******THIS IS A NEW APPENDIX. – PLEASE READ CAREFULLY. ******

Needs/Requirements Analysis

Iowa DOT Arrow-Board Communications Protocol

Version 1.0

Iowa DOT

SRF Consulting Group, Inc. | Engineers Planners Designers

August 14, 2019

SRF No. 12499

Definitions

For brevity, for the remainder of this document: “display” will mean “electronic arrow-board display”. “Controller” will mean “arrow-board display controller”. And “central software” will mean “central traffic management software”.

Acronyms

ASCII – American Standard Code for Information Interchange
GPS – Global Positioning System
IP – Internet Protocol
TCP – Transmission Control Protocol
UTC – Coordinated Universal Time

System Needs and Requirements

This document presents the system needs and high level-requirements. These are organized as a hierarchy with needs at the highest level and successive levels of requirements providing additional detail under each.

System needs describe the end goals the system is intended to achieve, or, put another way, the reason a system is being considered.

System requirements describe what a system should do to meet the needs articulated. Requirements are distinct from system specifications, which describe the materials, construction techniques and other details related to a design.

1. **Standardize communication between central software and displays using industry standard communications hardware and methods.**
   - 1.1. The controller shall accept a Transmission Control Protocol (TCP)/ Internet Protocol (IP) connection initiated by central software.
   - 1.2. The default TCP/IP port used for this communication shall be 23 (telnet). Note: The controller does not need to implement a full "telnet client". A raw TCP/IP port will suffice.
   - 1.3. A protocol shall be defined to communicate between central software and a controller using this port.
   - 1.4. All communication via this protocol shall be conveyed via 8-bit American Standard Code for Information Interchange (ASCII) text.
   - 1.5. The controller shall not echo command data.
   - 1.6. The controller shall provide a response to every protocol command.
2. **Permit remote monitoring of display's location, orientation, and arrow-board mode.**
   - 2.1. The protocol shall allow retrieving the display's:
     - 2.1.1. Latitude
     - 2.1.2. Longitude
     - 2.1.3. GPS-lock status
     - 2.1.4. Display orientation expressed as an azimuth (compass heading)
     - 2.1.5. Arrow-board mode (pattern) currently being displayed
       - 2.1.5.1. Flashing Left Arrow
       - 2.1.5.2. Flashing Right Arrow
       - 2.1.5.3. Flashing Double Arrow
       - 2.1.5.4. Sequential Left Chevron
       - 2.1.5.5. Sequential Right Chevron
       - 2.1.5.6. Sequential Left Arrow
       - 2.1.5.7. Sequential Right Arrow
       - 2.1.5.8. Four Flashing Corners (caution)
       - 2.1.5.9. Flashing Bar (caution)
       - 2.1.5.10. Alternating Diamond Caution
     - 2.1.6. UTC timestamp
     - 2.1.7. Age of latest GPS info
     - 2.1.8. Arrow Panel position (up or down)
     - 2.1.9. **FUTURE** - End of Lane Closure (not needed for arrow boards)
   - 2.2. The protocol may allow retrieving the display's
     - 2.2.1. Speed
     - 2.2.2. Direction of travel

3. The protocol shall allow service technician to perform on-site configuration and diagnostics without the need for custom software.

3.1. A service tech may initiate a communication session from a personal computer to a controller using a TCP/IP terminal program in raw mode.

3.2. All protocol commands shall be defined so that they can be generated from a standard ASCII keyboard.

3.3. All protocol responses shall be formatted to be human readable.

3.3.1. Responses shall be formatted as ASCII text.

3.3.2. Responses shall be descriptive.

3.4. All TCP/IP communication sessions shall be automatically disconnected, by the controller, after 60 seconds of no communication.

3.5. The protocol shall allow manual overriding of the reported location and/or orientation information where use of a GPS or a digital compass (parked in a tunnel or installed at a fixed location) isn't appropriate.

3.6. The protocol shall allow resuming normal reporting of location and/or orientation.

4. The controller shall allow the TCP communication port used to communicate with the controller to be configured after manufacture.

4.1. The protocol shall allow changing the TCP/IP port used for communicating with the controller. This change shall take effect automatically at the end of the communication session that initiates the change.

5. Allow retrieving/monitoring display maintenance information.

5.1. If a sign display controller is used, the protocol shall allow retrieving the display's:

5.1.1. Controller hardware version.

5.1.2. Controller software version.

5.1.3. Photocell data.

5.1.4. Current display brightness.

5.1.5. Functional status (*normal*, *locked on*, *locked off*, or *other error*) of each display element.

5.1.6. Power status (where applicable)

5.1.6.1. Battery voltage

5.1.6.2. Solar panel output voltage

5.1.6.3. Generator or vehicle power supply voltage

5.1.6.4. AC power voltage

6. Allow storing/retrieving secondary information about the display.

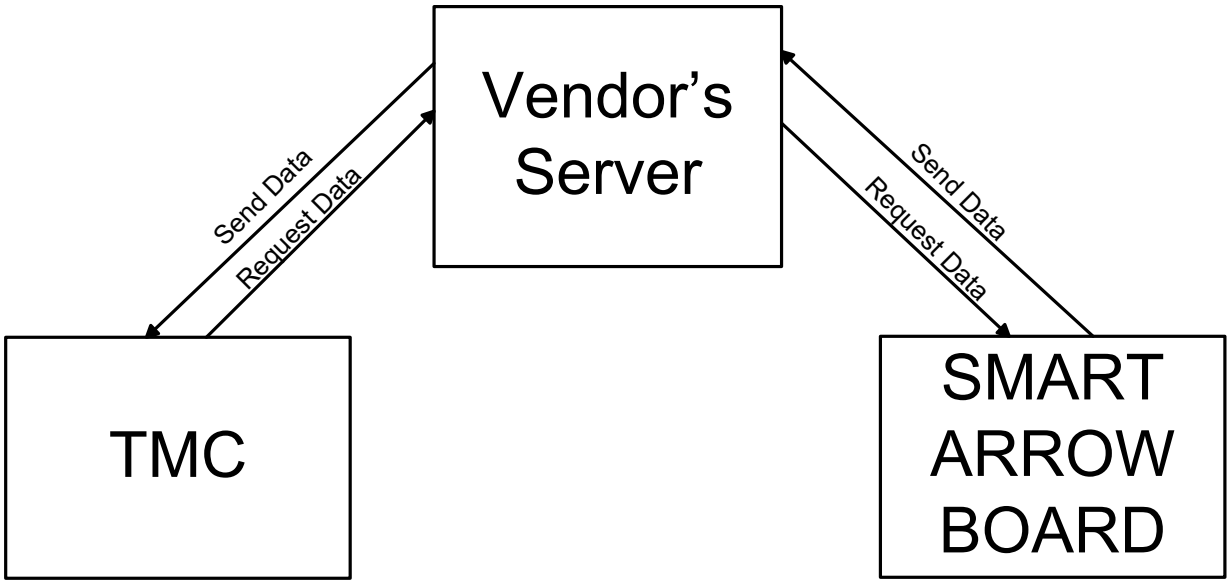
6.1. The protocol shall allow storing and retrieving a list of other hardware and/or project specific attributes.

6.1.1. <example optional attributes TBD with input from arrow board manufacturers>

Option 1

Smart Arrow Board Protocol - JSON

(Data received from an intermediary server)



SABP

Smart Arrow Board Protocol – JSON – Option 1

Version 1.0

Iowa Department of Transportation

Prepared by:

SRF

August 14, 2019

SRF No. 12499

TOC: Table of Contents

Introduction

Timely and accurate information on work zones is becoming increasingly important, not only to road authorities, but to other public and private stakeholders involved with managing road construction and maintenance activities.

Several problems in providing this information are:

- Collecting and reporting timely information is time consuming for staff and competes with other project administration duties.
- Road construction and maintenance activities that require lane or shoulder closures are not always reported to operations staff, resulting in inaccurate or no dissemination to traveler information systems and the traveling public.
- Stakeholders desire more detailed records on the start time, end time, and location of lane closures for improved analysis of the transportation management plan (TMP) and performance measurement.

The in-vehicle displays that will be offered by connected vehicles present an opportunity for improved safety through better lane closure notifications to drivers. These automated displays will require more accurate and timely data to function correctly, however.

Flashing Arrow Boards are routinely used in advance of active work zones to designate lane closures on multi-lane highways. By adding readily available technologies the location and state of Flashing Arrow Boards can be reported without adding to staff workload.

Several manufacturers have created their own solutions to monitoring arrow board deployment, but these solutions do not lend themselves to easily merging information from arrow boards of different manufacturers.

This document describes a standardized JSON data-interchange format for monitoring deployment and basic maintenance of smart arrow boards on all Iowa DOT projects on multi-lane highways.

The objective is for all manufacturers providing arrowboard monitoring services to provide HTTP access to data in this format.

This document is intended for use by arrow board manufacturers, client software authors, and DOT traffic engineers. It is assumed that the reader has some familiarity with arrow board terminology, software terms, HTTP communication, JSON data-interchange format, and procedures used for remote monitoring of field devices.

Definitions

For this document:

AB:
Arrowboard.

AB Server:
Arrowboard controller that can be polled directly for SABP information.

Agency:
Government or educational entity receiving arrowboard data for analysis.

Client:
Agency software which submits queries to one or more SABP servers and collects the responses.

Controller:
AB controller.

Consolidation Server:
Server polled for SABP info for one-or-more arrowboards.

Pretty-Printed SABP:
An SABP document with line breaks and indentation.

Property:
JSON name/value pair inside a SABP document.

Provider:
AB manufacturer or other entity that controls a consolidation server.

SABP:
Smart Arrow Board Protocol.

SABP Document:
Arrow-board data in JSON format returned in response to an SABP query.

SABP Query:
HTTP request for an SABP document.

SABP Server:
AB server or consolidation server that provides SABP documents.

Stringified SABP:
An SABP document with no line breaks or indentation.

Support tiers:
There will be two different tiers of support for this standard. Which tier will be used for each project will be determined on a project-by-project basis by agreement between the provider and the agency.

Tier 1: The client will obtain data by directly polling AB servers.
Tier 2: The client will obtain data by polling a consolidation server.

JSON Overview

All valid responses from a SABP server (SABP documents) will be in JSON format. JSON is a standard lightweight text-based data-interchange format.

For purposes of this document we will use:

1. The JSON description available at:
   http://www.json.org/
2. We will also adopt parts of the style guide available at:
   http://google.github.io/styleguide/jsoncstyleguide.xml
   (The parts before the "JSON Structure & Reserved Property Names" section.)

Libraries for parsing and manipulating JSON documents exist in many programming languages. A partial (but lengthy) list of available libraries is available at the json.org URL referenced above.

All JSON documents start with a left curly brace, end with a right curly brace, and contain a series of comma separated JSON properties.

Each property has a name. All property names begin with a letter and are then followed by a combination of letters, digits, and underscores. Property names are always surrounded by double quotes ("= 0x22) and are followed by a colon.

Each property name (with quotes and colon) is followed by a value. The value is one of the following: a number (integer or floating-point number), a text string, a boolean (true or false), a comma separated array of values, an object (a comma separated collection of properties), or a null.

All string values are quoted using double-quote characters. Special characters (some control characters, any backslashes, and any double quotes within a string must be replaced with the appropriate escape sequence. All non-string values are not quoted. (See the json.org URL for details.)

The following is a short example JSON document:

```
{
	"aString": "foo",
	"bNumber": 78,
	"cNumber": 62.878,
	"dBoolean": true,
	"eArray": [1, 2, 3, "bar"], "fObject": {
		"f1Something": 23,
		"f2SomethingElse": true
	},
	"gNull": null
}
```

(This is not an SABP document.)

Communication Process

SABP communication is a simple 3-step process:

1. The SABP client software submits a standard HTTP-GET request to an SABP server using a static URL specified by the provider.
2. The server responds with a stringified SABP document.
3. When the response is complete, server and client both terminate the HTTP connection.

If desired, communication security can be provided by using HTTPS instead of HTTP.

Structure of an SABP Document

Each SABP document consists of two parts. A "document" property which contains information about the document. And an "arrowboards" property which contains an array of arrowboard objects. Each arrowboard object contains properties describing that arrowboard and its status.

Example:

```
{
    "document": { "format":
        "SABP",
        <other document properties>
    },
    "arrowboards": [
        {
            "id": "Foont Road Signs;AB3;123-4275",
            <more properties for the first AB>
        },
        {
            "id": "Foont Road Signs;AB3;123-6275",
            <more properties for the second AB>
        },
        <And so on until…>
        {
            "id": "Foont Road Signs;AB3;123-9775",
            <more properties for the last AB>
        }
    ]
}
```

Appendix A provides a full example SABP response document containing one arrowboard object.

Appendix B provides an explanation of each SABP property in the same order as the example.

Appendix C provides additional details about formatting and purpose of various properties.

Appendix A: Example SABP Document

The following is an example SABP document containing information for one arrowboard in JSON pretty-print format. (A format used to allow people to read and understand a JSON document.)

The data returned from an SABP server may look like this or may be in raw serialized JSON format with no line breaks or indentation. (A long single line of text with no obvious structure.)

The client software that polls an SABP server is responsible for recognizing SABP documents in either format and for converting the document to whatever form the receiving agency finds most useful.

In a real SABP document, text shown between < and > brackets in the example is to be replaced with the appropriate value described and the brackets are to be removed.

```
{
    "document": {
        "format": "SABP",
        "version": "0.5",
        "tier": 1,
        "source": "<SABP server name>", "timestamp":
        "<timestamp of this document>"
    },
    "arrowboards": [
        {
            "id": "<manufacturer>;<model>;<serial number>", "firmware":
            "<firmware name>;<firmware version>", "owner": {
                "company": "<company name>", "contact": "<name
                of contact person>", "phone": "<phone number>",
                "email": "<email address>"
            },
            "gps": {
                "override": false,
                "tried": "<timestamp of last gps attempt>", "lock": 2,
                "sampled": "<timestamp of following sample>", "lat":
                44.979932,
                "lon": -93.464925
            },
```

```
                "display": {
                        "deployed": false,
                        "compass": 0,
                        "pattern": "Off"
                },
                "lampErrors": {
                        "count": 0,
                        "max": 15, "pattern":
                        null, "list": null
                },
                "voltage": 14.2,
                "temperature": {
                        "controller": 23.5,
                        "enclosure": 23.5,
                        "battery": 23.1,
                        "display": 25.8,
                        "ambient": 22.1
                },
                "errorCodes": null,
                "lastContact": "<timestamp of last communication with AB>"
        }
    ]
}
```

Appendix B: SABP Property Details

The following appendix provides detailed information for each SABP property. The entries in this appendix are in the same order as those in the example SABP document shown in Appendix A. The property names use dot notation to help show the position of each property in the document.

Optional properties not provided by a server -and- required properties with a null value may be omitted from the serialized JSON response sent from server to client. Software using SABP documents should assume all missing properties have a null value.

For all numerical sensor properties (compass, temperature, or voltage), a value of -999 is used to indicate that the sensor has malfunctioned.

Property: document Type:
required object
Description: Contains properties that identify the document.

Property: document.format Type:
required string
Description: Should always be "SABP" in an SABP document.

Property: document.version Type:
required string
Description: Indicates which version of the SABP standard is used by the server.

Property: document.tier
Type: optional integer
Description: Indicates if this is a tier 1 or tier 2 SABP document. The document is assumed to be tier 1 if this property is omitted. This is required for tier 2 documents.

Property: document.source
Type: required string
Description: Source of the SABP document. For tier 1 documents, this contains the arrowboard ID of the AB server. For tier 2 documents, this contains the name of the consolidation server.

Property: document.timestamp Type:
required string
Description: Timestamp showing when the contents of this document was last changed. This value is not updated when the document is requested by a client.

Property: arrowboards Type:
required array
Description: Array of arrowboard objects. For tier 1 documents, this will contain only one arrowboard object. For tier 2 documents, this may contain any number of arrowboard objects.

Note: For the following, we will use the property-name prefix "arrowboards[*]." to show properties that are members of objects in the arrowboards array.

Property: arrowboards[*].id Type:
required string
Description: Arrowboard ID. (More info in Appendix C section "Arrowboard Identifiers".)

Property: arrowboards[*].name Type:
optional string
Description: Defined placeholder for an optional assigned arrowboard name. This may be provided by the SABP server, may be added to the document by client software, or may be omitted. Names may be assigned based on: arrowboard ID, IP address, GPS location, or other criteria.

Property: arrowboards[*].firmware
Type: required string
Description: A string that combines an arrowboard controller's firmware name and firmware version into a single string containing two semicolon-separated attributes.

Property: arrowboards[*].owner Type:
optional object
Description: Contains information about the owner/operator of the arrowboard. If this property is provided, one or more of the following 4 optional strings is required.

Property: arrowboards[*].owner.company Type:
optional string
Description: Name of company.

Property: arrowboards[*].owner.contact Type:
optional string
Description: Name of contact person or department.

Property: arrowboards[*].owner.phone Type:
optional string
Description: Phone number for contact.

Property: arrowboards[*].owner.email
Type: optional string
Description: Email address for contact.

Property: arrowboards[*].gps Type:
required object
Description: Contains information about the GPS location of the arrowboard.

Property: arrowboards[*].gps.cycle Type:
optional number
Description: Number of seconds between attempts to detect GPS data. May be used to help determine when the client should next poll the SABP server.

Property: arrowboards[*].gps.override
Type: optional boolean
Description: This value is true if an override is in use to provide a location when an arrowboard is deployed where GPS cannot be used (e.g. in a tunnel). This value is false when an override is not in use. A null value indicates that the arrowboard does not support GPS override values.

Property: arrowboards[*].gps.tried Type:
required string
Description: Timestamp of latest attempt by the arrowboard to detect GPS data.

Property: arrowboards[*].gps.lock Type:
required number
Description: GPS lock status during the latest GPS detection attempt: 0 = no GPS lock, 1 = weak GPS lock, 2 = good GPS lock.

Note: The following three properties (sampled, lat, and lon) are only updated on GPS detection attempts when the gps.lock value is 1 (weak) or 2 (strong).

Property: arrowboards[*].gps.sampled
Type: required string
Description: Timestamp of most recent successful GPS attempt.

Property: arrowboards[*].gps.lat Type:
required number
Description: Most recent GPS latitude.

Property: arrowboards[*].gps.lon Type:
required number
Description: Most recent GPS longitude.

Property: arrowboards[*].display
Type: required object
Description: Contains information about the pattern shown on the arrowboard.

Property: arrowboards[*].display.deployed
Type: required boolean
Description: Deployed/stowed flag. If false, arrowboard is in its stowed position. If true, it is in its deployed position. A null value indicates that the arrowboard can't detect when the display is or isn't deployed.

Property: arrowboards[*].display.compass Type:
required number
Description: Digital compass value 0-360 indicating the direction traffic viewing the AB is moving. (Example: A south facing arrow board is viewed by northbound traffic so the value would be 0.)

Property: arrowboards[*].display.pattern Type:
required string
Description: Current pattern name. (See Appendix C for a list of pattern names.)

Property: arrowboards[*].lampErrors Type:
required object
Description: Contains information about lamp errors. This property is null if the arrowboard cannot detect lamp errors.

Property: arrowboards[*].lampErrors.count
Type: required number
Description: Count of lamps that have errors. Any lamp with a stuck on, stuck off, both, or other error counts as one towards this number. If the arrowboard can detect lamp errors, but is unable to obtain a count, this will show -1 any time there is a lamp error.

Property: arrowboards[*].lampErrors.max Type:
required number
Description: Total number of lamps on the arrowboard.

Property: arrowboards[*].lampErrors.pattern Type:
required string
Description: Name of most recent pattern displayed when a lamp failure was detected. This property is always null when lampErrors.count is zero.

Property: arrowboards[*].lampErrors.list
Type: optional array
Description: For arrowboards with enhanced lamp-error detection. Array contains array of lamp-identifier strings for lamps that have errors. Identifier strings are manufacturer- specific. This is null when there are no errors.

Property: arrowboards[*].voltage
Type: required number Description:
System voltage.

Property: arrowboards[*].temperature Type:
optional object
Description: Contains temperature information. If this property is null, that means the arrowboard contains no temperature sensors. If this object is not null, at least one of the following five "optional" temperature properties is required.

Note: All temperatures are in degrees Celsius.

Property: arrowboards[*].temperature.controller Type:
optional number
Description: Temperature of the controller.

Property: arrowboards[*].temperature.enclosure Type:
optional number
Description: Temperature of arrowboard enclosure.

Property: arrowboards[*].temperature.battery Type:
optional number
Description: Temperature of the battery.

Property: arrowboards[*].temperature.display Type:
optional number
Description: Temperature of the display.

Property: arrowboards[*].temperature.ambient Type:
optional number
Description: Ambient air temperature (shaded).

Property: arrowboards[*].errorCodes Type:
required array
Description: Array of short error-code strings. This is a catch-all for any detectable errors that are not listed elsewhere in this document. Any code placed in this list must automatically be removed a reasonable amount of time after the error condition is resolved. Each manufacturer is responsible for providing a list of errors that can appear in this list and the recommended steps for resolving each error. When there are no errors, this property will be null.

Property: arrowboards[*].lastContact Type:
required string
Description: Timestamp of last communication with arrowboard. For tier 1 documents, this is filled in by client software when it receives an SABP document from an arrowboard. For tier 2 documents, this is filled in by the consolidation server each time it successfully communicates with that specific arrowboard.

Appendix C: Special Property Values

Timestamp Values

SABP timestamps are string values containing date and time. They are formatted using the basic ISO 8601 standard (which is also the JavaScript toISOString() format). To avoid issues with configuring arrowboards for different time-zones, SABP timestamps are always formatted using the Zulu (UTC) timezone. A null value is used to represent a timestamp that is unknown.

Non-null timestamp values are formatted as:

"*yyyy-mm-ddThh:mm:ss.sssZ*"

Example:

"2012-04-23T18:25:43.500Z"

Additional info about ISO 8601 can be obtained at

http://en.wikipedia.org/wiki/ISO_8601.

GPS Coordinate Values

Latitude and longitude values are represented as signed decimal GPS coordinates.

Latitude values range from -90.0 to 90.0. Positive latitude values are above the equator. Negative latitude values are below the equator. Longitude values range from -180.0 to 180.0. Positive longitudes are east of the Prime meridian. Negative longitudes are west of the Prime Meridian.
Unknown GPS coordinates are represented by null values in both latitude and longitude properties.

Additional info about decimal GPS coordinates can be obtained at:

http://en.wikipedia.org/wiki/Decimal_degrees

Arrowboard Identifiers

Arrowboard IDs are strings that combine an arrowboard controller make, model, and serial number into a single string containing three semicolon-separated attributes.

Example:

"Foont Road Signs;AB3;1234-567-010"

These are used to uniquely identify each arrowboard controller without needing to manually assign a project or owner specific name to each arrowboard.

If a controller's firmware is unable to read an actual serial number, the first time the controller receives a GPS location, it will use the associated GPS timestamp to generate-and-save a unique pseudo-serial-number to be used for future SABP communication.

Arrowboard Patterns

The following is a list of pattern strings for use in the *pattern* property.

No pattern:
"Off"

Merge Right:
"Right Arrow, flashing" "Right Arrow, sequential"
"Right Stem Arrow, sequential" "Right Chevron, static"
"Right Chevron, flashing"
"Right Chevron, sequential"

Merge Left:
"Left Arrow, flashing" "Left Arrow, sequential"
"Left Stem Arrow, sequential" "Left Chevron, static"
"Left Chevron, flashing"
"Left Chevron, sequential"

Double-Arrow:
"Double Arrow, flashing"

Caution:
"Caution, Four Corner, flashing" "Caution, Bar, flashing"
"Caution, Alternating Diamonds, sequential"

Pattern used for testing the display:
"Test"

Notes:

1. When classifying patterns for the pattern property, sequential patterns that have an "all-off" pulse at the end of each pattern are considered equivalent to those that do not.
2. Depending on where you look, there are several interpretations for what a "stem arrow" is. For purposes of this protocol, the following pattern is considered a sequential arrow:

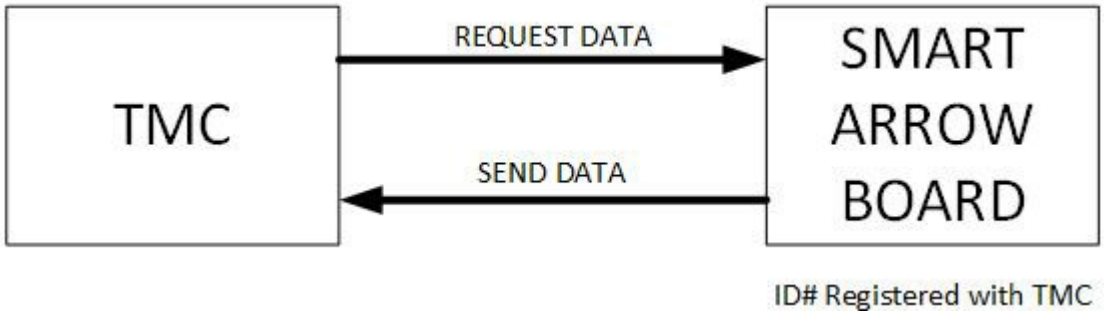
And the following is a sequential stem arrow:

3. If the AB has a pattern that does not exactly match, substitute the closest one of the given pattern names.

Option 2

Smart Arrow Board Protocol

(Data polled directly from the arrow board)



SABP

Smart Arrow Board Protocol – Option 2

Version 1.0

Iowa Department of Transportation

Prepared by:

SRF

August 14, 2019

SRF No. 12499

TOC Title: Table of Contents

Introduction

Timely and accurate information on work zones is becoming increasingly important, not only to road authorities, but to other public and private stakeholders involved with managing road construction and maintenance activities.

Several problems in providing this information are:

- Collecting and reporting timely information is time consuming for staff and competes with other project administration duties.
- Road construction and maintenance activities that require lane or shoulder closures are not always reported to operations staff, resulting in inaccurate or no dissemination to traveler information systems and the traveling public.
- Stakeholders desire more detailed records on the start time, end time, and location of lane closures for improved analysis of the transportation management plan (TMP) and performance measurement.

The in-vehicle displays that will be offered by connected vehicles present an opportunity for improved safety through better lane closure notifications to drivers. These automated displays will require more accurate and timely data to function correctly, however.

Flashing Arrow Boards are routinely used in advance of active work zones to designate lane closures on multi-lane highways. By adding readily available technologies the location and state of Flashing Arrow Boards can be reported without adding to staff workload.

Several manufacturers have created their own solutions to monitoring arrow board deployment, but these solutions do not lend themselves to easily merging information from arrow boards of different manufacturers.

This document describes a simple protocol for near-real-time deployment and basic maintenance monitoring of smart arrow boards on all Iowa DOT projects on multi-lane highways.

Scope

This document is intended for use by arrow board manufacturers, monitoring-software authors, DOT traffic engineers, and arrow board field technicians. It is assumed that the reader has some familiarity with software terms, TCP/IP communication, arrow board terminology, and procedures used for remote monitoring of field devices.

Definitions

For this document: "SABP" means "smart arrow board protocol". "AB" means "arrow board". "Client" means "central-monitoring or TCP-terminal software". And "controller" means "smart arrow board controller".

Communications Channel

It is assumed that all arrow boards supporting this protocol will use a wireless TCP/IP communication system determined by the arrow board manufacturer and the overseeing DOT. The only requirements placed on this channel by the protocol are:

A. A client must be able to open a channel to the arrow board via TCP/IP.
   (The default SABP port number is 23 (Telnet), but other port numbers could be used.)
B. The AB must be able to receive commands and send responses via the channel.
C. The channel must automatically close after 60 seconds of no data received or sent.

The diversity of wireless communications options places additional stipulations about the communications channel outside the scope of this document.

Communication Process

When a client (central software or technician at a TCP terminal) wishes to communicate with a smart arrow board, the client opens a TCP/IP connection to the arrow board's controller. The controller does not need to implement a full Telnet server, but must support use of the backspace character (0x08) to allow simple editing of commands.

The exchange of data between the two follows the standard client/server pattern with the client sending a command and the controller returning one or more lines of response. This process is repeated until the client has finished communicating with the arrow board. The client then closes the TCP connection.

To protect against orphaned TCP connections, if a TCP connection is not closed by the client, after 60 seconds of no-data (sent or received), the controller, or the modem at the controller end of the connection, automatically closes the TCP connection.

Objects and Object Groups

SABP uses a get/set object-based paradigm similar to SNMP and NTCIP. However, to simplify use of this protocol for controller firmware development and for use by field technicians without resorting to custom software: all commands are typeable ASCII, all objects have quasi-English language names, and all responses are (as much as possible) human-readable ASCII text.

Objects can be thought of as variables that reside in the controller. SABP commands allow the client to get or set the value of these objects. Some objects are read only. Some are read/write.

Each object has a name. All object names begin with a letter and are then followed by a combination of letters, digits, and underscores. The object name is used in protocol commands to address the individual object.

Each object holds an integer (int), a floating-point number (float), or an ASCII string (string). Any time a string value appears in a command or a response, it is quoted using double-quotes characters (" = 0x22). If a double-quote is included in a string, it is represented in a command or response by two adjacent double-quotes.

Objects can also be members of one or more object-groups. Each group has a name used in get commands to obtain information about a set of related objects. Naming conventions for object- groups are the same as those for object names. Examples of object group names are: **FIRMWARE, CONFIG, STATUS, PROJECT, GPS, POWER,** and **DISPLAY**. Some object groups also have shorter aliases. (See the "Object Groups" section of Appendix B for a list.)

The case of object and group names are ignored (case-insensitive) in commands. (*NAME*, *nAMe*, and *name* all address the same object.) Object and group names are always upper-case in responses.

An alphabetical list of object and group names with short descriptions is included in Appendix A. A list of objects, grouped by category, with more information about each is included in Appendix B.

Commands and Responses

There are four commands:

- Comment
- Are you there?
- Get
- Set

Each command line is terminated with a carriage-return (0x0D) character. Each line of text in a response is terminated with a carriage-return + linefeed (0x0D + 0x0A) pair of characters. For the remainder of this document, we won't mention the end-of-command or end-of-response-line characters. Just assume they are implicitly included in the descriptions of all SABP commands and responses.

Any whitespace characters in a command that are not part of a quoted string are ignored.

Each set of response lines from the controller is terminated by four minus characters ("----") on a line by themselves.

Examples of command/response exchanges are shown in Courier-New font with command lines prefixed by "< " and response lines prefixed by "> ". (The "< " and "> " are **NOT** part of the commands or responses. They are simply there to visually indicate the source (client or controller) of that line of text.)

Example:

```
< ?name
> NAME="Arrow Board 17"
> ----
```

Other than in the exchange examples, object names are shown in *ITALICS* and object groups are shown in **BOLD**.

If the controller determines that a command is incorrectly formatted, refers to a non-existent object, or contains an incorrect value, the controller responds with an error message prefixed by "!Error: ".

Examples:

```
< gps_cycle="Bar"
> !Error: GPS_CYCLE value must be an integer
> ----
< reboot=99
> !Error: REBOOT value must be in the range 0 to 1
> ----
< @baz
> !Error: Invalid command
> ----
```

There is a list of all expected error messages in Appendix D.

Comment Command

A comment command consists of a hash/pound (# = 0x23) character followed by any number of characters up to the next carriage return. This command is ignored by the controller and is intended to allow imbedded comments in a command-script or logfile. Because there is no response to this command, the controller does not send a "----" line to end the response.

"Are You There?" Command

This is an empty command (a simple carriage-return). The default response is the controller's assigned name and the current protocol (SABP version).

Example:

```
<
> NAME="Arrow Board 17"
> PROTOCOL="SABP 1.0"
> ----
```

The objects returned in response to this command can be changed or the response to this command can be suppressed entirely using the *ARE_YOU_THERE* object.

Get Command

The get command is used to retrieve the current value of object(s) from the controller. It consists of a question mark followed by one or more object names separated by commas. The response consists of one line for each object name, showing the object name, an equals symbol, and the current value of that object.

Example:

```
< ?name, gps_cycle
> NAME="Arrow Board 17"
> GPS_CYCLE=600
> ----
```

If a get command contains requests for unknown object(s), an error message flags each unknown object(s) and the current values of known object(s) are returned.

Example:

```
< ?name, foo,gps_cycle
> NAME="Arrow Board 17"
> !Error: FOO is not a known object
> GPS_CYCLE=600
> ----
```

Group names can be used to retrieve all objects in an object group.

Example:

```
< ?gps
(Controller responds with all GPS related objects and their current values.)
> ----
```

An intersection of more than one group can be retrieved using multiple group names, separated by an "and" (& = 0x26) character.

Example:

```
< ?gps&status
(Controller responds with all GPS-status objects and their current values.)
> ----
```

If the order of objects is different in the groups used to create an intersection, the order of objects in the list returned by the controller matches the order of objects in the first group in the intersection. In the above example, the order of objects would match that of the **GPS** group.

There are two special objects that can be used for object and group discovery. *OBJECTS* contains a string with a comma-separated list of all objects in the controller. *GROUPS* contains a string with a comma-separated list of all groups in the controller. (Note: The *OBJECTS* string is rather long. It currently contains over 32 object names. However, it does **NOT** reference the *OBJECTS* or *GROUPS* objects.)

Example:

```
< ?objects,groups
> OBJECTS="NAME,ARE_YOU_THERE,…"
> GROUPS="CONFIG,STATUS,HARDWARE,FIRMWARE,…"
> ----
```

Set Command

The set command consists of one or more, comma-separated, assignments. Each assignment consists of an object name, an equals symbol, and a value. The response consists of one line for each assignment, showing the object name, an equals symbol, and the new value of that object.

Example:

```
< name="Arrow Board 18", gps_cycle= 1200
> NAME="Arrow Board 18"
> GPS_CYCLE=1200
> ----
```

If there is an error in a set command:

- All assignments before the error are processed.
- An error message is shown, describing the error.
- If there were any assignments after the error, they are ignored and another error is shown.

Example:

```
< name="Arrow Board 18",foo=45,gps_cycle=1200
> NAME="Arrow Board 18"
> !Error: FOO is not a known object
> !Error: Assignment(s) were ignored
> ----
```

Appendix A: SABP Objects (Alphabetical)

Object Name	Type	Short Description	Required?	Category
ARE_YOU_THERE	string	"Are You There?" objects.	Required	General
CONFIG	group	Configuration group	Required	Groups
COMPASS	int	AB compass direction	Optional	Display
DEPLOYED	int	AB is deployed	Optional	Display
DISPLAY	group	Main display group	Required	Groups
ERROR_CODES	string	List of other errors.	Required	Errors
FACTORY_RESET	int	Factory reset flag	Required	Other
FAILED_COUNT	int	Number of failed lamps	Optional	Display
FAILED_LAMP	int	Failed lamp flag	Required	Display
FAILED_LIST	string	List of failed lamps	Optional	Display
FAILED_PATTERN	string	Failed lamp pattern	Required	Display
FIRMWARE	group	Firmware group	Required	Groups
FW_NAME	string	Firmware name	Required	Firmware
FW_VER	string	Firmware version	Required	Firmware
GPS	group	GPS group	Required	Groups
GPS_AGE	int	Age of GPS data (seconds).	Required	GPS
GPS_ATTEMPT	string	Datetime of last GPS polling attempt	Required	GPS
GPS_CYCLE	int	GPS polling cycle (seconds)	Required	GPS
GPS_LAT	float	GPS latitude	Required	GPS
GPS_LOCK	int	GPS lock status	Required	GPS
GPS_LON	float	GPS longitude	Required	GPS
GPS_OVERRIDE	string	GPS location override	Required	GPS
GPS_TIMESTAMP	string	Datetime of last GPS sample	Required	GPS
GROUPS	string	List of object-groups	Required	Other
HARDWARE	group	Hardware group	Required	Groups
HW_COMPANY	string	AB manufacturer name	Required	Hardware
HW_MODEL	string	AB model name	Required	Hardware
HW_SERIAL_NO	string	AB serial number	Optional	Hardware
HW_VERSION	string	AB version	Required	Hardware
JITTER_FILTER	int	Jitter filter (meters)	Required	GPS
LAMP_COUNT	int	Lamps on arrow board	Required	Hardware
NAME	string	Assigned arrow board name.	Required	General
OBJECTS	string	List of objects	Required	Other
OTHER	group	Other group	Required	Groups
PATTERN	string	Current pattern	Required	Display
POWER	group	Power group	Required	Groups

Object Name	Type	Short Description	Required?	Category
PROTOCOL	string	Protocol	Required	Firmware
REBOOT	int	Reboot flag	Required	Other
RTC_TIME	string	Current RTC datetime.	Required	Time
STATUS	group	Status group	Required	Groups
TEMP_AMBIENT	int	Air temperature.	Optional	Temperature
TEMP_BATTERY	int	Battery temperature.	Optional	Temperature
TEMP_CONTROLLER	int	Controller temperature.	Optional	Temperature
TEMP_DISPLAY	int	Display temperature.	Optional	Temperature
TEMP_ENCLOSURE	int	Enclosure temperature.	Optional	Temperature
TEMPERATURE	group	Temperature group	Optional	Groups
TIME	group	Time group	Required	Groups
TIME_ZONE	string	Local timezone offset.	Optional	Time
VOLTAGE	float	System voltage	Required	Power

Appendix B: SABP Object Details (Grouped by Category)

B.1 General

Object Name	Type	Description	Required	Default	Get/Set	Group(s)
NAME	string	Assigned name of the arrow board. This object is included as the first object in all object groups.	Required	""	Get/Set	**CONFIG**
ARE_YOU_THERE	string	Comma separated list of objects to be reported in response to the "Are you there?" command. If this object is empty, then nothing (not even the terminating "----") is returned in response to this command.	Required	"NAME,PROTOCOL"	Get/Set	**CONFIG**

B.2 Hardware

Object Name	Type	Description	Required	Default	Get/Set	Group(s)
HW_COMPANY	string	Name of arrow board manufacturer	Required	<manufacturer name>	Get	**HARDWARE**
HW_MODEL	string	Arrow board model name	Required	<AB model name>	Get	**HARDWARE**
HW_VERSION	string	Arrow board version	Required	<AB version>	Get	**HARDWARE**
HW_SERIAL_NO	string	Arrow board serial number	Optional	<AB serial number>	Get	**HARDWARE**
LAMP_COUNT	int	Number of lamps on the arrow board. (This is also the maximum possible *FAILED_COUNT* value.)	Required		Get	**DISPLAY, STATUS, HARDWARE**

B.3 Firmware

Object Name	Type	Description	Required	Default	Get/Set	Group(s)
FW_NAME	string	Name of controller firmware	Required	<firmware name>	Get	**FIRMWARE**
FW_VER	string	Version number of firmware.	Required	<firmware version>	Get	**FIRMWARE**
PROTOCOL	string	Protocol supported by this firmware.	Required	"SABP 1.0"	Get	**FIRMWARE, COMM**

B.4 GPS

Object Name	Type	Description	Required	Default	Get/Set	Object Group(s)
GPS_CYCLE	int	Number of seconds between GPS polling attempts. If set to 0, GPS polling is disabled.	Required	600	Get/Set	**GPS, CONFIG**
GPS_OVERRIDE	string	GPS location override. When set to a value other than "", this contains a quoted, comma separated "latitude, longitude" pair of float values that are placed in the *GPS_LAT* and *GPS_LON* objects instead of the values reported by the GPS. This is intended for use when an arrow board is deployed where GPS cannot be used (e.g. in a tunnel). This value does NOT affect updating any values except *GPS_LAT* and *GPS_LON.*	Required	""	Get/Set	**GPS, CONFIG**
JITTER_FILTER	int	When polling the GPS, if the GPS reports that the arrow board has moved less than this number of meters, the *GPS_LAT* and *GPS_LON* object values are not changed. This value does NOT affect updating any values except *GPS_LAT* and *GPS_LON.*	Required	100	Get/Set	**GPS, CONFIG**
GPS_LOCK	int	GPS lock status during the latest GPS polling attempt: 0 = no GPS lock, 1 = weak GPS lock, 2 = good GPS lock	Required	0	Get	**GPS, STATUS, ERRORS**
GPS_ATTEMPT	string	Datetime timestamp when latest GPS poll was attempted. The *TIME_ZONE* object (if set) is used when generating this string. (See Appendix E for notes on datetime string formats.)	Required	""	Get	**GPS, STATUS**
GPS_TIMESTAMP	string	Datetime timestamp from the most recent good GPS sample. The *TIME_ZONE* object (if set) is used when generating this string. (See Appendix E for notes on datetime string formats.)	Required	""	Get	**GPS, STATUS**
GPS_AGE	int	Number of seconds since last successful GPS poll. This is the difference between *RTC_TIME* and *GPS_TIMESTAMP* in seconds.	Required	0	Get	**GPS, STATUS, ERRORS**
GPS_LAT	float	Most recent GPS latitude in decimal degrees. A value of 91.0 indicates that no GPS sample has been obtained. The normal range of values is -90.0 to 90.0. (See *GPS_OVERRIDE* and *JITTER_FILTER* for additional information.)	Required	91.0	Get	**GPS, STATUS**
GPS_LON	float	Most recent GPS longitude in decimal degrees. A value of 181.0 indicates that no GPS sample has been obtained. The normal range of values is -180.0 to 180.0. (See *GPS_OVERRIDE* and *JITTER_FILTER* for additional information.)	Required	181.0	Get	**GPS, STATUS**
Note: See Appendix E for notes on datetime string formats.						

B.5 Display

Object Name	Type	Description	Required	Default	Get/Set	Object Group(s)
COMPASS	int	Digital compass value 0-360 indicating the direction traffic viewing the AB is moving. (Example: A south facing arrow board is viewed by northbound traffic so *COMPASS* value would be 0.) A value of 999 indicates an error or compass not-available.	Optional	999	Get	**DISPLAY, STATUS**
DEPLOYED	string	Deployed/stowed flag. If "No", AB is in its stowed position. If "Yes", AB is in its deployed position.	Optional	"Yes"	Get	**DISPLAY, STATUS**
PATTERN	string	Current pattern name. (See Appendix C for a list of pattern names.)	Required	""	Get	**DISPLAY, STATUS**
FAILED_LAMP	int	Basic failed-lamp flag. (Usually detected via current sensor.) Returns 0 if all lamps in use are functional. Returns 1 if one or more lamps have failed. Controller keeps track of which pattern was showing when the failure was detected and sets this value back to 0 when that pattern is shown without a failure.	Required	0	Get	**DISPLAY, STATUS, ERRORS**
FAILED_PATTERN	string	Name of pattern displayed when *FAILED_LAMP* condition was detected. This is always "" when *FAILED_LAMP* returns 0.	Required	""	Get	**DISPLAY, STATUS, ERRORS**
FAILED_COUNT	int	Enhanced lamp-failure. (For arrow boards with enhanced lamp-failure detection.) Set to the number of lamps currently failed. Any type of failure (stuck on, stuck off, or other) on one lamp counts as 1 towards this number.	Optional	0	Get	**DISPLAY, STATUS, ERRORS**
FAILED_LIST	string	Enhanced lamp-failure. (For arrow boards with enhanced lamp-failure detection.) String contains a semicolon separated list of failed-lamp identifiers. Identifiers are manufacturer-specific.	Optional	""	Get	**DISPLAY, STATUS, ERRORS**

B.6 Power

Object Name	Type	Description	Required	Default	Get/Set	Object Group(s)
VOLTAGE	float	System voltage (Returns -999.0 if sensor has malfunctioned.)	Required	0	Get	**POWER, STATUS**

B.7 Time

Object Name	Type	Description	Required	Default	Get/Set	Object Group(s)
TIME_ZONE	string	Local timezone offset in numerical ISO 8601 timezone format (example: "-5:00"). This is used to adjust *RTC_TIME, GPS_ATTEMPT,* and *GPS_TIMESTAMP* values returned via the get command. If this value is empty, the controller returns datetime values using the "Z" (Zulu) timezone.	Optional	""	Get/Set	**TIME, CONFIG**
RTC_TIME	string	Current controller RTC (real time clock) datetime. The RTC used to create this string is updated every GPS polling cycle when there is a GPS lock. This datetime string is generated on the fly from the controller's RTC when this object is requested via the get command. The *TIME_ZONE* object (if set) is used when generating this string.	Required	""	Get	**TIME, STATUS**
Note: See Appendix E for notes on timezone and datetime string formats.						

B.8 Temperature

Object Name	Type	Description	Required	Default	Get/Set	Object Group(s)
TEMP_CONTROLLER	int	Temperature of controller	Optional	0	Get	**TEMPERATURE, STATUS**
TEMP_ENCLOSURE	int	Temperature of controller enclosure	Optional	0	Get	**TEMPERATURE, STATUS**
TEMP_BATTERY	int	Temperature of battery	Optional	0	Get	**TEMPERATURE, STATUS**
TEMP_DISPLAY	int	Temperature of display	Optional	0	Get	**TEMPERATURE, STATUS**
TEMP_AMBIENT	int	Ambient air temperature (shaded)	Optional	0	Get	**TEMPERATURE, STATUS**
Note: All temperatures are in whole centigrade degrees. Returns -999 if sensor has malfunctioned						

B.9 Errors

Object Name	Type	Description	Required	Default	Get/Set	Object Group(s)
ERROR_CODES	string	Semicolon-separated list of short error codes. This is a catch-all for any detectable errors that are notlisted elsewhere in this document. Any code placed in this list must automatically be removed a reasonable amount of time after the error condition is resolved. Each manufacturer is responsible for providing a list of errors that can appear in this list and the recommended steps for resolving each error.	Required	""	Get	**ERRORS, STATUS**

B.10 Reboot/Reset

Object Name	Type	Description	Required	Default	Get/Set	Object Group(s)
REBOOT	int	When the TCP connection is closed, if this value is set to 1, the controller sets the value to 0 and then reboots.	Required	0	Get/Set	**OTHER**
FACTORY_RESET	int	When the TCP connection is closed, if this value is set to 1, the controller sets all objects to their default values and then reboots.	Required	0	Get/Set	**OTHER**

B.11 Discovery Objects

Object Name	Type	Description	Required	Default	Get/Set	Object Group(s)
OBJECTS	string	List of objects (does not include *OBJECTS* or *GROUPS* objects)	Required		Get	**OTHER**
GROUPS	string	List of object groups (does not include group aliases)	Required		Get	**OTHER**

B.12 Object Groups

Group Name	Type	Description	Required	Get/Set
CONFIG	group	Configuration (arrow board setup-objects) (alias = CFG)	Required	Get
STATUS	group	Status (arrow board monitoring-objects)	Required	Get
HARDWARE	group	Hardware Info (alias = HW)	Required	Get
FIRMWARE	group	Firmware Info (alias = FW)	Required	Get
TIME	group	Datetime & Timezone	Required	Get
DISPLAY	group	Main Arrow Board Display	Required	Get
GPS	group	GPS	Required	Get
POWER	group	Electrical Power	Required	Get
TEMPERATURE	group	Temperatures (alias = TEMP)	Optional	Get
OTHER	group	All objects not included in any other group	Required	Get

Appendix C: Arrow Board Patterns

The following is a list of pattern strings for use in the *PATTERN* object.

No pattern:
"Off"

Merge Right:
"Right Arrow, static"
"Right Arrow, flashing"
"Right Arrow, sequential"
"Right Stem Arrow, sequential"
"Right Chevron, static"
"Right Chevron, flashing"
"Right Chevron, sequential"

Merge Left:
"Left Arrow, static"
"Left Arrow, flashing"
"Left Arrow, sequential"
"Left Stem Arrow, sequential"
"Left Chevron, static"
"Left Chevron, flashing"
"Left Chevron, sequential"

Double-Arrow:
"Double Arrow, static"
"Double Arrow, flashing"

Caution:
"Caution, Four Corner, flashing"
"Caution, Bar, flashing"
"Caution, Alternating Diamonds, sequential"

Notes:
1. When classifying patterns for the PATTERN object, sequential patterns that have an "all- off" pulse at the end of each sequence are considered equivalent to those that do not.
2. Depending on where you look, there are several interpretations for what a "stem arrow" is. Or purposes of this protocol, the following pattern is considered a sequential arrow:

And the following is a sequential stem arrow:

Appendix D: Errors

The following is a list of known protocol errors:

!Error: <object name> value must be an integer
!Error: <object name> value must be a float
!Error: <object name> value must be a string
!Error: <object name> value must be in the range <minval> to <maxval>
!Error: <object name> value must be an ISO timestamp
!Error: <object name> value must be an ISO timezone offset
!Error: Invalid value for <object name>
!Error: <object name> is not a known object
!Error: Invalid command
!Error: Unbalanced string quotes
!Error: Assignment(s) were ignored

Appendix E: Additional Notes

E.1 General Smart Arrow Board Requirements

All Smart Arrow Boards are battery/solar-powered arrow panels that must meet the requirements found in the Manual on Uniform Traffic Control Devices (MUTCD), 2009, Part 6F.61. Arrow boards used on construction projects must also meet Iowa DOT Specification Article 2528.03.G.3 and be included on the Approved Products are listed in the Materials Approved Products Listing Enterprise (MAPLE).

E.2 Datetime and Time-zone Formats

Datetime (*RTC_TIME* and *GPS_TIMESTAMP* objects) strings and timezone (*TIME_*ZONE object) strings use extended ISO 8601 formats.

Datetime string values are formatted as:
"" (datetime is unknown),
"yyyy-mm-dd hh:mm:ssZ",
or
"yyyy-mm-dd hh:mm:ss±hh:mm".

Timezone string values are formatted as:
"" (Zulu timezone) or
"±hh:mm".

Additional info about ISO 8601 can be obtained at
http://en.wikipedia.org/wiki/ISO_8601.

E.3 Decimal Latitude and Longitude Coordinates

Info about decimal GPS coordinates can be obtained at
http://en.wikipedia.org/wiki/Decimal_degree.

E.4 Why not use XML, JSON, ASN.1, or DATEX?

While any of these standards could have been used for this protocol, all of them:

1. Make it more difficult for a human to issue ad-hoc commands or read results without use of custom software.
2. Can incur added development or operational costs (coding complexity, programming language constraints, or data transfer costs).
3. Require more effort to create scripts and scripting tools to monitor the arrow boards.

Bottom line is that many kinds of data transfer formats could be made to work with enough effort, but for the goals of an easy-to-implement/easy-to-debug/simple-to-use protocol, hierarchical or packed binary formats don't fit as well as simple ASCII text.

Testing Procedure (for either option)

Step 1 – Make sure device is off and at the start location

Make sure it is off and the data feed represents this

Step 2 – Right Chevron

Turn the chevron to right and wait at least 5 minutes so the data is archived

Step 3 – Left Chevron

Turn the chevron to left and wait at least 5 minutes so the data is archived

Step 4 – Move 500'

Move the arrow board at least 500' (try to minimize as much as possible)

Step 5 – Wait 5 minutes after 1st move

Wait 5 minutes to see if the location is refined

Step 6 – Move 500' again

Move the arrow board again at least 500' (try to minimize as much as possible)

Step 7 – Wait 5 minutes after 2nd move

Wait 5 minutes to see if the location is refined

Step 8 – Right Chevron again

Change to right chevron to make sure the device location and information is updated

Step 9 – Wait 1 hour (if on roadway wait as long as possible)

Wait 1 hour to see how much the check-in occurs

Step 10 – Turn Device Off

Turn the device off and record information.