Revision 1.1.0 - 7-Feb-2024

Hexagon’s Core Box is an Auto Steering module, capable to steer most tractors, self-propelled and harvesters' models in the market. This document described the API used to communicate with the Core Box for settings, tuning and operation.

Use the Revision History to check what was changed from the previous versions.

1 Overview

This document contains a draft of the Core Box External Communication API. Please consider the Core Box will be referred to as TiX or Titanium on multiple occasions.

The system is divided into objects, where each object represents a different logical component. These objects implement interfaces, which have methods that can be invoked, and signals that the object can emit.

Methods are used to execute commands in the Core Box. Signals are used to send information triggered by hardware or software events, and periodic data (such as GNSS frames).

Every object implements the Introspectable interface, which contains the Introspect method, which lists all available interfaces, methods, and signals.

Section 2 contains the description of the high-level communication protocol.

Section 3 describes the system objects and their interfaces.

Section 4 describes the introspection of the system.

A brief overview of basic steps to run Core Box are:

1. register for signal '*NewGNSSData' (see section 2.1)*
2. set the current vehicle (see section 4.6)
3. set Guidance Source File (see section 7.2.2)
4. register for signal 'SteeringStatus' (see section 3.5)
5. register for signal '*UpdateGuidance' (*see section 7.1)

OBJ

2 Communication Protocol

The communication protocol is based on the exchange of JSON messages over WebSocket.

All JSON messages have these common fields:

- **event:** indicates the type of JSON message.
- **id:** timestamp used to identify messages.
- **object:** represents the logical system component that will be manipulated.
- **interface:** the object's interface.
- **member:** the interface's member, which can be either a method or a signal.

2.1 Command message

This message is used to execute a method call in the Core Box. The Core Box will answer valid command messages with a command result message.

The params field can be omitted when the method does not have any input parameters. An empty array in the params field is also accepted.

Each parameter must have a "type" field, as well as a value field, because it is necessary to match the call to the underlying C++ method signature. The possible parameter types are listed below.

```
{
  "event":"call_method",
  "id":1455198438,
  "object":"/object_name",
  "interface":"interface_name",
  "member":"method_name",
  "params":[
          {
          "type":"int64",
          "value":100
          },
          {
          "type":"string",
          "value":"abcd"
          }
  ]
}
```

2.1.1 Parameter types

Name	Description
byte	8-bit unsigned integer, encoded as a JSON number.
boolean	JSON Boolean
int16	16-bit signed integer, encoded as a JSON number.
uint16	16-bit unsigned integer, encoded as a JSON number.
int32	32-bit signed integer, encoded as a JSON number.
uint32	32-bit unsigned integer, encoded as a JSON number.
int64	64-bit signed integer, encoded as a JSON number.
uint64	64-bit unsigned integer, encoded as a JSON number.
double	IEEE 754 double, encoded as a JSON number.
string	UTF-8 string
type array	The array *type* must be one of the above types. Arrays cannot have mixed types.

2.2 Command result message

This message returns the result of an executed command message.

The type of result will vary according to the method's return type but will always be the same for a combination of object + interface + method. The result type will be one of the types listed in section 2.1.1.

```
{
  "event":"method_result",
  "id":1455198438,
  "object":"/object_name",
  "interface":"interface_name",
  "member":"method_name",
  "result":"{"p1":result_value}"
}
```

2.3 Subscribe message

This message should be sent when the application wants to listen to an object's signal. Signals are asynchronous messages sent by the object, normally triggered by an event in the Core Box. For example, for every GNSS frame received by the Core Box, the GNSS object sends a signal with the new GNSS data.

The Core Box will reply to the subscribe message with a subscribe reply, and from that moment onwards will forward the desired signal messages to the application.

```
{
  "event":"subscribe",
  "id":1455198438,
  "object":"/object_name",
  "interface":"interface_name",
  "member":"signal_name"
}
```

2.4 Subscribe reply message

This message indicates whether a subscribe message was successful, a result value different than 0 means an error occurred and the signal will not be forwarded.

```
{
  "event":"subscribe_result",
  "id":1455198438,
  "object":"/object_name",
  "interface":"interface_name",
  "member":"signal_name",
  "result":0
}
```

2.5 Unsubscribe message

This message should be sent if you want to stop listening to an object's signal.

```
{
  "event":"unsubscribe",
  "id":1455198438,
  "object":"/object_name",
  "interface":"interface_name",
  "member":"signal_name"
}
```

2.6 Unsubscribe reply message

This message indicates whether an unsubscribe message was successful.

```
{
  "event":"unsubscribe_result",
  "id":1455198438,
  "object":"/object_name",
  "interface":"interface_name",
  "member":"signal_name",
  "result":0
}
```

2.7 Signal message

A signal is an asynchronous message sent from the Core Box to the application. The result field of a signal will always contain an array with the data sent by the signal. If the signal does not send any data, it will contain an empty array.

```
{
  "event":"signal",
  "id":1455198438,
  "object":"/object_name",
  "interface":"interface_name",
  "member":"signal_name",
  "result": "{
          \"p1\":100,
          \"p2\":\"abcd\",
          \"p3\":10.6
  }"
}
```

2.8 Error message

An error message will be sent whenever there is a problem with a received message. The result field will contain the description of the error, such as “Invalid event”, “Object does not exist”, “Method does not exist”, etc.

```
{
  "event":"error",
  "id":1455198438,
  "object":"/object_name",
  "interface":"interface_name",
  "member":"member_name",
  "result":"error_message"
}
```

3 System Objects

3.1 System

3.1.1 Request connection – Deprecated (>4.8.0)

This command is no longer used.

3.1.2 Get Generic params

To Get a Param

```
{
 "event":"call_method",
 "id":1455198438,
 "object":"/System",
 "interface":"br.com.arvus.TiX.System",
 "member":"GetParam",
 "params":[
        {
          "type": "uint16",
          "value" : param_code
        }
 ]
}
```

To Get a Param Response

```
{
 "event":"method_result",
 "id":1455198438,
 "object":"/System",
 "interface":"br.com.arvus.TiX.System",
 "member":"GetParam",
 "result":" {
      "{\"p1\"[param_code, param_value]}"
  }"
}
```

Param_code and param_value are strings

The field “param_code” could be one of them:

param_code (uint16)	description	Supported values	Default	Type	Can be set
0	TiX_FW_version	-	-	string	n
1	Tix_SERIAL_NUBER	-	-	string	n
2	TiX_MAC_ADDRESS	-	-	string	n
3	Core Box temperature (Celsius)	-	-	string	n
4	OEM_CODE	-	-	string	n
5	Steering FW version	-	-	string	n
6	Eth0 Mac address	-	-	string	n
-1	Error - Parameter not found	0	0	string	n

3.1.3 Buzzer Beep

The BuzzerBeep command depends on the buzzer state to be enabled (see 3.1.4). Otherwise, the buzzer shall not beep.

Buzzer beep command

```
{
 "event": "call_method",
 "id": 1455198438,
 "object": "/System",
 "interface": "br.com.arvus.TiX.System",
 "member": "BuzzerBeep",
 "params": [
          {
          "type": "uint16",
          "value": type
          }
          ]
}
```

Param Field	Description
type - uint16	Buzzer sound type • 1 - long beep (1 second) • 2 - short beep (300 milliseconds) • 3 - start continuous long beep (1 second between beeps) • 4 - stop continuous long beep

Buzzer beep command response

```
{
 "event": "method_result",
 "id": 1455198438,
 "object": "/System",
 "interface": "br.com.arvus.TiX.System",
 "member": "BuzzerBeep",
 "result": "{
   \"p1\" : result
  }"
}
```

The result (uint16) could be:

- 0 - success
- 1 - any buzzer problem (reserved)
- 2 - buzzer disabled (API commands)

3.1.4 Buzzer Enable

Buzzer enable command

```
{
 "event": "call_method",
 "id": 1455198438,
 "object": "/System",
 "interface": "br.com.arvus.TiX.System",
 "member": "BuzzerEnable",
 "params": [
          {
          "type": "uint16",
          "value": status
          }
          ]
}
```

Param Field	Description
type - uint16	• 0 - disabled • 1 - enabled – Core Box and API can send commands (this is the default) • 2 - enabled - only accept commands from the API

Buzzer enable command response

```
{
 "event": "method_result",
 "id": 1455198438,
 "object" : "/System",
 "interface" : "br.com.arvus.TiX.System",
 "member" : "BuzzerEnable",
 "result": "{
   \“p1\” : result
  }"
}
```

The result (uint16) could be:

- 0 - success
- 1 - any buzzer problem

3.1.5 Get Buzzer State

Get Buzzer State command

{
"event": "call_method",
"id": 1455198438,
"interface": "br.com.arvus.TiX.System",
"member": "GetBuzzerState",
"object": "/System"
}

Get Buzzer State command response

{
"**event**": "method_result",
"**id**": 1455198438,
"**object**" : "/System",
"**interface**" : "br.com.arvus.TiX.System",
"**member**" : "GetBuzzerState",
"**result**": "{
\“p1\” : result
}"
}
 	
The result (uint16) could be one of the buzzer states (see 3.1.4 – Get Buzzer State)

3.1.6 Core Box Alarms

The alarms_state is an uint64, and each bit of the variable holds the state of a type of alarm (1 - alarm is triggered, 0 - alarm is cleared)

Bit	Alarm description
1	CAN short circuit alarm
2	Low voltage alarm
3	High voltage alarm
4	CPU temperature warning
5	CPU temperature critical
6	Memory warning (RAM)
7	Memory critical (RAM)
8	Memory warning (SD)
9	Memory critical (SD)

Get Alarms command

{
"**event**": "call_method",
"**id**": 1455198438,
"**object**" : "/System",
"**interface**" : "br.com.arvus.TiX.System",
"**member**" : "GetAlarms"
}

Get Alarms command response

{
"**event**": "method_result",
"**id**": 1455198438,
"**object**" : "/System",
"**interface**" : "br.com.arvus.TiX.System",
"**member**" : "GetAlarms",
"**result**": "{
\“p1\” : alarms_state
}"
}
 	
The result (uint64)

3.1.7 Core Box Alarms Signal

If you want to receive automatic updates when an alarm state changes, you should register to listen to this signal. It is recommended to call the command “Get alarms” at least one time before registering for this signal.

To register

{
"**event**": "subscribe",
"**id**": 1455198438,
"**object**”: "/System",
"**interface**”: "br.com.arvus.TiX.System",
"**member**”: "AlarmsSignal"
}

Register Response
{

```
  "event": "subscribe_result",
  "id": 1455198438,
  "object”: "/System",
  "interface”: "br.com.arvus.TiX.System",
  "member”: "AlarmsSignal",
  “result”: 0
}
```


Periodic Message


```
{
  "event": "signal",
  "id": 1455198438,
  "object”: "/System",
  "interface”: "br.com.arvus.TiX.System",
  "member”: "AlarmsSignal",
  "result”: "{
    \“p1\”: alarms_state,
  }"
}
```


3.1.8 Power-off / reboot Core Box

Power commands


```
{
 "event": "call_method",
 "id": 1455198438,
 "object”: "/System",
 "interface”: "br.com.arvus.TiX.System",
 "member”: "Power",
"params": [
            {
                "type": "uint16",
                "value”: command
             }
          ]
}
```


Command could be:

- 0 - turn-off
- 1 - reboot

Power command response


```
{
```

```
"event": "method_result",
"id": 1455198438,
"object": "/System",
"interface": "br.com.arvus.TiX.System",
"member": "Power",
"result": "{
  \"p1\" : result
 }"
}
```

The result (uint16) could be:

- 0 - success
- 1 - any problem

3.1.9 Operation Completed Signal

The System Manager is a service that handles software updates, backups, and diagnostics. Since most of these operations involve the file system, they may take a few seconds or even minutes to complete, exceeding the DBus message timeout, therefore most of these operations are executed asynchronously. Async methods use signals, or a second method call to determine if the operation was completed and its result.

Most of the methods return one of the status codes in the table below, where negative values always indicate errors.

Value	Name	Description
1	InProgress	The async task was launched successfully.
0	Success	The task completed successfully.
-1	Failure	Unable to complete the task.
-2	InvalidName	The specified name does not exist
-3	NoUsbDevice	There is no available USB device
-4	MaxBackupsLimitReached	The maximum number of backups was reached
-5	Busy	There is another task in progress.

This signal is emitted every time an asynchronous task is completed and returns the result of the task.

Subscribe to Operation Completed Signal
{ "event":"subscribe",

```
  "id":1455198438,
  "object":"/Backup",
  "interface":"br.com.arvus.SystemManager.Backup",
  "member":"SignalOperationCompleted"
}
```

Subscribe Reply

```
{
  "event":"subscribe_result",
  "id":1455198438,
  "object":"/Backup",
  "interface":"br.com.arvus.SystemManager.Backup",
  "member":"SignalOperationCompleted"
}
```

Operation Completed Signal Message

```
{
  "event":"signal",
  "id":1455198438,
  "object":"/Backup",
  "interface":"br.com.arvus.SystemManager.Backup",
  "member":"SignalOperationCompleted",
  "result":"{ \"p1\": status_code}"
}
```

3.1.10 Factory Reset

Restores the Core Box to the original factory configuration. This is an asynchronous method, Operation Completed Signal will be emitted once the background task completes.

Factory reset command commands

```
{
  "event":"call_method",
  "id":1455198439,
  "object":"/Backup",
  "interface":"br.com.arvus.SystemManager.Backup",
  "member":"FactoryReset",
  "params":[
        {
        "type":"boolean",
        "value":true
        }
  ]
}
```

If the param is true Core Box will be rebooted automatically after the process

Power command response

{
"event":"method_result",
"id":1455198438,
"object":"/Backup",
"interface":"br.com.arvus.SystemManager.Backup",
"member":"FactoryReset",
"result":"{ \"p1\": *status_code*}"
}

3.2 GNSS

3.2.1 Signal GNSS Data Frame

This message is continuously sent from the Core Box's ECU at each position update (typical frequency of 20 Hz).

To receive this message the device must register to listen to the signal 'NewGNSSData*XXX'*.

The coordinate frame used to report the position is WGS-84 (degrees for latitude, long and meters for altitude).

The position reported is already corrected using roll/pitch angles and nudge.

The heading provided by this message has a range of 0-360 degrees (0 north, 90 east, 180 south, 270 west), even if reversing.

Information: The Core Box uses the GNSS frame data to update these signals, if the GNSS is not connected then every signal is updating every 2 seconds.

To Register

{
"event": "subscribe",
"id": 1455198438,
"object" : "/GNSS",
"interface" : "br.com.arvus.TiX.GNSS",
"member" : "*member_type*"
}

member_type	Description
NewGnssData1hz	Register for receiving messages with Frequency of 1 Hz
NewGnssData5hz	Register for receiving messages with Frequency of 5 Hz

NewGnssData10hz	Register for receiving messages with Frequency of 10 Hz
NewGnssData20hz	Register for receiving messages with Frequency of 20 Hz

Register Response

{
"**event**": "subscribe_result",
"**id**": 1455198438,
"**object**" : "/GNSS",
"**interface**" : "br.com.arvus.TiX.GNSS",
"**member**" : "member_type",
"**result**" : 0
}

Periodic Message

```
{
 "event": "signal",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : "member_type",
 "result" :
          "{
          \“p1\”: current_vehicle_latitude,
          \“p2\”: current_vehicle_longitude,
          \“p3\”: current_vehicle_altitude,
          \“p4\”: current_implement_latitude,
          \“p5\”: current_implement_longitude,
          \“p6\”: current_implement_altitude,
          \“p7\”: gnss_epoch,
          \“p8\”: roll_angle_value,
          \“p9\”: pitch_angle_value,
          \“p10\”: heading_angle,
          \“p11\”: vehicle_speed,
          \“p12\”: is_forward,
          \“p13\”: sat_count,
          \“p14\”: differential_age_value,
          \“p15\”: gnss_mode,
          \“p16\”: gnss_status
          }"
}
```

Result Fields	Description
p1 - current_vehicle_latitude (double)	Current vehicle latitude (degrees)
p2 - current_vehicle_longitude (double)	Current vehicle longitude (degrees)
p3 - current_vehicle_altitude (double)	Current vehicle altitude (meters)
p4 - current_implement_latitude (double)	Current implement latitude (degrees)
p5 -current_implement_longitude(double)	Current implement longitude (degrees)
p6 - current_implement_altitude (double)	Current implement altitude (meters)
p7 - gnss_epoch (double)	GNSS timestamp

p8 - roll_angle_value (double)	Current vehicle roll angle (degrees)
p9 - pitch_angle_value (double)	Current vehicle pitch angle (degrees)
p10 - heading_angle (double)	Current vehicle yaw angle (degrees)
p11 - vehicle_speed (double)	Current vehicle speed (meters/seconds)
p12 - is_forward (boolean)	true if vehicle is moving forward, false if is reversing
p13 - sat_count (uint16)	number of satellites used to determine the position of vehicle
p14 – differential_age_value (double)	Differential age
p15 - gnss_mode (uint16)	Reserved
p16 - gnss_status (uint16)	Could be one of them: • 1 - No communication with GNSS board • 2 - RTK not converged • 3 - TerraStar not converged • 4 - GLIDE not converged • 5 - SBAS not converged • 6 - GNSS Syncing • 7 - GNSS Sync • 8 - TerraStar subscription Missing

3.2.2 Set Nudge

- Nudge is an X offset / Y offset (meters).
- The Nudge value is global – it will apply to all path types until it is reset to 0.0
- Nudge will default to 0.0 at startup
- After setting a nudge, the position reported by *NewGnssData* signal will be corrected using nudge value
- Changing the position more than 10 cm (about 3.94 in) during a brief time could cause control instability and affect the safety of the product. Because of this, the new value could be rejected. With the system engaged, max. nudge is 0.02m and total position offset is 15m.

Set Nudge command
{

```
"event": "call_method",
"id": 1455198438,
"object" : "/GNSS",
"interface" : "br.com.arvus.TiX.GNSS",
"member" : "SetNudge",
"params" : [
        {
        "type" : "double",
        "value" : north_offset
        },
        {
        "type" : "double",
        "value" : east_offset
        }
        ]
}
```

Param Field	Description
north_offset (double)	North offset (meters)
east_offset (double)	East Offset (meters)

Set Nudge command response

```
{
 "event": "method_result",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : "SetNudge",
 "result": "{
   \"p1\" : result
  }"
}
```

The result (uint16) could be:

- 0 - success
- 1 - failed, steering is engaged, and nudge change is too high, consider reducing the amount change of nudge

3.2.3 Get Nudge

Get Nudge Param

```
{
 "event": "call_method",
 "id": 1455198438,
```

```
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : "GetNudge"
}
```

Get Nudge Param Response

```
{
 "event": "method_result",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : "GetNudge",
 "result" : "{
          \"p1\": [\"north_offset\", \"east_offset\"]
          }"
}
```

Result Type	
p1 (double array)	north offset in meters, east offset in meters

3.2.3 NTRIP

3.2.3.1 Get Configuration

NTRIP get configuration command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Ntripd",
"interface":"br.com.arvus.Ntrip.Ntripd",
"member":"GetConfiguration"
}
```

NTRIP get configuration response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Ntripd",
  "interface" : "br.com.arvus.Ntrip.Ntripd",
  "member" : "GetConfiguration",
  "result" : "{
          \"p1\": [\"host\", \"port\",\"user\", \"password\", \"stream\"]
  }"
}
```

Result Field	Description

p1 - string array	• host - first argument - current NTRIP host • port - second argument - current NTRIP server port • user - third argument - current username • password - fourth argument - current user password • stream - fifth argument - current stream (mount point)

3.2.3.2 Set Configuration

If the NTRIP is already connected, then it will disconnect and change the configuration.

NTRIP set configuration command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Ntripd",
"interface":"br.com.arvus.Ntrip.Ntripd",
"member":"SetConfiguration",
"params": [
        {
        "type" : "string",
        "value" : "host"
        },
        {
        "type" : "string",
        "value" : "port"
        },
        {
        "type" : "string",
        "value" : "user"
        },
        {
        "type" : "string",
        "value" : "password"
        },
        {
        "type" : "string",
        "value" : "stream"
        }
]
}
```

NTRIP set configuration response

```
{
  "event": "method_result",
```

```
  "id": 1455198439,
  "object" : "/Ntripd",
  "interface" : "br.com.arvus.Ntrip.Ntripd",
  "member" : "SetConfiguration",
  "result" : "{
    \“p1\” : result
  }"
}
```

Result Field	Description
p1 - uint16	• 0 – success • 1 – error (invalid argument)

3.2.3.3 NTRIP Get Available Streams (mount points)

NTRIP Get Streams command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Ntripd",
"interface":"br.com.arvus.Ntrip.Ntripd",
"member":"GetStreams",
"params":[ {
"type": "string",
"value" : "host"
},
{
"type": "string",
"value" : "port"
}
]
}
```

NTRIP Get Streams response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Ntripd",
  "interface" : "br.com.arvus.Ntrip.Ntripd",
  "member" : "GetStreams",
  "result" : "{
    \“p1\” :[\“stream1\”, \“stream2\”,\“stream3\”, …., \“stream-n\”]
  }"
}
```

Result Field	Description
p1 - string array	Array with available streams

3.2.3.4 Connect

The Core Box will use current configuration to connect to a NTRIP server.

This method has no effect if the NTRIP is already connected (status equal 0 or 1). It will, for now, instantly return “ok” so for further information the user must wait for the “status” signal.

NTRIP Connect command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Ntripd",
"interface":"br.com.arvus.Ntrip.Ntripd",
"member":"Connect"
}
```

NTRIP Connect response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Ntripd",
  "interface" : "br.com.arvus.Ntrip.Ntripd",
  "member" : "Connect",
  "result" : "{
    \“p1\”  : \“result\”
  }"
}
```

Result Field	Description
p1 - string	• “ok” - connected • other string = error msg

3.2.3.5 Disconnect

NTRIP Disconnect command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Ntripd",
"interface":"br.com.arvus.Ntrip.Ntripd",
"member":"Disconnect"
}
```

NTRIP Disconnect response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Ntripd",
  "interface" : "br.com.arvus.Ntrip.Ntripd",
  "member" : "Disconnect",
  "result" : "{
    \“p1\”  : result
  }"
}
```

Result Field	Description
p1 - uint16	0 - disconnected

3.2.3.6 Get NTRIP version

NTRIP Get Version command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Ntripd",
"interface":"br.com.arvus.Ntrip.Ntripd",
"member":"GetVersion"
}
```

NTRIP Get Version response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Ntripd",
  "interface" : "br.com.arvus.Ntrip.Ntripd",
  "member" : "GetVersion",
  "result" : "{
    \“p1\”  : \“result\”
  }"
}
```

Result Field	Description
p1 - string	“x.y.z”

3.2.3.7 NTRIP Status Signal

This is a periodic signal (10 seconds). There are some internal errors (status = 7) which are not critical so the NTRIP will try again to connect in the hope that the user does not feel the problem.

To Register


```
{
 "event": "subscribe",
 "id": 1455198438,
 "object" : "/Ntripd",
 "interface" : "br.com.arvus.Ntrip.Ntripd",
 "member" : "NtripStatus"
}
```


Register Response


```
{
 "event": "subscribe_result",
 "id": 1455198438,
 "object" : "/Ntripd",
 "interface" : "br.com.arvus.Ntrip.Ntripd",
 "member" : "NtripStatus",
 "result" : "{}"
 }
```


Periodic Message


```
{
 "event": "signal",
 "id": 1455198438,
 "object" : "/Ntripd",
 "interface" : "br.com.arvus.Ntrip.Ntripd",
 "member" : "NtripStatus",
 "result" :
          "{
          \“p1\”: connections_state,
           \“p2\”: base_station_longitude,
            \“p3\”: base_station_latitude,
           \“p4\”: last_gga_frame_timestamp,
           \“p5\”: last_correction_timestamp,
          }"
}
```


Result field	Description

p1 - uint16	status: • 0 connected • 1 connected (is not receiving position updates constantly) • 2 disconnected • 3 disconnected wrong password/user • 4 disconnected incorrect mount point (or too far) • 5 disconnected missing gga messages • 6 disconnected not configured • 7 disconnected Internal error (no internet connection or server response, could not open serial,others)
P2 - double	Base station longitude
P3 - double	Base station latitude
P4 - int64	Last gga frame timestamp
P5 - int64	Last correction timestamp

3.2.4 Get Configuration

This section describes how to change the GNNS board model, internal serial port, and other useful configurations.

Get Configuration

```
{
 "event": "call_method",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : "GetConfiguration"
}
```

Get Configuration command response

```
{
 "event": "method_result",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : "GetConfiguration",
 "result": "{
   \“p1\” : [ serial_port, gnss_board,min_speed]
```

```
  }"
}
```

- serial_port – Core Box serial port
  - 0 - for internal
  - 1 - for external
  - 2 - for simulator
  - 3 - for internal secondary
  - 4 - for TCP
  - 5 - for USB0
  - 6 - for USB1
  - 7 - for USB2
- gnss_board
  - 1 - Reserved
  - 2 - Reserved
  - 3 - Reserved
  - 4 - Reserved
  - 5 - Reserved
  - 6 - Reserved
  - 7 - Reserved
  - 8 - Reserved
  - 9 - Simulator
  - 10 - OEM7
  - 11 - OEM7 PPP
  - 12 - OEM 7 NTRIP
  - 13 - OEM 7 RTK
  - 14 - Reserved
  - 15 - Reserved
  - 16 - Reserved
  - 17 - External RMC
  - 18 - External GGA
  - 19 - OEM 7 PPP Basic
  - 20 - Reserved
  - 21 - OEM7 dual antenna NTRIP
  - 22 - OEM7 dual antenna PPP
  - 23 - OEM7 dual antenna PPP basic
- Min_speed - min speed to Core Box consider as a valid vehicle movement (default is 1.6km/h)

3.2.5 Set Configuration

Set configuration command

```
{
 "event": "call_method",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : "SetConfiguration",
  "params": [
    {
      "type": "uint16",
      "value" : serial_port
    },
    {
      "type": "uint16",
      "value": gnss_board
    },
    {
      "type": "double",
      "value": min_speed
    }
```

Set Configuration command response

```
{
   "event": "method_result",
   "id": 1461078402579,
   "interface": "br.com.arvus.TiX.GNSS",
   "member": "SetConfiguration",
   "object": "/GNSS",
   "result": {
        "p1": 0
   }
}
```

P1 could be one of them

- 0 - sucess
- 1 - invalid serial_port
- 2 - invalid gnss_board

3.2.6 Reserved

3.2.7 Set Antenna offset

Set AntennaOffset

```
{
 "event": "call_method",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : "SetAntennaOffset",
  "params": [
    {
```

```
      "type": "double",
      "value" : heading_offset
    },
    {
      "type": "double",
      "value": pitch_offset
    }
  ]
}
```

Heading_offset: heading offset (degrees)
Pitch_offset: pitch offset (degrees)

Set AntennaOffset command response

```
{
   "event": "method_result",
   "id": 1461078402579,
   "interface": "br.com.arvus.TiX.GNSS",
   "member": "SetAntennaOffset",
   "object": "/GNSS",
   "result": {
       "p1": 0
   }
}
```

P1 could be one of them

- 0 - success
- 1 - wrong GNSS board model
- 2 – Invalid antenna position
- 3 – Heading offset out of bounds
- 4 - Pitch offset out of bounds

3.2.8 Get GNSS Information

Get GNSS Information

```
{
"event":"call_method",
"id":1455198438,
"object":"/GNSS",
"interface":"br.com.arvus.TiX.GNSS",
"member":"GetGnssInfo",
"params": [ ]
}
```

Get GNSS Information Response

```
{
        "event": "method_result",
        "id": 1455198438,
        "interface": "br.com.arvus.TiX.GNSS",
```

```
            "member": "GetGnssInfo",
            "object": "/GNSS",
            "result": "{
             "p1":"{
                        "AppVersion": string_version,
                        "Model": string_model,
                        "PSN": string_serial_number
             }"
            }"
}
```

3.2.9 Activate GNSS

Activate GNSS

```
{
   "event":"call_method",
   "id":1455198438,
   "object":"/GNSS",
   "interface":"br.com.arvus.TiX.GNSS",
   "member":"ActivateGnss",
   "params": [
            {
                     "type": "string",
                     "value" : "string_auth_code"
            }
    ]
 }
```

Activate GNSS Response

```
{
  "event": "method_result",
  "id": 1455198438,
  "interface": "br.com.arvus.TiX.GNSS",
  "member": "ActivateGnss",
  "object": "/GNSS",
  "result": "{
     "p1": int_code
  }"
}
```

P1 could be one of them

- 0 - Success - The authcode was sent. A reboot is recommended.
- 1 - Undefined error
- 2 - Error to validate authentication code regex

3.2.10 Firmware Update

3.2.10.1 List of GNSS firmware available

Use this call to get the list of GNSS firmwares available in the flash drive and */home/titanium/repository/firmware_gps/*. Core Box version: 3.35.0

List of GNSS firmware available

```
{
"event":"call_method",
"id":1455198438,
"object":"/GNSS",
"interface":"br.com.arvus.TiX.GNSS",
"member":" GetGNSSFirmwareList",
}
```

List of GNSS firmware response

```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.GNSS",
"member": " GetGNSSFirmwareList",
"object": "/GNSS",
"result": "{
    \"p1\": [“firmware_path_1”,“firmware_path_2”,..,“firmware_path_ n”]
   }"
}
```

3.2.10.2 Send firmware path to update GNSS

Update GNSS firmware

```
{
 "event": "call_method",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : " UpdateGNSSFirmware",
  "params": [
        {
          "type" : "string",
          "value": "firmware_path"
        }
   ]
```

Update GNSS firmware response

```
{
   "event": "method_result",
   "id": 1461078402579,
   "interface": "br.com.arvus.TiX.GNSS",
   "member": "UpdateGNSSFirmware",
   "object": "/GNSS",
   "result": update_started
}
```

update_started (bool) can be:

- 0 – Update started successfully
- 1 – Update failed to start

3.2.10.3 Update Progress Changed Signal

Indicates that the progress of the GNSS firmware update process has changed. The progress will vary between 0 and 100%.

Subscribe to Update Progress Changed Signal

```
{
   "event":"subscribe",
   "id":1455198438,
   "interface":"br.com.arvus.TiX.GNSS",
   "member":"GNSSUpdateStatus",
   "object":"/GNSS"
}
```

Subscribe Reply

```
{
   "event":"subscribe_result",
   "id":1455198438,
```

```
  "interface":"br.com.arvus.TiX.GNSS",
  "member":"GNSSUpdateStatus",
  "object":"/GNSS",
  "result":0
}
```


Signal Update Status Message


```
{
  "event":"signal",
  "id":1470251948710,
  "interface":"br.com.arvus.TiX.GNSS",
  "member":" GNSSUpdateStatus",
  "object":"/GNSS",
  "result":"{
    \"p1\":status,
    \"p2\": progress_percentage
  }"
}
```


Result Field	Description
Status (int)	• 0 – Ok. Updated with success • 1 – Starting update • 2 – Transferring firmware • 3 – Writing firmware • 4 – Restarting GNSS • 5 – Communication timeout. No response received from GNSS module • 6 – Invalid header sync. Header sync bytes not found in GNSS response • 7 – Invalid CRC in GNSS response • 8 – Invalid header size in GNSS response • 9 – Command error received from GNSS module • 10 – Invalid firmware file. Unable to properly process it • 11 – Soft load reset failed • 12 – Unknown log state during soft load command • 13 – Soft load commit failed • 14 – Update canceled • 15 – Serial port error, probably not able to be opened
progress_percentage (int)	Update progress, with range between 0 and 100 (0% to 100%)

3.2.11 Get TerraStar Information

Use this call to check the TerraStar subscription and information needed to get the auth codes to use the correction. Core Box version: 2.10.0

TerraStar info

{
"event":"call_method",
"id":1455198438,
"object":"/GNSS",
"interface":"br.com.arvus.TiX.GNSS",
"member":"GetGnssTerrastarInfo",
}

TerraStar info Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.GNSS",
"member": "GetGnssTerrastarInfo",
"object": "/GNSS",
"result": "{\"p1\":\"{\\\"Board Version\\\":\\\"\\\",\\\"Freq\\\":\\\"\\\",\\\"PAC\\\":\\\"\\\",\\\"TS Signal\\\":\\\"Disabled\\\",\\\"TS Sub\\\":\\\"\\\",\\\"TS Sync\\\":\\\"No TERRASTAR (LBand)\\\"}\"}"
}

Result Field	Description
Board Version	The GNSS board version (can be asked for the subscription)
Freq	
PAC	
TS signal	
TS sub	
Ts Sync	

3.2.12 Invert motion direction

When using a single antenna solution, the system will consider the first detected movement direction as forward. If this is incorrect, you should call this method to correct the movement direction. The heading can be noisy some seconds after changing it. Do not change it when the steering system is engaged. Core Box version: 2.10.0

Invert motion direction

{
"event":"call_method",
"id":1455198438,
"object":"/GNSS",
"interface":"br.com.arvus.TiX.GNSS",
"member":"InvertMotionDirection"
}

Invert motion direction Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.GNSS",
"member": "InvertMotionDirection",
"object": "/GNSS",
"result" : "{
\“p1\” : result
}"
}

Result Field	Description
p1 - uint16	• 0 - success

3.2.13 Get SPAN Status

This message is continuously sent from the Core Box’s ECU at 1 Hz.

To Register

{
"event": "subscribe",
"id": 1455198438,
"object" : "/GNSS",
"interface" : "br.com.arvus.TiX.GNSS",
"member" : " NewSPANData1hz"
}

Register Response
{ "**event**": "subscribe_result", "**id**": 1455198438,

```
"object" : "/GNSS",
"interface" : "br.com.arvus.TiX.GNSS",
"member" : " NewSPANData1hz",
"result" : 0
}
```

Periodic Message

```
{
"event": "signal",
"id": 1455198438,
"object" : "/GNSS",
"interface" : "br.com.arvus.TiX.GNSS",
"member" : " NewSPANData1hz",
"result" :
        "{
\“p1\”: current_vehicle_latitude,
        \“p2\”: current_vehicle_longitude       }"
}
```

Result Fields	Description
p1 - rpy_std_magnitude (double)	Current magnitude of RPY standard deviation
p2 - span_status (int)	Current SPAN status

3.2.14 Get GNSS Feature Enabled

This method is used to know if a feature form SMART7 is enabled.

Get GNSS Feature Enabled

```
{
"event": "call_method",
"id": 1455198438,
"object" : "/GNSS",
"interface" : "br.com.arvus.TiX.GNSS",
"member" : "GetFeatureEnabled",
"params": [
    {
        "type": "uint32",
```

```
            "value": feature_id
        }
 ]
}
```

- feature_id
  - 1 - Omnistar
  - 2 - TerraStar-C PRO (PPP)
  - 3 - TerraStar-L (PPP Basic)
  - 4 - SBAS
  - 5 - NTRIP
  - 6 - RTK
  - 7 - ALIGN
  - 8 - Beidou
  - 9 - No glide
  - 10 – SPAN
  - 11 - Silent Mode

Get GNSS Feature Enabled command response

```
{
 "event": "method_result",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : " GetFeatureEnabled ",
 "result": "{
    \"p1\": feature_enabled
  }"
}
```

- P1 (boolean) - Feature enabled or not. Returns false if antenna model is not SMART7

3.2.15 Set GNSS Feature Enabled

This method to set a feature form SMART7 enabled.

Set GNSS Feature Enabled

```
{
 "event": "call_method",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : "SetFeatureEnabled",
  "params": [
    {
      "type": "uint32",
      "value" : feature_id
    },
```

```
    {
      "type": "boolean",
      "value": feature_enabled
    }
 ]
}
```

Set GNSS Feature Enabled command response

```
{
   "event": "method_result",
   "id": 1461078402579,
   "interface": "br.com.arvus.TiX.GNSS",
   "member": "SetFeatureEnabled",
   "object": "/GNSS",
   "result": {
       "p1": 0
   }
}
```

P1 could be one of them

- 0 - sucess
- 1 – Error: antenna is not SMART7 or feature combination is not possible.

3.2.16 Get ALIGN Status

This message is continuously sent from the Core Box's ECU at 1 Hz.

To Register

```
{
 "event": "subscribe",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : " ALIGNData1hz "
}
```

Register Response

```
{
 "event": "subscribe_result",
 "id": 1455198438,
 "object" : "/GNSS",
 "interface" : "br.com.arvus.TiX.GNSS",
 "member" : " ALIGNData1hz ",
 "result" : 0
```

}

Periodic Message

{
"**event**": "signal",
"**id**": 1455198438,
"**object**" : "/GNSS",
"**interface**" : "br.com.arvus.TiX.GNSS",
"**member**" : " ALIGNData1hz ",
"**result**" :
"{
\“p1\”: current_vehicle_latitude,
\“p2\”: current_vehicle_longitude }"
}

Result Fields	Description
p1 - orientation_std_magnitude (double)	Current magnitude of orientation standard deviation
p2 - align_status (int)	Current ALIGN status

[0] - Solution computed
[1] - Insufficient observations
[2] - No convergence
[3] - Singularity at parameters matrix
[4] - Covariance trace exceeds maximum (trace > 1000 m)
[5] - Test distance exceeded (maximum of 3 rejections if distance > 10 km)
[6] - Not yet converged from cold start
[7] - Height or velocity limits exceeded (in accordance with export licensing restrictions)
[8] - Variance exceeds limits
[9] - Residuals are too large
[10] - Delta position is too large
[11] - Negative variance
[12] - Reserved
[13] - Large residuals make position unreliable (Integrity warning)
[14] - Reserved
[15] - Reserved
[16] - Reserved

[17] - Reserved
[18] - When a FIX POSITION command is entered, the receiver computes its own position and determines if the fixed position is valid
[19] - The fixed position, entered using the FIX POSITION command, is not valid
[20] - Position type is unauthorized
[21] - Antenna warning
[22] - The selected logging rate is not supported for this solution type

3.3 Vehicle

A vehicle is unique in the system with its own steering control. When you set a vehicle as active, all steering changes will only apply to this vehicle. It also has the following properties, with all distances measured in meters:

- **id:** uniquely identifies the vehicle (automatically generated by the Core Box when creating a new vehicle) (string)
- **name:** vehicle name (string)
- **steerType:** -1 unknown, 0 for front steer or 1 for rear steer (int)
- **wheelbase:** distance from rear to front axle (string)
- **antennaAxisDistance:** relative position from the antenna (string)
  - **Front steer:** relative position to rear axle, positive if antenna in front of rear axle.
  - **Rear steer:** relative position to front axle, positive value if antenna is in front of front axle.
- **antennaHeight:** antenna height (meters) (string)
- **antennaOffset:** antenna lateral offset (meters), positive to right (string)

3.3.1 To list all available vehicles

Vehicle index command

```
{
   "event": "call_method",
   "id": 1455198438,
   "interface": "br.com.arvus.TiX.Vehicle",
   "member": "Index",
   "object": "/Vehicle"
}
```

Vehicle index response

```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.Vehicle",
"member": "Index",
```

"object": "/Vehicle",
"result":
"{\"p1\":\"[{\\\"Vehicle\\\":{\\\"antennaAxisDistance\\\":\\\"2.56\\\",\\\"antennaHeight\\\":\\\"2.73\\\",\\\"antennaOffset\\\":\\\"-0.15\\\",\\\"id\\\":\\\"coZy9B55pNyhLwH\\\",\\\"name\\\":\\\"novo_name\\\",\\\"steerType\\\":\\\"1\\\",\\\"vehicleAxisToPivot\\\":\\\"3.47\\\",\\\"vehicleType\\\":\\\"3\\\",\\\"wheelbase\\\":\\\"2.73\\\"}},{\\\"Vehicle\\\":{\\\"antennaAxisDistance\\\":\\\"2.56\\\",\\\"antennaHeight\\\":\\\"2.73\\\",\\\"antennaOffset\\\":\\\"-0.15\\\",\\\"id\\\":\\\"3Cakw91Um3jXNLd\\\",\\\"name\\\":\\\"teste\\\",\\\"steerType\\\":\\\"1\\\",\\\"vehicleAxisToPivot\\\":\\\"3.47\\\",\\\"vehicleType\\\":\\\"1\\\",\\\"wheelbase\\\":\\\"2.73\\\"}},{\\\"Vehicle\\\":{\\\"antennaAxisDistance\\\":\\\"2.56\\\",\\\"antennaHeight\\\":\\\"2.73\\\",\\\"antennaOffset\\\":\\\"-0.15\\\",\\\"id\\\":\\\"cmQNQw1OkGAB7s8\\\",\\\"name\\\":\\\"excavator\\\",\\\"steerType\\\":\\\"1\\\",\\\"vehicleAxisToPivot\\\":\\\"3.47\\\",\\\"vehicleType\\\":\\\"2\\\"}}]\"}"
}

3.3.2 Create vehicle

Create vehicle command

```
{
    "event": "call_method",
    "id": 1455198438,
    "interface": "br.com.arvus.TiX.Vehicle",
    "member": "Create",
    "object": "/Vehicle",
    "params": [
        {
            "type": "string",
            "value":
"{\"Vehicle\":{\"name\":\"new_vehicle_name\",\"vehicleType\":\"1\",\"vehicleAxisToPivot\":\"3.47\",\"steerType\":\"1\",\"wheelbase\":\"2.73\",\"antennaAxisDistance\":\"2.56\",\"antennaHeight\":\"2.73\",\"antennaOffset\":\"-0.15\"}}"
        }
    ]
}
```

Create Vehicle response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.Vehicle",
"member": "Create",
"object": "/Vehicle",
"result":
"{\"p1\":\"{\\\"Vehicle\\\":{\\\"antennaAxisDistance\\\":\\\"2.56\\\",\\\"antennaHeight\\\":\\\"2.73\\\",\\\"antennaOffset\\\":\\\"-0.15\\\",\\\"id\\\":\\\"KiqQ6RNBg8KgyTM\\\",\\\"name\\\":\\\"asdasdsad_vehicle_name\\\",\\\"steerType\\\":\\\"1\\\",\\\"vehicleAxisToPivot\\\":\\\"3.47\\\",\\\"vehicleType\\\":\\\"1\\\",\\\"wheelbase\\\":\\\"2.73\\\"}}\"}"

}

The p1 (string) field could be one of them

- String with the vehicle parameters - Success
- "p1": “{ \“error\” : \“msg\” }”

3.3.3 Update vehicle

Vehicle update command

{
"event": "call_method",
"id": 1455198438,
"interface": "br.com.arvus.TiX.Vehicle",
"member": "Update",
"object": "/Vehicle",
"params": [
{
"type": "string",
"value":
"{\"Vehicle\":{\"id\":\"J0wNC6x0u3vghIE\",\"steerType\":\"1\",\"wheelbase\":\"2.83\",\"vehicleAxisToPivot\":\"3.57\",\"antennaAxisDistance\":\"1.56\",\"antennaHeight\":\"1.73\",\"antennaOffset\":\"-1.15\"}}"
}
]
}

Vehicle Update Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.Vehicle",
"member": "Update",
"object": "/Vehicle",
"result":
"{\"p1\":\"{\\\"Vehicle\\\":{\\\"antennaAxisDistance\\\":\\\"1.56\\\",\\\"antennaHeight\\\":\\\"1.73\\\",\\\"antennaOffset\\\":\\\"-1.15\\\",\\\"id\\\":\\\"J0wNC6x0u3vghIE\\\",\\\"name\\\":\\\"new_vehicle_name0\\\",\\\"steerType\\\":\\\"1\\\",\\\"vehicleAxisToPivot\\\":\\\"3.57\\\",\\\"vehicleType\\\":\\\"0\\\",\\\"wheelbase\\\":\\\"2.83\\\"}}\"}"
}

The p1 (string) field could be one of them

- String with the vehicle parameters - Success
- Empty ("p1": "") - Error

3.3.4 Delete vehicle

To delete a vehicle command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Vehicle",
"interface":"br.com.arvus.TiX.Vehicle",
"member":"Delete",
"params": [
        {
          "type" : "string",
          "value": "vehicle_id_to_delete"
        }
    ]
}
```

To delete a vehicle response

```
{
    "event": "method_result",
    "id": 1461078402579,
    "interface": "br.com.arvus.TiX.Vehicle",
    "member": "Delete",
    "object": "/Vehicle",
    "result": “{
        "p1": 0
    }"
}
```

The “p1” (uint16) field could be one of them

- 0 - success
- 1 - error, does not found a vehicle with id

3.3.5 Show vehicle

To show a vehicle

```
{
"event":"call_method",
"id":1455198438,
"object":"/Vehicle",
"interface":"br.com.arvus.TiX.Vehicle",
"member":"Show",
"params": [
          {
            "type" : "string",
            "value": "vehicle_id_to_show"
        }
    ]
}
```

To show a vehicle response

```
{
"event": "method_result",
"id": 1461078638689,
"interface": "br.com.arvus.TiX.Vehicle",
"member": "Show",
"object": "/Vehicle",
"result": “{
          \"p1\": "{\\\"Vehicle\\\":{
          \\\"antennaAxisDistance\\\":”antennaAxisDistance”,
          \\\"antennaHeight\\\":”antennaHeight”,
          \\\"antennaOffset\\\":”antennaOffset”,
          \\\"id\\\":\\\"id\\\",
          \\\"name\\\":\\\"name\\\",
          \\\"steerType\\\":steerType,
          \\\"wheelbase\\\":”wheelbase”}
  }"
 }”
}
```

If result is an empty object **“”**, the Core Box cannot find a vehicle with provided id

3.3.6 Set active vehicle

This command will set the vehicle with the specified "id" as active, and its steering control parameters will be used. The command does not update the vehicle, and if any parameter is incorrect, an error will be returned.

Command to set active

```
{
"event":"call_method",
"id":1455198438,
"object":"/Vehicle",
"interface":"br.com.arvus.TiX.Vehicle",
"member":"SetActive",
"params": [
          {
            "type" : "string",
           "value": "vehicle_id_to_set_as_active"
        }
     ]
 }
```

Command set vehicle response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Vehicle",
  "interface" : "br.com.arvus.TiX.Vehicle",
  "member" : "SetActive",
  "result" : “{
   \“p1\” : result_code
 }”
}
```

result_code could be one of them:

- 0 - success setting the current vehicle
- 1 - error, does not found a vehicle with id
- 2 - error, steering is engaged

3.3.7 Get active vehicle

Command to get active vehicle

```
{
"event":"call_method",
"id":1455198438,
"object":"/Vehicle",
"interface":"br.com.arvus.TiX.Vehicle",
"member":"GetActiveVehicle"
 }
```

Command get active vehicle response

```
{
"event": "method_result",
"id": 1461078638689,
"interface": "br.com.arvus.TiX.Vehicle",
"member": "GetActiveVehicle",
"object": "/Vehicle",
```

```
"result": “{
          \"p1\": "{\\\"Vehicle\\\":{
          \\\"antennaAxisDistance\\\":\”antennaAxisDistance\”,
          \\\"antennaHeight\\\":\”antennaHeight\”,
          \\\"antennaOffset\\\":\”antennaOffset\”,
          \\\"id\\\":\\\"id\\\",
          \\\"name\\\":\\\"name\\\",
          \\\"steerType\\\":\”steerType\”,
          \\\"wheelbase\\\":\”wheelbase\”}
  }"
 }”
}
```

result_code could be one of them:

- 0 - success setting the current vehicle
- 1 - error, does not found a vehicle with id
- 2 - error, steering is engaged

3.4 Steering

A Steering configuration belongs to a vehicle. If you change any steering params, it is only applied for the current vehicle.

3.4.1 Steering Status Signal

This message is continuously sent from the ECU.

To receive this message, the device must register to listen to the signal ‘SteeringStatus’.

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "member_type"
}
```

member_type	Description
SteeringStatus1hz	Register for receiving messages with Frequency of 1hz
SteeringStatus5hz	Register for receiving messages with Frequency of 5hz

SteeringStatus10hz	Register for receiving messages with Frequency of 10hz

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "SteeringStatus",
  “result” : 0
 }
```

Periodic Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "SteeringStatus",
  "result" :”{
    \“p1\”: status_code,
    \“p2\”: current_token,
    \“p3\”: current_vehicle_id,
  }”
}
```

Result Fields	Description
Status_code (uint16)	• 0 - ready to engage (disengaged) • 1 - engaged - this state indicates that the steering is operating • 2 - blocked,recording a path • 3 - blocked, cannot steer now (path error or angle too high) • 4 - no communication with ECU • 5 - blocked, unknown motive • 6 - calibration mode

	• 7 - blocked, poor signal precision (see param 23 section 3.4.4) • 8 - auto steering disabled (feature) • 9 - without wheel sensor • 10 - without external power • 11 - invalid line • 12 - Heading solution has not converged yet (OEM-617 dual antenna only)
current_token (string)	Current Guide Token
Current_vehicle_id (string)	Current Vehicle Id

3.4.2 Engage Auto Steering

To Engage

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "Engage",
  "params":[
          {
          "type": "string",
          "value" : "token"
          },
          {
          "type": "string",
          "value" : "vehicle_id"
          }
          ]
}
```

Engage Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "Engage",
  "result": "{
     \“p1\”: status_code
  }"
}
```


The result status_code (uint16) will be one of the following:

- 0 - engaged successfully
- 1 - not engaged, wrong token
- 2 - not engaged, too far from line
- 3 - not engaged, angle too high
- 4 - not engaged, no communication with ECU
- 5 - not engaged, GNSS not ready
- 6 - not engaged, heading too noisy
- 7 - not engaged, poor signal precision (set/get steering param 23)
- 8 - not engaged, vehicle not selected
- 9 - not engaged, wrong vehicle
- 10 - not engaged, invalid line
- 11 - not engaged, feature not enabled
- 12 - not engaged, above max speed
- 13 - not engaged, under min speed

3.4.3 Disengage Auto Steering

To Disengage


```
{
 "event": "call_method",
 "id": 1455198439,
 "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "Disengage"
}
```


Disengage Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
 "member" : "Disengage",
 "result": "{ \“p1\” : 0 }"
}
```

3.4.4 Steering SetParams

All set/get methods will return/set the values of the current vehicle.

To Set a Param

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "SetParam",
  "params": [
    {
      "type": "uint16",
      "value" : param_code
    },
    {
      "type": "double",
      "value": param_value
    }
  ]
}
```

Set Param Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "SetParam",
  "result": "{
    \"p1\": status
   }"
}
```

The status (uint16) field could be one of them

- 0 - success
- 1 - param not set, invalid parameter

The field “paramCode” could be one of them:

paramCode (uint16)	description	Supported values	Default	Type	Can be set
0	aggressiveness	0-200	100	double	y
1	sensibility	0-200	100	double	y
2	overshoot	0-200	0	double	y
3	actuator_kp	0-200	30	double	y
4*	max_wheel_position_low_speed	0-45 (degrees)	40	double	y
5*	max_wheel_position_high_speed	0-45 (degrees)	40	double	y
6*	max_wheel_delta	1-45 (degrees)	20	double	y
7*	line_ki	5-50	10	double	y
8*	int_threshold	0 - 500	15	double	y
9*	drift_filter	0 - 1000	250	double	y
10	line_error_to_engage	0 - 90	1	double	y
11	angle_error_to_engage	0 - 90	30	double	y
12	actuator_dead_band_left	0-1000	280	double	y
13	actuator_dead_band_right	0-1000	280	double	y
14	gyro_x_bias	-	-	double	y
15	gyro_y_bias	-	-	double	y
16	gyro_z_bias	-	-	double	y
17	roll_offset	-	-	double	y
18	pitch_offset	-	-	double	y

19	roll_angle	-	-	double	n
20	pitch_angle	-	-	double	n
21	heading_angle	-	-	double	n
22	steering_type	1 - Hydraulic Steering. 2 – Track Controller 3 - Steer Direct 4 - Hydraulic without wheel sensor 7 – PVED-CLS 11 – PVED-CL 13 – Track controller Next-Gen 17 – Fort VSC (Must restart application after setting it)	1	double	y
23	min_gnss_quality	0 - Autonomous 1 - Precision (Glide) 2 - High precision (SBAS, TerraStar) 3 - RTK	1	double	y
24	Encoder x0	-	0	double	y
25	Encoder x1	-	1	double	y
26	Encoder x2	-	0	double	y
27	Board position	0 - horizontal front 1 - vertical back 2 - vertical front 3 - vertical right 4 - vertical left 5 - horizontal back	0	double	y
28	IMU accel gain	0-100	10	double	yes
29	Encoder raw data	-	-	double	no
30	Wheel position (degrees)	-	-	double	no
31	Motor ppr (MDU)	-	-	double	yes
32	Speed gain scaler	10 - 200	100	double	yes

33	Max operation speed	0-100 (km/h) - does not accept float values	50	double	yes
34	Min operation speed	0-100 (km/h)) - does not accept float values	0	double	yes
35	Override gain	0-1000	1000	double	yes
36	Align roll offset	-90 - 90 (degrees)	-	double	no
37	Core Box's ECU logs	1 - IMU log	0 (disabled)	double	yes
38	Performance log frequency**	1, 2, 4 ,5,10 or 20 (Hz)	1 Hz	double	Yes
39	Steering session id (used for logs)	-	-	double	no
40	IMU type (deprecated)	• 0 - Default IMU • 1 - ALIGN IMU	0	double	Yes
41	Override Read	-	-	double	no
48	Analog valve neutral output	0-100 (PWM %)	50.0	double	yes
49	Analog valve low output	0-100 (PWM %)	25.0	double	yes
50	Analog valve high output	0-100 (PWM %)	75.0	double	yes
51	Kp vel	-	-	double	yes
52	Ki vel	-	-	double	yes
53	Kp pos	-	-	double	yes
54	Speed reference percentage	0-100	-	double	yes
55	Curve aggressiveness	0-200	100	double	yes
56	INS type	0 – Kalman 1 – ALIGN	0	double	yes

		2 – Can (private) 3 – Kalman + ALIGN 4 – Adaptative Kalman (private) 5 - Span			

* These parameters are changed for advanced setup only, normally you do not have to work with them

** The parameter will not be saved, so if you reboot the Core Box you must set it again if you need

3.4.5 Steering GetParams

The reference for the fields param_code and param_value can be found in the topic 3.4.3 Steering Set Params.

To Get a Param

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "GetParam",
  "params": [
    {
      "type": "uint16",
      "value" : param_code
    }
  ]
}
```

Get a Param Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "GetParam",
  "result" : "{   \"p1\" : [param_code, param_value]   }"
}
```

If the param is not found, [-1,0] is returned.

3.4.6 Steering Interruption Signal

This message is sent from the Core Box's ECU if any critical situation occurs (like remote switch pressed, end of guidance …)

To receive this message, the device must register to listen to the signal 'SteeringInterruption'.

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "SteeringInterruption"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "SteeringInterruption",
  "result" : 0
 }
```

Periodic Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "SteeringInterruption",
  "result" : "{
    \"p1\": status_code
  }"
}
```

Result Fields	Description

status_code (uint16)	• 1 - wrong token (token sent by display does not match with Core Box current info) • 2 - too far from line • 3 - angle too high • 4 - without communication with ECU • 5 - end of guidance • 6 - heading too noisy (rough terrain alarm) • 7 - poor signal precision (set/get steering param 24) • 8 - external button pressed, system is disengaged • 9 - external button pressed, decide to engage • 10 - User Steering Wheel Disengage (user turned the steering wheel) • 11 - GNSS dropout (lost RTK/TerraStar/SBAS) • 12 - Above maximum speed • 13 - under minimum speed • 14 - Loss of communication • 15 - Lost ALIGN precision • 16 – PVES Fault signal

If you receive this message with status "2" and the steering system is disengaged, the display will determine whether it is safe to engage the system. If so, it will send the "Engage" command to the Core Box. If the external button is pressed while the steering is engaged, the system will automatically disengage and send this message with code "3".

3.4.7 Steering Calibration

There are 4 steps (some tests have subroutines) to calibrate the steering system

- Step 1 - set board position
- Step 2 - estimate board offset
- Step 3 - calibrate gyro bias
- Step 4 - calibration wheel encoder

3.4.7.1 Emergency stop

To stop all hydraulics and calibration logic, you can send the command ‘Calibration’ with code 18.

Emergency stop

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
```

```
  "member": "Calibration",
  "params": [
    {
      "type": "uint16",
      "value" : 18
    }
  ]
}
```

3.4.7.2 Signal Steering Calibration

This signal will be sent after a successful step calibration. We do not save these parameters automatically, so you must set them to confirm that you want to use these values (section 3.4.4).

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "SteeringCalibration"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "SteeringCalibration",
  “result” : 0
 }
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "SteeringCalibration",
```

"**result**" : "{
\“p1\”: calibration_step_code,
\“p2\”: calibration_step_response_double_array
}"
}

Calibration_step_code:

- -1 - Error
- 1 - Offset calibration position 1
- 2 - Offset calibration position 2 (must perform the Calibration_step_code 1 first)
- 3 - Gyro bias calibration
- 4 - Encoder Center
- 5 - Encoder Left
- 6 - Encoder Right
- 7 - Actuator Test
- 8 - Start GNSS Calibration
- 9 - Pause Encoder Calibration
- 10 - Destroy All Data Encoder Calibration
- 9 - Encoder Coefs
- 10 - Encoder Feedback
- 12 - Dead Band
- 13 - Actuator Controller Test
- 14 - Prepare For Calibration
- 15 - Restore Saved Data
- 16 - Actuator Gain Test
- 17 - Use Encoder Pre Center
- 18 - Emergency Stop
- 19 - Apply PWM
- 20 - Send Wheel to Center
- 21 - Send Wheel to Right
- 22 - Send Wheel ToLeft
- 23 - Start Counter Clock Wise Motor Calibration

Calibration_step_response_double_array, depends on the calibration_step_code (double array always)

- For calibration_step_code = 1 it will be an empty array
- For calibration_step_code = 2 it will be [roll_offset, pitch_offset] (degrees)
- For calibration_step_code = 3 it will be [gyro_x_bias, gyro_y_bias, gyro_z_bias] (degrees/seconds)
- For calibration_step_code = 4, 5 and 6 it will be [encoder_value]
- For calibration_step_code = 7 it will be [sensor_installation, actuator_installation]
    - For sensor_installation: 0 to inform that the sensor is in the default installation, 1 for inverse
    - For actuator_installation: 0 to inform that the actuator is in the default installation, 1 for inverse

- For calibration_step_code = 8 it will be an empty array. Just inform that we finished the clockwise test
- For calibration_step_code = 9 it will be [x0, x1, x2] it will be the calibration params
- For calibration_step_code =10 it will be [path_number, path_percent] it will be feedback of the current test.
- For calibration_step_code = -1 it means that we have an error during calibration, the calibration_step_response_double_array will be an array with error code
    - 0 - without ECU communication
    - 1 - wheel control problem

3.4.7.3 Step 1 - board position

For this step you only have to set the correct board position (section 3.4.4 parameter 27).

3.4.7.4 Step 2 - board offset

For this calibration, the vehicle must be in flat ground and totally stopped. You must perform two terrain reads at same location. Each test takes 20s and when finished the signal “Calibration Status will be sent”.

Warning: You must first perform the calibration with code 1, followed by the calibration with code 2. These values are not saved automatically; you need to issue the set command to save them (param code 17 for roll and param code 18 for pitch).

To start

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "Calibration",
  "params": [
    {
      "type": "uint16",
      "value" : calibration_step_code
    }
  ]
}
```

Calibration_step_code

- 1 - offset calibration position 1
- 2 - offset calibration position 2 (must be performed after the calibration step 1)

To start response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "Calibration",
  "result" :"{
    \"p1\": 0
  }"
}
```


- 0 - success
- 1 - error - another calibration in progress
- 2 - timeout
- 3 - must perform calibration step 1 first
- 4 - communication error

3.4.7.5 Step 3 - Gyro bias

For this calibration, the vehicle must be stationary. The test will take 20 seconds to complete. You need to send a message to instruct the Core Box to save the calibration data (set params).

To start


```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "Calibration",
  "params": [
    {
      "type": "uint16",
      "value" : calibration_step_code
    }
  ]
}
```


Calibration_step_code

- 3 - Gyro bias calibration

3.4.7.6 Step 4 - Wheel sensor calibration

For this calibration, you must first configure a pre-setup with the maximum wheel positions and perform an actuator test. Once this setup is complete, the system will

automatically execute several curves to calibrate the steering system. During this process, the "Calibration Status" signal will be sent with data to keep the user updated on the process status. At the end, an array (Double) with the sensor configuration will be sent.

3.4.7.6.1 Setup

Please provide us with three-wheel positions and send us the corresponding data. Once we receive this, we will test the actuator and let you know if everything is functioning correctly.

To report a wheel position, ask the user to move the wheel to the correct position and then send us the corresponding command.

Before starting any test of wheel sensor calibration (section 3.4.6.4), you must send the command “Calibration” with code 14 to prepare all our internal variables to a calibration process. When you finish the test (or cancel it) we recommend you send the command with code 15 to reset these variables to valid values.

To start

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "Calibration",
  "params": [
    {
      "type": "uint16",
      "value" : calibration_step_code
    }
  ]
}
```

Calibration_step_code

- 4 - Save center position
- 5 - Save max wheel left position
- 6 - Save max wheel right position
- 14 - Prepare system variables for calibration
- 15 - Reset system variables

To start response

```
{
        "event": "method_result",
        "id": 1455198439,
        "interface": "br.com.arvus.TiX.Steering",
        "member": "Calibration",
```

"object": "/Steering",
"result": "{\"p1\":7}"
}

p1:

- -1 - In the case you are using TrackController or TrackController next gen and try to set the right wheel position before the left one. It will not save the position.
- 7 - position stored

Once the three wheel positions are set, initiate a quick actuator test (the wheel will move during this test). When the test is complete, you will receive the SteeringCalibration signal with calibration_step_code = 7.

To start

{
"**event**": "call_method",
"**id**": 1455198439,
"**object**": "/Steering",
"**interface**": "br.com.arvus.TiX.Steering",
"**member**": "Calibration",
"**params**": [
{
"**type**": "uint16",
"**value**" : calibration_step_code
}
]
}

Calibration_step_code

- 7 - start actuator test

3.4.7.6.2 Automatic test

This test performs some curves in clockwise and counterclockwise. You can start/resume, stop or destroy (delete all data) of a test. First, you must perform the test in a clockwise direction. During the test, Core Box will send messages with test progress feedback.

When the test is finished, the Core Box will send SteeringCalibration signal with code 9 (and the calibration coefficients as data, being X0, X1 and X2 for the hydraulic pilot and just X0 for the TrackController type), you have to send a message to inform the Core Box to save these parameters (set params variables 24, 25, 26).

To start

{
"**event**": "call_method",
"**id**": 1455198439,
"**object**": "/Steering",

```
  "interface": "br.com.arvus.TiX.Steering",
  "member": "Calibration",
  "params": [
    {
      "type": "uint16",
      "value" : calibration_step_code
    }
  ]
}
```


Calibration_step_code (these commands does not have a response)

- 8 - start/resume clockwise test
- 9 - pause a test
- 10 - destroy all data from the automatically test
- 23 – start/resume counter-clockwise test (only TrackController and TrackController-next Gen)

To start


```
{
"event": "method_result",
"id": 1455198439,
"interface": "br.com.arvus.TiX.Steering",
"member": "Calibration",
"object": "/Steering",
"result": "{\"p1\":0}"
}
```


- P1 = 0 : success
- P1 = -1: In the case of TrackController or TrackController-next gen, it is required that the clock-wise calibration has been performed before

3.4.7.6.3 Move to preset positions

These tests can only be used after you set the center, left, right positions and actuator logic.

You have three codes to move the wheels to a specific position:

- 20: will move the wheels to center position
- 22: will move the wheels to left position
- 21: will move the wheels to right position

To start


```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "Calibration",
```

```
  "params": [
    {
      "type": "uint16",
      "value" : calibration_step_code
    }
  ]
}
```

Calibration_step_code (these commands does not have a response)

- 20 -move wheel to center position)
- 21 - move the wheels to right position
- 22 - move the wheel to left position

3.4.7.6.4 Apply PWM

This command will apply (during 2s) a PWM signal. This command accepts values between -1000 and 1000 (1000 = 12v). If you send a positive value, the wheels should move to the right. If you apply a value < than the valve dead band the wheel will not move.

To start

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "ActuatorPWMTest",
  "params": [
    {
      "type": "int16",
      "value" : pwm_ref
    }
  ]
}
```

3.4.7.7 Valve Calibration

3.4.7.7.1 Signal Actuator Calibration

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "ActuatorCalibration"
```

}

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "ActuatorCalibration",
  “result” : 0
 }
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "ActuatorCalibration",
  "result" : "{
    \“p1\”: calibration_step_code,
    \“p2\”: calibration_step_response_double_array
  }"
}
```

Calibration_step_code:

- 1 - dead band right
- 2 - dead band left
- 3 - controller test
- 4 - dead band test feedback
- 5 - actuator gain test feedback
- 6 - actuator gain test result
- 8 - dead band end
- 9 - current dead band

Calibration_step_response_int_array, depends on the calibration_step_code (int array always)

- For calibration_step_code = 1 it will be [dead_band_right]
- For calibration_step_code = 2 it will be [dead_band_left]
- For calibration_step_code = 3 it will be [postion_ref,current_position]
- For calibration_step_code = 4 (feedback of calibration codes 1 and 2) it will be [direction,current_dead_band], direction 1 = right, 2 = left

- direction : 1 to right, 2 to left
- Current_dead_band current pwm applied to the valve

- For calibration_step_code = 8 it will be the final dead_band value [for the actuators that do not use the left-right logic]
- For calibration_step_code = 9 it will be the current dead_band [for the actuators that do not use the left-right logic]

3.4.7.7.2 Dead Band Calibration

This calibration automatically applies a ramp to the valve to determine the dead band value. It is an automatic test that first tests to the right and then to the left. You will receive status feedback through the previous signal (3.4.6.6.1) with code = 4. When the test is complete, the calibration to the right will send you a signal with code = 1 and the value found and will automatically start the test to the left.

We do not save these parameters automatically, so you must set them to confirm that you want to use these values (section 3.4.4)

To start

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "ActuatorDeadBandCalibration",
  "params": [
    {
      "type": "uint16",
      "value" : calibration_step_code
    }
  ]
}
```

Calibration_step_code

- 1 - will start the test (if you resend this message with this same code will stop the dead band calibration

To start response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "ActuatorDeadBandCalibration",
  "result" :" {
    "p1": 0
  }"
}
```

P1 will be:
- 0 : success, calibration started
- -1: failure, generic error
- -2: failure, encoder calibration positions not set

3.4.7.7.3 Controller gain test

This calibration automatically estimates the best valve gain for the system. It will move the wheels during the calibration.

We will send feedback messages (signal ActuatorCalibration with code 5) and the result (signal ActuatorCalibration with code 6) during the process.

To start

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": "ActuatorGainTest"
}
```

To start response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "ActuatorGainTest",
  "result" :” {
    “p1”: 0
  }”
}
```

P1 will be:
- 0 : success, calibration started
- -1: failure, generic error
- -2: failure, encoder calibration positions not setted

3.4.7.7.4 Controller test

The purpose of this calibration is to test the local controller of the valve and determine the optimal configuration of "actuator_kp" for the installed hydraulic system. You should send

this message periodically. If we do not receive it for more than 2 seconds, the test will be stopped.

We will send the Feedback signal with code = 3 periodically with the current wheel position and reference.

To start

{
"**event**": "call_method",
"**id**": 1455198439,
"**object**": "/Steering",
"**interface**": "br.com.arvus.TiX.Steering",
"**member**": "ActuatorControllerTest",
"**params**": [
{
"**type**": "int16",
"**value**" : wheel_ref
}
]
}

• 1 - reference for the wheels (positive value to the right) in degree
Response

{
"event": "method_result",
"id": 1455198439,
"object" : "/Steering",
"interface" : "br.com.arvus.TiX.Steering",
"member" : "ActuatorControllerTest",
"result" :" {
"p1": error_code
}"
}

error_code • -1: Only if you are using TrackController or TrackController next gen and you try to set the wheel reference without have set the maximum left and right position before. You can find how to set these maximum values at **3.4.6.6.1** setup topic. • 0: Success

3.4.7.7.5 ActuatorPWMTest

This test is simply to determine whether the valve cables are crossed. We will control the wheels for 2 seconds (after which we will stop moving them). We are not checking if the wheels have moved to the correct position; this should be verified by the user.

To start

{
"**event**": "call_method",
"**id**": 1455198439,
"**object**": "/Steering",
"**interface**": "br.com.arvus.TiX.Steering",
"**member**": "ActuatorPWMTest",
"**params**": [
{
"**type**": "int16",
"**value**" : pwm_ref
}
]
}

• Pwm_ref: should be a value between -1000 and 1000 (-1000 or 1000 are the maximum possible values). A negative value should move the wheels to left, positive to right.
Response

{
"event": "method_result",
"id": 1455198439,
"object" : "/Steering",
"interface" : "br.com.arvus.TiX.Steering",
"member" : "ActuatorControllerTest",
"result" :" {
“p1”: status
}”
}

• Status: ○ -1 error (another calibration is running) ○ 0 success

3.4.8 Steering Statistics

The following calls can be used to retrieve the latest histogram from the configuration file generated for the current active job.

3.4.8.1 Get last generated histogram

Message
{ "**event**": "call_method", "**id**": 1455198439,

```
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": " GetLastHistogram ",
  "params": [ ]
}
```

Message result

```
{
        "event": "method_result",
        "id": 1455198439,
        "interface": "br.com.arvus.TiX.Steering",
        "member": " GetLastHistogram",
        "object": "/Steering",
        "result": "{"p1":{
                        "Data:":"b64 _encoded_data",
                        "Name:":"filename"
                        } }"
}
```

3.4.8.2 Get last generated configuration file

Message

```
{
"event":"call_method",
"id":1455198438,
"object":"/Steering",
"interface":"br.com.arvus.TiX.Steering",
"member":" GetLastConfig",
"params": [  ]
 }
```

Message result

```
{
        "event": "method_result",
        "id": 1455198438,
        "interface": "br.com.arvus.TiX.Steering",
        "member": " GetLastConfig",
        "object": "/Steering",
        "result": "{
         "{"p1":{
                        "Data:":"b64 _encoded_data",
                        "Name:":"filename"
                } }"
}
```

3.4.9 Clear Steering alarms

To Clear

```
{
"event":"call_method",
"id":1455198438,
"object":"/Steering",
"interface":"br.com.arvus.TiX.Steering",
"member":"ClearAlarms"
 }
```

Response

```
{
        "event": "method_result",
        "id": 1455198438,
        "interface": "br.com.arvus.TiX.Steering",
        "member": "ClearAlarms",
        "object": "/Steering",
        "result": "{\"p1\":0}"
}
```

3.4.10 Steer ready

For the steer ready steering types there are status information provided by the valves and methods to change the valves configuration.

3.4.10.1 PVED-CL

3.4.10.1.1 Get Parameter

To get parameters from the valve call the Get command, using the PVED-CL User Manual to get the parameter index. The value will be received on the GetParamerResponse signal.

Get Parameter

```
{
"event": "call_method",
"id": 1455198438,
"object": "/PvedCl",
"interface": "br.com.arvus.TiX.PvedCl ",
"member": "GetParameter",
"params": [
   {
      "type": "uint32”,
      "value" : parameter_index
   }
]
}
```

Get Parameter Response

```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCl",
"member": "GetParameter",
"object": "/PvedCl",
"result": "{}"
}
```


3.4.10.1.2 Get Parameter Response Signal

To register

{
 "**event**": "subscribe",
 "**id**": 1455198438,
 "**object**": "/PvedCl",
 "**interface**": "br.com.arvus.TiX.PvedCl",
 "**member**": "GetParameterResponse"
 }

Register Response

{
 "**event**": "subscribe_result",
 "**id**": 1455198438,
 "**object**": "/PvedCl",
 "**interface**": "br.com.arvus.TiX.PvedCl",
 "**member**": "GetParameterResponse",
 “**result**”: 0
 }

Message

{
 "**event**": "signal",
 "**id**": 1455198438,
 "**object**": "/PvedCl",
 "**interface**": "br.com.arvus.TiX.PvedCl",
 "**member**": "GetParameterResponse",
 "**result**": "{
 \“p1\”: parameter_index (uint32),

```
    \“p2\”: value (uint32)
  }"
}
```

3.4.10.1.3 Set Parameter

To set parameters from the valve, call the Set command, using the PVED-CL User Manual to get the parameter index and value range. The method will return true if the command was sent correctly. The confirmation that the value was set to the valve will be received by the SetParameterResponse signal.

Set Parameter

```
{
"event": "call_method",
"id": 1455198438,
"object": "/PvedCl",
"interface": "br.com.arvus.TiX.PvedCl",
"member": "SetParameter",
"params": [
   {
      "type": "uint32",
      "value": parameter_index
   },
   {
      "type": "uint32",
      "value": value
   }
]
}
```

Set Parameter Response

```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCl",
"member": "SetParameter",
"object": "/PvedCl",
"result": {\"p1\": success (bool)}"
}
```

3.4.10.1.4 Set Parameter Response Signal

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "SetParameterResponse"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "SetParameterResponse",
  “result”: 0
 }
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "SetParameterResponse",
  "result": "{
    \“p1\”: parameter_index (uint32),
    \“p2\”: value (uint32)
  }"
}
```

3.4.10.1.5 Commit Data

To persist the data, the commit data method must be called. Similar to the set method, it returns true if the command was sent successfully. The CommitDataResponse signal will then provide the status and error codes.

Commit Data

{
"**event**": "call_method",
"**id**": 1455198438,
"**object**": "/PvedCl",
"**interface**": "br.com.arvus.TiX.PvedCl",
"**member**": "CommitData",
"**params**": []
}

Commit Data Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCl",
"member": "CommitData",
"object": "/PvedCl",
"result": {\"p1\": success (bool)}"
}

3.4.10.1.6 Commit Data Response Signal

To register

{
"**event**": "subscribe",
"**id**": 1455198438,
"**object**": "/PvedCl",
"**interface**": "br.com.arvus.TiX.PvedCl",
"**member**": "CommitDataResponse"
}

Register Response

{
"**event**": "subscribe_result",
"**id**": 1455198438,
"**object**": "/PvedCl",
"**interface**": "br.com.arvus.TiX.PvedCl",
"**member**": "CommitDataResponse",
“**result**”: 0
}

Message

{
"**event**": "signal",
"**id**": 1455198438,
"**object**": "/PvedCl",
"**interface**": "br.com.arvus.TiX.PvedCl",
"**member**": "CommitDataResponse",
"**result**": "{
\"p1\": status (uint8),
\"p2\": error code (uint16)
}"
}

Commit status can be: • 0 – ready to perform commit operation • 1 – commit operation has just begun • 2 – the operation succeeded • 3 – the operation failed • 4 – parameter cross-check failed • 5 – commit is already in progress Error codes can be found in the PVED-CL Communication Protocol document.

3.4.10.1.7 Valve status

This status can be used to check if the valve is on and ready to operate.

Is Online

{
"**event**": "call_method",
"**id**": 1455198438,
"**object**": "/PvedCl",
"**interface**": "br.com.arvus.TiX.PvedCl",
"**member**": "IsOnline",
"**params**": [ ]
}

Is Online Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCl",
"member": "IsOnline",
"object": "/PvedCl",
"result": {\"p1\": result (bool)}"
}

Is Operational

{
"**event**": "call_method",
"**id**": 1455198438,
"**object**": "/PvedCl",
"**interface**": "br.com.arvus.TiX.PvedCl",
"**member**": "IsOperational",
"**params**": [ ]
}

Is Operational Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCl",
"member": "IsOperational",
"object": "/PvedCl",
"result": {\"p1\": result (bool)}"
}

3.4.10.1.8 Became Online Signal

The following signals will be sent only when the valve goes online or offline, but they will not be transmitted periodically.

To register

{
"**event**": "subscribe",
"**id**": 1455198438,

```
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "BecameOnline"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "BecameOnline",
  “result”: 0
 }
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "BecameOnline",
  "result": "{ }"
}
```

3.4.10.1.9 Became Offline Signal

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "BecameOffline"
}
```

Register Response

```
{
```

```
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "BecameOffline",
  “result”: 0
}
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "BecameOffline",
  "result": "{ }"
}
```

3.4.10.1.10 Status information Enable

The status information messages provide additional details from the valve. To receive them, the message must be configured. Use the StatusInformationEnable method to configure it, specifying the desired dataset. Information about datasets can be found in the PVED-CL Communication Protocol.

Status Information Enable

```
{
"event": "call_method",
"id": 1455198438,
"object": "/PvedCl",
"interface": "br.com.arvus.TiX.PvedCl",
"member": "StatusInformationEnable",
"params": [
   {
      "type": "uint32",
      "value": dataset
   }
]
}
```

Data set options:

- 0 – stop status information messages
- 1 – request status dataset 1
- 2 – request status dataset 2
- 3 – request status dataset 3
- 4 – request status dataset 4

Status Information Enable Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCl",
"member": "StatusInformationEnable",
"object": "/PvedCl",
"result": { }"
}

3.4.10.1.11 Status information Signal

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "StatusInformation"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "StatusInformation",
  "result": 0
 }
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "StatusInformation",
  "result": "{
     \"p1\": dataset (uint32)
     \"p2\": [data_0 (uint8), data_1 (uint8), ....., data_7 (uint8)]
  }"
}
```

3.4.10.1.12 Operation Status Signal

To register

```
{
  "event": "subscribe",
```

"**id**": 1455198438,
"**object**": "/PvedCl",
"**interface**": "br.com.arvus.TiX.PvedCl",
"**member**": "OperationStatus"
}

Register Response

{
"**event**": "subscribe_result",
"**id**": 1455198438,
"**object**": "/PvedCl",
"**interface**": "br.com.arvus.TiX.PvedCl",
"**member**": "OperationStatus",
"**result**": 0
}

Periodic Message

{
"**event**": "signal",
"**id**": 1455198438,
"**object**": "/PvedCl",
"**interface**": "br.com.arvus.TiX.PvedCl",
"**member**": "OperationStatus",
"**result**": "{
\"p1\": current_mode (uint8)
\"p1\": selected_device (uint8)
}"
}

Current mode can be:

- 0xAA – Operational
- 0x55 – Calibration
- 0xAF – Reduced
- 0xFF – Fault

Selected device can be:

- 0 – No device selected
- 1 – steering wheel
- 2 – reserved
- 3 – high priority steering device
- 4 – low priority steering device

- 5 – high priority external set-point controller

3.4.10.1.13 Valve Mode Changed Signal

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "ValveModeChanged"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "ValveModeChanged",
  “result”: 0
 }
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCl",
  "interface": "br.com.arvus.TiX.PvedCl",
  "member": "ValveModeChanged",
  "result": “{
     \“p1\”: current_mode (uint8)
  }”
}
```

Current mode can be:

- 0xAA – Operational
- 0x55 – Calibration
- 0xAF – Reduced

- 0xFF – Fault

3.4.10.2 PVED-CLS

3.4.10.2.1 Get Parameter

To get parameters from the valve, call the Get command using the PVED-CLS User Manual to get the parameter address and size. The value will be received on the GetParamerResponse signal.

Get Parameter

```
{
"event": "call_method",
"id": 1455198438,
"object": "/PvedCls",
"interface": "br.com.arvus.TiX.PvedCls",
"member": "GetParameter",
"params": [
   {
      "type": "uint32",
      "value" : parameter_address
   },
   {
      "type": "uint32",
      "value" : bytes_to_read
   }
]
}
```

Get Parameter Response

```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCls",
"member": "GetParameter",
"object": "/PvedCls",
"result": "{}"
}
```

3.4.10.2.2 Parameter Read Signal

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "ParameterRead"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "ParameterRead",
  “result”: 0
 }
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "ParameterRead",
  "result": "{
    \“p1\”: success (bool),
    \“p2\”: parameter_address (uint16),
    \“p3\”: value (uint32),
    \“p4\”: read_bytes(uint8)
  }"
}
```

3.4.10.2.3 Set Parameter

To set parameters on the valve, call the Set command, referring to the PVED-CLS User Manual for the parameter address, size, and value range. The method returns true if the command is sent successfully. If an error occurs during transmission, the SetParameterError signal will be emitted. The Set method will automatically enter bootloader mode to write the new value and exit once the process is complete.

Set Parameter


```
{
"event": "call_method",
"id": 1455198438,
"object": "/PvedCls",
"interface": "br.com.arvus.TiX.PvedCls",
"member": "SetParameter",
"params": [
   {
      "type": "uint32",
      "value": parameter_address
   },
   {
      "type": "uint32",
      "value": value
   }
]
}
```


Set Parameter Response


```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCls",
"member": "SetParameter",
"object": "/PvedCls",
"result": {\"p1\": success (bool)}"
}
```


3.4.10.2.4 Set Parameter Error Signal

To register


```
{
   "event": "subscribe",
   "id": 1455198438,
   "object": "/PvedCls",
   "interface": "br.com.arvus.TiX.PvedCls",
```

"**member**": "SetParameterError"
}

Register Response

{
"**event**": "subscribe_result",
"**id**": 1455198438,
"**object**": "/PvedCls",
"**interface**": "br.com.arvus.TiX.PvedCls",
"**member**": "SetParameterError",
“**result**”: 0
}

Message

{
"**event**": "signal",
"**id**": 1455198438,
"**object**": "/PvedCls",
"**interface**": "br.com.arvus.TiX.PvedCls",
"**member**": "SetParameterError",
"**result**": "{
\“p1\”: controller (uint8),
\“p2\”: state (uint8),
\“p3\”: data (uint64)
}"
}

Controller can be:

- 0 – main controller,
- 1 – safety controller

3.4.10.2.5 Bootloader mode enter

In bootloader mode, the valve's configuration can be modified. Setting new values will automatically trigger bootloader mode. Exiting this mode will clear all pending processes.

Enter Bootloader Mode

{
"**event**": "call_method",
"**id**": 1455198438,
"**object**": "/PvedCls",
"**interface**": "br.com.arvus.TiX.PvedCls",
"**member**": "EnterBootloaderMode",
"**params**": [ ]
}

Enter Bootloader Mode Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCls",
"member": "EnterBootloaderMode",
"object": "/PvedCls",
"result": {\"p1\": success (bool)}"
}

3.4.10.2.6 Bootloader mode exit

Exit Bootloader Mode

{
"**event**": "call_method",
"**id**": 1455198438,
"**object**": "/PvedCls",
"**interface**": "br.com.arvus.TiX.PvedCls",
"**member**": "ExitBootloaderMode",
"**params**": [ ]
}

Exit Bootloader Mode Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCls",
"member": "ExitBootloaderMode",
"object": "/PvedCls",
"result": { }"
}

3.4.10.2.7 Bootloader mode entered Signal

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "EnteredBootloaderMode"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "EnteredBootloaderMode",
  “result”: 0
 }
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "EnteredBootloaderMode",
  "result": "{
    \“p1\”: controller (uint8)
  }"
}
```

Controller can be:

- 0 – main controller,
- 1 – safety controller

3.4.10.2.8 Bootloader mode exited Signal

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "ExitedBootloaderMode"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "ExitedBootloaderMode",
  “result”: 0
 }
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "ExitedBootloaderMode",
  "result": "{
    \“p1\”: controller (uint8)
  }"
}
```

Controller can be:

- 0 – main controller,
- 1 – safety controller

3.4.10.2.9 Valve status

This status can be used to check if the valve is on and ready to operate.

Is Online

{
"**event**": "call_method",
"**id**": 1455198438,
"**object**": "/PvedCls",
"**interface**": "br.com.arvus.TiX.PvedCls",
"**member**": "IsOnline",
"**params**": [ ]
}

Is Online Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCls",
"member": "IsOnline",
"object": "/PvedCls",
"result": {\"p1\": result (bool)}"
}

Is Operational

{
"**event**": "call_method",
"**id**": 1455198438,
"**object**": "/PvedCls",
"**interface**": "br.com.arvus.TiX.PvedCls",
"**member**": "IsOperational",
"**params**": [ ]
}

Is Operational Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCls",
"member": "IsOperational",
"object": "/PvedCls",
"result": {\"p1\": result (bool)}"
}

3.4.10.2.10 Became Online Signal

The following signals will be sent when the valve goes online or offline but will not be sent periodically.

To register


```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "BecameOnline"
}
```


Register Response


```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "BecameOnline",
  “result”: 0
 }
```


Message


```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "BecameOnline",
  "result": "{ }"
}
```


3.4.10.2.11 Became Offline Signal

To register


```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "BecameOffline"
}
```


Register Response


```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "BecameOffline",
  “result”: 0
 }
```


Message


```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "BecameOffline",
  "result": "{ }"
}
```


3.4.10.2.12 Status messages

Status messages provide additional information from the valve. To receive them, the message must be configured. Use the EnableStatusMessages method to configure it. Details about the messages can be found in the PVED-CLS Communication Protocol.

Enable Status Messages

```
{
"event": "call_method",
"id": 1455198438,
"object": "/PvedCls",
"interface": "br.com.arvus.TiX.PvedCls",
"member": "EnableStatusMessages",
"params": [
   {
      "type": "boolean",
      "value": enable_all
   }
]
}
```


Enable Status Messages Response


```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.PvedCls",
"member": "EnableStatusMessages",
"object": "/PvedCls",
"result": { }"
}
```


3.4.10.2.13 Status Information Signal

To register


```
{
   "event": "subscribe",
   "id": 1455198438,
   "object": "/PvedCls",
   "interface": "br.com.arvus.TiX.PvedCls",
   "member": "StatusInformation"
}
```


Register Response


```
{
   "event": "subscribe_result",
   "id": 1455198438,
   "object": "/PvedCls",
   "interface": "br.com.arvus.TiX.PvedCls",
   "member": "StatusInformation",
```

```
  “result”: 0
}
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "StatusInformation",
  "result": “{
     \“p1\”: status_pgn (uint16)
     \“p1\”: [\“data_0 (uint8)\”, \”data_1 (uint8)\”, ....., \”data_7 (uint8)\”]
  }”
}
```

3.4.10.2.14 Operation Status Signal

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "OperationStatus"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "OperationStatus",
  “result”: 0
}
```

Periodic Message

<table>
<tr><td>{
"event": "signal",
"id": 1455198438,
"object": "/PvedCls",
"interface": "br.com.arvus.TiX.PvedCls",
"member": "OperationStatus",
"result": “{
\“p1\”: operation_state (uint8)
\“p2\”: service_mode_state (uint8)
}”
}</td></tr>
<tr><td>Operation state can be:
0x00 – On-Road
0x10 – Off-road reaction
0x11 – Off-road non-reaction
0x40 – GPS-Steering
0x41 – GPS2-Steering
0xD0 – Off road safety check
0xE0 – Direct output control
0xE1 – Service mode – Wheel angle sensor calibration
0xE2 – Service mode – Spool calibration
0xFF – Safe state
More states are described on the PVED-CLS Communication protocol
Service mode state can be:
0x00 – Direct output control reset
0x01 – Direct output control active
0x11 – WAS calibration in progress
0x1F – WAS calibration completed
0x1E – WAS calibration failure
0x21 – Spool calibration inactive
0x2E – Spool calibration failure
0xFE – No WAS configurated
0xFF – Not available
More states are described on the PVED-CLS Communication protocol</td></tr>
</table>

3.4.10.2.15 DM1 Message Signal

This message is sent every 1 second or when a change occurs.

To register


```
{
  "event": "subscribe",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "DM1Message"
}
```


Register Response


```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "DM1Message",
  “result”: 0
 }
```


Periodic Message


```
{
  "event": "signal",
  "id": 1455198438,
  "object": "/PvedCls",
  "interface": "br.com.arvus.TiX.PvedCls",
  "member": "DM1Message",
  "result": “{
     \“p1\”: error (uint8),
     \“p2\”: fmi (uint8),
     \“p3\”: spn(uint32)
  }”
}
```


Error can be:

- 0x00 – No active failure
- 0x04 – Warning info errors
- 0x10 – Critical/Severe errors

SPN and FMI can be found on PVED-CLS User Manual, where SPN represents the error category and FMI the failure mode.

3.4.11 Get Track Controller next gen version

To Get a Param

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Steering",
  "interface": "br.com.arvus.TiX.Steering",
  "member": " GetTrackControllerNextGenVersion"
}
```

Get a Param Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Steering",
  "interface" : "br.com.arvus.TiX.Steering",
  "member" : "GetParam",
  "result" : "{   \"p1\" : version  }"
}
```

3.5 Implement

3.5.1 Set implement parameters

The Core Box saves the implement configuration, allowing it to use the last valid settings at the next start-up. An implement includes the following parameters:

- **width:** implement width (meters)
- **vehicleDistance:** the distance between the centerline of the implement and the vehicle's pivot in meters
- **lateralOffset:** the distance in meters between the implement's centerline and the vehicle's centerline. This value is used to adjust the position for the steering system's control.

Command to set implement parameters

```
{
"event":"call_method",
"id":1455198438,
"object":"/Implement",
"interface":"br.com.arvus.TiX.Implement",
"member":"SetDefault",
"params": [
```

```
            {
                "type": "double",
                "value" : implement_width
             },
             {
                "type": "double",
                "value": implement_vehicle_distance
             },
             {
                "type": "double",
                "value": implement_lateral_offset
             }
    ]
}
```

Set implement response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Implement",
  "interface" : "br.com.arvus.TiX.Implement",
  "member" : "SetDefault",
  "result" : “{
    \“p1\”: status
  }”
}
```

Status (uint16) could be one of them:

- 0 - success setting the current vehicle
- 1 - error

3.5.2 Get implement parameters

Get Implement Parameters

```
{
"event":"call_method",
"id":1455198438,
"object":"/Implement",
"interface":"br.com.arvus.TiX.Implement",
"member":"GetDefault"
 }
```

Get Implement Parameters response

```
{
"event": "method_result",
"id": 1455198438,
```

```
"object" : "/Implement",
"interface" : "br.com.arvus.TiX.Implement",
"member" : "GetDefault",
"result" : “{
        \“p1\”: [implement_width, implement_vehicle_distance, implement_lateral_offset]
        }”
}
```

Result Type	double array
p1 (double array)	Data in meters

3.6 Guidance

3.6.1 Guidance Update Signal

This message is continuously transmitted by the ECU.
To receive it, the device must register to listen for the 'UpdateGuidance' signal.
The coordinate frame used to report the position is WGS-84 (degrees for lat, long and meters for altitude).
Each row has a unique id (token), which is sent with the 'UpdateGuidance' signal. This token must match the one sent with the "Engage" command (Steering object); otherwise, the system will not engage.

To register

```
{
"event": "subscribe",
"id": 1455198438,
"object" : "/Guidance",
"interface" : "br.com.arvus.TiX.Guidance",
"member" : "member_type"
}
```

member_type	Description
UpdateGuidance1hz	Register for receiving messages with Frequency of 1hz
UpdateGuidance5hz	Register for receiving messages with Frequency of 5hz

UpdateGuidance10hz	Register for receiving messages with Frequency of 10hz

Register Response

{
"event": "subscribe_result",
"id": 1455198438,
"object" : "/Guidance",
"interface" : "br.com.arvus.TiX.Guidance",
"member" : "member_type",
"result" : 0
}

Periodic message

{
"event": "signal",
"id": 1455198438,
"object" : "/Guidance",
"interface" : "br.com.arvus.TiX.Guidance",
"member" : "member_type",
"result" :
{
"p1": token,
"p2": nearest_point_long,
"p3": nearest_point_lat,
"p4": length,
"p5": distance_from_line,
"p6": reserved
}
}

Result Fields	Description
p1 - token (string)	Current path token
p2 - nearest_point_long (double)	Longitude of nearest point
p3 - nearest_point_lat (double)	Latitude of nearest point
p4 - length (double)	Path remaining length (distance until the end of line)
p5 - distance_from_line (double)	Current path error (meters) (+ to right)
p6 - guidance_type (uint16)	• 0 – Curve AB • 1 – Straight AB • 2 – Pivot

	• 3 – Straight A+ heading • 4 – Adaptative • 5 – Imported shapefile • 6 – Points feed • 7 – Unknown

3.6.2 Guidance Type Points Feed

This message defines a series of points that outline a path for the steering control system to follow.

If the steering is engaged and the token sent with this message does not match the current token, the steering will be disengaged. If there is a significant distance (over 5 meters) between the new and old segments, the issue will be indicated in the returned status. However, the steering will remain engaged to allow recovery from the problem. The points must be sent in the correct order to construct the path properly.

There is no limit to the number of points that can be sent, but it is recommended to include some repeated points from the last segment. Sending fewer than four points (x, y) may not work in some cases, so this is the minimum recommended.

Points Feed command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Guidance",
"interface":"br.com.arvus.TiX.Guidance",
"member":"PointsFeed",
"params": [
            {
                "type": "string",
                "value" : "token"
             },
             {
                "type": "uint16",
                "value":points_order_to_be_followed
             },
             {
                "type": "double array",
                "value": [points_array]
             }
    ]
}
```

Params	Description
token (string)	Path token

points_order_to_be_followed (uint16)	Points order to following. Could be one of them: 0 - any heading order 1 - only points heading to steer 2 - only opposite points heading to steer Known issue: Points ordered not being used in all Software version, to be fixed in version 4.9.0
points_array (double array)	Segment points, the first double it is the x position (longitude) and the second is the y (latitude) => [x1,y1,x2,y2,...,xn,yn]

Points Feed command response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Guidance",
  "interface" : "br.com.arvus.TiX.Guidance",
  "member" : "PointsFeed",
  "result" : “{
    \“p1\”: status
  }”
}
```

Status:

- 0 - success - points accepted
- 1 - rejected - incorrect number of points (odd or too few)
- 2 - rejected - No GPS
- 3 - rejected - Wrong token
- 4 - rejected - Too far from last points

3.6.3 SetParameters

To Set a Param

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Guidance",
  "interface": "br.com.arvus.TiX.Guidance",
  "member": "SetParam",
  "params": [
    {
      "type": "uint16",
      "value" : param_code
    },
    {
```

<table>
<tr><td>

<pre>
      "type": "double",
      "value": param_value
    }
  ]
}
</pre>

</td></tr>
<tr><th>Set Param Response</th></tr>
<tr><td>

<pre>
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Guidance",
  "interface" : "br.com.arvus.TiX.Guidance",
  "member" : "SetParam",
  "result": "{
    \"p1\": status
   }"
}
</pre>

</td></tr>
<tr><td>

The status (uint16) field could be one of them

- 0 - success
- 1 - param not set, invalid parameter

</td></tr>
</table>

The field “paramCode” could be one of them:

paramCode (uint16)	description	Supported values	Default	Type	Can be set
0	Curve filter (zero delay filter)	1-100 (% passing band)	30	double	yes
1	Enable Curve filter	0-1	0	double	yes
2	Curve Distance filter	0.1-10 (meters)	3	double	yes
3	Enable Distance filter	0-1	0	double	yes

3.6.4 Get Parameters

For the fields param_code and param_value, you can refer to the topic 3.6.3 Set Parameters.

To Get a Param

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Guidance",
  "interface": "br.com.arvus.TiX.Guidance",
  "member": "GetParam",
  "params": [
    {
      "type": "uint16",
      "value" : param_code
    }
  ]
}
```

Get a Param Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object": "/Guidance",
  "interface": "br.com.arvus.TiX.Guidance",
  "member": "GetParam",
  "result": "{   \"p1\" : [param_code, param_value]   }"
}
```

If the param is not found, [-1,0] is returned.

3.6.5 Guidance lines CRUD

3.6.5.1 Start Guidance line Creation

The guidance lines work exactly as in Hexagon's displays. This call is equivalent to pressing button "A" when creating a guidance line. This call must be used in the starting position of the guidance to be created. Note that this D-BUS does not check GNSS synchronization.

Start Guidance Line Creation

```
{
  "event": "call_method",
  "id": 1455198439,
  "interface": "br.com.arvus.TiX.WaylineCrud",
  "member": "StartWaylineCreation",
  "object": "/WaylineCrud",
```

```
   "params": [
      {
         "type": "int32",
         "value": wayline_type
      }
   ]
}
```

Start Guidance Line Creation Response

```
{
   "event": "call_method",
   "id": 1455198439,
   "interface": "br.com.arvus.TiX.WaylineCrud",
   "member": "StartWaylineCreation",
   "object": "/WaylineCrud",
   "result": "{
      \"p1\": status
    }"
}
```

The status (uint16) field could be one of them

- 0 – success
- 1 – Generic error
- 2 – Creation not started
- 3 – Invalid Guidance line
- 4 – Missing Activation
- 5 – Invalid guidance line type

Guidance type	Description
0	Curve
1	Liner
2	Pivot
3	Angle
4	Adaptive

3.6.5.2 Stop Guidance Line Creation

This call must be made when the vehicle is on the end of the guidance to be created.

Stop Guidance Line Creation

```
{
```

```
   "event": "call_method",
   "id": 1455198439,
   "interface": "br.com.arvus.TiX.WaylineCrud",
   "member": "StopWaylineCreation",
   "object": "/WaylineCrud",
   "params": [
      {
         "type": "int32",
         "value": wayline_type
      }
   ]
}
```

Stop Guidance Line Creation Response

```
{
   "event": "call_method",
   "id": 1455198439,
   "interface": "br.com.arvus.TiX.WaylineCrud",
   "member": "StopWaylineCreation",
   "object": "/WaylineCrud",
   "result": "{
      "p1":
           "{
                \"name\": \"wayline_name\",
                \"status\": status
            }"
    }"
}
```

3.6.5.3 Cancel Guidance Line Creation

Cancel Guidance Line Creation

```
{
   "event": "call_method",
   "id": 1455198439,
   "interface": "br.com.arvus.TiX.WaylineCrud",
   "member": "CancelWaylineCreation",
   "object": "/WaylineCrud",
   "params": [
      {
         "type": "int32",
         "value": wayline_type
      }
   ]
}
```

Cancel Guidance Line Creation Response

```
{
   "event": "call_method",
```

```
    "id": 1455198439,
    "interface": "br.com.arvus.TiX.WaylineCrud",
    "member": "CancelWaylineCreation",
    "object": "/WaylineCrud",
    "result": "{
        \"p1\": status
     }"
}
```

3.7 Data transfer

3.7.1 List

This command lists all available files.

File List command

```
{
"event":"call_method",
"id":1455198438,
"object":"/DataTransfer",
"interface":"br.com.arvus.TiX.DataTransfer",
"member":"Index",
"params": [
        {
          "type": "uint16",
          "value" :file_type
         },
        {
          "type": "uint16",
          "value": location
         }
     ]
}
```

Params	Description
file_type	0 - list all operation map files 1 - list all prescription map files 2 - list all map markers files 3 - list all guidance line files 4 - list all settings files 5 - list all steering files 6 - list all available FW updates .ati files (from flash-drive plugged at titanium/) 7 - list all available log files
location	1 - Flash drive files 2 - Core Box files

File List command response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/DataTransfer",
  "interface" : "br.com.arvus.TiX.DataTransfer",
  "member" : "Index",
  "result" : “{
    \“p1\”: [\“file_name_1\”,\”file_name_2\”,.....,\”file_name_n\”]
  }”
}
```

- P1 is a string array with available file names

3.7.2 Export

File export command

```
{
"event":"call_method",
"id":1455198438,
"object":"/DataTransfer",
"interface":"br.com.arvus.TiX.DataTransfer",
"member":"Export",
"params": [
        {
          "type": "uint16",
          "value" :file_type
         },
        {
          "type": "string array",
          "value" : ["file_names"]
         },
        {
          "type": "uint16",
          "value" :export_file_extension
         },
        {
          "type": "uint16",
          "value" : where_to_export
         }
     ]
 }
```

Params	Description
file_type	See section 3.7.1
file_names	Array with file names to be exported. If you send an empty array all files of the selected file_type will be transferred

export_file_extension	For operation map files you can set the type of exported file • 0 - for .ti (Saig - hxgn software) • 1 - for .kml • 2 - for .shp
where_to_export	• 0 - for flash drive (Core Box) • 1 - for the API (Base64 encoded)

File export command response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/DataTransfer",
  "interface" : "br.com.arvus.TiX.DataTransfer",
  "member" : "Export",
  "result" : “{
          \"p1\":
                    "[{ "File":{
                              "Data:":"b64 _encoded_data_1",
                              "Name:":"filename_1"
                              }
                    },
                    { "File":{
                              "Data:":"b64_encoded_data_2",
                              "Name:":"filename_2"
                              }
                    }
                    ...
                    { "File":{
                              "Data:":"b64_encoded_data_n",
                              "Name:":"filename_n"
                              }
                    }]"
  }
}
```

- P1 is a string, will be empty if you do not select the field where_to_export = 1

3.7.3 Delete

File delete command

```
{
"event":"call_method",
"id":1455198438,
```

```
"object":"/DataTransfer",
"interface":"br.com.arvus.TiX.DataTransfer",
"member":"Delete",
"params": [
        {
          "type": "uint16",
          "value" : file_type
         },
        {
          "type": "string array",
          "value" : ["file_names"]
         }
     ]
 }
```

Params	Description
file_type	See section 3.7.1
file_names	Array with file names to be deleted

File export command response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/DataTransfer",
  "interface" : "br.com.arvus.TiX.DataTransfer",
  "member" : "Delete",
  "result" : “{
    \“p1\”: code
  }”
}
```

- P1 is a uint16
  - 0 success
  - 1 - error - file(s) not found

3.7.4 Import

Import the selected files from the plugged flash drive to the Core Box. These files must be firmware updates (.ati files).

File import command

```
{
"event":"call_method",
"id":1455198438,
"object":"/DataTransfer",
"interface":"br.com.arvus.TiX.DataTransfer",
```

```
"member":"Import",
"params": [
        {
          "type": "uint16",
          "value" : file_type
         },
        {
          "type": "string array",
          "value" : ["file_names"]
         }
    ]
}
```

Params	Description
file_type	See section 3.7.1
file_names	Array with file names to be imported. If you send an empty array all files of the selected file_type will be transferred

File import command response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/DataTransfer",
  "interface" : "br.com.arvus.TiX.DataTransfer",
  "member" : "Import",
  "result" : “{
    \“p1\”: code
  }”
}
```

- P1 is a uint16
  - 0 success
  - 1 - error - without flash drive

3.8 Networking

3.8.1 3G

3.8.1.1 Modem Signal Quality Changed Signal

To register

```
{
  "event": "subscribe",
```

```
  "id": 1455198438,
  "object" : "/Network",
  "interface" : "br.com.arvus.TiX.Network",
  "member" : "ModemSignalQualityChanged"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object" : "/Network",
  "interface" : "br.com.arvus.TiX.Network",
  "member" : "ModemSignalQualityChanged",
  “result” : 0
}
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object" : "/Network",
  "interface" : "br.com.arvus.TiX.Network",
  "member" : "ModemSignalQualityChanged",
  "result" : "{
    \“p1\”: signal_quality,
  }"
}
```

- Signal_quality int16 - signal quality (RSSI 0-100)

3.8.1.2 Get Connection state

To Get Connection State

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Network",
  "interface": "br.com.arvus.TiX.Network",
  "member": "ModemConnectionState",
  "params": [ ]
}
```


Set Response


```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Network",
  "interface" : "br.com.arvus.TiX.Network",
  "member" : "ModemConnectionState",
  "result": "{
    \"p1\": status
   }"
}
```


Status: (uint16)

- -1 - undefined
- 0 - disconnected
- 1 - connecting (prepare)
- 2 - connecting (getting IP)
- 3 - connected

3.8.1.3 Get connection settings

To Get Connection Settings


```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Network",
  "interface": "br.com.arvus.TiX.Network",
  "member": "Get3gConfiguration",
  "params": [ ]
}
```


Get Response


```
{
```

```
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.Network",
"member": "Get3gConfiguration",
"object": "/Network",
"result": "{\"p1\":
   "{
      "APN": apn,
      "Password": password,
      "User": username
   }"
}"
}
```

- P1 will be empty if there is no 3g available

3.8.1.4 Set Configuration

To Set Connection Settings

```
{
   "event":"call_method",
   "id":1455198438,
   "object":"/Network",
   "interface":"br.com.arvus.TiX.Network",
   "member":"Set3gConfiguration",
   "params":[
            {
            "type":"string",
            "value": “apn”
            },
            {
            "type":"string",
            "value": “user”
            },
            {
            "type":"string",
            "value": “password”
            }
   ]
}
```

Set Response

```
{
"event": "method_result",
```

"id": 1455198438,
"interface": "br.com.arvus.TiX.Network",
"member": "Set3gConfiguration",
"object": "/Network",
"result": "{\"p1\": status }"
}
status: (uint16)

- 0 - success
- 1 - error
- 3 - no 3g available

3.8.1.5 Get Connection Enabled

To Get 3g Enable State

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Network",
  "interface": "br.com.arvus.TiX.Network",
  "member": "Get3gEnabled",
  "params": [ ]
}
```

Get Response

{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.Network",
"member": "Get3gEnabled",
"object": "/Network",
"result": "{\"p1\": state(bool) }"
}

If “state-enabled” is equal to false, the Core Box will disconnect and disable auto-connect on startup, otherwise it will try to connect and enable auto-connect.

To Set 3g Enable

```
{
   "event":"call_method",
   "id":1455198438,
   "object":"/Network",
   "interface":"br.com.arvus.TiX.Network",
   "member":"Set3gEnabled",
   "params":[
            {
            "type":"boolean",
            "value":state-enabled
            }
   ]
}
```


Set Response


```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.Network",
"member": "Set3gEnabled",
"object": "/Network",
"result": "{\"p1\": status }"
}
```


status: (uint16)

- 0 - success
- 1 – error
- 3 - no 3g available

3.8.1.6 Connection Information

To Get Connection Information


```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Network",
  "interface": "br.com.arvus.TiX.Network",
  "member": "Get3gInformation",
  "params": [ ]
}
```


Get Response

```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.Network",
"member": "Get3gInformation",
"object": "/Network",
"result": "{\"p1\":
   \"{
      \"Access technologies\": \"HSDPA\",
      \"Imei\": \"354043052234749\",
      \"Manufacturer\": \"Telit\",
      \"Model\": \"HE910-EUD\",
      \"Operator name\": \"VIVO\",
      \"Revision\": \"12.00.226\",
      \"own\":\"008523645478\",
      \"SIM connected\": true,
      \"Signal quality\": \"45\"
   }\"
}"
}
```


3.8.2 Wi-Fi

3.8.2.1 Signal Wi-Fi Strength Changed

To register

{
 "event": "subscribe",
 "id": 1455198438,
 "object" : "/Network",
 "interface" : "br.com.arvus.TiX.Network",
 "member" : "WifiSignalStrengthChanged"
 }

Register Response
{

```
  "event": "subscribe_result",
  "id": 1455198438,
  "object" : "/Network",
  "interface" : "br.com.arvus.TiX.Network",
  "member" : "WifiSignalStrengthChanged",
  “result” : 0
}
```

Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object" : "/Network",
  "interface" : "br.com.arvus.TiX.Network",
  "member" : "WifiSignalStrengthChanged",
  "result" : "{
    \“p1\”: signal_quality,
  }"
}
```

- Signal_quality int16 - signal quality

3.8.2.2 AP configuration

You must use this command to change SSID or password of your AP. The default values for these fields will be:

- SSID: hex-ag-SN
- PASSWORD: tixSN

Tip: SN refers to the Core Box serial number, which can be found on the back of the enclosure.

To Set

```
{
```

```
  "event": "call_method",
  "id": 1455198439,
  "object": "/Network",
  "interface": "br.com.arvus.TiX.Network",
  "member": "SetAPConfiguration",
  "params": [
    {
      "type": "string",
      "value": "SSID"
    },
    {
      "type": "string",
      "value": "password"
    }
  ]
}
```

Set Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Network",
  "interface" : "br.com.arvus.TiX.Network",
  "member" : "SetAPConfiguration",
  "result": "{
    \"p1\": status
   }"
}
```

Status:

- 0 - success
- 1 - invalid password (must have at least 8 digits)
- 2 - no Wi-Fi available

3.9 ECU Firmware update

3.9.1 List CAN ECUs

This command lists all available ECUs for update, referred to as drivers here.

To List

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Updater",
  "interface": "br.com.arvus.TiX.Updater",
  "member": "IndexCANDrivers",
  "params": [ ]
}
```

To List Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Updater",
  "interface" : "br.com.arvus.TiX.Updater",
  "member" : "IndexCANDrivers",
  "result": "{
    \"p1\": [“driver_name_1”,“driver_name_2”,..,“driver_name_n”]
   }"
}
```

- P1 - string array with ECU name

3.9.2 List CAN ECU With Update

This command list only ECU with an update available. The ECU is referred as driver.

To List

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Updater",
  "interface": "br.com.arvus.TiX.Updater",
  "member": "IndexCANDriversHasUpdate",
  "params": [ ]
}
```

To List Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Updater",
```

```
  "interface" : "br.com.arvus.TiX.Updater",
  "member" : "IndexCANDriversHasUpdate",
  "result": "{
    \"p1\": [“driver_name_1”,“driver_name_2”,..,“driver_name_n”]
   }"
}
```

- P1 - string array with drivers’ names

3.9.3 List Available Firmwares

This command lists all available firmwares for an ECU.

To FW List

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Updater",
  "interface": "br.com.arvus.TiX.Updater",
  "member": "IndexDriverFW",
  "params": [
     {
        "type": "string",
        "value" : "driver_name"
     }
  ]
}
```

To FW List Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Updater",
  "interface" : "br.com.arvus.TiX.Updater",
  "member" : "IndexDriverFW",
  "result": "{
    \"p1\": [“firmware_name_1”,“ firmware_name _2”,..,“ firmware_name _n”]
   }"
}
```

- P1 - string array with available firmwares

3.9.4 Start an update

This command starts an ECU update. To monitor the update progress or receive notifications of any errors, listen to the "FW update feedback" signal (3.10.4). Once the update starts, the process cannot be stopped.

To Start an update

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Updater",
  "interface": "br.com.arvus.TiX.Updater",
  "member": "UpdateDriver",
  "params": [
    {
      "type": "string",
      "value" : "driver_name"
    },
    {
      "type": "string",
      "value" : " firmware_name "
    }]
}
```

- Driver_name is the drivers' name (3.10.1)
- firmware_name is the firmware name (3.10.2)

To Start an update Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Updater",
  "interface" : "br.com.arvus.TiX.Updater",
  "member" : "UpdateDriver",
  "result": "{
    \"p1\": code
   }"
}
```

- Code (uint16)
  - 0 - update started
  - 1 - invalid driver_name

- 2 - invalid fw name
- 3 - already has an update in progress

3.9.5 Update All

This command starts the update for all ECUs. If you want to know the update progress of any kind of error during the update, you should listen to the signal “FW update feedback” (3.10.4). After you start an update, the process cannot be stopped.

To Start an update

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/Updater",
  "interface": "br.com.arvus.TiX.Updater",
  "member": "UpdateAll",
  "params": [
    ]
}
```

To Start an update Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/Updater",
  "interface" : "br.com.arvus.TiX.Updater",
  "member" : "UpdateAll",
  "result": "{
   \"p1\": code
   }"
}
```

- Code (uint16)
  - 0 - update started
  - 1 - invalid driver_name
  - 2 - invalid fw name
  - 3 - already has an update in progress
  - 4 - all drivers updated

3.9.6 Signal Firmware Update Feedback

This signal will be sent periodically every 1 second during a firmware update.

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "object" : "/Updater",
  "interface" : "br.com.arvus.TiX.Updater",
  "member" : "FWUpdate"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "object" : "/Updater",
  "interface" : "br.com.arvus.TiX.Updater",
  "member" : "FWUpdate",
  “result” : 0
 }
```

Periodic Message

```
{
  "event": "signal",
  "id": 1455198438,
  "object" : "/Updater",
  "interface" : "br.com.arvus.TiX.Updater",
  "member" : "FWUpdate",
  "result" : "{
    \“p1\”: status_code,
    \“p2\”: update_progress,
    \“p3\”: driver_name,
    \“p4\”:  firmware_name,
  }"
}
```

Result Fields	Description
status_code (uint16)	• 0 - update finished

	• 1 - update in progress • 2 - update error - cannot enter in bootloader mode
Update_progress (uint16)	Update progress (%)
Driver_name (string)	ECU name
firmware_name	FW that is being saved

3.10 Keep Alive

3.10.1 Signal Keep Alive

This signal will be sent every 60 seconds.

To register

```
{
  "event": "subscribe",
  "id": 1455198438,
  "interface": "br.com.arvus.TiX.KeepAlive",
  "member": "KeepAliveSignal",
  "object": "/KeepAlive"
}
```

Register Response

```
{
  "event": "subscribe_result",
  "id": 1455198438,
  "interface": "br.com.arvus.TiX.KeepAlive",
  "member": "KeepAliveSignal",
  "object": "/KeepAlive",
  "result": 0
}
```

Periodic Message

```
{
  "event": "signal",
  "id": 1470251948710,
  "interface": "br.com.arvus.TiX.KeepAlive",
  "member": "KeepAliveSignal",
  "object": "/KeepAlive",
  "result": "{}"
```

```
}
```

3.10.2 Keep Alive Command

This command must be called once every second.

Command

```
{
  "event": "call_method",
  "id": 1455198438,
  "interface": "br.com.arvus.TiX.KeepAlive",
  "member": "KeepAlive",
  "object": "/KeepAlive"
}
```

Command Response

```
{
  "event": "method_result",
  "id": 1455198438,
  "interface": "br.com.arvus.TiX.KeepAlive",
  "member": "KeepAlive",
  "object": "/KeepAlive",
  "result": "{}"
}
```

3.10.3 Keep Alive Set Params

All set/get methods will return/set the values of the current vehicle.

To Set a Param

```
{
  "event": "call_method",
  "id": 1455198439,
  "object": "/KeepAlive",
  "interface": "br.com.arvus.TiX.KeepAlive",
  "member": "SetParam",
  "params": [
    {
      "type": "uint16",
      "value" : param_code
    },
    {
      "type": "double",
      "value": param_value
    }
```

```
  ]
}
```

Set Param Response

```
{
  "event": "method_result",
  "id": 1455198439,
  "object" : "/KeepAlive",
  "interface" : "br.com.arvus.TiX.KeepAlive",
  "member" : "SetParam",
  "result": "{
    \"p1\": status
   }"
}
```

The status (uint16) field could be one of them

- 0 - success
- 1 - param not set, invalid parameter

The field “paramCode” could be one of them:

paramCode (uint16)	description	Default	Type	unit
0	disengage	5	double	seconds
1	shutdown	3600	double	seconds

3.10.4 Keep Alive Get Params

For a reference on the fields param_code and param_value, please refer to the topic 3.10.3 Keep Alive Set Params.

To Get a Param

```
{
   "event": "call_method",
   "id": 1455198439,
   "object": "/KeepAlive",
   "interface": "br.com.arvus.TiX.KeepAlive",
   "member": "GetParam",
   "params": [
     {
```

```
        "type": "uint16",
        "value" : param_code
      }
   ]
}
```

Get a Param Response

```
{
   "event": "method_result",
   "id": 1455198439,
   "object" : "/KeepAlive",
   "interface" : "br.com.arvus.TiX.KeepAlive",
   "member" : "GetParam",
   "result" : "{   \"p1\" : [param_code, param_value]   }"
}
```

If the param is not found, [-1,0] is returned.

3.11 CAN Settings

The CAN Settings has methods to configure the can interface. This interface is available from version 3.26.0.

3.11.1 Get Available Interfaces

It returns a list of all available interfaces, including the bitrate and interface name for physical interfaces, and the IP and port for virtual interfaces.

Get Available Interfaces Command

```
{
    "event":"call_method",
    "id":1455198438,
    "interface":"br.com.arvus.TiX.Can",
    "member":"GetAllInterfaces",
    "object":"/Can"
}
```

Get Available Interfaces Command Response

```
{
"event": "method_result",
"id": 1455198438,
```

```
"interface": "br.com.arvus.TiX.Can",
"member": "GetAllInterfaces",
"object": "/Can",
"result": "{\"p1\": [
   {
      "Bitrate":can_bitrate,
      "Interface":"can_name",
      "Type":"Physical"
   },
   {
      "IP":"virtual_can_ip",
      "Port":virtual_can_port,
      "Type":"Virtual"
   }
]}
```


3.11.2 Get Available Features

It returns a list of all configured features.

Get Available features Command


```
{
   "event":"call_method",
   "id":1455198438,
   "interface":"br.com.arvus.TiX.Can",
   "member":"GetAllFeatures",
   "object":"/Can"
}
```


Get Available Features Command Response


```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.TiX.Can",
"member": "GetAllFeatures",
"object": "/Can",
"result": {
      "p1": [
         {
            "Feature": "hxgn",
            "Interface": [
               selected_interface
            ]
         },
         {
            "Feature": "isobus",
            "Interface": [
               selected_interface
            ]
```

```
        },
        {
            "Feature": "planting_head",
            "Interface": [
                selected_interface
            ]
        },
        {
            "Feature": "isobus_vt_tc",
            "Interface": [
                selected_interface
            ]
        },
        {
            "Feature": "steer_ready",
            "Interface": [
                selected_interface
            ]
        }
    ]
}
```

Result Fields	Description
selected_interface (string array)	Indicates a list of can interfaces. For virtual interfaces, it will return <IP>:<PORT>.

3.11.3 Set CAN bitrate value

Sets the physical interface bitrate values.

Set CAN bitrate value Command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Can",
"interface":"br.com.arvus.TiX.Can",
"member":"SetBitRate",
"params": [
    {
        "type" : "string",
        "value" : "interface_name"
    },
    {
        "type" : "int32",
        "value" : port
```

```
    }
]
}
```

Set CAN bitrate value Command Response

```
{
    "event": "method_result",
    "id": 1455198439,
    "object" : "/Can",
    "interface" : "br.com.arvus.TiX.Can",
    "member" : "SetBitRate",
    "result" : "{
        \“p1\” : result
    }"
}
```

Result Fields	Description
p1 - uint32	• 0 – Success. • 1 – Fail. The bitrate value is incorrect, or the interface chosen is not a physical interface, or this interface is used by hxag or isobus_vt_tc.

3.11.4 Set a feature as None

Set a feature to use no CAN.

Set a feature as None Command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Can",
"interface":"br.com.arvus.TiX.Can",
"member":"SetDummyFeature",
"params": [
    {
        "type" : "uint32",
        "value" : feature_option
    }
]
}
```

Result Fields	Description
feature_option	Indicates the features. 1. HxGN Ag bus

	2. Monitoring 3. ISOBUS VT/TC 4. Steer Ready

Set a feature as None Command Response

```
{
   "event": "method_result",
   "id": 1455198439,
   "object":"/Can",
   "interface":"br.com.arvus.TiX.Can",
   "member" : "SetDummyFeature",
   "result" : "{
      \“p1\” : result
   }"
}
```

Result Fields	Description
p1 - uint32	• 0 – Success. • 1 – Fail. Incorrect feature chosen.

3.12.5 Set a physical interface to a feature

Set a feature to use a chosen physical CAN interface.

Set a physical interface Command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Can",
"interface":"br.com.arvus.TiX.Can",
"member":"SetPhysicalFeature",
"params": [
   {
      "type" : "uint32",
      "value" : feature_option
   },
   {
      "type" : "string array",
      "value" :  [physical_interface]
   }
]
}
```

Result Fields	Description
feature_option	Indicates the features.

	5. HxGN Ag bus 6. Monitoring 7. ISOBUS VT/TC 8. Steer Ready
physical_interface	The string name of the physical interfaces as a list.

Set a physical interface Command Response

```
{
   "event": "method_result",
   "id": 1455198439,
   "object":"/Can",
   "interface":"br.com.arvus.TiX.Can",
   "member" : "SetPhysicalFeature",
   "result" : "{
      \“p1\” : result
   }"
}
```

Result Fields	Description
p1 - uint32	• 0 – Success. • 1 – Fail. Incorrect feature chosen or is not a valid physical interface.

3.12.6 Set a virtual interface to a feature

Set a feature to use a chosen virtual interface.

Set a virtual interface Command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Can",
"interface":"br.com.arvus.TiX.Can",
"member":"SetVirtualFeature",
"params": [
   {
      "type" : "uint32",
      "value" : feature_option
   },
   {
      "type" : "string",
      "value" : "interface_ip"
   },
   {
      "type" : "int32",
      "value" : interface_port
```

```
    }
]
}
```

Result Fields	Description
feature_option	Indicates the features. 9. HxGN Ag bus 10. Monitoring 11. ISOBUS VT/TC 12. Steer Ready
interface_ip	The virtual interface IP address.
interface_port	The virtual interface port.

Set a virtual interface Command Response

```
{
   "event": "method_result",
   "id": 1455198439,
   "object":"/Can",
   "interface":"br.com.arvus.TiX.Can",
   "member" : "SetVirtualFeature",
   "result" : "{
      \“p1\” : result
   }"
}
```

Result Fields	Description
p1 - uint32	• 0 – Success. • 1 – Fail. Incorrect feature chosen or is not a valid virtual interface.

3.12.7 Set internal CAN terminator value

Set the physical interface internal terminator values.

Set CAN bitrate value Command

```
{
"event":"call_method",
"id":1455198438,
"object":"/Can",
"interface":"br.com.arvus.TiX.Can",
"member":"SetTerminator",
"params": [
      {
       "type" : "boolean",
```

```
        "value" : terminator_value
    },
    {
        "type" : "string",
        "value" : "interface_name"
    }
]
}
```

Set CAN bitrate value Command Response

```
{
    "event": "method_result",
    "id": 1455198439,
    "object" : "/Can",
    "interface" : "br.com.arvus.TiX.Can",
    "member" : " SetTerminator ",
    "result" : "{
        \“p1\” : result
    }"
}
```

Result Fields	Description
p1 - uint32	• 0 – Success. • 1 – Fail. The interface chosen is not a physical interface, or this interface is used by hxag or isobus_vt_tc.

3.12 Remote Access Control

The Remote Access Control interface has methods to obtain the access code and allow touching the screen, as well as signals to obtain the requests received.

3.12.1 Get current access code

It returns the current access code for remote access.

Get access code Command

```
{
    "event": "call_method",
    "id": 1455198438,
    "interface": "br.com.arvus.Wanda.WandaAuthDbus",
    "member": "GetAccessCode",
    "object": "/WandaAuthDbus"
```

```
}
```

Get access code Command Response

```
{
   "event": "method_result",
   "id": 1455198438,
   "interface": "br.com.arvus.Wanda.WandaAuthDbus",
   "member": "GetAccessCode",
   "object": "/WandaAuthDbus",
   "result" : "{
      \“p1\” : "000000+aaaaa”
   }"
]}
```

Result Fields	Description
p1 - string	• String with the current access code.

3.12.2 Set remote access operation permission

This allows the screen to be clicked when accessing remotely.

Set remote access operation permission Command

```
{
   "event": "call_method",
   "id": 1455198438,
   "interface": "br.com.arvus.Wanda.WandaAuthDbus",
   "member": "SetOperationPermission",
   "object": "/WandaAuthDbus",
   "params": [
      {
         "type": "boolean",
         "value": permission
      }
   ]
}
```

Field	Description
permission	Enable or disable.

Set remote access operation permission Command Response

```
{
"event": "method_result",
"id": 1455198438,
"interface": "br.com.arvus.Wanda.WandaAuthDbus",
"member": "SetOperationPermission",
"object": "/WandaAuthDbus",
"result": "{}"
}
```


3.12.3 Signal Streaming Signal

This signal returns when there is a request.

To Register

{
"event": "subscribe",
"id": 1455198438,
"interface": "br.com.arvus.Wanda.WandaAuthDbus",
"object": "/WandaAuthDbus",
"member": “SignalStreaming”
 	
}

Register Response

{
"event": "subscribe_result",
"id": 1455198438,
"interface": "br.com.arvus.Wanda.WandaAuthDbus",
"member": " SignalStreaming",
"object": "/WandaAuthDbus",
"result": 0
}

Periodic Message

```
{
   "event": "signal",
   "id": 1706556906255,
   "interface": "br.com.arvus.Wanda.WandaAuthDbus",
   "member": "SignalStreaming",
   "object": "/WandaAuthDbus",
   "result":"{
      \"p1\": is_viewing,
      \"p2\": is_controlling
   }"
}
```

Result Fields	Description
p1 - is_viewing (boolean)	Indicates if a remote user is viewing the Core Box.
p2 - is_controlling (boolean)	Indicates if a remote user is controlling the Core Box.

3.12.4 Signal Request Operation Permission

Register to receive messages when a request for operation permission arrives.

To Register

```
{
"event": "subscribe",
"id": 1455198438,
"interface": "br.com.arvus.Wanda.WandaAuthDbus",
"member": "SignalRequestOperationPermission",
"object": "/WandaAuthDbus"
}
```

Register Response

```
{
"event": "subscribe_result",
"id": 1455198438,
"interface": "br.com.arvus.Wanda.WandaAuthDbus",
"member": "SignalRequestOperationPermission",
"object": "/WandaAuthDbus",
"result": 0
}
```

Periodic Message

```
{
   "event": "signal",
   "id": 1706556906255,
   "interface": "br.com.arvus.Wanda.WandaAuthDbus",
   "member": "SignalRequestOperationPermission",
   "object": "/WandaAuthDbus",
   "result":"{
      \"p1\": request_timeout_s,
      \"p2\": permission_timeout_s
   }"

}
```

Result Fields	Description
p1 - request_timeout_s (int)	Request timeout (seconds)
p2 - permission_timeout_s (int)	Permission timeout (seconds)

3.12.5 Signal Cancel Request Operation Permission

Register to receive messages when a cancel request arrives.

To Register

```
{
"event": "subscribe",
"id": 1455198438,
"interface": "br.com.arvus.Wanda.WandaAuthDbus",
"member": "SignalCancelRequestOperationPermission",
"object": "/WandaAuthDbus"
}
```

Register Response

```
{
"event": "subscribe_result",
"id": 1455198438,
"interface": "br.com.arvus.Wanda.WandaAuthDbus",
"member": "SignalCancelRequestOperationPermission",
"object": "/WandaAuthDbus",
"result": 0
}
```

Periodic Message

```
{
"event": "signal",
"id": 1706556906255,
"interface": "br.com.arvus.Wanda.WandaAuthDbus",
"member": "SignalCancelRequestOperationPermission",
"object": "/WandaAuthDbus",
"result": "{}"
}
```


4 Introspection

Calling the introspect method without specifying an object will return a list of all available objects. The result is a string that contains a JSON object.

Introspect


```
{
  "event":"call_method",
  "id":1455198438,
  "object":"/",
  "interface":"Introspectable",
  "member":"Introspect"
}
```


Introspect Reply


```
{
  "event":"method_result",
  "id":1455198438,
  "object":"/",
  "interface":"Introspectable",
  "member":"Introspect",
  "result":"result string"
}
```


Result String


```
{
   "objects":[
      "GNSS",
      "Guidance",
```

```
    "Implement",
    "System",
    "Steering",
    "Vehicle"
  ]
}
```

To see the capabilities of the GNSS object, we again call the introspect method:

Introspect

```
{
  "event":"call_method",
  "id":1455198438,
  "object":"/GNSS",
  "interface":"Introspectable",
  "member":"Introspect"
}
```

Introspect Reply

```
{
  "event":"method_result",
  "id":1455198438,
  "object":"/GNSS",
  "interface":"Introspectable",
  "member":"Introspect",
  "result":"result string"
}
```

Result String

```
{
  "br.com.arvus.TiX.GNSS":{
    "signals":{
      "NewGNSSData":[
        {
          "name":"longitude",
          "type":"double"
        },
        {
          "name":"latitude",
          "type":"double"
        },
        {
```

```
            "name":"altitude",
            "type":"double"
          },
          {
            "name":"speed_ms",
            "type":"double"
          },
          {
            "name":"direction",
            "type":"double"
          },
          {
            "name":"satellites",
            "type":"int32"
          },
          {
            "name":"is_synced",
            "type":"boolean"
          },
          {
            "name":"epoch",
            "type":"int64"
          }
        ]
      },
      "methods":{
      }
    }
}
```


The GNSS object contains 1 interface br.com.arvus.TiX.GNSS, this interface has no methods and 1 signal NewGNSSData. The NewGNSSData signal has a list of parameters and their respective types.

Document History

Version	Date	Who	Comments
1.0.5	31 Jan 2024	Alan Tabata	Extend CAN settings for multiple interfaces (section 3.13.2 and 3.13.5)
1.0.6	15 April 2024	Rafael Berto	Added Steer ready documentation
1.0.7	24 May 2024	Alan Valmorbida	Fix Calibration_step_codes of SteeringCalibration signal Add communication error return to steering calibration method
1.0.8	24 May 2024	Alan Tabata	Added explicit chapter for each message. Update summary accordingly
1.0.9	13 June 2024	Alan Tabata	Fixed chapter enumeration and title; added missing items to system Get ParamCode table (section 3.1.2)
1.0.10	25 June 2024	Alan Tabata	Fixed parameters guidance type on Guidance update Signal (section 3.6.2)
1.0.11	22 Aug 2024	Mathias Nichele	Added two new parameters to steering Enum and minor doc fixes
1.0.12	27 Aug 2024	Diesson Allebrandt	Changed the return and argument parameters when setting the antenna offset (from uint32 to double).
1.1.0	7-Feb-2025	Hugo Fagundes	1. Deprecated method 3.1.1, no longer used. 2. AP default password updated at 3.8.2.2 2. GNSS receivers update changes 3. ALIGN status messages added 4. Added local guidance line creation messages 5. General improvements in descriptions