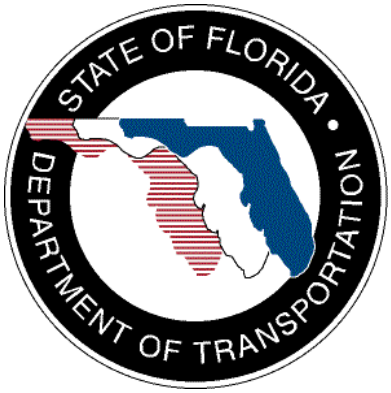
SunGuide®:

Center-to-Center
Interface Control Document

SunGuide-C2C-ICD-4.0.12



Prepared for:

Florida Department of Transportation
Traffic Engineering and Operations Office
605 Suwannee Street, M.S. 90
Tallahassee, Florida 32399-0450
(850) 410-5600

October 15, 2010

Document Control Panel			
File Name:	SunGuide-C2C-ICD-4.0.12(Draft).docx		
File Location:	SunGuide CM Repository		
CDRL:	2-7.1		
Name		**Initial**	**Date**
Created By:	John Brisco, SwRI	JSB	12/11/07
Reviewed By:	Steve Dellenback, SwRI	SWD	12/12/07
	Steve Dellenback, SwRI	SWD	12/19/07
	Steve Dellenback, SwRI	SWD	2/08/08
	John Brisco, SwRI	JSB	07/25/08
	Steve Dellenback, SwRI	SWD	09/03/08
	Steve Dellenback, SwRI	SWD	01/21/09
	Steve Dellenback, SwRI	SWD	02/05/09
	Steve Dellenback, SwRI	SWD	02/18/09
	Steve Dellenback, SwRI	SWD	09/08/09
	Steve Dellenback, SwRI	SWD	12/08/09
	Steve Dellenback, SwRI	SWD	01/19/10
	Robert Heller, SwRI	RWH	10/15/10
Modified By:	John Brisco, SwRI	JSB	12/11/07
	John Brisco, SwRI	JSB	12/19/07
	John Brisco, SwRI	JSB	02/08/08
	John Brisco, SwRI	JSB	07/23/08
	John Brisco, SwRI	JSB	09/03/08
	John Brisco, SwRI	JSB	01/21/09
	John Brisco, SwRI	JSB	02/05/09
	Steve Dellenback, SwRI	SWD	02/18/09
	John Brisco, SwRI	JSB	09/08/09
	John Brisco, SwRI	JSB	12/08/09
	John Brisco, SwRI	JSB	01/19/10
	John Brisco, SwRI	JSB	10/13/10
Completed By:			

Table of Contents

List of Figures

Page

List of Acronyms

ATIS.....Advanced Traveler Information System
ATMS .....Advanced Traffic Management Systems
C2C .....Center-to-Center
CCTV.....Closed Circuit Television
DMS.....Dynamic Message Sign
DOM .....Document Object Model
FDOT .....Florida Department of Transportation
GIS .....Geographic Information System
GUI .....Graphical User Interface
HAR .....Highway Advisory Radio
HTTP.....Hyper Text Transfer Protocol
ICD.....Interface Control Document
ITN.....Invitation to Negotiate
ITS.....Intelligent Transportation Systems
MAS.....Message Arbitration Susbsystem
NTCIP.....National Transportation Communications for ITS Protocol
SOAP .....Simple Object Access Protocol
SwRI .....Southwest Research Institute®
TCP/IP.....Transport Control Protocol Internet Protocol
TMC.....Transportation Management Center
TSS.....Transportation Sensor Subsystem
TxDOT.....Texas Department of Transportation
URL.....Uniform Resoure Locator
VDD.....Version Description Document
XML.....Extensible Markup Language

Revision History

Revision	Date	Changes
Pre 4.0.0	Many dates	TxDOT published many versions of this document and SunGuide previously used the TxDOT version of the document.
4.0.0-Draft	December 12, 2007	Updated to support the FDOT Data Fusion requirements for Release 4.0 of SunGuide
4.0.1-Draft	December 19, 2007	Updated to add support for: • Dual language Floodgate messages • Inventory of roadways (to support FL-ATIS project) • Inventory of event locations (to support FL-ATIS project)
4.0.2-Draft	February 8, 2008	Updated for incident and closure unification: • Removed incident type • Removed closure type • Renamed travelEvent type to event • Removed extraneous elements
4.0.3	July 25, 2008	Updated for minor document fix and revised XML schemas.
4.0.4	September 3, 2008	Update made: • Edits to Floodgate status data following FL-ATIS PDR • Edits to Event and Location status data resulting from DTN schema changes, FL-ATIS PDR, and SGV4.0 FAT • Added Floodgate command request and response • Added Event command request and response following SGV4.0 FAT
4.0.5	January 21, 2009	Updates: • The data type of the sort order element of the location element was changed from xs:short to xs:integer
4.0.6	February 5, 2009	Updates: • Added a “can publish to the public” attribute for CCTV, DMS and Travel Time data.
4.0.7	February 18, 2009	Updates: • Added a field for Travel Time length in the travel time data
4.0.8	March 11, 2009	Updates: • Added a “can publish” element to roadway link and traffic conditions data types.

Revision	Date	Changes
4.0.9	September 8, 2009	Updates: • Added county name to equipment location element, making the county info available in DMS, HAR, CCTV, and ESS data types. • Added county name to roadway link data type, based on the location of the start node. • Added county name as an optional element to event location data.
4.0.10	December 8, 2009	Updates: • Added new "traffic speed" data type that can be configured to be sent less often that normal traffic updates (added to slow down the amount of data being sent to FLATIS).
4.0.11	January 19, 2010	Added the new ATIS Severity to the event status as a new element. Replaced the TMC Severity in the event command (sent to FLATIS) with the ATIS Severity.
4.0.12	October 13, 2010	Updates: • Added message queue schema and message queue data type to status data types. • Added new "restricted use" optional attribute to ID element attribute group.

1. Scope

1.1 Document Identification

This Interface Control Document (ICD) describes the Center-to-Center Status and Command interfaces to SunGuide®.

1.2 Project Overview

The Florida Department of Transportation (FDOT) is conducting a program that is developing SunGuide software. The SunGuide software is a set of Intelligent Transportation System (ITS) software that allows the control of roadway devices as well as information exchange across a variety of transportation agencies. The goal of the SunGuide software is to have a common software base that can be deployed throughout the state of Florida. The SunGuide software development effort is based on ITS software available from the state of Texas; significant customization of the software is being performed as well as the development of new software modules. The following figure provides a graphical view of the software to be developed:

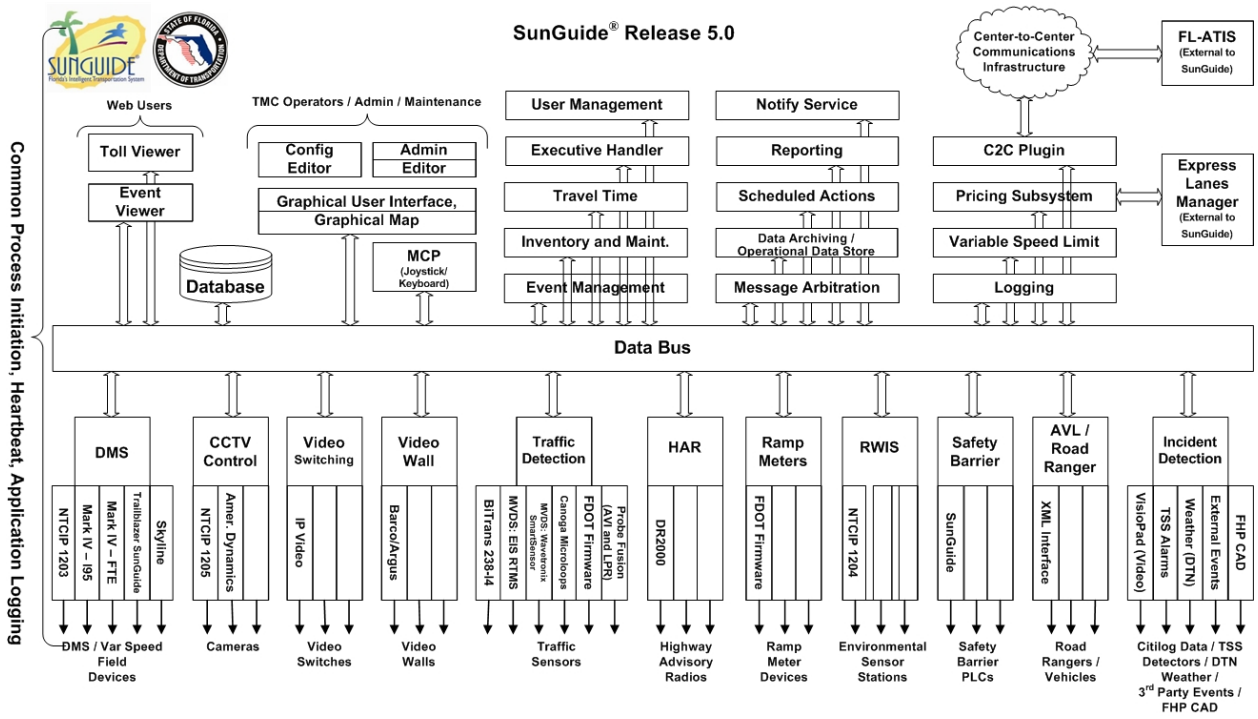


Figure 1-1 - High-Level Architectural Concept

1.3 Background

The Texas Department of Transportation (TxDOT) operates a class of Intelligent Transportation Systems (ITS) termed Advanced Traffic Management Systems (ATMS). ATMSs generate a significant amount of data that is useful to the traveling public as well as to other agencies managing the transportation infrastructure. An important component for the long-term deployment of these systems was to develop a common data format utilizing C2C (Center-to-

Center) concepts so that information can be exchanged between centers operated by any agency, this data could simply be "status" data (informational) or it could be "control" data.

TxDOT began development of their C2C project began in 1999, it was originally implemented using the evolving ITS Traffic Management Data Dictionary (TMDD) standard and the message sets associated with TMDD as well as protocols (e.g. DATEX) from the National Transportation Communications for ITS Protocol (NTCIP) C2C Working Group. A significant upgrade in functionality was initiated in 2003 and the underlying protocol was changed to XML/SOAP (Extensible Markup Language/Simple Object Access Protocol). At the time the C2C interfaces were defined and the underlying support software was developed, XML schemas were not available from TMDD. The developers developed a set of XML schema that were optimal for the TxDOT project. Once TMDD publishes reasonable XML schema (expected in TMDD 3.0 to be released in early 2008), it is envisioned the C2C infrastructure will be updated to use the published schema from TMDD.

This document is based on FDOT's reuse of the TxDOT C2C infrastructure. FDOT has customized what data is needed to support operational requirements throughout the state of Florida, this ICD provides the spefic C2C components supported by the SunGuide implementation of C2C.

The C2C project provides an "infrastructure" for information exchange. This infrastructure has a well-defined interface that allows systems to deposit and retrieve data in a highly predictable fashion. The infrastructure does not permanently store information, all data is kept in memory and is lost when a software process is restarted.

1.4 Operational Concept

The C2C infrastucture must have the capability to interconnect similair or dissimilar traffic management systems. In order to create the C2C infrastructure, interfaces to the existing systems must be created. The data being deposited into the C2C infrastructure will be converted to a standard format (ITS standards based). The C2C infrastructure is built using a series of building blocks. These building blocks allow the software to be utilized in a number of configurations (by simply altering the configuration parameters of the software). The building blocks include:

- Data Provider: A building block that receives data from an ITS system in a format defined in Sections 2, 3, and 4. The Data Provider converts the data to a standard format and transmits it to other blocks.
- Data Collector: A building block that data from multiple sources (both Data Providers and Data Collectors) in a standard format and stores the data in local memory. Data Extractor blocks subscribe to this block to receive the stored data in a standard format.
- Data Extractor: A building block that receives data from the Data Collector block in a format defined in Sections 2, 3, and 4.
- Command Receiver: A building block that interfaces to an ITS system to receive command/control requests for ITS equipment. The interface and data format is specified Sections 5, 6, and 7.

The capabilities are implemented using four building blocks described above, these blocks are installed and configured to provide the services that a center expects to receive or provide. A

deployment will typically have multiple instances of each block. Figure 1-2 contains a depiction of a simple deployment between two centers. Note that all data that passes through the C2C infrastructure is formatted using XML which is described in a set of XML schema files (provided in a separate zipped file).

The C2C Status XML schemas describe how status information is formatted. The schemas describe both OUTBOUND status data (data being sent to outside organizations) as well as INBOUND status data (data being received to be processed locally).

The C2C Command XML schemas describe how commands to devices are formatted. The schemas describe both the sending of OUTBOUND commands (commands being sent to other centers) as well as INBOUND commands (commands being received for local processing).

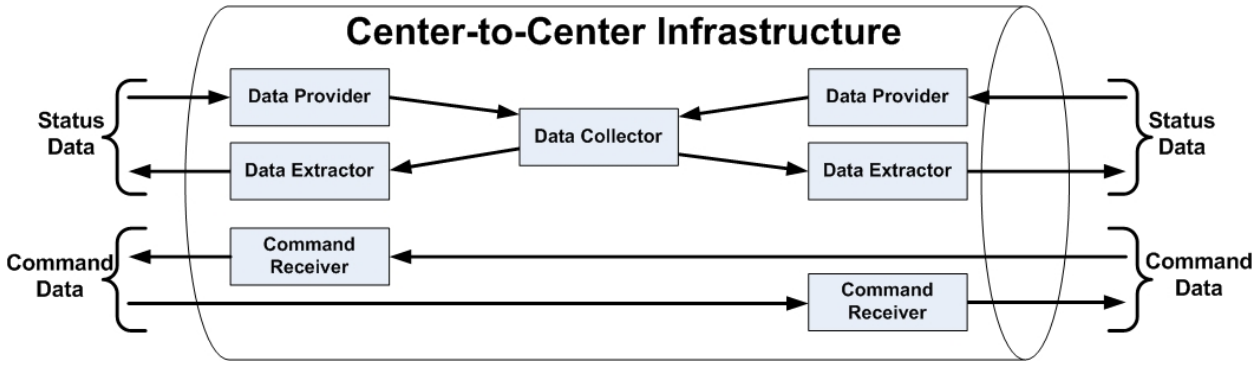


Figure 1-2 - Center-to-Center Components

In order to deploy the C2C infrastructure, it is necessary to have TCP/IP connectivity between the agencies wishing to utilize the C2C functionality. The C2C infrastructure is implemented using Web Services so any network appliances must be configured to allow HTTP communication. Figure 1-3 provides a depiction of how C2C could be used to connect two centers.

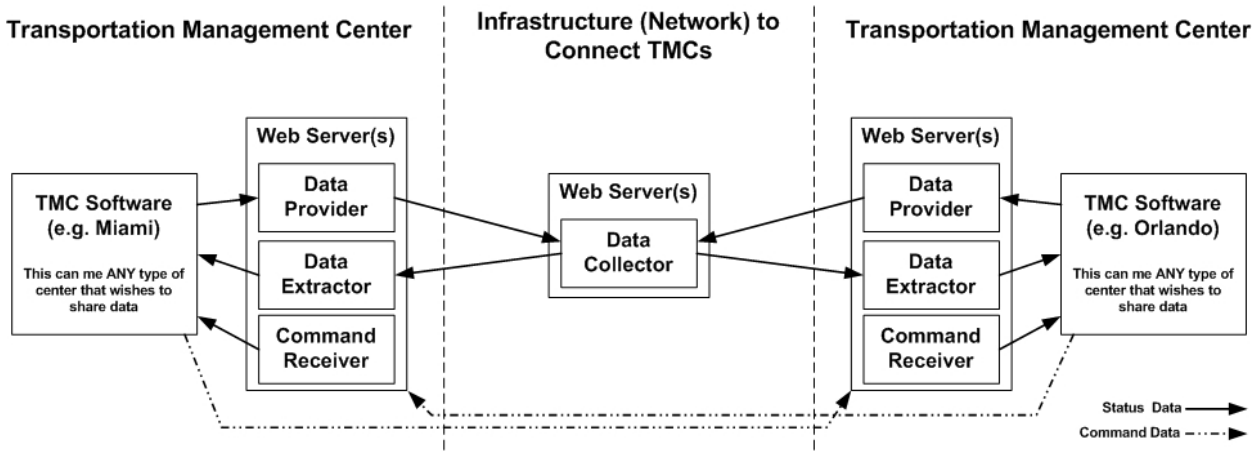


Figure 1-3 - Deployment of the C2C Infrastructure

Once a client data connection has been established, client data requests or client data subscriptions can be made. Client data requests allow clients to get the complete current status of a particular type of data. Client data requests can be made at any time (with or without a

subscription) after the connection has been established. Client data subscriptions allow clients to get the current complete status (one time) of a particular type of data followed by updates for that type of data. Once a subscription is in place and until it is removed, the data provider will provide asynchronous updates for each type of data subscribed to.

A region establishes a C2C infrastructure by deploying multiple instances of the building blocks to exchange information. The primary advantage if the C2C infrastructure is that only a single interface to a system needs to be developed, once the system is connected to the C2C infrastructure any system, with appropriate privileges, operating within the C2C infrastructure can gain access to data from the system using the single interface point established. Organizations can develop custom applications to process the data available from the C2C infrastructure, an example of a custom application would be a web server that provides a visual view of the regional traffic conditions.

1.5 How to Use This Document

This document needs to be used in conjunction with the XML schema files and an XML view tool (XMLSpy is recommended) so that the specific contents of the messages to be exchanged can be reviewed along with the the interface discussions contained in this document. Access to the C2C XML schemas (provided in a zipped file) is needed to fully interstand this ICD document.

1.6 Related Documents

The following documents were used to develop this document:

- FDOT Scope of Services: *BDQ69, Standard Written Agreement for SunGuide Software Support, Maintenance, and Development, Exhibit A: Scope of Services.* July 1, 2010.
- Notice to Proceed: Letter to SwRI for BDQ69, July 1, 2010
- Letter of Authorization 001: Letter to SwRI for BDQ69, July 1, 2010.
- Letter of Authorization 002: Letter to SwRI for BDQ69, August 3, 2010.
- Letter of Authorization 003: Letter to SwRI for BDQ69, August 19, 2010.
- SunGuide Project website: http://sunguide.datasys.swri.edu.

1.7 Contacts

The following are contact persons for the SunGuide software project:

- Elizabeth Birriel, ITS Section, Traffic Engineering and Operations Office, elizabeth.birriel@dot.state.fl.us, 850-410-5606
- Arun Krishnamurthy, FDOT SunGuide Project Manager, arun.krishnamurthy@dot.state.fl.us, 850-410-5615
- Khue Ngo, PBS&J Project Manager, khue.ngo@dot.state.fl.us, 850-410-5579.
- David Chang, PBS&J Project Advisor, David.Chang@dot.state.fl.us, 850-410-5622
- Robert Heller, SwRI Project Manager, rheller@swri.org, 210-522-3824
- Tucker Brown, SwRI Software Project Manager, tbrown@swri.com, 210-522-3035

2. Status Interface

The following sections describe the Status Interfaces to a Traffic Management Center (TMC). The Status Interface defines the mechanism by which TMCs communicate traffic and roadway status information (such as events, travel speeds, and DMS content) to external entities. A TMC will receive requests from data providers and provide responses to them through this interface. This interface does not provide roadside device control capability.

2.1 Overview

The FDOT C2C infrastructure provides the following status information:

- Roadway network:
  - Layout (GIS-based)
  - Current conditions
  - Travel Times
- Locale data:
  - Roadway inventory
  - Location inventory
- Events:
  - Events Messages (incidents, closures, weather alerts)
  - Floodgate Messages
- Field Devices:
  - DMS
  - CCTV
  - Environmental Sensors
  - HAR
- Remote Messages (in SunGuide this is the contents of the MAS queues)

2.2 Client Data Requests

Once a client is connected, it can initiate a request to provide a complete list of any of the following data:

- Current roadway network data (this includes roadway node, roadway link, and travel time segment definitions): these messages allow the physical description of the roadway network to be retrieved from a center. Because traffic conditions are identified by Link ID and travel times are defined and identified by link collections, the Roadway Network should be retrieved first so that the locations corresponding to the traffic conditions and travel times are known.

- Current locale data (this includes roadway inventory and location inventory): these messages allow the roadways and location descriptions to be retrieved from a center. Because Traveler Events may be identified by Location ID, the Locale data should be retrieved first so that the locations corresponding to the traveler events are known.

- Current traffic condition data (this includes speed, occupancy, volume, level of service and classification) for all roadway links: these messages allow the list of current traffic

conditions to be retrieved from a center. Because traffic conditions are identified by Link ID, the Roadway Network should be retrieved first so that the locations corresponding to the traffic conditions are known.

- Current traffic speed data (this includes speed data only) for all roadway links: these messages allow the list of current traffic speeds to be retrieved from a center. Because traffic speeds are identified by Link ID, the Roadway Network should be retrieved first so that the locations corresponding to the traffic speeds are known.
- Current DMS status data: these messages allow the list of current DMS statuses to be retrieved from a center.
- Current CCTV status data: these messages allow the list of current CCTV statuses to be retrieved from a center.
- Current CCTV snapshot data: these messages allow the list of current CCTV snapshots to be retrieved from a center.
- Current HAR status data: these messages allow the current Highway Advisory Radio data to be retrieved from a center.
- Current ESS status data: these messages allow the list of current ESS statuses to be retrieved from a center.
- Current Remote Message data: these messages allow the Remote Message statuses to be retrieved from a center.
- Current Floodgate Message data: these messages allow the list of current Floodgate messages to be retrieved from a center.
- Current Travel Time status data: these messages allow the list of current Travel Time values to be retrieved from a center. Because links are identified by Link ID and travel times are defined and identified by link collections, the Roadway Network should be retrieved first so that the locations corresponding to the links and travel times are known.
- Current Event message data: these messages allow the list of current event messages to be retrieved from a center. Events can include at least incidents/accidents, closures, and weather alerts.

2.3 Subscription Request

The client will use this message for subscribing to or removing a subscription to various data types. Once the subscription is activated, data for that subscription will be transmitted asynchronously with the initial publication being the current list as defined in Section 2.2, and subsequent publications updates as defined in Section 2.4. Any client subscribing for data must include roadway network data in the subscription. The roadway network data subscription is required because it is the mechanism used to indicate loss of connection to individual networks and that any prior data related to that network is no longer valid.

2.4 Asynchronous Server Updates

The server will transmit asynchronous updates (either periodic or event–driven) to the client. These updates are activated and deactivated through the Subscription Request message, and will include:

- Roadway network updates (typically used to indicate loss of data for a particular network)
- Locale data updates (typically used to indicate loss of data for a particular network)
- Traffic conditions data updates (such as speed, occupancy, travel time, volume), this data will be identified by link identifiers
- Traffic speed data updates (speed only), this data will be identified by link identifiers
- DMS updates: updated status data for a DMS
- CCTV updates: updated status and/or snapshot data for a CCTV
- HAR updates: updated status data for a HAR
- ESS updates: updated status data for an ESS
- Remote Message updates: updated status data for a Remote Message
- Floodgate Message updates: updated status for a Floodgate Message
- Travel Time updates: updated travel time data
- Event updates: new or updated data or notification to cancel the event

The server may elect to send one update per message (an event–driven implementation), or multiple updates per message (a periodic update implementation). For data types that result in a large number of updates or frequent updates (e.g., traffic conditions), it is recommended that the updates be sent as a list containing multiple data items rather than in individual messages. This is necessary to reduce overhead and improve performance throughout the C2C system environment.

There are two types of asynchronous server update messages – add/modify messages and delete messages. The add/modify message is sent when a new data element is to be added to the existing elements or when a change has been made to a previously defined element. The delete message is sent when the element is to be removed from the list of elements in the client's database. For instance, a delete event message would be sent if a previously posted event should no longer be displayed on a web map. For deletions, only the network identifier and the identifier of the item to be deleted are transmitted.

2.4.1 Roadway Network Update

This message allows updates of the roadway network to be sent by a center.

If the update is a deletion, only the network identifiers will be sent for each deleted network. The deletion of a network is used to indicate loss of data from a center. If a network deletion message is received, none of the other data associated with that network (e.g., events, traffic conditions, etc.) will be valid until new update messages for that data are received.

2.4.2 Locale Data Update

This message allows updates of the locale data to be sent by a center.

If the update is a deletion, only the network identifiers will be sent for each deleted locale data. The deletion of a network is used to indicate loss of data from a center. If a network deletion message is received, none of the other data associated with that network (e.g., events, traffic conditions, etc.) will be valid until new update messages for that data are received.

2.4.3 Traffic Condition Update

This message allows updates of traffic conditions to be sent by a center. Because traffic conditions are identified by Link ID, the Roadway Network should be retrieved first so that the locations corresponding to the traffic conditions are known.

If the update is a deletion, only the Network and Link identifier data items will be sent for each deleted traffic data element.

2.4.4 Traffic Speed Update

This message allows updates of traffic speeds to be sent by a center. Because traffic speeds are identified by Link ID, the Roadway Network should be retrieved first so that the locations corresponding to the traffic speeds are known. Traffic speed data is published periodically at an interval configured by the originating center.

If the update is a deletion, only the Network and Link identifier data items will be sent for each deleted traffic data element.

2.4.5 DMS Update

This message allows new DMSs or updates to existing DMSs to be sent by a center. Although additions and deletions are supported, a center would typically send only modifications to existing DMSs.

2.4.6 CCTV Status Update

This message allows new CCTVs or updates to existing CCTVs to be sent by a center. Although support for updates is specified here, it is unlikely that a consumer would be interested in status updates once the lists of CCTV IDs and locations have been received.

If the update is a deletion, only the Network and CCTV identifier data items will be sent for each deleted CCTV.

2.4.7 CCTV Snapshot Update

This message allows new CCTV snapshots or updates to existing snapshots to be sent by a center.

If the update is a deletion, only the Network and CCTV identifier data items will be sent for each deleted CCTV snapshot.

2.4.8 HAR Update

This message allows new HAR Data or updates to existing HAR data to be sent by a center.

If the update is a deletion, only the Network and HAR identifier data items will be sent for each deleted HAR.

2.4.9 ESS Update

This message allows new ESS Data or updates to existing ESS data to be sent by a center.

If the update is a deletion, only the Network and ESS identifier data items will be sent for each deleted ESS.

2.4.10 Remote Message Interface Update

This message allows new Remote Message Data or updates to existing Remote Message data to be sent by a center.

If the update is a deletion, only the Network and Remote Message identifier data items will be sent for each deleted Remote Message.

2.4.11 Floodgate Message Update

This message allows new floodgate messages or updates to previously posted floodgate messages to be sent by a center.

If the update is a deletion, only the Network and Floodgate Message identifier data items will be sent for each deleted floodgate message.

2.4.12 Travel Time Status Update

This message allows updates of travel time data to be sent by a center. Because links are identified by Link ID and travel times are defined and identified by link collections, the Roadway Network should be retrieved first so that the locations corresponding to the links and travel times are known.

2.4.13 Event Update

This message allows new events or updates to previously posted events to be sent by a center.
If the update is a deletion, only the Network and Event identifier data items will be sent for each deleted event.

3. Status XML Schema

The following sections detail the status information which can be retrieved from or injected into the C2C infrastructure using the C2C web services.

The web methods that are used to inject data into the C2C infrastructure contain a parameter for passing in an XML document in text format. The C2C Infrastructure stores the XML in local DOM (Document Object Model) databases. When clients retrieve data from the infrastructure, it is provided as an XML document in text format. The XML schemas files are provided in a zipped file.

3.1 XML Status

The status element is used to report all status data to a client using the C2C web services interface. Each of the data type children of this element may contain zero or more net subelements, which contain the actual data elements for the parent network.

The schema allows for multiple data types, e.g. event, DMS, etc., to be sent in a single XML document. This provides for additional flexibility in how data is sent, e.g. as periodic updates or event-driven updates. The structure allows for a single event, an entire network's events, or multiple networks' events to be sent in a single message. As another example, events and travel times could be sent in a single update message. In general, it is preferable to bundle information and send it as a single update vs. sending an update message for each element. For instance, traffic conditions data updates would occupy significantly more bandwidth and seriously impede system performance if each traffic link were sent individually. The recommended approach is to combine all updated traffic links into one message for transmission.

3.2 Adding or Modifying Status

When the status of any existing object changes, the new information will be reflected in the next status update sent to the client. If a new object with status is added to the system, that object will be added to the status data sent to the client. An added object will have a unique network ID and object ID pair which was not previously provided to the C2C infrastructure. When the infrastructure receives an update, it will check if the element with the specific data type/network/ id combination exists. If it exists already, the entire data type element and its subtree are replaced, e.g. all status information for an **event** is replaced at the **event** level – child elements of the event are not individually updated. Therefore, an event must be sent in its entirety when an update occurs.

3.3 Deleting Status

A separate schema is used to perform deletions of traffic and device status data. A list of elements is sent, each containing a data type, element name, element id and network id attribute. Any objects matching these attributes are deleted from the DOM databases stored in the C2C infrastructure. If an object with status has been deleted from the system, it will be reported by sending a `delete` element to the client.

4. Status XML Interface

The following sections describe the interfaces to the C2C XML infrastructure from both the TMC update server side and the client applications side. The TMC update server is a data provider, i.e. it injects data into the C2C infrastructure, where the data may be combined with data from other TMCs. The client applications side extracts data from the C2C XML infrastructure for purposes such as displaying traffic and device information on a Web page.

4.1 TMC Server Interface

The following sections describe how a TMC update server can inject traffic and status information into the C2C XML Infrastructure. In general, TMC data is injected by 'pushing' the XML data to a C2CProvider Web Service. The Web Service is part of an application that maintains the data in an XML DOM database, and delivers subscribed-for or requested data to its clients. The C2CProvider application domain consists of two web services – ProviderClient and ProviderServer. The TMC update server makes web method calls into the ProviderClient, and passes in the XML data that contains the traffic and status information as defined by the XML Schema. Remote clients can log into the ProviderServer to request subscriptions for the data stored at the C2CProvider.

4.1.1 ProviderClient Web Methods

The following sections describe the ProviderClient web methods that are used to inject data into the C2C infrastructure.

GetSubscriptions

Web Method Name:	GetSubscriptions
Description:	This web method is the first method that must be invoked by a data provider external to the C2C XML Infrastructure. The return value from this method is a string containing all of the XML dataType names for which the C2CProvider will maintain current status. The names are delimited by commas, spaces, or tab characters. Only those data types that are returned by this method should be sent by a TMC update server. This allows different Providers to be configured to support specific data types. The C2CProvider's subscription data types are defined in its web.config file.
Prototype:	string GetSubscriptions ()
Parameters:	{none}
Return Value:	A string containing the delimited list of data types that the C2CProvider is configured to maintain in its database.

KeepAlive

Web Method Name:	KeepAlive
Description:	This web method is invoked on a periodic basis by the server to maintain the update session with the client. If there are no updates to be sent, the KeepAlive keeps the session from timing out. The KeepAlive call should be made at approximately 30-second intervals, because the shortest session timeout that can be set by a Web Service is one minute.
Prototype:	void KeepAlive ()
Parameters:	{none}
Return Value:	{none}

RegisterUpdateSession

Web Method Name:	RegisterUpdateSession
Description:	This web method is called from within the server's Login web method. It is used within the C2C infrastructure by the C2CProvider and the C2CCollector during the Login process to register the update session with its clients (Collectors or Extractors). When a client makes a Login call, it provides the URI where the traffic updates should be sent. The server then makes a proxy call to this method at the specified URI, and the session ID created by the call is returned. The session ID then becomes the return value of the Login method. See section 0 for a description of the Login method.
Prototype:	string RegisterUpdateSession ()
Parameters:	{none}
Return Value:	A string containing the session ID that was created by the invocation of this method. The session created by this method call is the same session that will be used for sending traffic and device status updates and deletions to a client.

SendStatusData

Web Method Name:	SendStatusData
Description:	This web method is invoked to send the current status data from the TMC update server to the C2CProvider's client web service. An XML document in text format is sent to the web service. The document may contain all current status information for the TMC, or it may contain current status of a particular data type. Multiple web method calls must be made to send the data by type, whereas a single call can be made if all of the data types are represented in the XML document.
Prototype:	int SendStatusData (string sXmlString)
Parameters:	string sXmlString – an XML string containing the server's current traffic and device status information.
Return Value:	An integer containing either the size of the XML string received by the web method (returned when the call was successful), or a -1 (returned when an error is encountered within the method).

SendStatusDeletions

Web Method Name:	SendStatusDeletions
Description:	This web method is invoked to send deleted status data to the client. An XML document in text format is sent to the client web service. The document contains whatever deletions need to be made by the C2CProvider. For instance, a **delete** element is sent when an event no longer exists. The XML document may contain multiple deletions for a data type, as well as multiple data types' deletion information.
Prototype:	int SendStatusDeletions (string sXmlString)
Parameters:	string sXmlString – an XML string containing the items that were deleted by the server.
Return Value:	An integer containing either the size of the XML string received by the web method (returned when the call was successful), or a -1 (returned when an error is encountered within the method).

SendStatusUpdates

Web Method Name:	SendStatusUpdates
Description:	This web method is invoked to send the updated status data from the TMC to the client web service. An XML document in text format is sent to the web service. The document contains whatever updates need to be sent to the client. Note that the C2C infrastructure only updates data at the typeData level. The business logic for the client will locate the data element having the specified identifier and replace all child elements and attributes with those sent in the update document. Therefore, even if a particular data item for a type is unchanged, it must be sent with the update, or it will be deleted when the typeData information is replaced.
Prototype:	int SendStatusUpdates (string sXmlString)
Parameters:	string sXmlString – an XML string containing the items that were updated at the server.
Return Value:	An integer containing either the size of the XML string received by the web method (returned when the call was successful), or a -1 (returned when an error is encountered within the method).

Shutdown

Web Method Name:	Shutdown
Description:	This web method is invoked when the TMC server is able to notify the C2CProvider that it is shutting down, e.g. for maintenance, failure, etc. When the C2CProvider receives a Shutdown, it will delete all information from its database and inform its clients that the network information is no longer available. Issuance of the Shutdown call is the preferred way to notify the C2CProvider that the server is shutting down. The C2CProvider can also detect a loss of data by a session timeout, which would occur if the KeepAlive updates were discontinued. However, the session timeout may be significantly longer than the KeepAlive interval, so the loss of connectivity would be undetected for a longer interval.
Prototype:	void Shutdown ()
Parameters:	{none}
Return Value:	{none}

4.1.2 ProviderServer and CollectorServer Web Methods

The following sections describe the web methods that are used to retrieve data from the C2C infrastructure at the C2CProvider or C2CCollector level.

Login

Web Method Name:	Login
Description:	This web 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 data server and subscribe for specific data types. The client must also provide the location where it will accept the data updates.
Prototype:	string Login (string sUpdatesURI)
Parameters:	string sUpdatesURI – a string containing the URI where traffic and status updates should be sent. The URI should point to the appropriate C2C client-side web service for receiving the updates.
Return Value:	A string containing the session ID that was created by the caller's update web service when the update session was registered. A null is returned if the Login was unsuccessful.

Logout

Web Method Name:	Logout
Description:	This web method should be invoked when the client is ready to disconnect from the server. If status data is no longer required, or the client is getting ready to shut down, it should make a Logout method call to notify the server that further updates are no longer required, and that the client session should be removed from the server's tables. The Logout method contains no parameters. The return value from this method indicates the success of the call.
Prototype:	boolean Logout ()
Parameters:	{none}
Return Value:	The return value from this method is a boolean, where a true indicates a valid method call, and a false indicates an invalid call. The call should only be made after a successful Login. If the caller receives a return value of false, the session was not recognized by the server – no further action is required by the client.

KeepAlive

Web Method Name:	KeepAlive
Description:	This web method is invoked on a periodic basis by the client to keep the HTTP session with the ProviderServer or CollectorServer maintained. If the HTTP session is allowed to time out, the server will assume that the client connection has been lost, and will discontinue sending updates to the client. This method should be invoked at 30-second intervals, to prevent the session from timing out. The return value from this method indicates the success of the call.
Prototype:	boolean KeepAlive ()
Parameters:	{none}
Return Value:	The return value from this method is a boolean, where a true indicates a valid method call, and a false indicates an invalid call. The call can only be made after a successful Login. If the caller receives a return value of false, the caller should reattempt the Login process to reestablish the connection.

Subscribe

Web Method Name:	Subscribe
Description:	This web method is invoked to subscribe for specific data types stored in the web service's database. A list of data types to be subscribed for is provided, along with an indication of whether the subscription is persistent. The subscribed-for data is returned to the caller. This method should only be invoked after a successful Login method call.
Prototype:	string Subscribe (string sSubscriptionDataTypes, boolean bPersistent)
Parameters:	string sSubscriptionDataTypes – a string containing a delimited list of XML dataType element names, e.g. "**eventData trafficCondData**". Only a single delimiter is allowed between data types. Valid delimiters are a comma, space, or tab. Multiple Subscribe calls can be made, and any new data types will be added to the list maintained at the server. Any data type can be requested at any time, and its status data will be returned. However, a persistent subscription for a data type will only be recognized once - it must be cancelled before resubscribing. boolean bPersistent – a boolean that is true for subscriptions, or false for single requests. If true, the current data will be returned to the caller, and subsequent updates for the data types will be sent to the client at the URI specified during the Login process.
Return Value:	An XML string containing the current status data for the requested data types is returned. The return value is null if an error occurs during the subscription process, such as a Subscribe call being made prior to a Login call.

CancelSubscriptions

Web Method Name:	CancelSubscriptions
Description:	This web method is invoked to cancel subscriptions for specific data types. A string containing a delimited list of XML data types elements, e.g. "**eventData trafficCondData**" is passed to the web service. The server will cancel the specified subscriptions, and no further updates will be sent for the specified data types. The valid delimiters are a single space, comma, or tab character between the data type names. Return value from this method is a boolean, indicating the success of the call.
Prototype:	boolean CancelSubscriptions(string sSubscriptionDataTypes)

Parameters:	string sSubscriptionDataTypes – a string containing a delimited list of data types, e.g. "**eventData trafficCondData**". Only a single delimiter is allowed between data types. Valid delimiters are a comma, space, or tab.
Return Value:	The return value from this method is a boolean, where a true indicates a valid method call, and a false indicates an invalid call. The call should only be made after a successful Login. If the caller receives a return value of false, the client session has been abandoned by the web server, so the client will need to reestablish the session through the Login/Subscribe calls if continued status updates are desired.

4.2 C2C Extractor Interface

The following sections describe how socket-based XML data can be obtained from the C2C infrastructure through the C2CExtractor web services. The socket-based interface provides the same XML data that the C2CProvider and C2CCollector applications do, but this interface does not require a web service for sending status updates. Updates are sent via a standard TCP/IP socket port.

4.2.1 ExtractorServer Web Methods

This section describes the web methods that are used to connect to the C2CExtractor and obtain traffic and status data.

Login

Web Method Name:	Login
Description:	This web method is invoked to connect to the infrastructure. Once the login call is made, the C2CExtractor will immediately attempt to connect to the listener socket at the specified network address and port number. It will also connect to its parent C2CCollector, which has been defined in the C2CExtractor's web.config file.
Prototype:	bool Login(string sHostName, int nPort)
Parameters:	sHostName – the network name or address where the XML client listener socket is located, e.g. "127.0.0.1" or "localhost". This address will be used by the C2CExtractor to make a socket connection to the XML client. All XML status data will be sent over this connection in the format listed in section 0. nPort – the integer port number where the converter listener socket can be contacted, e.g. "8080". This port, along with the above address, will be used by the C2CExtractor to make a socket connection to the XML client.
Return Value:	If the client and collector connections are made successfully, the web method will return true. If an error occurs during processing, it will return false. Once a single converter is connected, all other calls to

	Login will return false, until the current client logs out or is disconnected due to an error.

Logout

Web Method Name:	Logout
Description:	This web method is invoked to signal the intention to disconnect from the infrastructure. When called, the C2CExtractor will close its side of the socket connection and become available again to service a single client.
Prototype:	bool Logout()
Parameters:	{none}
Return Value:	If the disconnections occur successfully, the method will return true. If an error occurs during processing, or the client is not recognized, it will return false.

Subscribe

Web Method Name:	Subscribe
Description:	This web method is invoked to subscribe for specific data types provided by the C2C infrastructure. A list of data types to be subscribed for, along with an indication of whether the subscription is persistent, are passed as parameters. The status data will be returned to the client over the socket connection created during the Login. Multiple calls to Subscribe are allowed. Each successive call will have its data types added to the client's current subscription. This method should only be invoked after a successful Login call.
Prototype:	bool Subscribe(string sSubscriptionDataTypes, bool bPersistent)
Parameters:	sSubscriptionDataTypes – a string containing a delimited list of XML data type elements, e.g. "**eventData trafficCondData**". Only a single delimiter is allowed between data types. Valid delimiters are comma, space, or tab. bPersistent – a boolean that is true for subscriptions, or false for single requests. If true, the caller will receive any update messages over the socket connection as well as the initial status data.
Return Value:	This function will return a boolean value that is true if the data was retrieved from the infrastructure and sent to the socket connection successfully. If an error occurs during processing, or the client is not recognized, it will return false.

CancelSubscriptions

Web Method Name:	Cancel Subscriptions
Description:	This web method is invoked to cancel subscriptions to data types. This method will cause the infrastructure to cease sending status updates for the listed data types. Multiple calls to this method are allowed. Each successive call will have its data types removed from the client's current subscription. This method should be invoked only after both Login and Subscription calls have been made.
Prototype:	bool CancelSubscriptions(string sSubscriptionDataTypes)
Parameters:	sSubscriptionDataTypes – a string containing a delimited list of data types, e.g. "**eventData trafficCondData**". Only a single delimiter is allowed between data types. Valid delimiters are comma, space, or tab.
Return Value:	When called, this function will attempt to unsubscribe the listed data types, and if successful will return true. If an error occurs during processing, or the client is not recognized, it will return false.

KeepAlive

Web Method Name:	Keep Alive
Description:	This method must be called periodically by the client in order to keep its connection with the C2CExtractor from timing out. It should be called every 30 seconds. This method should be invoked after the Login call has returned successfully.
Prototype:	bool KeepAlive()
Parameters:	{none}
Return Value:	This function will return a boolean value that is true if the call is recognized to come from the same client that is currently logged in. If the client cannot be identified false is returned.

C2C Extractor Socket Response Message Format

This section describes the socket interface that is used by the C2CExtractor to transmit XML data to the client.

Byte	1	2	3	4	5	6	7	8	9-N
Content	Message ID				Data Length				Data

Message ID = 4-byte integer. This field identifies the type of message to follow.

ID	Message Type
2001	Current Status Data
2002	Status Update
2003	Status Deletion
2004	Network Deletion

Data Length = 4-byte integer. This field indicates the number of bytes contained in the data portion of the message. This value will be 0 for a shutdown message, because there is no data sent on shutdown.

Data = variable length. This field is the XML formatted ICD message. It is empty when the message type is 'Shutdown'. Refer to XML schema document for more information.

Current Status Data This message returns an initial list of XML packaged ICD status data over a socket. The data contains all the current information available at the server.

Status Update This message returns a list of XML packaged ICD status updates over a socket.

Status Deletion This message returns a list of XML packaged ICD status deletions over a socket.

Network Deletion This message returns a string representing the ID of a network to be deleted.

4.3 C2C Status Data Flow

This section provides a high-level description of the data flow between data publishers and data consumers via the C2C Infrastructure web services.

1. The publisher plug-in application gathers local status data and delivers it to C2C Provider web service using the "ProviderClient" interface.
2. The C2C Provider stores the status data in a DOM tree and waits for a connection from a C2C Collector.
3. The consumer plug-in application connects to a C2C Extractor web service using the "ExtractorServer" interface. A socket address on which the consumer is listening must be supplied to deliver status data back to the consumer.

4. The C2C Extractor connects to its configured C2C Collector web service using the "CollectorServer" interface. The Extractor provides an update URI to Collector.
5. The C2C Collector connects to its list of configured C2C Provider web services using the "ProviderServer" interface. The Collector provides its update URI to each connected Provider.
6. The C2C Provider delivers data to each connected C2C Collector web service using the "CollectorClient" interface and the provided update URI. The Collector stores the received data in a DOM tree.
7. The C2C Collector delivers updated data to connected C2C Extractor web services using the "ExtractorClient" interface and the provided update URI.
8. The C2C Extractor delivers data to the consumer plug-in application using the supplied socket address.

5. Command Interface

The following sections describe the Command Interfaces to a Traffic Management Center (TMC). The Command Interface defines the mechanism by which a TMC can receive commands from other TMCs (such as posting a DMS message, posting a HAR message, repositioning a CCTV).

5.1 Overview

The FDOT C2C infrastructure provides the following status and command capability:

- DMS
- HAR
- CCTV (continuous motion not supported)

5.2 Client Status Requests

Once a client is connected, it can request status for all devices of each device type or the status of any single device:

- Current status of all DMS, HAR, and CCTV devices. This is typically performed once at startup to retrieve the device inventory.
- Current status of a single DMS.
- Current status of a single HAR.
- Current status of a single CCTV.
- Current snapshot from a single CCTV.

5.3 Client Command Requests

Once a client has retieved the device inventory, it can submit command requests for any single device:

- Change the message and beacon of a single DMS.
- Change the message and beacon of a single HAR.
- Obtain or release a lock on a single CCTV.
- Change the preset of a single CCTV.
- Change the absolute PTZ settings of a single CCTV.
- Change the relative PTZ settings of a single CCTV.

6. Command XML Schema

The following sections detail the command and control information which can be exchanged with the C2C infrastructure using XML web services. Command/control requests (formatted as XML) are sent from the client application to the TMC Web Service and responses (also formatted as XML) are returned from the TMC Web Service to the application. The XML schemas files are provided in a zipped file.

6.1 XML Commands

The root `cmd` element is the parent element for all command and control requests and responses. Schema depictions of the command and control requests and responses may be viewed in the XML schema document.

7. Command XML Interface

The following sections describe the interfaces between Command/Control client applications, the TMC XML Web Service, and TMC Server applications.

7.1 TMC Server Interface

The Command Receiver Web Service will submit the XML request to the TMC Server and receive the response from the TMC Server using a TCP/IP connection. The XML request string from the Web Service client and the XML response string from the TMC Server must be must be formatted as defined in the C2C XML Command schema. The format of the request and response messages on the TCP/IP connection is defined in the following table.

Byte	1	2	3	4	5-N
Content	Data Length				Data

Data Length: 4-byte integer. This field indicates the number of bytes contained in the data portion of the request and response messages.

Data: Variable length. This field is the XML-formatted request or response message.

7.2 Web Service Interface

The Command Receiver Web Service is accessed by Command/Control client applications using a web service proxy. The Command Receiver Web Service exposes methods available to the proxy as defined in the following sections.

7.2.1 Transact Command

Web Method Name:	TransactCommand
Description:	This web method is invoked by the Command/Control client application to send a command request to the Web Service and receive a command response.
Prototype:	string TransactCommand (string sXmlRequest)
Parameters:	string sXmlRequest – an XML string containing a command request formatted according to the C2C XML Command Request schemas.
Return Value:	An XML string containing the command response formatted according to the C2C XML Command Response schemas.

8. Plug-in Development

This section provides a brief outline of the steps required to develop the "plug-in" applications that bridge between data providers and data consumers via the C2C Infrastructure. In most cases the majority of the work involves converting C2C-fomatted data to/from the local format and accessing local system resources. Typically the code needed to parse and generated the C2C XML messages and communicate with C2C web services can be reused from previous developments. This document is accompanied by a Microsoft Visual Studio 2005 C# solution in a zipped file that implements the basic connection handling between a C2C plug-in and the Extractor, Provider, and Command Receiver C2C web services.

8.1 Status Plug-in

The following sections provide a brief description of the steps that need to be performed to develop a C2C Status plugin.

8.1.1 Publisher Plug-in Development

A Publisher Plug-in is responsible for gathering local data to be published, converting or transforming the data between the local format and the C2C XML format, and establishing the connection with the C2C Provider component of the C2C Infrastructrure. The process is described in general terms below:

- Identify local device and event data types to be supported by plug-in.
- Use C2C Status ICD and XML Schemas to identify required and optional data elements for each data type to be supported.
- Develop code that retrieves data from local TMC components.
- Develop code that parses and stores local device and event data.
- Develop code that interfaces with the C2C Provider using the "ProviderClient" interface.
- Develop code that generates C2C XML messages for current status, status updates, and status deletes.
- Develop code that manages local TMC connection and C2C Provider connection.

8.1.2 Subscriber Plug-in Development

A Subscriber Plug-in is responsible for establishing the connection with the C2C Extractor component of the C2C Infrastructrure, converting or transforming the data between the the C2C XML format and the local format, and delivering the converted data to the local system components. The process is described in general terms below:

- Identify C2C device and event data types to be supported by plug-in.
- Use C2C Status ICD and XML Schemas to identify required and optional data elements for each data type to be supported.
- Develop code that parses C2C device and event data for current status, status updates, and status deletes.
- Use the C2C device and event data elements as new local data types, OR
- Convert the C2C device and event data into local TMC equivalents.
- Develop code that delivers remote device and event data to local TMC components.

- Develop code that interfaces with the C2C Extractor using the “ExtractorServer” interface.
- Develop code that manages local TMC connection and C2C Extractor connection.

8.2 Command Plug-in

A Subscriber Plug-in is responsible for establishing the connection with the C2C Extractor component of the C2C Infrastructrure, converting or transforming the data between the the C2C XML format and the local format, and delivering the converted data to the local system components. The process is described in general terms below:

- Identify local device types to be supported for remote commands by plug-in.
- Use C2C Command ICD and XML Schemas to identify required and optional command elements for each device type to be supported.
- Usually developed to be part of C2C Status Publisher Plug-in for local data cache sharing.
- Develop code that generates local TMC command messages for supported device types.
- Develop code that accepts messages from the C2C Command Receiver.
- Develop code that parses the C2C Command request messages.
- Develop code that generates C2C Command response messages.
- Develop code that manages local TMC connection and C2C Command Receiver connection.