Cummins Inc.

Next Gen Data Ingestion Interface

SDK – Specifications & Requirements

Full Document Name:
Next Gen Data Ingestion Interface (v1.1.0)

Last Update:
January 29, 2019

Parent Document:
None

I. TABLE OF CONTENTS

II. PURPOSE

This document explains the operations of the Next Gen Data Ingestion interface, (NGDI) providing both Specifications and Requirements for its implementation.

A. Overview

The Next Gen Data Ingestion interface should be used by telematics providers who need to:

1. Send engine/component data to Cummins Connected Solutions for the purposes of communicating equipment fault and/or parameter data periodically or upon the occurrence of a fault or other defined event.
2. Receive data collection definition commands from Cummins Connected Solutions for the purposes of configuring engine/component data collection.
3. Receive data collection control commands from Cummins Connected Solutions for the purposes of activating/deactivating engine/component data collection.
4. Negotiate encryption schemes with Cummins Connected Solutions.

B. Terminology

In the context of this document, there are notable terms that have specific, special meanings:

- **Provider**
  Telematics and real-time data integration solution providers.

- **Customer**
  The end user, or whomever is delegated to represent the customer.

- **Fleet Manager**
  A type of customer role; one who manages a fleet of vehicles.

- **Operator**
  A type of customer role; one who operates a vehicle in a fleet.

- **Calibration**
  The complete file to be loaded into the ECM, which includes software code, factory-set parameters, and tool-adjustable parameters (also known as “Trims”).

- **Tool-adjustable Parameters**
  ECM software parameters that can be adjusted using a service tool (in this document, a telematics device acting as a service tool).

- **Buffering**
  Transferring the new calibration or parameter write package from telematics device to the ECM memory.

- **HMI**
  Human Machine Interface

- **UDS**
  Unified Diagnostic Services

III. SPECIFICATIONS

A. Next Gen Data Ingestion Interface Details

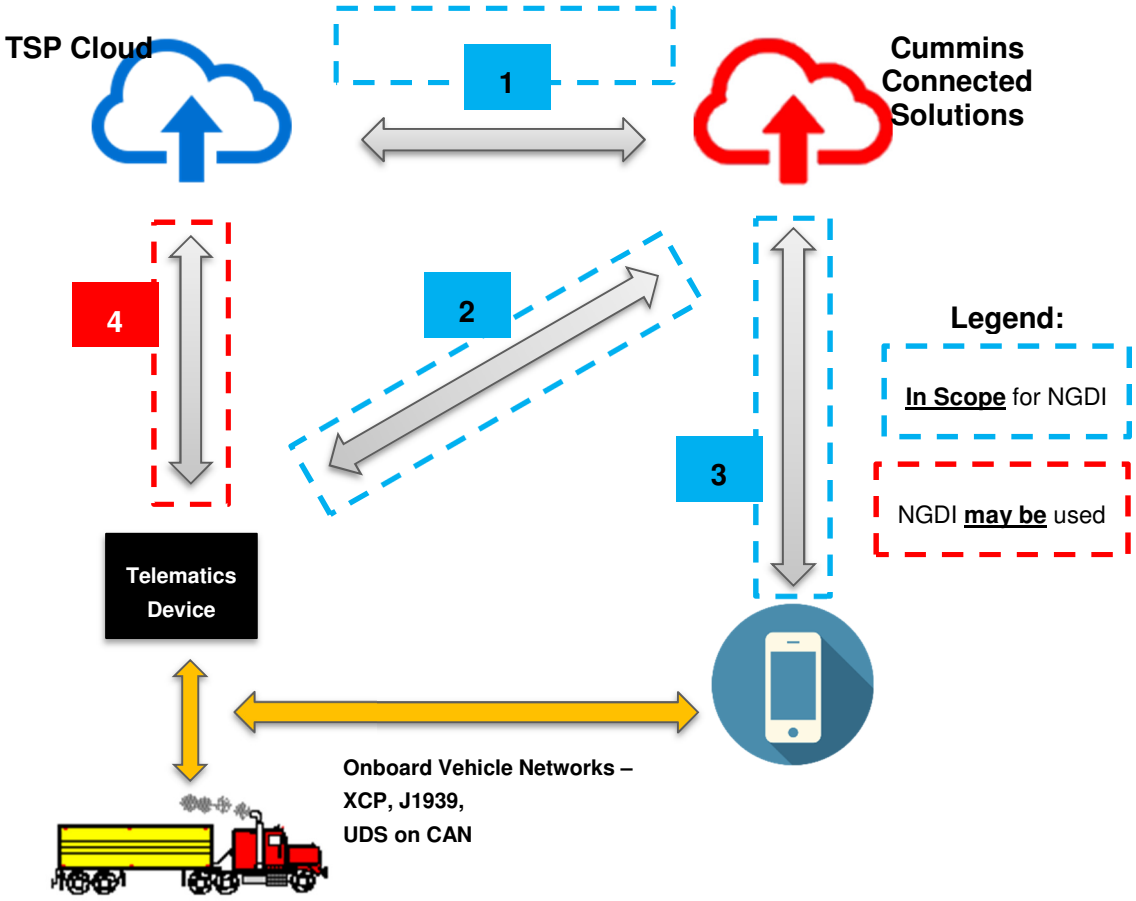
NGDI is a set of API methods that facilitate the exchange of data between clouds and between clouds and devices ("things").

NGDI uses JavaScript Object Notation (JSON) as the primary input/output format.

NGDI is hosted on a Secure Sockets Layer (SSL) connection. Messages POSTed from the TSP to Cummins Connected Solutions must include the parameter "AuthToken" (with the appropriate authentication token) as described in the Cummins Connected Solutions Integration SDK. Please see the Cummins Connected Solutions Integration SDK for full authentication details. Messages POSTed from Cummins Connected Solutions to the TSP endpoint must include TBD token / certificate as agreed by Cummins and the TSP.

Cummins ECMs are keyed by two primary values. Those are the Engine Serial Number (ESN) of the engine on which that ECM resides, and the Source Address on the J1939 network of the ECM. This is to protect for the case where there is a multi-ECM engine. It is highly recommended that the telematics provider take these two values and properly send and receive data to/from this ECM through their telematics solution by mapping it to a telematics device within their system.

The diagram below illustrates a high-level representation of the overall system and application of NGDI:



The “In Scope for NGDI” (blue dotted boxed) arrows represent data exchange between Cummins Connected Solutions and:

1. TSP cloud
2. TSP telematics devices
3. “BYOD” telematics devices, such as a cell phone

NGDI uses a common set of API methods for all three of these paths (though some fields are used exclusively for cloud or device).

Data exchange between TSP cloud and TSP telematics devices (path #4 in the diagram) is, of course, at the TSP’s discretion. NGDI may be used for this path as well.

NOTE:
This document does not go into detail regarding the specifications and requirements for the “Onboard Vehicle Networks” (gold arrows in the diagram above). They are outside the scope of this document.

IV. REQUIREMENTS

The requirements that follow in this section must be satisfied by the parties concerned, in order to ensure proper system functionality. The minimum system implementation consists of “B. Data Collection Activation” (starting on page 21~~21~~) and “C. Data Collection” (starting on page 24~~25~~).

A. Data Collection Configuration

The first group of API methods are used to **configure** data collection. Cummins uses these methods to POST messages requesting the device to set up and store a data collection configuration for future use. The device must respond to the request with “accepted”, “modified”, or “rejected”, depending on its capability to implement the request (more details below). Cummins may also POST messages to the TSP cloud. The TSP cloud must store the data collection configuration and send it to the devices in its control. It must also respond to the request on behalf of the devices in its control (more details below).

There are six data collection configuration API methods, each of which are fully explained below:

1. /defineDataContentSpec
2. /defineDataSamplingSpec
3. /defineSimpleEvent
4. /defineCompositeEvent
5. /defineDataSamplingConfig
6. /forgetDefinition

1. Data Content Specifications

A Data Content Specification is a list of data parameters to be sampled from various sources/ECUs, to which the telematics device is connected. Cummins sends /defineDataContentSpec messages to define new Data Content Specifications to be saved in the telematics device. Each message includes a unique SpecID to identify the new Data Content Specification, along with definition of the data parameters to be sampled. The sampled data parameters are divided into Single Sample and All Sample Parameters. Each of these groups is further divided into Telematics Device Parameters and Equipment Parameters (which include Fault Codes).

Destination Devices

The telematics device(s) that are being requested to save the Data Content Specification are called “destination devices.” The destination devices are defined as follows:

1.1 DestinationIDType: In recognition that the telematics device function may be contained in other devices, and also that equipment and engines/components are identified in different ways, the DestinationIDType allows for various ways of identifying the destination devices. Choices are: EquipmentID, VIN, ComponentSerialNumber, UnitNumber, TelematicsDeviceID, GatewayID, and BrokerID.

1.2 DestinationIDs: This is an array of DestinationIDs (of the type specified in DestinationIDType) that are being requested to save the Data Content Specification. Additionally, the following “special” DestinationID is available:

1.2.1 When sending the request to the TSP cloud (not telematics devices), Cummins may send a single-element array with the string “all” to request ALL destination devices to save the Data Content Specification.

Single Sample vs. All Sample Parameters

1.3 Single Sample Parameters must be sampled and included only once per data set (see section 2.1.1 below for data set details). Typically, parameters that are static or seldom change would be specified as Single Sample Parameters.

1.4 All Sample Parameters must be sampled and included in each sample in a data set (see section 2.1.1 below for data set details). Typically, dynamic parameters would be specified as All Sample Parameters.

Telematics Device Parameters

1.5 Telematics Device Parameters are generated from the telematics device/cloud, rather than being sampled from the sources/ECUs on the equipment. Examples are telematics device serial number, wireless signal strength, or GPS parameters (if they are generated by the telematics device instead of being read from some other source).

1.6 Each Telematics Device Parameter is defined by a parameter name string.

Equipment Parameters

1.7 Equipment Parameters are sampled from various sources/ECUs on the equipment. Examples are engine speed (from an engine ECU), fuel pressure (from a fuel system ECU), alternator voltage (from a generator ECU), etc.

1.8 Each Equipment Parameter is defined by the following attributes:

- 1.8.1 Protocol: the datalink protocol from which the parameter should be sampled (example: J1939). A list of supported protocols is available in the Appendix.
- 1.8.2 NetworkID: the ID of the network from which the parameter should be sampled (example: CAN1). This is needed for equipment that has more than one network.
- 1.8.3 DeviceID: the ID of the device/ECU from which the parameter should be sampled. For J1939, this would be the source address.
- 1.8.4 Parameters: the IDs of the parameters as defined by the Protocol. For J1939, this would be the SPN. In the case of Manufacturer Defined PGNs, the PGN is used instead of the SPN, preceded by 'P'. Example: P65479. Additionally, the following "special" ParameterIDs are available:
  - 1.8.4.1 ActiveFaultCodes
  - 1.8.4.2 InactiveFaultCodes
  - 1.8.4.3 PendingFaultCodes

    NOTE: If included in the Data Content Specification, these "special" parameters must be included in the data sent to the Cummins cloud, as described in the "Data Collection" section below.

1.9 Engineering Access Level (EAL) is a special way of accessing Equipment Parameters from the ECU. EAL is defined as follows:

- 1.9.1 Protocol: the protocol will be set to "EAL". This means that the telematics device must use the ECU's UDS interface to establish an EAL session with the ECU to get the data. More details are provided in the Data Collection section below.
- 1.9.2 NetworkID: same as above
- 1.9.3 DeviceID: same as above
- 1.9.4 Parameters: for EAL, the "ParameterIDs" contain the information the telematics device needs to dynamically define a data grouping on the ECU. Each "ParameterID" consists of four hex characters (representing the dynamically defined data identifier or "DDID"), followed by a colon (:) delimiter, followed by a hex string (representing information about the data to be

included in the data grouping).

1.10 Some ECUs store engine operational data collected over time. This "Trip Data" is another special way of accessing Equipment Parameters from the ECU. Trip Data is defined as follows:
- 1.10.1 Protocol: the protocol will be set to "TripData". This means that the telematics device must use the ECU's UDS interface to establish an EAL session with the ECU to retrieve the data. More details are provided in the Data Collection section below.
- 1.10.2 NetworkID: same as above
- 1.10.3 DeviceID: same as above
- 1.10.4 Parameters: for Trip Data, the "Parameters" section contains a single "ParameterID", either "Partial" (for partial data extraction) or "Full" (for full data extraction). More details are provided in the Data Collection section below.

Responses to Data Content Specification Define Messages

1.11 The device must respond to the define message with "accepted", "modified", or "rejected".
- 1.11.1 An "accepted" response means that the Data Content Specification is supported by the device(s) and has been or will be saved on the device(s) for future activation.
- 1.11.2 A "modified" response means that the Data Content Specification is supported by the device(s) only after excluding certain Telematics Device Parameters and/or Equipment Parameters. For example, if the Data Content Specification includes "Altitude" as a parameter, but the device does not support it, the device must send a "modified" response. A "modified" response means that the modified Data Content Specification has been or will be saved on the device(s) for future activation. A "modified" response must include information about the excluded parameter(s) as follows:
  - 1.11.2.1 ReasonCode = 500 (Parameter(s) not supported)
  - 1.11.2.2 Parameters = List of parameter ID(s) that were excluded
- 1.11.3 A "rejected" response means that the Data Content Specification is not supported by the device(s) and will not be saved on the device(s). For example, if the device does not support the Protocol included in the Data Content Specification, the device must send a "rejected" response. A "rejected" response must include information about the reason for rejection as follows:
  - 1.11.3.1 ReasonCode = 501 (Setting(s) not supported), or 503 (Destination(s) do not exist)

1.11.3.2 Parameters = List of setting(s) that are not supported by the destination device(s), or destination(s) that do not exist

Predefined Data Content Specifications

In addition to dynamically defined Data Content Specifications (using /defineDataContentSpec messages), a set of pre-defined Data Content Specifications will be developed, each with its own unique ID.

1.12 All relevant pre-defined Data Content Specifications must be loaded into devices during initial programming or provisioning so they can be activated later without the need for /defineDataContentSpec messages.
1.13 TSPs are strongly encouraged to support the /defineDataContentSpec API method, but it is not required so long as the TSP is willing to be limited to only pre-defined Data Content Specifications.
1.14 A list of pre-defined Data Content Specifications is provided in the Appendix.

2. Data Sampling Specifications

A Data Sampling Specification is a group of settings that define how data parameters should be sampled from various sources/ECUs, to which the telematics device is connected, and then transmitted to the Cummins cloud. Cummins sends /defineDataSamplingSpec messages to define new Data Sampling Specifications to be saved in the telematics device. Each message includes a unique SpecID to identify the new Data Sampling Specification, along with the settings defining how the data parameters are to be sampled, as follows:

Destination Devices

The telematics device(s) that are being requested to save the Data Sampling Specification are called “destination devices.” The destination devices are defined in the same manner as the define Data Content Specification messages.

Data Sampling and Transmission

There are three settings that control data sampling and transmission: SamplingPeriod, MaxTransmitPeriod, and MaxSetSize:

2.1 "samplingPeriod" is simply the time between samples. All the specified data parameters should be sampled simultaneously every sampling period. If any of the specified data parameters are sent "on request," the device must send the request in time to include the parameter values in the sample.

2.1.1 For Trip Data (see section 1.10 above), samplingPeriod is the time between data retrieval sequences, **not** the time between individual data blocks retrieved from the ECU. More details are provided in the Data Collection section below.

2.2 "maxTransmitPeriod" and "maxSetSize" together specify when the sampled data should be transmitted. They configure the device to store the data samples until at least one of these Max settings is reached, at which time the device would transmit the "set" of data samples. For example, if SamplingPeriod is set to 5 sec, MaxTransmitPeriod is set to 120 sec, and MaxSetSize is set to 12, then MaxSetSize would typically be the limiting factor causing the device to transmit the data set (every 60 sec as 12 samples are collected). However, if some samples were lost or sampling was slowed down, the MaxTransmitPeriod would serve as a backup factor to ensure the device would transmit the data set after 120 sec even if the 12-sample quota were not reached. ("maxSetSize" = 1 means that each sample should be transmitted immediately.) Note: Any sample containing only Single Sample Parameters should NOT be counted toward the MaxSetSize. See section 1.3 above.

2.2.1 For Trip Data (see section 1.10 above), maxSetSize is defined differently. It represents the number of individual data blocks (retrieved from the ECU) the telematics device should store before beginning to overwrite data blocks, starting with the oldest (a "FIFO rolling buffer").

2.2.1.1 If the MaxSetSize represents more storage space than the telematics device has allocated, the telematics device should store all of the data blocks from the ECU until the allocated storage space becomes full, at which time the oldest data must be overwritten first (a "FIFO rolling buffer").

2.2.1.2 Regardless of the MaxSetSize setting, the telematics device shall retrieve and transmit **all** of the data blocks in the data retrieval sequence. More details are provided in the Data Collection section below.

Trigger Types

There are two types of sampling triggers: Periodic and EventDriven:

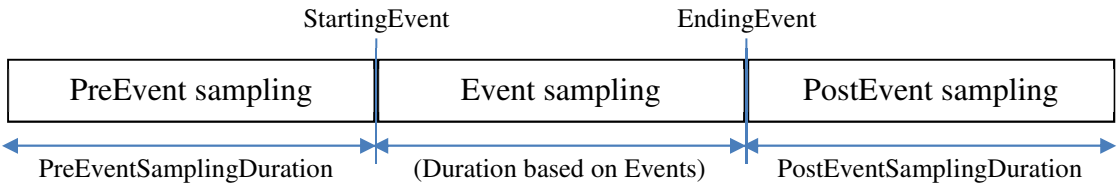
2.3 "periodic" simply means that the device would be configured to sample and transmit data (as specified in 2.1 and 2.1.1), indefinitely.

2.4 “eventDriven” means that the device would be configured to sample and transmit data (as specified in 2.1 and 2.1.1) only when a specified Event is detected. This is called an “EventDriven data sampling sequence.” See section 3: “Events” for more details about different types of Events that can be used as EventDriven triggers.

EventDriven Data Sampling Sequence

There are three phases in an EventDriven data sampling sequence:

1) PreEvent sampling phase (optional)
2) Event sampling phase
3) PostEvent sampling phase (optional)



2.5 The device would sample and transmit data as specified in 2.1 and 2.1.1 for all three phases of the EventDriven data sampling sequence, including the following considerations:

2.5.1 MaxSetSize = 0 means that the entire EventDriven data sampling sequence would be accumulated before transmitting.

2.5.2 If the EventDriven data sampling sequence exceeds the MaxTransmitPeriod, the device would transmit the set of data samples accumulated to that point. It would then start accumulating a new data set to be transmitted when the EventDriven data sampling sequence ends, or when the MaxTransmitPeriod is exceeded (again), whichever occurs first. It would continue in this manner until the EventDriven data sampling sequence ends.

2.6 When the device detects the StartingEvent, it would start an EventDriven data sampling sequence, including the following considerations:

2.6.1 If a PreEventSamplingDuration is specified, the device would include samples for PreEventSamplingDuration time **before** the StartingEvent occurs. (This would typically be accomplished by using a "rolling buffer" of data samples that would be saved for transmission upon detection of the StartingEvent.)

2.6.2 If the number of data samples accumulated during the PreEventSamplingDuration time exceeds the MaxSetSize, or if the PreEventSamplingDuration time exceeds the MaxTransmitPeriod, the device would transmit (as a data set) all

of the data samples accumulated immediately upon detection of the StartingEvent. It would then continue sampling and transmitting data as described in 2.5.

2.7 When the device detects the EndingEvent, it would continue sampling and transmitting data (as described in 2.5) for PostEventSamplingDuration time before ending the EventDriven data sampling sequence and transmitting all of the remaining (un-transmitted) accumulated data samples.

2.7.1 Specifying an EndingEvent is optional. If no EndingEvent is specified, the StartingEvent also serves as the EndingEvent, meaning that the EventDriven data sampling sequence consists of a single sample at the time of the StartingEvent, plus any PreEvent or PostEvent samples (as described above).

Responses to Data Sampling Specification Define Messages

2.8 The device must respond to the define message with “accepted”, “modified”, or “rejected”.

2.8.1 An “accepted” response means that the Data Sampling Specification is supported by the device(s) and has been or will be saved on the device(s) for future activation.

2.8.2 A “modified” response means that the Data Sampling Specification is supported by the device(s) only after excluding certain settings. The only settings that can be excluded are PreEventSamplingDuration and PostEventSamplingDuration. If the Data Sampling Specification includes one or both of these settings, but the device either does not support the specified function(s) or does not support the specified time period(s), the device must send a “modified” response. A “modified” response means that the modified Data Sampling Specification has been or will be saved on the device(s) for future activation. A “modified” response must include information about the excluded setting(s) as follows:

2.8.2.1 ReasonCode = 501 (Setting(s) not supported), or 502 (Function(s) not supported)

2.8.2.2 Parameters = “preEventSamplingDuration” and/or “postEventSamplingDuration”

2.8.3 A “rejected” response means that the Data Sampling Specification is not supported by the device(s) and will not be saved on the device(s). If the device does not support any setting other than those listed in 2.8.2, the device must send a “rejected” response. A “rejected” response must include information about the reason for rejection as follows:

2.8.3.1 ReasonCode = 501 (Setting(s) not supported), or 502 (Function(s) not supported), or

503 (Destination(s) do not exist)

2.8.3.2 Parameters = List of setting(s)/function(s) that are not supported by the destination device(s), or destination(s) that do not exist

Predefined Data Sampling Specifications

In addition to dynamically defined Data Sampling Specifications (using /defineDataSamplingSpec messages), a set of pre-defined Data Sampling Specifications will be developed, each with its own unique ID.

2.9 All relevant pre-defined Data Sampling Specifications must be loaded into devices during initial programming or provisioning so they can be activated later without the need for /defineDataSamplingSpec messages.

2.10 TSPs are strongly encouraged to support the /defineDataSamplingSpec API method, but it is not required so long as the TSP is willing to be limited to only pre-defined Data Sampling Specifications.

2.11 A list of pre-defined Data Sampling Specifications is provided in the Appendix.

3. Events

An Event is a defined occurrence that can be used to trigger an EventDriven data sampling sequence. NGDI enables definition of “Simple” Events and “Composite” Events (details below). Cummins sends /defineSimpleEvent and /defineCompositeEvent messages to define new Events to be saved in the telematics device. Each message includes a unique EventID to identify the new Event, along with the settings defining the Event, as follows:

Destination Devices

The telematics device(s) that are being requested to save the Event are called “destination devices.” The destination devices are defined in the same manner as the other data collection configuration messages.

Simple Events – Fault-based

Simple Events are classified into one of four EventTypes: ActiveFault, InactiveFault, PendingFault, and ParameterCompare. The first three EventTypes use the following settings to define the Simple Event:

3.1 ActiveFault-, InactiveFault-, and PendingFault-type Simple Events use a single argument to define which fault(s) to monitor, defined by the following attributes:
  3.1.1 Protocol: the datalink protocol to be monitored for ActiveFault(s) (example: J1939). A list of supported protocols is available in the Appendix. There is also a special Protocol setting called “allProtocols”.
  3.1.2 NetworkID: the ID of the network to be monitored for ActiveFault(s) (example: CAN1). This is needed for equipment that has more than one network. There is also a special NetworkID setting called “allNetworks”.
  3.1.3 DeviceID: the ID of the device/ECU to be monitored for ActiveFault(s). For J1939, this would be the source address. There is also a special DeviceID setting called “allDevices”.
  3.1.4 ParameterID: the ID of the fault parameter to be monitored as defined by the Protocol. For J1939, this would be the SPN. There is also a special ParameterID setting called “allParameters”.
  3.1.5 FailureModeID: the ID of the failure mode to be monitored as defined by the Protocol. For J1939, this would be the FMI. There is also a special FailureModeID setting called “allFMIs”.
3.2 Some Protocols may have a single identifier for a fault, rather than a separate ParameterID and FailureModeID. If there is only a single identifier, it should be sent in the ParameterID field and the FailureModeID field should be omitted.
3.3 Any or all of the special “all___” settings can be used to define a Simple Event based on multiple faults.
3.4 The Simple Event occurs whenever the specified fault(s) become active, pending, or inactive (based on the EventType).

Simple Events – Parameter Compare

The fourth EventType, ParameterCompare, uses the following settings to define the Simple Event:

3.5 ParameterCompare-type Simple Events use two arguments and an operator to define a parameter and a point of comparison. The first argument is similar to (but not the same as) the argument used for fault-based Simple Events, and is defined by the following attributes:
  3.5.1 Protocol: the datalink protocol of the parameter to be monitored (example: J1939). A list of supported protocols is available in the Appendix.
  3.5.2 NetworkID: the ID of the network of the parameter to be monitored (example: CAN1). This is needed for equipment that has more than one network.

3.5.3 DeviceID: the ID of the device/ECU of the parameter to be monitored. For J1939, this would be the source address.
3.5.4 ParameterID: the ID of the parameter to be monitored as defined by the Protocol. For J1939, this would be the SPN.

3.6 Note: The special “all___” settings **cannot** be used for ParameterCompare-type Simple Events.

3.7 The operator determines how the first argument is to be compared to the second argument. Options are equal, not equal, greater than, greater/equal, less than, less/equal.

3.8 The second argument can be one of two types: Parameter or Constant. A Parameter-type argument is defined by the same attributes listed in sections 3.5.1 through 3.5.4. A Constant-type argument is defined by the following attributes:
3.8.1 DataType: the type of data constant. May be “float”, “integer”, or “string”.
3.8.2 ArgumentValue: the value of the constant.

3.9 The Simple Event occurs whenever the first argument (parameter) meets the specified comparative relationship with the second argument (parameter or constant). For example, the Simple Event might be configured to occur when engine speed (first argument) is greater than (operator) 600 RPM (second argument).

3.10 Note: A "parameter change" event can be defined by using ParameterCompare type and setting the parameter 'not equal to' itself. This may require the telematics device to store a parameter value for change comparison.

Composite Events

A Composite Event is the logical combination of two or more already-defined Simple Events. An array is used to define the Composite Event, as follows:

3.11 The array must have at least two elements. Each element of the array contains two attributes: the EventID of a Simple Event, and a logical operator (AND, OR, or NONE).

3.12 The first element of the array contains the first Simple Event and the logical operator must be either AND or OR, which links the Simple Event to the second Simple Event (contained in the second element of the array).

3.13 Each subsequent element of the array contains another Simple Event, and a logical operator linking that Simple Event to the **next** element in the array. In this way, two, three, or more Simple Events can be logically linked together.

3.14 The last element of the array must have the logical operator set to NONE (since there is no next element to link to).

3.15 The logical linkage shall be performed in the order of the elements of the array. The first Simple Event shall be AND'd / OR'd with the second Simple Event, and the result shall be AND'd / OR'd with the third Simple Event, and so on. (AND shall **not** take precedence over OR).

Responses to Event Define Messages

3.16 The device must respond to the define message with "accepted" or "rejected".
- 3.16.1 An "accepted" response means that the Event (Simple or Composite) is supported by the device(s) and has been or will be saved on the device(s) for future use.
- 3.16.2 A "rejected" response means that the Event (Simple or Composite) is not supported by the device(s) and will not be saved on the device(s). A "rejected" response must include information about the reason for rejection as follows:
  - 3.16.2.1 ReasonCode = 501 (Setting(s) not supported), or 502 (Function(s) not supported), or 503 (Destination(s) do not exist)
  - 3.16.2.2 Parameters = List of setting(s)/function(s) that are not supported by the destination device(s), or destination(s) that do not exist
- 3.16.3 Note: A "modified" response shall not be sent in response to an Event Define Message.

Predefined Events

In addition to dynamically defined Events (using /defineSimpleEvent or /defineCompositeEvent messages), a set of pre-defined Events will be developed, each with its own unique ID.

3.17 All relevant pre-defined Events must be loaded into devices during initial programming or provisioning so they can be activated later without the need for /defineSimpleEvent or /defineCompositeEvent messages.

3.18 TSPs are strongly encouraged to support the /defineSimpleEvent and /defineCompositeEvent API methods, but it is not required so long as the TSP is willing to be limited to only pre-defined Events.

3.19 A list of pre-defined Events is provided in the Appendix.

4. Data Sampling Configurations

A Data Sampling Configuration is a pairing of one Data Content Specification (see section 1) with one Data Sampling Specification (see section 2). A Data Sampling Configuration therefore contains everything the telematics device needs to know to sample and transmit data parameters from various sources/ECUs to the Cummins cloud. Cummins sends /defineDataSamplingConfig messages to define new Data Sampling Configurations to be saved in the telematics device for future activation. Each message includes a unique ConfigID to identify the new Data Sampling Configuration, along with the Data Content and Data Sampling SpecIDs.

Destination Devices

The telematics device(s) that are being requested to save the Data Sampling Configuration are called “destination devices.” The destination devices are defined in the same manner as the other data collection configuration messages.

Responses to Data Sampling Configuration Define Messages

4.1 The device must respond to the define message with “accepted” or “rejected”.
- 4.1.1 An “accepted” response means that the Data Sampling Configuration is supported by the device(s) and has been or will be saved on the device(s) for future use.
- 4.1.2 A “rejected” response means that the Data Sampling Configuration is not supported by the device(s) and will not be saved on the device(s). For example, if the device does not support the Data Content Specification included in the Data Sampling Configuration, the device must send a “rejected” response. Even if the device supports the included Data Content and Data Sampling Specifications, it may not support the **combination** thereof (e.g. too many parameters at too fast a sampling/transmit rate). A “rejected” response must include information about the reason for rejection as follows:
  - 4.1.2.1 ReasonCode = 501 (Setting(s) not supported), or 503 (Destination(s) do not exist)
  - 4.1.2.2 Parameters = List of setting(s) that are not supported by the destination device(s), or destination(s) that do not exist

4.2 Note: A “modified” response shall not be sent in response to a Data Sampling Configuration Define Message.

Predefined Data Sampling Configurations

In addition to dynamically defined Data Sampling Configurations (using /defineDataSamplingConfig messages), a set of pre-defined Data Sampling Configurations will be developed, each with its own unique ID.

4.3 All relevant pre-defined Data Sampling Configurations must be loaded into devices during initial programming or provisioning so they can be activated later without the need for /defineDataSamplingConfig messages.
4.4 TSPs are strongly encouraged to support the /defineDataSamplingConfig API method, but it is not required so long as the TSP is willing to be limited to only pre-defined Data Sampling Configurations.
4.5 A list of pre-defined Data Sampling Configurations is provided in the Appendix.

5. Forgetting Data Collection Configurations

If for any reason a data collection configuration (Data Content Specification, Data Sampling Specification, Event, or Data Sampling Configuration) is no longer needed, Cummins may send a /forgetDefinition message to direct the device to delete the unneeded data collection configuration from memory. Each message includes a DefinitionType (may be DataContentSpec, DataSamplingSpec, Event, or DataSamplingConfig) and a DefinitionID (may be a SpecID, EventID, or ConfigID) to delete.

Destination Devices

The telematics device(s) that are being requested to forget the data collection configuration are called “destination devices.” The destination devices are defined in the same manner as the other data collection configuration messages.

Responses to Forget Data Collection Configuration Messages

5.1 The device must respond to the forget message with “accepted” or “rejected”.
  5.1.1 An “accepted” response means that the data collection configuration has been or will be deleted from the device(s).
  5.1.2 A “rejected” response means that the data collection configuration will not be deleted from the device(s). For example, if the device does not support deletion of data collection configurations, the device must send a “rejected” response. A “rejected” response must include information about the reason for rejection as follows:

5.1.2.1 ReasonCode = 501 (Setting(s) not supported), or
502 (Function(s) not supported), or
503 (Destination(s) do not exist)

5.1.2.2 Parameters = List of setting(s)/function(s) that are not supported by the destination device(s), or destination(s) that do not exist

5.2 Note: A “modified” response shall not be sent in response to a Forget Data Collection Configuration Message.

B. Data Collection Activation

The second group of API methods are used to **activate and deactivate** data collection. Cummins uses these methods to POST messages requesting that one or more Data Sampling Configurations (see section 4 above) be activated (start collecting data) or deactivated (stop collecting data) on a device. The device must respond to the request with “accepted” or “rejected”, depending on its capability to implement the request (more details below). Cummins may also POST messages to the TSP cloud. The TSP cloud must store the data collection activation/deactivation request and send it to the specified devices in its control. It must also respond to the request on behalf of the devices in its control (more details below).

There are two data collection activation API methods, each of which are fully explained below:

1. /activateDataCollection
2. /deactivateDataCollection

1. Activating Data Collection

Cummins sends /activateDataCollection messages to request one or more devices to start collecting data based on one or more Data Sampling Configurations previously saved on the device(s) or on the TSP cloud. Each message includes additional data collection settings. Any previously activated Data Sampling Configurations should continue unaffected. A detailed description follows:

Destination Devices

The telematics device(s) that are being requested to start collecting data are called “destination devices.” The destination devices are defined in the same manner as the data collection configuration messages.

Data Collection Settings

Other settings are included in the /activateDataCollection message. These settings further specify how the data should be collected, as follows:

1.1 DataSamplingConfigIDs: a list (array) of Data Sampling Configurations to be activated. Telematics devices must be able to have more than one Data Sampling Configuration active simultaneously.
1.2 ResumeMode: if set to “true”, the telematics device must automatically resume (reactivate) the DataSamplingConfigID(s) after the device is power cycled (powered off, then powered on).
1.3 DataPriority: indicates the priority of the Data Sampling Configuration(s) that are being activated. When multiple Data Sampling Configurations are active on a device (e.g. from previous /activateDataCollection messages), higher priority data should be sampled and transmitted first, followed by lower priority data. DataPriority may be High, Normal, or Low.
1.4 DataTransmitMethod: indicates how the sampled data should be transmitted for the Data Sampling Configuration(s) that are being activated. DataTransmitMethod may be Stream, File, or DelayedStream, defined as follows:
    1.4.1 Stream: the sampled data should be transmitted using the /sendSingleDataSet or /sendMultipleDataSets API methods, as described in the Data Collection section below. If no over-the-air connection is available, the sampled data is lost.
    1.4.2 File: the sampled data should be compiled into a file and uploaded, as described in the “NGDI File Upload Process” SDK document. See the Data Collection section below.
    1.4.3 DelayedStream: same as Stream mode, except that if no over-the-air connection is available, the sampled data should be stored and Streamed (as individual samples) when the connection is available.

    If the DataTransmitMethod is not specified, Stream mode is the default setting.
1.5 AutoActivateMode: when sending the request to the TSP cloud (not telematics devices), Cummins may include AutoActivateMode set to “true”. This indicates that the TSP cloud should automatically activate data collection (including all specified settings) on any newly discovered/activated/registered telematics devices that support the request. This may mean sending the necessary /define messages to the telematics devices before sending the /activateDataCollection message.

Responses to Activate Messages

1.6 The response consists of an ActionResponses array. Each element of the array has a Response, an optional array of Reasons, and a list of destination devices (using DestinationIDType and DestinationIDs as described in sections **Error! Reference source not found.** and **Error! Reference source not found. Error! Reference source not found.**).
- 1.6.1 For a response sent by a TSP cloud, the ActionResponses array may consist of multiple elements, each element representing a different group of destination devices with a different Response and/or Reasons.
- 1.6.2 For a response sent by a TSP cloud, the ActionResponses array may consist of a single element with the "special" DestinationID "all" to indicate that the Response and Reasons apply to ALL the destination devices.
- 1.6.3 For a response sent by a telematics device, the ActionResponses array would typically have a single element corresponding to the (single) destination device.
  - 1.6.3.1 (An example of an exception might be for a single telematics device connected to multiple components, and DestinationIDType = ComponentSerialNumber. In this case, the telematics device may need to send a multi-element ActionResponses array if the multiple components require different Responses and/or Reasons.)

1.7 The Response value must be set to "accepted" or "rejected".
- 1.7.1 An "accepted" response means that the Data Sampling Configuration(s) are supported by the destination device(s) and have been or will be activated on the destination device(s).
- 1.7.2 A "rejected" response means that the Data Sampling Configuration(s) will not be activated on the destination device(s). For example, if the destination device(s) do not support the listed Data Sampling Configuration(s), if they do not support any functions or settings in the message, or the DestinationID(s) are invalid, a "rejected" response must be sent. A "rejected" response must include information about the reason for rejection as follows:
  - 1.7.2.1 ReasonCode = 501 (Setting(s) not supported), or 502 (Function(s) not supported), or 503 (Destination(s) do not exist)
  - 1.7.2.2 Parameters = List of setting(s)/function(s) that are not supported by the destination device(s), or destination(s) that do not exist

2. Deactivating Data Collection

Cummins sends /deactivateDataCollection messages to request one or more devices to stop collecting data based on one or more Data Sampling Configurations previously activated on the device(s). Any previously activated Data Sampling Configurations that are not being deactivated should continue unaffected. A detailed description follows:

Destination Devices

The telematics device(s) that are being requested to stop collecting data are called “destination devices.” The destination devices are defined in the same manner as the activate data collection messages.

Data Collection Settings

The only setting included in the /deactivateDataCollection message is the DataSamplingConfigIDs: a list (array) of Data Sampling Configurations to be deactivated.

Responses to Deactivate Messages

2.1 The response consists of an ActionResponses array. The ActionResponses array is defined in the same manner as the activate data collection messages.
2.2 The Response value must be set to “accepted” or “rejected”.
- 2.2.1 An “accepted” response means that the Data Sampling Configuration(s) have been or will be deactivated on the destination device(s).
- 2.2.2 A “rejected” response means that the Data Sampling Configuration(s) will not be deactivated on the destination device(s). For example, if the destination device(s) do not have the listed Data Sampling Configuration(s) active or the DestinationID(s) are invalid, a “rejected” response must be sent. A “rejected” response must include information about the reason for rejection as follows:
  - 2.2.2.1 ReasonCode = 501 (Setting(s) not supported), or 502 (Function(s) not supported), or 503 (Destination(s) do not exist)
  - 2.2.2.2 Parameters = List of setting(s)/function(s) that are not supported by the destination device(s), or destination(s) that do not exist

C. Data Collection

The third group of API methods are used to **send** data to the Cummins cloud. Telematics devices and TSP clouds use these methods to stream data collected according to one or more activated Data Sampling Configurations.

There are two data collection API methods, each of which are fully explained below:

1. /sendSingleDataSet
2. /sendMultipleDataSets

There is also an alternative process for data collection, where instead of streaming data, telematics devices and TSP clouds can upload a data file to the Cummins cloud. This data file upload process is described in the "NGDI File Upload Process" SDK document. The data transmit method to be used is specified in the /activateDataCollection message.

In general, file upload is the preferred method of transmitting data to Cummins Connected Solutions. If the telematics device and/or TSP cloud is not capable of performing the NGDI File Upload Process, it should default to the DelayedStream method.

Depending on the type of data being collected, the telematics device may need to establish a communication session with the ECU and/or provide an access key to collect the data.

A summary of the data collection methods is shown in the following table:

Data Type	Data Collection Configuration	ECU Access	Data Transmit Method
Public	NGDI section A (e.g. J1939)	None	Stream / DelayedStream (NGDI section C.2 and C.3) OR NGDI File Upload Process
Proprietary	NGDI section A (e.g. UDS)	NGDI section C.1 (UDS)	Stream / DelayedStream (NGDI section C.2 and C.3) OR NGDI File Upload Process
Engineering	NGDI section A (EAL)	NGDI section C.1 (EAL)	Stream / DelayedStream (NGDI section C.2 and C.3) OR NGDI File Upload Process
Trip Data	NGDI section A (Trip Data)	NGDI section C.1 (EAL)	NGDI File Upload Process only

1. Establishing ECU Access

NOTE: This step is not necessary for public (broadcast) data.

UDS Default Session Access

When the Data Content Specification contains parameters with Protocol = UDS, the following steps must be followed:

1.1 Telematics device establishes a default session with the ECU.
1.2 Telematics device uses ReadDataByIdentifier service ($22) to read the parameters (that is, the UDS data identifiers) specified in the Data Content Specification, when prescribed in the Data Sampling Specification.

Engineering Access Level

When the Data Content Specification contains parameters with Protocol = EAL, the following steps must be followed:

1.3 Telematics device establishes a default session with the ECU.
1.4 If the telematics device does not have a valid Access Key for EAL, it must initiate a Learning Session per section 4.1 of the "Telematics Partner Specification" document.
  1.4.1 The Provider Telematics Device shall only initiate (or retry) the Learning Access Process (see the Telematics Partner Specification) when it has good over-the-air coverage. **The entire Learning Access process, from the time the ECM receives the Seed request, until the ECM receives the Session Key, must be completed in 5 minutes or less.** If the time is exceeded, the process shall be retried until it is completed within the time limit. In addition, no other UDS commands or operations (except "Tester Present") shall be sent to the ECM until the Learning Access process is complete.
  1.4.2 If the ECM sends "Negative Response Code" NRC-35 ("invalidKey (IK)"), the Provider Telematics device shall retry the Learning Access Process once. If the Learning Session still cannot be established, the Provider Telematics Device shall abort the process.
1.5 After the Learning Access Process is successfully executed (or if the telematics device bypassed the Learning Access Process because it already had a valid Access Key for EAL), the telematics device must initiate a telematics session with the ECU and request Engineering Access Level per section 4.5 of the "Telematics Partner Specification" document.
  1.5.1 If the ECU denies Engineering Access Level, then the telematics device shall abort the process.
  1.5.2 In case of an error in establishing the session or granting access, a "Negative Response Code" NRC-35 ("invalidKey

(IK)") will be sent by the ECU. In this case, the telematics device shall retry once. If the session still cannot be established, the telematics device shall abort the process.

1.6 After the Engineering Access Level is established, the telematics device shall define data group(s), as specified in the Data Content Specification, using the DynamicallyDefineDataIdentifier service per section 4.5 of the "Telematics Partner Specification" document.
- 1.6.1 The telematics device must define each data group separately.
- 1.6.2 The ECU shall normally return a positive response to each define command.
- 1.6.3 In case of an error in data group definition, a "Negative Response Code" (NRC) will be sent by the ECU. Refer to the below NRC table for details and actions to be taken.

1.7 After the Dynamically Defined Data Identifier(s) (DDIDs) are established, the telematics device shall request the data using either the ReadDataByIdentifier service or the ReadDataByPeriodicIdentifier service per section 4.5 of the "Telematics Partner Specification" document.
- 1.7.1 The telematics device shall use ReadDataByPeriodicIdentifier where applicable (e.g. a Data Sampling Specification with TriggerType = Periodic), to avoid excess datalink traffic from repeated ReadDataByIdentifier requests. "SendAtSlowRate" setting shall be used unless the Data Sampling Specification requires a faster rate.
- 1.7.2 The telematics device must request each Dynamically Defined Data Identifier (DDID) separately.
- 1.7.3 The ECU shall normally return a positive response to each request.
- 1.7.4 In case of an error in the data request, a "Negative Response Code" (NRC) will be sent by the ECU. Refer to the below NRC table for details and actions to be taken.
- 1.7.5 The ECU shall return all the parameter values in a given request as a single data block. Separate requests will return separate data blocks.
- 1.7.6 The telematics device shall pass the data to the Cummins cloud unchanged. Each sample shall be formatted: "<DDID>": "<Data_Block_Received>"

Trip Data

When the Data Content Specification contains parameters with Protocol = TripData, the following steps must be followed:

1.8 Telematics device establishes a default session with the ECU.

1.9 If the telematics device does not have a valid Access Key for EAL, it must initiate a Learning Session per section 4.1 of the "Telematics Partner Specification" document.

1.9.1 The Provider Telematics Device shall only initiate (or retry) the Learning Access Process (see the Telematics Partner Specification) when it has good over-the-air coverage. **The entire Learning Access process, from the time the ECM receives the Seed request, until the ECM receives the Session Key, must be completed in 5 minutes or less.** If the time is exceeded, the process shall be retried until it is completed within the time limit. In addition, no other UDS commands or operations (except "Tester Present") shall be sent to the ECM until the Learning Access process is complete.

1.9.2 If the ECM sends "Negative Response Code" NRC-35 ("invalidKey (IK)"), the Provider Telematics device shall retry the Learning Access Process once. If the Learning Session still cannot be established, the Provider Telematics Device shall abort the process.

1.10 After the Learning Access Process is successfully executed (or if the telematics device bypassed the Learning Access Process because it already had a valid Access Key for EAL), the telematics device must initiate a telematics session with the ECU and request Engineering Access Level per section 4.5 of the "Telematics Partner Specification" document.

1.10.1 If the ECU denies Engineering Access Level, then the telematics device shall abort the process.

1.10.2 In case of an error in establishing the session or granting access, a "Negative Response Code" NRC-35 ("invalidKey (IK)") will be sent by the ECU. In this case, the telematics device shall retry once. If the session still cannot be established, the telematics device shall abort the process.

1.11 After the Engineering Access Level is established, the telematics device shall retrieve the Trip Data using the RoutineControl service per section 4.5.3 of the "Telematics Partner Specification" document.

1.11.1 The telematics device shall include the RoutineControlOption that matches the Data Content Specification in each request ("Partial" extraction or "Full" extraction).

1.11.2 The ECU shall normally return a positive response to each request.

1.11.3 After requesting the data and receiving the positive response, the telematics device shall repeat the data request and receive another positive response. The telematics device shall repeat this request-response sequence until the ECU returns the "no more data" positive response (0xFF 0xFF). This constitutes a

"data retrieval sequence" (see Data Sampling Specifications section above).

1.11.4 In case of an error in the data request, a "Negative Response Code" (NRC) will be sent by the ECU. Refer to the below NRC table for details and actions to be taken.

1.11.5 The ECU shall return all the parameter values in a given request as a single data block. Separate requests will return separate data blocks.

1.11.6 The telematics device shall pass the data to the Cummins cloud unchanged, omitting the "no more data" indication (0xFF 0xFF). Each block of trip data retrieved shall be a separate sample, date-timestamped when the trip data block was retrieved from the ECU. Each sample shall be formatted: "<partial/full>": "<Data_Block_Retrieved>"

1.11.7 To minimize the amount of stored trip data, the telematics device **may** request trip data four times, transmit those four data blocks, and then continue the "data retrieval sequence". If this method is used, the trip data must be requested and sent in groups of **four** data blocks.

NRC	Description	Action To Be Taken
0x12	**sub-functionNotSupported (SFNS)**	Check/correct system settings before retrying.
0x13	**incorrectMessageLengthOrInvalidFormat (IMLOIF)**	Check/correct system settings before retrying.
0x22	**conditionsNotCorrect (CNC)**	Retry 3 times.
0x24	**requestSequenceError (RSE)**	Check/correct system settings before retrying (possibly increase time between repeated requests).
0x31	**requestOutOfRange (ROOR)**	Check/correct system settings before retrying.
0x33	**securityAccessDenied (SAD)**	Retry 1 time.
0x72	**generalProgrammingFailure (GPF)**	Retry 1 time.

2. Sending a Single Data Set

A “data set” is one transmitted set of data, from one source, based on one Data Sampling Configuration (that is, its parameter content is based on one Data Content Specification and it was sampled and transmitted based on the settings in one Data Sampling Specification). Telematics devices and TSP clouds can send single data sets using the /sendSingleDataSet API method. Multiple data sets can be sent using repeated /sendSingleDataSet calls or using the /sendMultipleDataSets API method (see section 3 below). The data set message has the following attributes:

Data Source Identification

The data set must contain several parameters to identify the source of the data, as follows:

2.1 TelematicsPartnerName: the name of the TSP the message is being sent from.
2.2 CustomerReference: the identifier of the customer the message is being sent from. This would typically be the owner (or possibly the lessee) of the equipment.
2.3 ComponentSerialNumber: the serial number of the component (e.g. engine, transmission, motor, etc.) the message is being sent from.
2.4 EquipmentId: the Customer's ID/name for the equipment/asset/vehicle.
2.5 VIN: Vehicle Identification Number (or equivalent for off-highway markets).
2.6 TelematicsDeviceId: the identifier/serial number of the telematics device the message is being sent from.
2.7 AdditionalSourceIds: a list (array) of optional additional source identification fields. Each element of the array has the following fields:
  2.7.1 SourceIDType: In recognition that equipment, engines/components, and telematics devices are identified in different ways, the SourceIDType allows for various ways of identifying the source of the data. Choices are: TelematicsPartnerName, CustomerReference, ComponentSerialNumber, EquipmentID, VIN, TelematicsDeviceID, UnitNumber, GatewayID, and BrokerID.
  2.7.2 SourceID: This is the SourceID (of the type specified in SourceIDType) that the data set is being sent from.

Data Set Attributes

Other attributes are included in the /sendSingleDataSet message to enable the Cummins cloud to interpret the data, as follows:

2.8 DataSamplingConfigID: the Data Sampling Configuration that the data was collected based on.
  2.8.1 Note: If a telematics device has more than one Data Sampling Configuration active simultaneously, it must send the data sets separately (using separate /sendSingleDataSet calls or the /sendMultipleDataSets API method).

2.9 DataEncryptionSchemeID: the Data Encryption Scheme that was used to encrypt the transmitted data, as agreed on by mutual consent or by using the Encryption Negotiation process (see section D below).

2.10 NumberOfSamples: the number of data samples in the data set. This will typically depend on the MaxTransmitPeriod and MaxSetSize settings in the Data Sampling Specification, but should always reflect the actual number of data samples in the data set. Note: Any sample containing only Single Sample Parameters SHOULD be counted toward the NumberOfSamples. For example, if there are 12 AllSampleParameters samples, the SingleSampleParameters sample (if there is one) will make 13. See also section 2.1.1 on page 12~~on page 12~~.

2.11 Samples: an array with NumberOfSamples elements, each element being a single data sample, consisting of:
  2.11.1 DateTimestamp: The date and time the sample was taken, in ISO 8601 format in the UTC time zone.
  2.11.2 ConvertedDeviceParameters: These are parameters generated from the telematics device / TSP cloud, converted and scaled. Each parameter is sent as a parameter name string + parameter value pair, as defined in the Data Content Specification.
  2.11.3 RawEquipmentParameters: These are parameters sampled from the equipment sources/ECU(s), in raw hex (not converted or scaled). Each parameter is defined by the same attributes used in the Data Content Specification (Protocol, NetworkID, DeviceID, and ParameterID) and includes the raw parameter value.
  2.11.4 RawEquipmentFaultCodes: If any of the “special” Parameter IDs were included in the Data Content Specification, the applicable Fault Codes should be sent as follows:
    2.11.4.1 Include the Protocol, NetworkID, and DeviceID as described above.
    2.11.4.2 ActiveFaultCodes: Send the entire list of active fault codes as a raw hex string (as received from the

source/ECU). For J1939, this would be the DM1 message.

2.11.4.3 InactiveFaultCodes: Send the entire list of previously active fault codes as a raw hex string (as received from the source/ECU). For J1939, this would be the DM2 message.

2.11.4.4 PendingFaultCodes: Send the entire list of pending fault codes as a raw hex string (as received from the source/ECU). For J1939, this would be the DM27 or DM6 messages.

2.11.5 ConvertedEquipmentParameters: These are parameters sampled from the equipment sources/ECU(s), converted and scaled. Each parameter is defined by the same attributes used in the Data Content Specification (Protocol, NetworkID, DeviceID, and ParameterID) and includes the converted parameter value.

2.11.6 ConvertedEquipmentFaultCodes: If any of the “special” Parameter IDs were included in the Data Content Specification, the applicable Fault Codes should be sent as follows:

2.11.6.1 Include the Protocol, NetworkID, and DeviceID as described above.

2.11.6.2 ActiveFaultCodes: Send the entire list of active fault codes by SPN, FMI, and Occurrence Count as shown below. For J1939, the source/ECU sends these on the DM1 message. For fault codes that have only one identifier, use the SPN field.

```
"activeFaultCodes": [
      {
            "spn": "689",
            "fmi": "1",
            "count": "3"
      },
      {
            "spn": "102",
            "fmi": "18",
            "count": "1"
      }
]
```

2.11.6.3 InactiveFaultCodes: Send the entire list of previously active fault codes by SPN, FMI, and Occurrence Count as shown below. For J1939, the source/ECU sends these on the DM2 message. For fault codes that have only one identifier, use the SPN field.

```
"inactiveFaultCodes": [
      {
            "spn": "689",
```

```
                    "fmi": "1",
                    "count": "3"
               },
               {
                    "spn": "102",
                    "fmi": "18",
                    "count": "1"
               }
          ]
```

2.11.6.4 PendingFaultCodes: Send the entire list of pending fault codes by SPN, FMI, and Occurrence Count as shown below. For J1939, the source/ECU sends these on the DM27 and/or DM6 messages. For fault codes that have only one identifier, use the SPN field.

```
     "pendingFaultCodes": [
               {
                    "spn": "689",
                    "fmi": "1",
                    "count": "3"
               },
               {
                    "spn": "102",
                    "fmi": "18",
                    "count": "1"
               }
          ]
```

2.11.7 Note: Each Equipment Parameter should be sent raw (in RawEquipmentParameters) **or** converted (in ConvertedEquipmentParameters), but not both.

2.11.8 Note: Each Equipment Fault Code should be sent raw (in RawEquipmentFaultCodes) **or** converted (in ConvertedEquipmentFaultCodes), but not both.

Responses to Sent Data Set

2.12 If the data set is successfully received, Cummins will respond with an HTTP status code 200 (with no JSON object).

2.13 If there is an error in receiving the data, Cummins will respond with an appropriate non-200 HTTP status code and a JSON error object containing details of the error, similar to the Company and Application Registration Interfaces. See the “Connected Solutions Integration Interfaces” and “Next Gen Data Ingestion API” documentation for details.

2.14 If Cummins sends a non-200 HTTP status code, the telematics device or TSP cloud shall take one of the following actions:

2.14.1 If the HTTP status code (typically in the 400s range) indicates a problem with the **content** of the message, the provider shall attempt to correct the issue (working with Cummins as necessary) before possibly retrying.

2.14.2 If the HTTP status code (typically in the 500s range) indicates a problem with the **transmission** of the message (e.g. server is down), the Provider environment should retry the message. The Provider environment should wait 5 minutes between each retry, retry up to a maximum of 5 times, then contact Cummins if still unsuccessful.

3. Sending Multiple Data Sets

Telematics devices and TSP clouds can send multiple data sets using the /sendMultipleDataSets API method. The multiple data set message is basically just a grouping of single data sets. The multiple data set message has the following attributes:

Multiple Data Set Attributes

3.1 NumberOfDataSets: the number of (single) data sets in the message.

3.2 DataSets: an array with NumberOfDataSets elements, each element being a single data set, exactly as described in section 2 above.

Responses to Sent Data Sets

3.3 See section 2 above.

D. Encryption Negotiation

Because telematics devices and TSP clouds may support different methods of encrypting data (“Data Encryption Schemes”), an encryption negotiation process enables telematics devices / TSP clouds and Cummins to agree on a Data Encryption Scheme so that data can be securely exchanged.

The encryption negotiation process uses two API methods, each of which are fully explained below:

1. /interrogateEncryption
2. /setEncryption

1. Interrogating a Telematics Device or TSP Cloud

Cummins “interrogates” a telematics device or a TSP cloud to determine what Data Encryption Scheme it supports. Cummins uses the /interrogateEncryption API method to do this, and the telematics devices and/or TSP cloud respond with the **best** supported Data Encryption Scheme, based on the list in the Appendix.

Destination Devices

The telematics device(s) that are being interrogated are called “destination devices.” The destination devices are defined in the same manner as described above. The interrogation message simply consists of a list of destination devices.

Responses to Interrogation Messages

1.1 The response consists of an EncryptionResponses array. Each element of the array has a DataEncryptionSchemeID and a list of destination devices (using DestinationIDType and DestinationIDs as described above).
- 1.1.1 For a response sent by a TSP cloud, the EncryptionResponses array may consist of multiple elements, each element representing a different group of destination devices with a different DataEncryptionSchemeID.
- 1.1.2 For a response sent by a TSP cloud, the EncryptionResponses array may consist of a single element with the “special” DestinationID “all” to indicate that the DataEncryptionSchemeID applies to ALL the destination devices.
- 1.1.3 For a response sent by a telematics device, the EncryptionResponses array would typically have a single element corresponding to the (single) destination device.
  - 1.1.3.1 (An example of an exception might be for a single telematics device connected to multiple components, and DestinationIDType = ComponentSerialNumber. In this case, the telematics device may need to send a multi-element EncryptionResponses array if the multiple components require different DataEncryptionSchemeIDs.)

1.2 The DataEncryptionSchemeID value must contain the **best** Data Encryption Scheme that the device supports based on the list in the Appendix.

2. Setting the Data Encryption Scheme

After learning which Data Encryption Scheme the telematics device or TSP cloud supports, Cummins will set the Data Encryption Scheme to be used by the telematics device or TSP cloud when sending data to Cummins (using the /sendSingleDataSet or /sendMultipleDataSets API methods). Cummins uses the /setEncryption API method to do this, as follows:

Set Encryption Messages

2.1 The set encryption message consists of an EncryptionSettings array. Each element of the array has a DataEncryptionSchemeID and a list of destination devices (using DestinationIDType and DestinationIDs in the same way as the interrogation messages).
- 2.1.1 When sent to a TSP cloud, the EncryptionSettings array may consist of multiple elements, each element representing a different group of destination devices with a different DataEncryptionSchemeID to be set.
- 2.1.2 When sent to a TSP cloud, the EncryptionSettings array may consist of a single element with the “special” DestinationID “all” to indicate that the DataEncryptionSchemeID should be set on ALL the destination devices.
- 2.1.3 When sent to a telematics device, the EncryptionSettings array would typically have a single element corresponding to the (single) destination device.
  - 2.1.3.1 (An example of an exception might be for a single telematics device connected to multiple components, and DestinationIDType = ComponentSerialNumber. In this case, Cummins may need to send a multi-element EncryptionSettings array if the multiple components require different DataEncryptionSchemeIDs.)

Responses to Set Encryption Messages

2.2 The response consists of an ActionResponses array. The ActionResponses array is defined in the same manner as the activate and deactivate data collection messages.

2.3 The Response value must be set to “accepted” or “rejected”.
- 2.3.1 An “accepted” response means that the Data Encryption Scheme(s) have been or will be set on the destination device(s).
- 2.3.2 A “rejected” response means that the Data Encryption Scheme(s) will not be set on the destination device(s). For example, if the destination device(s) do not support the requested Data Encryption Scheme(s) or the DestinationID(s) are invalid, a “rejected” response must be sent. A “rejected” response must include information about the reason for rejection as follows:

2.3.2.1 ReasonCode = 501 (Setting(s) not supported), or
502 (Function(s) not supported), or
503 (Destination(s) do not exist)

2.3.2.2 Parameters = List of setting(s)/function(s) that are not supported by the destination device(s), or destination(s) that do not exist

V. APPENDIX

A. Supported Datalink Protocols for Data Content Specifications

Protocol String	Description
Currently Supported	
J1939	SAE Surface Vehicle Recommended Practice J1939
UDS	ISO 14229: Road vehicles – Unified diagnostic services (UDS) ISO 15765: Road vehicles – Diagnostic communication over Controller Area Network (DoCAN)
EAL	Cummins Engineering Access Level
TripData	Cummins Trip Data
Possible Future Support	
J1587	SAE Surface Vehicle Recommended Practice J1587
N2K	NMEA 2000 – IEC 61162-3
ModRTU	Modbus RTU
ModTCP	Modbus TCP/IP

B. Reason/Error Codes for Message Responses

Code	Description
16	NRC: generalReject (GR)
18	NRC: sub-functionNotSupported (SFNS)
19	NRC: incorrectMessageLengthOrInvalidFormat (IMLOIF)
34	NRC: conditionsNotCorrect (CNC)
36	NRC: requestSequenceError (RSE)
49	NRC: requestOutOfRange (ROOR)
51	NRC: securityAccessDenied (SAD)
53	NRC: invalidKey (IK)
55	NRC: requiredTimeDelayNotExpired (RTDNE)
114	NRC: generalProgrammingFailure (GPF)

115	NRC: wrongBlockSequenceCounter (WBSC)
120	NRC: requestCorrectlyReceived-ResponsePending (RCRRP)
300	No response from ECM
301	Slow / intermittent response from ECM
302	Invalid response from ECM
303	Calibration/Write Invalid response from ECM
304	Rollback Expired response from ECM
305	ECM running from Bootloader
306	Unexpected/Unknown Calibration version from ECM
310	No on-vehicle datalink connection
311	Slow / intermittent on-vehicle datalink connection
320	No over-the-air connection (cellular)
321	Slow / intermittent over-the-air connection (cellular)
330	No over-the-air connection (satellite)
331	Slow / intermittent over-the-air connection (satellite)
340	No over-the-air connection (other)
341	Slow / intermittent over-the-air connection (other)
350	No response from Connected Solutions
351	Slow / intermittent response from Connected Solutions
352	Invalid response from Connected Solutions
359	Connected Solutions system failure – indeterminate
360	No response from Operator / HMI
361	Slow / intermittent response from Operator / HMI
362	Invalid response from Operator / HMI
363	Deny limit exceeded by Operator / HMI
364	Write cancelled due to insufficient keyswitch off time
370	Telematics device / system error
371	Telematics environment / system error
379	Telematics system failure – indeterminate
400	Provider not certified
401	Provider Device hardware / software not capable
402	Provider Device hardware / software combination causes an exception
410	App / Feature not available in ECM calibration for associated customer
411	App / Feature disabled in ECM calibration for associated customer
412	App / Feature not available in ECM due to Bootloader Version
420	ECM calibration capable of App / Feature but not compatible with Provider Device hardware / software
430	App / Feature not recognized
431	App / Feature not supported
432	Disable request for an App / Feature that is already disabled
433	App / Feature subscribed to by another Provider on the same equipment
434	Date specified is in the Past
435	End Date does not comply with commercial agreement
440	Request cancelled by customer

441	Request timeout
442	Request refused
443	Equipment / Engine / Component not recognized
444	System failure – indeterminate
445	Unrecognized Customer
446	Trial subscription not available
447	Trial subscription duration does not comply with commercial agreement
448	Trial subscription count exceeded
481	Capability not approved by Cummins
500	Parameter(s) not supported
501	Setting(s) not supported
502	Function(s) not supported
503	Destination(s) do not exist

C. Pre-defined Data Collection Specifications

ID#	Description
Data Content Specifications	
DC5000	Connected Diagnostics Fault Code (FC) Snapshot: <15L diesel with aftertreatment (on highway)
DC5001	Connected Diagnostics Heart Beat (HB) Snapshot: <15L diesel with aftertreatment (on highway)
DC5002	Connected Diagnostics Fault Code (FC) Snapshot: <15L natural gas (on highway)
DC5003	Connected Diagnostics Heart Beat (HB) Snapshot: <15L natural gas (on highway)
DC5004	Connected Diagnostics Fault Code (FC) Snapshot: V16+ diesel without aftertreatment
DC5005	Connected Diagnostics Heart Beat (HB) Snapshot: V16+ diesel without aftertreatment
DC5006	Connected Diagnostics Fault Code (FC) Snapshot: V16+ diesel with aftertreatment
DC5007	Connected Diagnostics Heart Beat (HB) Snapshot: V16+ diesel with aftertreatment
DC5008	Filtration System Specific
DC5009	Mining X Engine Parameters Message
DC5010	Mining X Fault Codes Message
DC5011	Mining X Slow Data Message
DC5012	Connected Software Updates UDS Reads
DC5013	EAL Default
DC5014	Trip Data Partial Extraction
DC5015	Trip Data Full Extraction
Data Sampling Specifications	

DS5000	Connected Diagnostics Fault Code (FC): All Active faults – single snapshot at time of fault
DS5001	Connected Diagnostics Fault Code (FC): All Active faults – pre fault (15 sec) video snapshot (1 sec sampling)
DS5002	Connected Diagnostics Fault Code (FC): All Inactive faults – single snapshot at time of fault
DS5003	Connected Diagnostics Fault Code (FC): All Inactive faults – pre fault (15 sec) video snapshot (1 sec sampling)
DS5004	Connected Diagnostics Heart Beat (HB): Single snapshot every 15 min
DS5005	Connected Diagnostics Heart Beat (HB): Single snapshot every 60 min
DS5006	Next Gen Connected Advisor Heart Beat (HB): 1 sec sampling, 1 sec transmit
DS5007	Next Gen Connected Advisor Heart Beat (HB) / Mining X Engine Parameters Message: 5 sec sampling, 5 sec transmit
DS5008	Mining X Fault Codes Message: All Active & Inactive faults – single snapshot at time of fault
DS5009	Mining X Slow Data Message: 5 min sampling, 5 min transmit
DS5010	Connected Software Updates UDS Reads: Single read every 3 hours
DS5011	EAL Slow Rate: 1 sec sampling, 5 min transmit
DS5012	EAL Medium Rate: 500 msec sampling, 5 min transmit
DS5013	EAL Fast Rate: 100 msec sampling, 5 min transmit
DS5014	Trip Data at Key On
Events	
EV5000	Simple Event: All Active Faults
EV5001	Simple Event: All Inactive Faults
EV5002	Simple Event: All Pending Faults
EV5003	Simple Event: Engine Speed > 400 RPM (engine running)
EV5004	Simple Event: Engine Speed < 50 RPM (engine stopped)
EV5005	Composite Event: EV5000 + EV5001
EV5006	Simple Event: Key On
Data Sampling Configurations	
SC5000	DC5000 + DS5000
SC5001	DC5000 + DS5001
SC5002	DC5000 + DS5002
SC5003	DC5000 + DS5003
SC5004	DC5001 + DS5004

SC5005	DC5001 + DS5005
SC5006	DC5002 + DS5000
SC5007	DC5002 + DS5001
SC5008	DC5002 + DS5002
SC5009	DC5002 + DS5003
SC5010	DC5003 + DS5004
SC5011	DC5003 + DS5005
SC5012	DC5004 + DS5001
SC5013	DC5004 + DS5002
SC5014	DC5005 + DS5006
SC5015	DC5005 + DS5007
SC5016	DC5006 + DS5001
SC5017	DC5006 + DS5002
SC5018	DC5007 + DS5006
SC5019	DC5007 + DS5007
SC5020	DC5008 + DS5006
SC5021	DC5008 + DS5007
SC5022	DC5009 + DS5007
SC5023	DC5010 + DS5008
SC5024	DC5011 + DS5009
SC5025	DC5012 + DS5010
SC5026	DC5013 + DS5011
SC5027	DC5014 + DS5014

D. Supported Data Encryption Schemes

ID#	Description	Rank
IMPORTANT: The higher the Data Encryption Scheme rank, the better		
ES0	No encryption	0
ES1	TLS 1.2	1

VI. DOCUMENT UPDATES

Version Number	Change Date	Modified By	Reviewed By	Changes Made

1.1.0	January 29, 2019	Michael F. Mattern		• Added EAL • Added Trip Data • Added DataTransmitMethod • Added reference to NGDI File Upload Process • Changed “batch” to “set”
1.0.2	August 15, 2018	Michael F. Mattern		• Added several line items to Appendix
1.0.1	July 23, 2018	Michael F. Mattern		• Updated Fault Code content for incoming data • Corrected case on various quoted strings
1.0.0	July 13, 2018	Michael F. Mattern		• Document created

VII. REQUIRED FILES

The following items are essential supplementary documents, APIs, schema, additional data, miscellaneous necessary files, etc.

Document Name	Description	Item Type
Next Gen Data Ingestion API (v1.1.0)	Application Program Interface (API) description for communication between the telematics devices / TSP cloud and Cummins in support of Next Gen Data Ingestion Interface	API