a component of the *i2* Traffic Management System.

Primary Flow

1. The use case starts when the operator starts *i2* NCTIP C2C Service.
2. The server will listen to port 3500 for a connection from aC2C client

3. When a connection is established, the C2C service checks the subscription status of the requesting client. If the client has an active connection subscription, the C2C service will accept command messages.
4. Wait for a new connection (step 2).

Alternative Flow(s):

3. (replace Primary Flow 3.) If there is no valid subscription for the requesting client, the service will return an error message and request that the client provide valid security information to establish a connection.

Precondition

ADOT IT Staff must provide the port number that the ADOT Traffic Management System (*i2*) that has been configured to for NTCIP C2C services.

Post condition

A valid subscription will exist and the client will be able to send ramp metering rates as desired.

Supplementary Requirements

SR.1. The communication protocol for NTCIP C2C service as implemented by Siemens ITS is described in Appendix 7.3.

Auxiliary Information

None

Name: Read FMS Detector Data (Case 3.0)

Brief Description:

The MILOS Algorithms will read detector data from the MILOS Data Service once per minute as needed by the MILOS algorithms

Primary Flow

1. The Use Case begins every minute when the MILOS algorithms requests to read detector data
2. The MILOS algorithms will call a special function that will read data from a port specified on the MILOS data service..

Alternative Flow(s):

None

Precondition

The MILOS Data Service is running (Use Case 1.0)

Post condition

MILOS Algorithms will have read the most recent 1 minute detector data.

Supplementary Requirements

None

Auxiliary Information

None

Name: Run MILOS Algorithms (Case 4.0)

Brief Description:

The MILOS Algorithms run in MATLAB – a computational software package used for engineering analysis and design.

Primary Flow

1. The Use Case begins when the MILOS Operator starts MATLAB
2. The MILOS Operator sets MATLAB to the "No Display" mode (to disable graphic output from MILOS)
3. The MILOS Operator calls the MATLAB function 'runsim_NEW"
4. The Use Case ends when the MILOS Operator terminates the "runsim_NEW" operations (Ctl-C).

Alternative Flow(s):

None

Precondition

The MILOS Data Service is running (Use Case 1.0)

Post condition

The active working directory must be MILOS_EB for the eastbound implementation and MILOS_WB for the westbound implementation. Each working director must have the configuration for the MILOS Network.

Supplementary Requirements

None

Auxiliary Information

None

Name: Compute Optimal Ramp Metering Rates (Case 5.0)

Brief Description:

The MILOS Algorithms in MATLAB utilize the CPLEX Optimization library to solve the mathematical programming optimization model.

Primary Flow

1. The Use Case begins when the MILOS algorithm calls the CPLEX Optimization library
2. The MILOS algorithm passes the optimization formulation to CPLEX
3. CPLEX returns the optimal ramp metering rates to MILOS.
4. The use case ends.

Alternative Flow(s):

None

Precondition

CPLEX must be installed and available to MATLAB.

Post condition

The MILOS algorithms will know the current and new ramp metering rates.

Supplementary Requirements

None

Auxiliary Information

None

Name: Set Ramp Metering Rates (Case 6.0)

Brief Description:

Optimal ramp metering rates as computed by MILOS nee to be set on the field controllers.

Primary Flow

1. The Use Case begins when the MILOS algorithm calls the Set Rate utility/
2. The Set Rate utility establishes the NTCIP C2C connection to the ADOT Traffic Management System (*i2*) C2C interface
3. The Set Rate utility sends the set rate message with the new rates to the *i2* system.
4. The Set Rate utility receives an acknowledgement for the *i2* C2C interface.
5. The Use Case ends.

Alternative Flow(s):

1. None

Precondition

MILOS must be running and active.

Post condition

The desired ramp metering rates have been sent to the ADOT Traffic Management System (and to the ramps).

Supplementary Requirements

None

Auxiliary Information

None

References

Appendix 7.3 – Siemens ITS C2C Specification

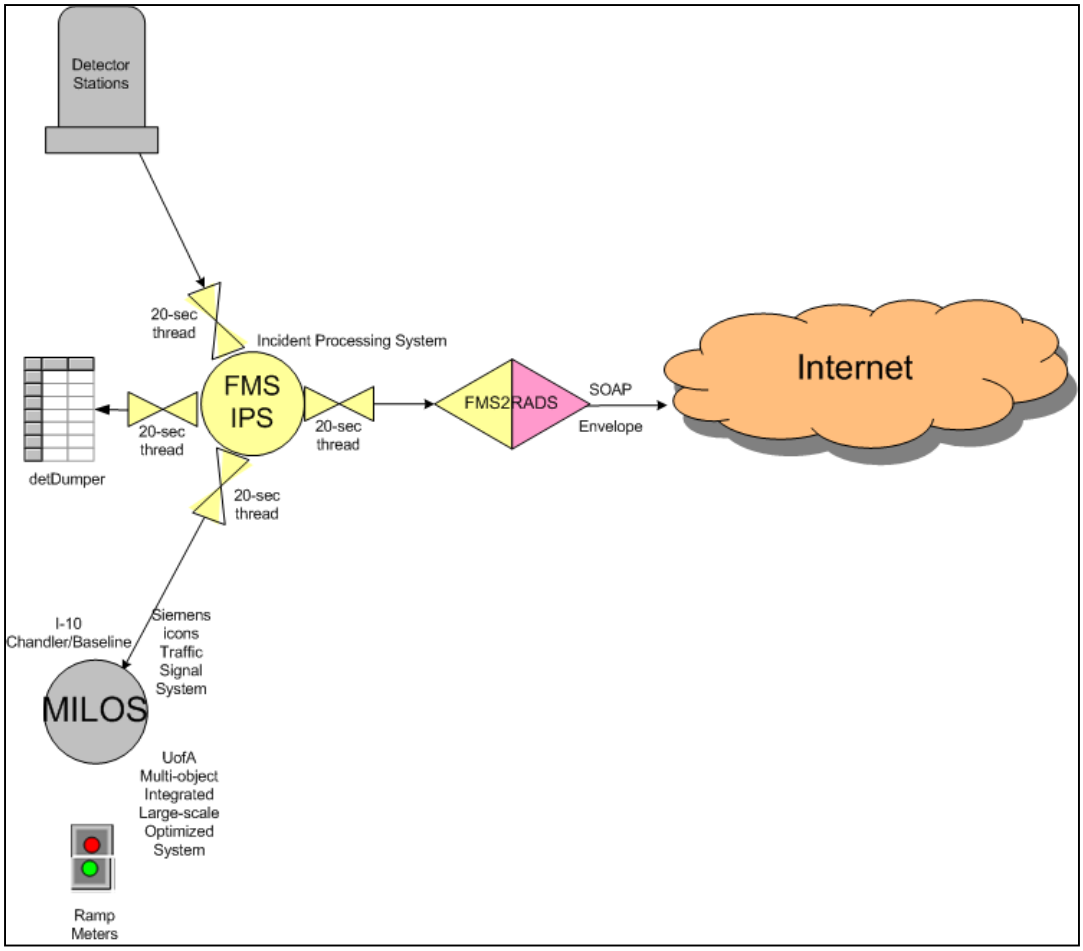
Appendix 7.2 FMS DETECTOR DUMPER PROTOCOL DESCRIPTION

(The text of this section was provided by Oz Engineering, LLC, on behalf of ADOT)

7.2.1 Introduction

This document describes the content of the message sent by the FMS Incident Processing System (IPS) to various subsystems. A simplified data flow diagram is shown below:

1. Detector Station data, which includes volume, occupancy and trap travel time (described in next section) is received by the FMS IPS process every 20 seconds.
2. When this 20-second data is received, the IPS spawns a thread and writes this data to a file on disk; this is known as the detDumper data.
3. A second thread is spawned by the IPS process and it sends the detDumper data message to the FMS2RADS process. This process, in turn, places a SOAP message envelope around the message body and sends it to the Regional Archived Data Server (RADS) through the Internet.
4. A third IPS process thread is spawned and it sends the detDumper data message to a MILOS process. This message does not have a SOAP envelope around it. The contents of this message are described in the following section.



To receive a detDumper data message, the recipient must follow the following procedures:

1. Accept a socket connection on a designated port: xxx.
2. The message will arrive every 20 seconds; message contents described in next section.
3. If the recipient is not available to accept a socket connection, the IPS thread will drop the data packet and not send any data.

7.2.1 FMS Detector Station Data

Each detector station may have up to 8 lanes of detectors. Each lane is normally configured as a dual-loop speed trap, although single-loop arrangements are also accommodated. The structure of the returned data is the same for both loop configurations with only the data values changed as appropriate. Two standard detector assignment conventions shall be used. In the first case, detectors are numbered by lane, from 1 to 8, with lane 1 the rightmost mainline lane, lane 6 the leftmost mainline lane, and lanes 7 and 8 the rightmost and leftmost exit lanes respectively. The second case shall apply to freeway to freeway connector ramps. In such cases, two ramps with up to 4 lanes in each shall be accommodated. Detectors on the first ramp shall be assigned as lanes 1 to 4, with lane 1 the right most lane, while the detectors on the second ramp shall be assigned as lanes 5 to 8, with lane 5 the rightmost lane.

The detector station function may coexist in the Model 179 controller unit with other functions and is assigned logical address 4 in the communications protocol. When the Model 179 controller is set up to support both detector station and ramp meter functions, the mainline detectors shall be used both for detector station accumulations as described in this section, and as input into the ramp metering algorithm as described under the ramp metering sections.

(A) . *Detector Station Protocol Dumper Data Format:*

The packet is comprised of a header followed by an array of 'Detector Station Protocol Dumper Data'. The header contains a four-byte length field that is the summed count of all bytes contained in the array plus the four-byte integer indicating the message length. The format of the array elements is described in section B below. The number of Detector Stations contained in the transmission can be obtained from:

DetStnCnt = (arrayByteCount – 4) / 68

Table 7.1. Detector Station Data Dumper Auto Feed Protocol

Byte	Bit7 (MS)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	ARRAY BYTE COUNT HIGH BYTE							
1	“							
2	“							
3	ARRAY BYTE COUNT LOW BYTE							
4	START FIRST DETECTOR STATION DATA							
	“							
	“							
71	END FIRST DETECTOR STATION DATA							
72	START SECOND DETECTOR STATION DATA							
	“							
	“							
139	END SECOND DETECTOR STATION DATA							
	“							
	“							
Ln1	START nth DETECTOR STATION DATA							
	“							
	“							
Ln2	END nth DETECTOR STATION DATA							

Note: Ln1 = (arrayByteCount - 67) + 4.
Ln2 = arrayByteCount + 4.

(B) . *Detector Station Protocol Dumper Data Format*:

The detector station protocol dumper data format is 68 bytes in length and is shown in the following Table 7.2. In addition to the station id bytes, seven bytes are returned for each lane (i.e. each pair of detectors). The data returned is as follows:

DIRn - Direction.
One bit is provided for each lane to return the currently operating flow direction. Although not initially used, each detector station supports reverse flow through the two-loop detector traps. When the direction bit is set, the detector station Model 179 processes the corresponding detector in reverse flow order.

VOLUME - Lane Volume.
This is a one-byte field for each lane which returns the volume across the two-loop trap accumulated since the last time the detector station was polled. This field can accommodate values 0 to 255, which is adequate to store volume data for at least 6 minutes between pollings without data loss. When both loops are operating, the volume shall reflect the number of vehicles that properly crossed both detectors, the maximum volume count of the two loops, the up-stream loop count, or the downstream loop count, as determined by the

configuration of the detector processing software (VOL MODE, as defined for Format 2). If one of the loops in the pair is failed, the volume shall be the volume across the working loop. Volume shall be measured when a vehicle enters the loop detection zone, i.e. from the start of a vehicle.

TRUCK 1 VOLUME - Truck 1 Volume.
TRUCK 2 VOLUME - Truck 2 Volume
These two three-bit counters shall return the number of trucks that crossed the trap since the last time the detector station was polled. Truck 1 shall be defined as any vehicle with a length equal to or greater than the value of truck 1 length and less than truck 2 length. Truck 2 volume shall be any vehicle with a length equal to or greater than the value of truck 2 length. The truck 1 and truck 2 lengths are programmed in the detector station database. As long as both loops are working, the length of individual vehicles shall be determined from a combination of single-loop and dual-loop measurements using the formula:

*VLENGTH = (TSPEED * SLTIME) - LLENGTH*

where:

VLENGTH is the calculated vehicle length in feet
TSPEED is the vehicle speed across the trap in feet/second
SLTIME is the time to cross the single loop in seconds
LLENGTH is the length of the single loop

While both loops in the speed trap are operating, the Model 179 detector station controller unit shall maintain a continually updated historical database of average vehicle lengths by lane for 96 fifteen minute periods covering the 24 hour day. The historical vehicle length database shall be used in the event of a failure of one of the two trap detectors in each lane. Initially, all entries in the historical vehicle length database shall default to 17 feet.

PRESU - Presence Upstream Detector.
PRESD - Presence Downstream Detector.
These two bits are each set when the accumulated presence for the corresponding loop equals or exceeds the PRESENCE DURATION threshold programmed in the detector station database. The accumulated presence and the PRESENCE DURATION threshold are measured in ticks. A tick is defined as the detector station's detector scanning rate. The designations "U" and "D" represent the upstream detector and downstream detector for the designated lane. The correspondence of bits "U" and "D" to the physical loops is changed when the direction of traffic flow is reversed, as designated by the DIRn bits. When the PRESU or PRESD bit is set, the occupancy level for the corresponding loop exceeds the programmed presence duration threshold. This occupancy is measured over the communications polling interval and is adjusted for the length of the loop.

TRAVEL TIME - Trap Travel Time (High and Low).
This value, expressed as a 16-bit quantity in two successive bytes of data, is the accumulated travel time for all vehicles that crossed the speed trap since the last time the detector station

was polled. The travel time is represented in ticks, as defined above. If one of the two loops in the lane is failed, travel time shall be the accumulation of individual derived travel times. The travel times shall be derived from the single loop occupancy using the historical average vehicle length for the current 15-minute period of the day. This historical average vehicle length shall be generated from an average of measured vehicle lengths and maintained by the detector station for each 15-minute period in the day.

OCCUPANCY - Lane Occupancy (High and Low).
This value, expressed as a 16-bit quantity in two successive bytes of data, is the accumulated time that a single loop has presence. The presence accumulation is represented in ticks. When both loops are operating, the occupancy shall reflect the average occupancy of both detectors, the maximum occupancy of the two detectors, the occupancy from the upstream detector, or the occupancy from the downstream detector, as determined by the configuration of the detector processing software (OCC MODE, as defined for Format 2). When one of the loops is failed, the occupancy shall be the occupancy from the working loop.

FAILU - Failure Flag Upstream Detector.
FAILD - Failure Flag Downstream Detector.
When set, these two bits indicate that the upstream ("U") detector or the downstream ("D") has been declared as failed by the Model 179 field processor. These bits are an indication that the data returned from the corresponding loop detector is unreliable, and are not being used. Data substitution, as described above under VOLUME, TRUCK 1 OR 2 VOLUME, TRAVEL TIME, and OCCUPANCY, shall be put into effect by the Model 179 controller when either of these two failure flags is set. The correspondence of bits U and D to the physical loops is changed when the direction of traffic flow is reversed, as designated by the DIRn bits. When the Model 179 controller determines that the detector operation has returned to normal, the corresponding FAIL bit shall be cleared.

ERROR COUNT - Error Count.
This six-bit value provides the total number of errors detected from the two detectors (trap) in the corresponding lane. The error count field is a total of all detected errors regardless of the nature of the error or on which loop it is detected. Once a given loop detector has been declared as failed by the field station, no additional errors shall be accumulated from the failed loop in the error count. When the detector failure condition is reset, error accumulation shall resume. A maximum of 63 errors can be reported per lane per polling interval. The field device shall incorporate safeguards to prevent overflow during periods of erratic detector operation.

RAMP ENTRANCE VOLUME – Ramp Volume.
The ramp entrance volume.

SEQUENCE NUMBER- Sequence Counter
This byte is assigned by the Field Device to synchronize with the Node Processor. Number will increase then roll over.

The following four bits are used to return to central when someone is accessing the Model 179 or there has been a power outage detected locally. When the bits are returned, the central will automatically upload the parameters using the command message format 2U. The central system will determine if any of the parameters have changed locally. When a change is detected the system will notify the operator of the change. The operator will than make a decision on which database is proper and change the other database to match. The change will require either uploading the controller's parameters or downloading the central parameters.

PWR OUT - This bit returns a 1 when there has been a power outage at the Model 179. This will also be returned when a new Model 179 is installed in the Cabinet.

LAP TOP - The laptop computer has been plugged in and is accessing the Model 179.

DOOR OPEN - The cabinet door is open.

KYBD ENTY - The keypads on the Model 179 are being accessed.

The format contains a trailer that consists of a problem code, and a communications status. The problem code contains a device specific problem designation. The assigned values are TBD by each subsystem along with the field device manufacturer. The Communications Status Code contains status information from the FD or the NP. This area is normally reserved for use by the FD. However, when the FD does not actually generate a Feedback Message, the NP uses this area to communicate FD specific, Command Message specific, or Circuit Number specific error status information. Values from zero (0) through 127 inclusive are reserved as non-error status codes originating in the FD. Values from 128

Table 7.2. Detector Station Protocol Dumper Data Format

Byte	Bit7 (MS)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	DETECTOR STN ID HIGH BYTE							
1	“							
2	“							
3	DETECTOR STN ID LOW BYTE							
4	DIR8	DIR7	DIR6	DIR5	DIR4	DIR3	DIR2	DIR1
5	VOLUME (1)							
6	PRESU	PRESD	TRUCK 1 VOLUME(1)	TRUCK 2 VOLUME (1)				
7	TRAVEL TIME (1) HIGH BYTE							
8	TRAVEL TIME (1) LOW BYTE							
9	OCCUPANCY (1) HIGH BYTE							
10	OCCUPANCY (1) LOW BYTE							
11	FAILU	FAILD	ERROR COUNT (1)					
12	VOLUME (2)							
13	PRESU	PRESD	TRUCK 1 VOLUME(2)	TRUCK 2 VOLUME (2)				
14	TRAVEL TIME (2) HIGH BYTE							
15	TRAVEL TIME (2) LOW BYTE							
16	OCCUPANCY (2) HIGH BYTE							
17	OCCUPANCY (2) LOW BYTE							
18	FAILU	FAIL	ERROR COUNT (2)					
:	:							
:	:							
54	VOLUME (8)							
55	PRESU	PRESD	TRUCK 1 VOLUME(8)	TRUCK 2 VOLUME (8)				
56	TRAVEL TIME (8) HIGH BYTE							
57	TRAVEL TIME (8) LOW B							
58	OCCUPANCY (8) HIGH BYTE							
59	OCCUPANCY (8) LOW BYTE							
60	FAILU	FAILD	ERROR COUNT (8)					
61	RAMP ENTRANCE VOLUME							
62	LAST FRAME DURATION HIGH BYTE							
63	LAST FRAME DURATION LOW BYTE							
64	SEQUENCE NUMBER							
65	NOT USED		PWR OUT	LAP TOP	DOOR OPEN	KYBD ENTY		
66	PROBLEM CODE							
67	COMM STATUS							

Appendix 7.3 SIEMENS ITS *I2* CENTER-TO-CENTER SPECIFICATIONS

(This document was provided by Siemens ITS)

Siemens ITS Center-To-Center Specification

Revision History

Revision	Date	Changes
0.1	30 May 2005	Initial Draft Outline
0.2	06 June 2005	Corrected/Modified/Added XML message names. Added detail to Signal Inventory Message.
0.3	20 June 2005	Added ramp metering functionality.

1 SCOPE

1.1. Identification/Purpose

This document specifies the interface between Siemens ITS Center-To-Center system interface and an external system. This interface is used both for the dissemination of status information and the introduction of system commands.

1.2. Background/Standards Development and Current Status

In Early 2002, Siemens investigated the various national standards as it applies to Center-to-Center (C2C) communications. At that time, Siemens reported current status of the Working Group responsible for NTCIP 1601 and 1106 (CORBA Near Real-Time Data and Base Communications Standards, respectively) and also identified what elements of the standards are stationary and what elements remain in the working group or pending working group vote.

Working in conjunction with these two groups was the development of the GRM, or Generic Reference Model, which began in August of 2002. The GRM would develop NTCIP 1602 and would serve as the one of two reference models for C2C communications. The GRM will be a protocol-independent model, suitable for use with both object-oriented implementations and for message-based implementations, such as DATEX-ASN. The CORBA-Specific Reference Model (CSRM) will be derived from the GRM, with CORBA-specific features added to enable interoperable CORBA implementations.

The other group that should be mentioned here is the Traffic Management Data Dictionary (TMDD). The TMDD Steering Committee started meeting in 2000, In October 2000, Version 1.0 of the TMDD guide[1] was released for comment.

Figure 1 below identifies how the various working groups and Steering Committees were working together.

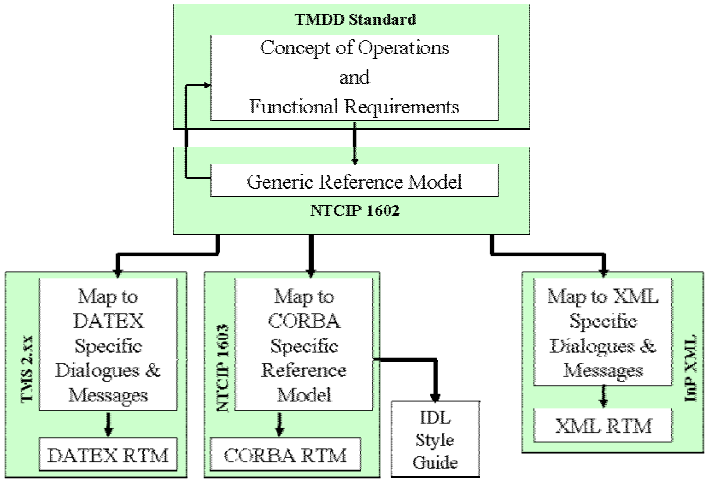


Figure 1. NTCIP C2C Application to Communications Standards Development Process

[1] Standard for Functional-level traffic management data dictionary (TMDD) and message sets for external traffic management center communications.

During 2003, there was much interaction between the various working groups and committees. There was also a large interest to pursue XML as the standard for C2C over the other two protocols – DATEX and CORBA – for a variety of reasons, most emphasis on the fact that XML is not as complex of a standard to implement. SOAP would be used was the transport for XML messages.
The NTCIP Center-to-Center working group has since published a user comment draft (UCD) of NTCIP 2306 which specifies the transport layer for implementing an XML-based, Center-to-Center protocol.
The TMDD Steering Committee, the group responsible for specifying the content layer of the specifications, plans to release (as of September 2004) version 2.1 as a provisional standard. There are many outstanding issues to be resolved regarding how the TMDD can be modified such that it fits within the efforts of the GRM. There is some discussion regarding whether the GRM and Version 3.0 should be rolled into one document.
In summary, the transport layer, while in UCD form, is still a little ways away from being formalized. On the content side, the development of the GRM is currently on hold pending future discussions about future releases of the TMDD. It is clear that the SOAP/XML means for accomplishing C2C is gaining in popularity and almost all transportation deployments of C2C are going XML and/or are moving in the XML direction. The C2C standards remain to be in an evolving state – mostly due to the complexity of the subject and diversity of needs.

1.3. System Overview

The Siemens ITS Center-To-Center (SITS-C2C) Interface is based on the NTCIP 2306 Standard. As of this version SITS-C2C interface, NTCIP 2306 was in user comment draft form. As such, this implementation is an extrapolation of that interface based on our knowledge of the interface.
NTICP 2306 specifies the use of WSDL. However, WSDL is not used throughout the interface as the operation input and output may contain XML documents or fragments. As such, it is an implementation that may not be completely intuitive from a software perspective. However, in order to maintain consistency, the NTCIP 2306 approach is implemented in the SITS-C2C interface.

1.4. Operational Concept

Typically, an external interface will connect to the SITS-C2C interface by first logging into the interface. This creates a session ID. At this point, the external system may request static (“inventory”) data or dynamic (“status”) data. In addition, the external system may subscribe to dynamic data. Once the subscription is established, updates will then be sent to the external system. In order for the subscription mechanism to work, the external system must implement the specified client WSDL interface. Finally, when the external system has completed its processing, it may log off the SITS-C2C interface.

1.5. Goals and Objectives

The Goals of SITS-C2C interface this document is the following.

- Provide a standards-based method for data exchange.
- Describe the nature and use of the SITS-C2C interface.
- Specify the Interaction with the SITS-C2C interface.

1.6. Document Overview

This document contains the following parts.
Section 2 of this document describes the important parts of the SITS-C2C interface necessary to successfully access the interface.
Appendix 2.2.1 includes the WSDL specific to the interface
Appendix 3 includes the XML schema used to define the XML data exchanged as part of the interface.

1.7. Related Documents

The following documents are related to this document.

- NTCIP 2306 v00.50 National Transportation Communications for ITS Protocol (NTCIP) Application Profile for XML Message Encoding and Transport in ITS Center-To-Center Communications (NTCIP C2C XML) March 01, 2005.

2. SITS-C2C DESCRIPTION

This section describes the SITS-C2C interface.

2.1. Server

2.1.1. OP_Login

This method is the first method that must be invoked by a client. In order to obtain traffic and device status data, a client must first log in to the server and subscribe for specific data types. The client must also provide the location where it will accept subscription updates. Note that a login is only required if data updates or command access is requested.
Parameters: An XML string, MSG_LoginRequest, which contains the login request. See APPENDIX B – XSD for details of the XML string.
Return Value: An XML string containing a session ID. A null is returned if the Login was not successful. See APPENDIX B – XSD for details of the XML string.

2.1.2. OP_Logout

This method is invoked when the client is ready to disconnect from the server. If status data updates are no longer required or if the client is getting ready to shut down, the client should make a logout request to notify the server that updates are no longer required.
Parameters: None.
Return value: A Boolean is returned. True indicates that the call was successful.

2.1.3. OP_KeepAlive

This method make be invoked on a periodic basis to maintain the update session with the server. If there are no updates sent, the KeepAlive may prevent a session from timing out. The KeepAlive call should be made at approximately thirty second intervals because the shortest session timeout that can be set is one minute.
Parameters: None
Return Value: True if timeout was reset successfully and the session was maintained.

2.1.4. OP_GetSubscriptions

This method will provide a list of the existing subscriptions for the client. This method is typically the first call after the login to ascertain the existence of any persisted subscriptions.
Parameters: None.
Return Value: An XML string, MSG_GetSubscriptionsReturn, which contains the extent subscriptions for the client. The definition of this document can be found in APPENDIX B – XSD

2.1.5. OP_Subscribe

This method is invoked to subscribe for specific data types. A reply to a successful subscription request contains the current data for the devices in the subscription request. After this, updates are sent to the client includes a sequence number that is incremented with update. In this way, it is possible to tell if an update was missed. See APPENDIX B – XSDfor details.
Parameters: An XML string, MSG_SubscribeRequest, which contains details of the subscription request. The definition of this document can be found in APPENDIX B – XSDThe requests contain client provided ID that may be used to reference the subscription when canceling a subscription.
Return Value: An XML string, MSG_SubscribeReturn, which contains the current status of the devices found in the subscription request.

2.1.6. OP_CancelSubscriptions

This method is used to cancel subscriptions.
Parameters: An XML string, MSG_CancelSubscriptionRequest containing subscriptions to be cancelled. See APPENDIX B – XSD for details.
Return Value: An XML string, MSG_CancelSubscriptionReturn, indicating if the cancel request was successful. See APPENDIX B – XSD for details.

2.1.7. OP_GetAllInventoryData

This method requests the static data for all devices in the system.
Parameters: None.
Return Value: An XML string, MSG_GetAllInventoryDataReturn, containing device inventory data. See APPENDIX B – XSD for details of the XML string.

2.1.8. OP_GetAllStatusData

This method requests the dynamic data for all devices in the system.
Parameters: None.
Return Value: An XML string, MSG_GetAllStatusDataReturn, containing device status data. See APPENDIX B – XSD for details of the XML string.

2.1.9. OP_GetSignalInventoryData

This method requests the static data for all signal devices in the system.
Parameters: None.
Return Value: An XML string, MSG_GetSignalInventoryDataReturn, containing signal inventory data. See APPENDIX B – XSD for details of the XML string.

2.1.10. OP_GetSignalStatusData

This method requests the dynamic data for all signal devices in the system.
Parameters: None.
Return Value: An XML string, MSG_GetSignalStatusDataReturn, containing signal status data. See APPENDIX B – XSD for details of the XML string.

2.1.11. OP_SignalControl

TBD.

2.1.12. OP_GetRampInventoryData

This method requests the static data for all ramp devices in the system.
Parameters: None.
Return Value: An XML string, MSG_GetRampInventoryDataReturn, containing signal inventory data. See APPENDIX B – XSD for details of the XML string.

2.1.13. OP_GetRampStatusData

This method requests the dynamic data for all ramp devices in the system.
Parameters: None.
Return Value: An XML string, MSG_GetRampStatusDataReturn, containing signal status data. See APPENDIX B – XSD for details of the XML string.

2.1.14. OP_RampControl

This method contains control request for ramp devices in the system.
Parameters: An XML string, MSG_RampControlRequest, containing ramp control request. See APPENDIX B – XSD for details of the XML string.
Return Value: Results of the requests.

2.2. Client

2.2.1. OP_InventoryDataUpdate

This method is called by the server to provide device inventory update. A sequence number is included so that missed updates can be identified.
Parameters: An XML string, MSG_InventoryDataUpdateRequest, containing the inventory data update. See APPENDIX B – XSD for details of the XML string.
Return Value: None.

2.2.2. OP_StatusDataUpdate

This method is called by the server to provide device status update. A sequence number is included so that missed updates can be identified.
Parameters: An XML string, MSG_StatusDataUpdateRequest, containing the status data update. See APPENDIX B – XSD for details of the XML string.
Return Value: None.

3. XML DEFINITIONS

3.1. Server

3.1.1. MSG_LoginRequest

This string contains a single element, updateUri, which contains the URI where traffic and status updates should be sent. The URI should point to an appropriate C2C client-side web service for receiving data updates. A sample entry is as follows.

```
<MSG_LoginRequest updateUri="http://client.com/c2c"/>
```

3.1.2. MSG_LoginReturn

This string contains the session ID. A sample entry is as follows.

```
<MSG_LoginReturn sessionId="session-id-001"/>
```

3.1.3. MSG_GetSubscriptionsReturn

This string contains the existing subscriptions.

3.1.4. MSG_SubscribeRequest

This string contains a subscription request.

3.1.5. MSG_SubscribeReturn

This string contains a subscription reponse.

3.1.6. MSG_CancelSubscriptionRequest

This string contains a cancel subscription request. It contains elements, each of which identifies a subscription to be cancelled. A sample entry follows.

```
<MSG_CancelSubscriptionRequest>
      <subscription id="a001"/>
      <subscription id="b002"/>
</MSG_CancelSubscriptionRequest>
```

3.1.7. MSG_GetSignalInventoryDataReturn

3.1.8. MSG_GetSignalStatusDataReturn

3.1.9. MSG_GetRampInventoryDataReturn

3.1.10. MSG_GetRampStatusDataReturn

3.1.11. MSG_RampControlRequest

3.1.12. MSG_RampControlReturn

3.2. Client

3.2.1. MSG_ InventoryDataUpdateRequest

3.2.2. MSG_ StatusDataUpdateRequest

4. APPENDIX A – WSDL

Below is the WSDL document that defines the SITS-C2C interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="c2c"
xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="c2c"
xmlns:intf="c2c" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.2
Built on May 03, 2005 (02:20:24 EDT)-->
 <wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="c2c"
xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="loginRequest" type="xsd:string"/>
   <element name="OP_LoginReturn" type="xsd:string"/>
   <element name="OP_LogoutReturn" type="xsd:boolean"/>
   <element name="OP_KeepAliveReturn" type="xsd:boolean"/>
   <element name="OP_GetSubscriptionsReturn" type="xsd:string"/>
   <element name="subscribeRequest" type="xsd:string"/>
   <element name="OP_SubscribeReturn" type="xsd:string"/>
   <element name="cancelSubscriptionsRequest" type="xsd:string"/>
   <element name="OP_CancelSubscriptionsReturn" type="xsd:string"/>
   <element name="deviceType" type="xsd:string"/>
   <element name="OP_GetInventoryDataReturn" type="xsd:string"/>
   <element name="deviceType1" type="xsd:string"/>
   <element name="OP_GetStatusDataReturn" type="xsd:string"/>
   <element name="OP_GetAllInventoryDataReturn" type="xsd:string"/>
   <element name="OP_GetAllStatusDataReturn" type="xsd:string"/>
   <element name="OP_GetSignalInventoryDataReturn" type="xsd:string"/>
   <element name="OP_GetSignalStatusDataReturn" type="xsd:string"/>
   <element name="OP_GetRampInventoryDataReturn" type="xsd:string"/>
   <element name="OP_GetRampStatusDataReturn" type="xsd:string"/>
   <element name="OP_RampControlRequest” type=”xsd:string”/>
   <element name="OP_RampControlReturn” type=”xsd:string”/>
   <element name="id" type="xsd:string"/>
   <element name="subscribeReturn" type="xsd:string"/>
```

```
  </schema>
 </wsdl:types>
   <wsdl:message name="OP_GetSignalInventoryDataResponse">
      <wsdl:part element="impl:OP_GetSignalInventoryDataReturn"
name="OP_GetSignalInventoryDataReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_LogoutRequest">
   </wsdl:message>
   <wsdl:message name="OP_GetStatusDataResponse">
      <wsdl:part element="impl:OP_GetStatusDataReturn"
name="OP_GetStatusDataReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_LoginRequest">
      <wsdl:part element="impl:loginRequest" name="loginRequest"/>
   </wsdl:message>
   <wsdl:message name="OP_GetAllInventoryDataRequest">
   </wsdl:message>
   <wsdl:message name="OP_GetInventoryDataRequest">
      <wsdl:part element="impl:deviceType" name="deviceType"/>
   </wsdl:message>
   <wsdl:message name="OP_LoginResponse">
      <wsdl:part element="impl:OP_LoginReturn" name="OP_LoginReturn"/>
   </wsdl:message>
   <wsdl:message name="subscribeRequest">
      <wsdl:part element="impl:id" name="id"/>
   </wsdl:message>
   <wsdl:message name="OP_GetSubscriptionsResponse">
      <wsdl:part element="impl:OP_GetSubscriptionsReturn"
name="OP_GetSubscriptionsReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_CancelSubscriptionsRequest">
      <wsdl:part element="impl:cancelSubscriptionsRequest"
name="cancelSubscriptionsRequest"/>
   </wsdl:message>
   <wsdl:message name="OP_GetAllStatusDataRequest">
   </wsdl:message>
   <wsdl:message name="OP_KeepAliveRequest">
   </wsdl:message>
   <wsdl:message name="subscribeResponse">
      <wsdl:part element="impl:subscribeReturn"
name="subscribeReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_GetSignalStatusDataRequest">
   </wsdl:message>
   <wsdl:message name="OP_GetRampInventoryDataRequest">
   </wsdl:message>
   <wsdl:message name="OP_GetRampStatusDataRequest">
   </wsdl:message>
   <wsdl:message name="OP_GetRampControlRequest">
      <wsdl:part element="impl:OP_GetRampControlRequest"
name="OP_GetRampControlRequest"/>
   </wsdl:message>
   <wsdl:message name="OP GetRampControlReturn">
      <wsdl:part element="impl:OP_GetRampControlReturn"
name="OP_GetRampControlReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_GetAllStatusDataResponse">
```

```
      <wsdl:part element="impl:OP_GetAllStatusDataReturn"
name="OP_GetAllStatusDataReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_SubscribeResponse">
      <wsdl:part element="impl:OP_SubscribeReturn"
name="OP_SubscribeReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_KeepAliveResponse">
      <wsdl:part element="impl:OP_KeepAliveReturn"
name="OP_KeepAliveReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_GetSubscriptionsRequest">
   </wsdl:message>
   <wsdl:message name="OP_SubscribeRequest">
      <wsdl:part element="impl:subscribeRequest"
name="subscribeRequest"/>
   </wsdl:message>
   <wsdl:message name="OP_GetSignalStatusDataResponse">
      <wsdl:part element="impl:OP_GetSignalStatusDataReturn"
name="OP_GetSignalStatusDataReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_CancelSubscriptionsResponse">
      <wsdl:part element="impl:OP_CancelSubscriptionsReturn"
name="OP_CancelSubscriptionsReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_GetRampStatusDataResponse">
      <wsdl:part element="impl:OP_GetRampStatusDataReturn"
name="OP_GetRampStatusDataReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_GetRampInventoryDataResponse">
      <wsdl:part element="impl:OP_GetRampInventoryDataReturn"
name="OP_GetRampInventoryDataReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_GetAllInventoryDataResponse">
      <wsdl:part element="impl:OP_GetAllInventoryDataReturn"
name="OP_GetAllInventoryDataReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_LogoutResponse">
      <wsdl:part element="impl:OP_LogoutReturn"
name="OP_LogoutReturn"/>
   </wsdl:message>
   <wsdl:message name="OP_GetSignalInventoryDataRequest">
   </wsdl:message>
   <wsdl:message name="OP_GetStatusDataRequest">
      <wsdl:part element="impl:deviceType1" name="deviceType"/>
   </wsdl:message>
   <wsdl:message name="OP_GetInventoryDataResponse">
      <wsdl:part element="impl:OP_GetInventoryDataReturn"
name="OP_GetInventoryDataReturn"/>
   </wsdl:message>
   <wsdl:portType name="NtcipC2CXmlAdapter">
      <wsdl:operation name="OP_Login" parameterOrder="loginRequest">
         <wsdl:input message="impl:OP_LoginRequest"
name="OP_LoginRequest"/>
         <wsdl:output message="impl:OP_LoginResponse"
name="OP_LoginResponse"/>
      </wsdl:operation>
```

```
      <wsdl:operation name="OP_Logout">
         <wsdl:input message="impl:OP_LogoutRequest"
name="OP_LogoutRequest"/>
         <wsdl:output message="impl:OP_LogoutResponse"
name="OP_LogoutResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_KeepAlive">
         <wsdl:input message="impl:OP_KeepAliveRequest"
name="OP_KeepAliveRequest"/>
         <wsdl:output message="impl:OP_KeepAliveResponse"
name="OP_KeepAliveResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_GetSubscriptions">
         <wsdl:input message="impl:OP_GetSubscriptionsRequest"
name="OP_GetSubscriptionsRequest"/>
         <wsdl:output message="impl:OP_GetSubscriptionsResponse"
name="OP_GetSubscriptionsResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_Subscribe"
parameterOrder="subscribeRequest">
         <wsdl:input message="impl:OP_SubscribeRequest"
name="OP_SubscribeRequest"/>
         <wsdl:output message="impl:OP_SubscribeResponse"
name="OP_SubscribeResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_CancelSubscriptions"
parameterOrder="cancelSubscriptionsRequest">
         <wsdl:input message="impl:OP_CancelSubscriptionsRequest"
name="OP_CancelSubscriptionsRequest"/>
         <wsdl:output message="impl:OP_CancelSubscriptionsResponse"
name="OP_CancelSubscriptionsResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_GetInventoryData"
parameterOrder="deviceType">
         <wsdl:input message="impl:OP_GetInventoryDataRequest"
name="OP_GetInventoryDataRequest"/>
         <wsdl:output message="impl:OP_GetInventoryDataResponse"
name="OP_GetInventoryDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_GetStatusData"
parameterOrder="deviceType">
         <wsdl:input message="impl:OP_GetStatusDataRequest"
name="OP_GetStatusDataRequest"/>
         <wsdl:output message="impl:OP_GetStatusDataResponse"
name="OP_GetStatusDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_GetAllInventoryData">
         <wsdl:input message="impl:OP_GetAllInventoryDataRequest"
name="OP_GetAllInventoryDataRequest"/>
         <wsdl:output message="impl:OP_GetAllInventoryDataResponse"
name="OP_GetAllInventoryDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_GetAllStatusData">
         <wsdl:input message="impl:OP_GetAllStatusDataRequest"
name="OP_GetAllStatusDataRequest"/>
         <wsdl:output message="impl:OP_GetAllStatusDataResponse"
name="OP_GetAllStatusDataResponse"/>
```

```
      </wsdl:operation>
      <wsdl:operation name="OP_GetSignalInventoryData">
         <wsdl:input message="impl:OP_GetSignalInventoryDataRequest"
name="OP_GetSignalInventoryDataRequest"/>
         <wsdl:output message="impl:OP_GetSignalInventoryDataResponse"
name="OP_GetSignalInventoryDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_GetSignalStatusData">
         <wsdl:input message="impl:OP_GetSignalStatusDataRequest"
name="OP_GetSignalStatusDataRequest"/>
         <wsdl:output message="impl:OP_GetSignalStatusDataResponse"
name="OP_GetSignalStatusDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_GetRampInventoryData">
         <wsdl:input message="impl:OP_GetRampInventoryDataRequest"
name="OP_GetRampInventoryDataRequest"/>
         <wsdl:output message="impl:OP_GetRampInventoryDataResponse"
name="OP_GetRampInventoryDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_GetRampStatusData">
         <wsdl:input message="impl:OP_GetRampStatusDataRequest"
name="OP_GetRampStatusDataRequest"/>
         <wsdl:output message="impl:OP_GetRampStatusDataResponse"
name="OP_GetRampStatusDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="OP_RampControl">
         <wsdl:input message="impl:OP_RampControlRequest"
name="OP_GetRampControlRequest"/>
         <wsdl:output message="impl:OP_RampControlResponse"
name="OP_RampControlResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="C2CBridgeSoapBinding"
type="impl:NtcipC2CXmlAdapter">
      <wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="OP_Login">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="OP_LoginRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="OP_LoginResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="OP_Logout">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="OP_LogoutRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="OP_LogoutResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="OP_KeepAlive">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="OP_KeepAliveRequest">
```

```
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_KeepAliveResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_GetSubscriptions">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_GetSubscriptionsRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_GetSubscriptionsResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_Subscribe">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_SubscribeRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_SubscribeResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_CancelSubscriptions">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_CancelSubscriptionsRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_CancelSubscriptionsResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_GetInventoryData">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_GetInventoryDataRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_GetInventoryDataResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_GetStatusData">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_GetStatusDataRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_GetStatusDataResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_GetAllInventoryData">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_GetAllInventoryDataRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_GetAllInventoryDataResponse">
```

```
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_GetAllStatusData">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_GetAllStatusDataRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_GetAllStatusDataResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_GetSignalInventoryData">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_GetSignalInventoryDataRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_GetSignalInventoryDataResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_GetSignalStatusData">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_GetSignalStatusDataRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_GetSignalStatusDataResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_GetRampInventoryData">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_GetRampInventoryDataRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_GetRampInventoryDataResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_GetRampStatusData">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_GetRampStatusDataRequest">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_GetRampStatusDataResponse">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
<wsdl:operation name="OP_RampControl">
   <wsdlsoap:operation soapAction=""/>
   <wsdl:input name="OP_ RampControl Request">
      <wsdlsoap:body use="literal"/>
   </wsdl:input>
   <wsdl:output name="OP_RampControl Response">
      <wsdlsoap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
```

```
   </wsdl:binding>
   <wsdl:service name="NtcipC2CXmlAdapterService">
      <wsdl:port binding="impl:C2CBridgeSoapBinding" name="C2CBridge">
         <wsdlsoap:address
location="http://localhost:8080/axis/services/C2CBridge"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

5. APPENDIX B – XSD

Below is a the XSD document that defines the XML message contained in the SITS-C2C document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by glenn
massarano (Eagle Traffic Control Systems) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:simpleType name="messageType">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <xs:simpleType name="statusType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="on-line"/>
      <xs:enumeration value="available"/>
      <xs:enumeration value="commFailure"/>
      <xs:enumeration value="deviceFailure"/>
      <xs:enumeration value="unknown"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="deviceTypeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="signal"/>
      <xs:enumeration value="station"/>
      <xs:enumeration value="ramp"/>
      <xs:enumeration value="meter"/>
      <xs:enumeration value="sign"/>
      <xs:enumeration value="other"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="signalStateType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="free"/>
      <xs:enumeration value="coordinated"/>
      <xs:enumeration value="preempted"/>
      <xs:enumeration value="flash"/>
      <xs:enumeration value="policeOveride"/>
      <xs:enumeration value="other"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="subscriptionActionType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="subscribe"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="meteringActionType">
    <xs:restriction base="xs:string">
```

```
      <xs:enumeration value="dark"/>
      <xs:enumeration value="restInGreen"/>
      <xs:enumeration value="fixedRate"/>
      <xs:enumeration value="trafficResponsive"/>
      <xs:enumeration value="emergencyGreen"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="OP_LoginRequest">
    <xs:complexType>
      <xs:attribute name="updateUri" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_LoginReturn">
    <xs:complexType>
      <xs:attribute name="loginResults" type="xs:boolean"
use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_GetSubscriptionsReturn">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="deviceTypeSubscription">
          <xs:complexType>
            <xs:attribute name="deviceType" type="deviceTypeType"
use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="subscriptionId" type="xs:string"
use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_SubscribeRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="deviceTypeSubscription">
          <xs:complexType>
            <xs:attribute name="deviceType" type="deviceTypeType"
use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="subscriptionId" type="xs:string"
use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_SubscribeReturn">
    <xs:complexType>
      <xs:choice>
        <xs:element ref="signalStatus" maxOccurs="unbounded"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_CancelSubscriptionsRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="cancelSubscription" maxOccurs="unbounded">
```

```
          <xs:complexType>
            <xs:attribute name="subscriptionId" type="xs:string"
use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_CancelSubscriptionReturn">
    <xs:complexType>
      <xs:attribute name="cancelSubscriptionResponse" type="xs:boolean"
use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_GetSignalInventoryDataReturn">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="signalInventory" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_GetSignalStatusDataReturn">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="signalStatus" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_InventoryDataUpdateRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="signalInventory" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element ref="rampInventory" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_StatusDataUpdateRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="signalStatus" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element ref="rampStatus" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_GetRampInventoryDataReturn">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="rampInventory" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_GetRampStatusDataReturn">
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element ref="rampStatus" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="OP_RampControlRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="rampControl" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="signalStatus">
    <xs:complexType>
      <xs:attribute name="deviceId" type="xs:string" use="required"/>
      <xs:attribute name="status" type="statusType" use="required"/>
      <xs:attribute name="pattern" type="xs:integer" use="required"/>
      <xs:attribute name="patternDescription" type="xs:string"
use="optional"/>
      <xs:attribute name="patternExpiration" type="xs:dateTime"
use="optional"/>
      <xs:attribute name="signalState" type="signalStateType"
use="required"/>
      <xs:attribute name="alarms" type="xs:string" use="required"/>
      <xs:attribute name="timestamp" type="xs:dateTime"
use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="signalInventory">
    <xs:complexType>
      <xs:attribute name="signalId" type="xs:string" use="required"/>
      <xs:attribute name="signalDescription" type="xs:string"
use="required"/>
      <xs:attribute name="signalName" type="xs:string" use="required"/>
      <xs:attribute name="latitude" type="xs:float" use="required"/>
      <xs:attribute name="longitude" type="xs:float" use="required"/>
      <xs:attribute name="locationDescription" type="xs:string"
use="required"/>
      <xs:attribute name="organizationInformation" type="xs:string"
use="required"/>
      <xs:attribute name="cycleTime" type="xs:integer" use="optional"/>
      <xs:attribute name="offsetTime" type="xs:integer"
use="optional"/>
      <xs:attribute name="timingPlanId" type="xs:integer"
use="optional"/>
      <xs:attribute name="timingPlanName" type="xs:string"
use="optional"/>
      <xs:attribute name="firmware" type="xs:string" use="optional"/>
      <xs:attribute name="firmwareVersion" type="xs:string"
use="optional"/>
      <xs:attribute name="controllerModel" type="xs:string"
use="optional"/>
      <xs:attribute name="controllerModelSerialNumber" type="xs:string"
use="optional"/>
      <xs:attribute name="lastUpdateTime" type="xs:dateTime"
use="required"/>
```

```
      <xs:attribute name="contactDetails" type="xs:string"
use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="rampInventory">
    <xs:complexType name="rampName">
      <xs:attribute name="rampId" type="xs:string" use="required"/>
      <xs:attribute name="rampDescription" type="xs:string"
use="required"/>
      <xs:attribute name="rampName" type="xs:string" use="required"/>
      <xs:attribute name="latitude" type="xs:float" use="required"/>
      <xs:attribute name="longitude" type="xs:float" use="required"/>
      <xs:attribute name="locationDescription" type="xs:string"
use="required"/>
      <xs:attribute name="organizationInformation" type="xs:string"
use="required"/>
      <xs:attribute name="meteredLanes" type="xs:integer"
use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="rampStatus">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="rampLaneStatus" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="laneNumber" type="xs:integer"
use="required"/>
            <xs:attribute name="meteringRate" type="xs:integer"
use="required"/>
            <xs:attribute name="meteringAction"
type="meteringActionType" use="required"/>
            <xs:attribute name="meteringPlan" type="xs:integer"
use="required"/>
            <xs:attribute name="vehiclesPerGreen" type="xs:integer"
use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="rampId" type="xs:string" use="required"/>
      <xs:attribute name="status" type="statusType" use="required"/>
      <xs:attribute name="timestamp" type="xs:dateTime"
use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="rampControl">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="rampLaneControl" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="laneNumber" type="xs:integer"
use="required"/>
            <xs:attribute name="meteringRate" type="xs:integer"
use="optional"/>
            <xs:attribute name="meteringAction"
type="meteringActionType" use="required"/>
            <xs:attribute name="meteringPlan" type="xs:integer"
use="optional"/>
```

```
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="rampId" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```