**EAST AFRICAN COMMUNITY**

EAC

JUMUIYA YA AFRIKA MASHARIKI

# Communication Protocol Guidelines for EAC RECTS Integration

# INTRODUCTION

The East African Community (EAC) Regional Electronic Cargo Tracking System (RECTS) relies on secure, interoperable, and standardized communication protocols to support real-time cargo monitoring, bonded transit enforcement, and fuel accountability across Partner States.

These **Communication Protocol Guidelines** define the technical framework for integrating RECTS-enabled IoT devices such as **eSEAL, eFUEL, and eLOCK** with central customs and regulatory platforms. The guidelines provide a unified approach for device-to-server data transmission using both **persistent TCP channels** and **HTTP REST interfaces**, ensuring adaptability across diverse operational environments and network conditions.

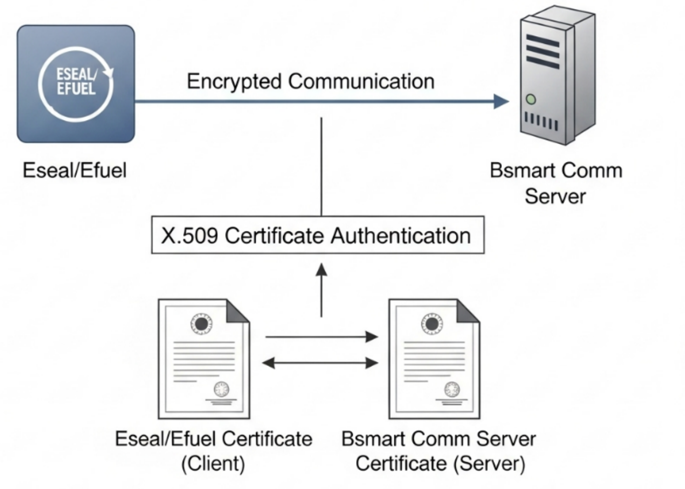
By establishing consistent protocols for operational reporting, exception alerts, failover resilience, and secure command execution, these guidelines support a coordinated and trusted integration of RECTS devices across the region enhancing cargo integrity, customs compliance, and cross-border digital supervision within the EAC Single Customs Territory. EAC will continue to upgrade the protocol as system improvement

## OVERVIEW OF ESEAL/EFUEL SERIES OVER-THE-AIR COMMUNICATION

The RECTS eSEAL / eFUEL system is an integrated IoT solution designed for secure seal management, fuel monitoring, and regulatory data reporting.
The system enables field devices to transmit operational data to central server platforms using both TCP and HTTP REST communication protocols, ensuring reliability, flexibility, and network adaptability across different deployment environments.
Both protocols are designed to carry the same unified JSON payload structure, allowing consistent data processing on the server side regardless of the transport mechanism.

## COMMUNICATION ARCHITECTURE

The RECTS integration architecture supports dual uplink communication paths from device to server with encrypted TCP-based data reporting,


ESEAL/ EFUEL
Encrypted Communication
Eseal/Efuel
Bsmart Comm Server
X.509 Certificate Authentication
Eseal/Efuel Certificate (Client)
Bsmart Comm Server Certificate (Server)

TCP Protocol (Persistent Data Channel)

The TCP protocol provides a persistent, connection-oriented channel for device data transmission.

Key Characteristics

- Long-lived TCP connections
- TLS or mutual TLS (mTLS) encryption
- Low-latency and reliable delivery
- Efficient for frequent or continuous data reporting

Payload Model

JSON data model is adopted in the payload formation with following benefits:

- Consistent data interpretation
- Unified server-side validation and processing logic
- Simplified protocol maintenance and versioning
- Seamless switching between transport mechanisms

Device Authentication and encryption

To guarantee the security of our communication, we employ X.509 certificate authentication. This method provides strong mutual authentication between the eFuel/eSeal/eLock devices and the Comm Server. Each device and the server possess unique digital certificates issued by RECTS operation. During the connection handshake, these certificates are exchanged and verified, ensuring that only authorized entities can establish communication. This approach effectively prevents unauthorized access and man-in-the-middle attacks, safeguarding the sensitive data transmitted between our devices and the platform

On top of that, the Electronic Seal must be integrated with ECTS enabled Blockchain to validate the encrypted off-chain command with ECTS Device Smart Contract Registry to perform (but not limited to) Activation/Deactivation procedure.

## DOMAIN NAME–BASED SERVER RESOLUTION

To avoid dependency on fixed IP addresses, eFuel,eSeal,eLock devices are configured to use domain names (DNS) to identify backend servers.

- Devices store backend endpoints as fully qualified domain names (FQDNs).
- During connection establishment, the device resolves the domain name to obtain the current server IP address.
- This design allows backend server IP addresses to change due to maintenance, scaling, or infrastructure migration without requiring firmware updates or device reconfiguration.
- Using DNS-based resolution ensures long-term flexibility and reduces operational risk in dynamic infrastructure environments.

## MULTI-SERVER CONFIGURATION AND AUTOMATIC FAILOVER

In addition to DNS-based resolution, eFuel,eSeal,eLock devices support configuration of multiple backend servers to ensure service continuity in case of server unavailability.

- Devices maintain a prioritized list of backend servers, each identified by a domain name.

- The primary server is used under normal operating conditions.
- If connection attempts to the primary server fail due to timeout, network error, or service unavailability, the device automatically attempts to connect to secondary or fallback servers.
- Failover is handled internally by the device firmware and requires no external intervention.
  This mechanism ensures uninterrupted data transmission, command reception, and monitoring even during backend outages.

Device reporting events

Overview

This chapter defines the unified device reporting framework for the RECTS eSEAL and eFUEL systems.

Both device types use a common reporting architecture and payload model, enabling consistent data ingestion, auditing, and regulatory supervision while serving different operational domains.

The reporting mechanism is designed to support customs control, cargo security, fuel accountability, and energy auditing, in alignment with government and regulatory requirements.

Reporting Scope

The RECTS device reporting framework covers three major categories:

1. Regular Operational Reporting
2. Event-Based Reporting
3. Alert and Exception Reporting

These categories apply consistently to both eSEAL and eFUEL devices, ensuring standardized handling and prioritization across the platform.

Report data payloads

Overall data payload model

| **Field Name** | **Description** | **Required** | |
|---|---|---|---|
| latitude | Latitude (in decimal degrees) | Yes | decimal degrees |
| longitude | longitude (in decimal degrees) | Yes | decimal degrees |
| altitude | Altitude | Yes | Integer |
| timestamp | GPS Time stamp in milliseconds | Yes | Long |
| speed | Speed (km/h) | Yes | decimal degrees |
| bearing | Direction (degree) | Yes | decimal degrees |
| satellite count | Receivable Number of satellites | Yes | Integer |
| HDOP | Horizontal Dilution Of Precision (Quality of GPS signal) | Yes | decimal degrees |
| d2d3 | Satellite mode 2D or 3D | Yes | 2 or 3 |
| RSSI | Received Signal Strength Indication | Yes | integer |
| LAC | Local Area Code | Yes | integer |
| Cell_ID | Cell ID | Yes | integer |
| MCC | Mobile Country Code | Yes | integer |
| MGS_ID | Unique data running number (64bits) | Yes | integer |
| Activity_id | Activity ID for the message, see Activity Message Description | Yes | integer |

| | | | |
|---|---|---|---|
| addon_info | Add-on information for related activity in JSON format depends on the activity ID | Optional | addon_info |
| fuel_info | Fuel data information object when activity id is 12(activity = 12) | | fuel_info |

Example for Activity 1 GPS reporting:

```
{
  "latitude": "-6.79124",
  "longitude": "39.1",
  "altitude": "21",
  "timestamp": "1541603095967",
  "horizontal_speed": "80",
  "vertical_speed": "0",
  "bearing": "150",
  "satellite_count": "9",
  "HDOP": "1",
  "d2d3": "3",
  "RSSI": "0",
  "LAC": "123",
  "Cell_ID": "12345",
  "MGS_ID": "12345",
  "MCC": "635",
  "activity_id": "001"
}
```

Activity Messages

*Activity ID: 001 – Movement / Logging*

| | |
|---|---|
| Activity Name | Movement / Logging |
| Activity Category | Regular Reporting |
| Description | Periodic logging of GPS position, speed, direction, and device status during transit. |
| Trigger Condition | Device operating normally within reporting interval. |
| Device Functionality | Collects GPS and operational data. |
| Severity Level | Informational |
| Regulatory Impact | Trip traceability |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 002 – Fuel Data Reporting*

| | |
|---|---|
| Activity Name | Fuel Data Reporting |
| Activity Category | Event |
| Description | Regular reporting of fuel level and sensor health data. |
| Trigger Condition | Scheduled fuel reporting interval. |
| Device Functionality | Reads fuel sensor and transmits data. |
| Severity Level | Informational |
| Regulatory Impact | Fuel accountability |
| Applicable Device | eFuel |

*Activity ID: 003 – Enter Checkpoint*

| | |
|---|---|
| Activity Name | Enter Checkpoint |
| Activity Category | Event |
| Description | Device enters predefined checkpoint or geofence. |
| Trigger Condition | GPS enters checkpoint boundary. |
| Device Functionality | Detects geofence entry. |
| Severity Level | Informational |
| Regulatory Impact | Customs compliance |
| Applicable Device | eFuel |

*Activity ID: 004 – Leave Checkpoint*

| | |
|---|---|
| Activity Name | Leave Checkpoint |
| Activity Category | Event |
| Description | Device exits predefined checkpoint or geofence. |
| Trigger Condition | GPS exits checkpoint boundary. |
| Device Functionality | Detects geofence exit. |
| Severity Level | Informational |
| Regulatory Impact | Transit compliance |
| Applicable Device | eFuel |

*Activity ID: 005 – Fuel Loading Detected*

| | |
|---|---|
| Activity Name | Fuel Loading Detected |
| Activity Category | Event |
| Description | Normal fuel loading activity detected. |
| Trigger Condition | Fuel level increases beyond threshold. |
| Device Functionality | Detects fuel increase. |
| Severity Level | Informational |
| Regulatory Impact | Fuel audit |
| Applicable Device | eFuel |

*Activity ID: 006 – Fuel Offloading Detected*

| | |
|---|---|
| Activity Name | Fuel Offloading Detected |
| Activity Category | Event |
| Description | Normal fuel offloading activity detected. |
| Trigger Condition | Fuel level decreases beyond threshold. |
| Device Functionality | Detects fuel decrease. |
| Severity Level | Informational |
| Regulatory Impact | Fuel audit |
| Applicable Device | eFuel |

*Activity ID: 007 – GPS Signal Not Available*

| | |
|---|---|
| Activity Name | GPS Signal Not Available |
| Activity Category | Alert |
| Description | GPS signal loss detected. |
| Trigger Condition | No GPS fix for configured duration. |
| Device Functionality | Monitors GPS module status. |
| Severity Level | Warning |
| Regulatory Impact | Tracking continuity |

| | |
|---|---|
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 008 – Speeding*

| | |
|---|---|
| Activity Name | Speeding |
| Activity Category | Alert |
| Description | Vehicle exceeds configured speed limit. |
| Trigger Condition | Speed above threshold (eg > 80km/h). |
| Device Functionality | Monitors speed. |
| Severity Level | Warning |
| Regulatory Impact | Road safety |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 009 – Harsh Braking*

| | |
|---|---|
| Activity Name | Harsh Braking |
| Activity Category | Alert |
| Description | Sudden deceleration detected. |
| Trigger Condition | Acceleration below braking threshold. |
| Device Functionality | Uses accelerometer. |
| Severity Level | Warning |
| Regulatory Impact | Driver behavior |
| Applicable Device | eFuel |

*Activity ID: 010 – Harsh Turning*

| | |
|---|---|
| Activity Name | Harsh Turning |
| Activity Category | Alert |
| Description | Sharp turning detected. |
| Trigger Condition | Lateral acceleration exceeds limit. |
| Device Functionality | Monitors accelerometer. |
| Severity Level | Warning |
| Regulatory Impact | Driver behavior |
| Applicable Device | eFuel |

*Activity ID: 011 – Harsh Acceleration*

| | |
|---|---|
| Activity Name | Harsh Acceleration |
| Activity Category | Alert |
| Description | Rapid acceleration detected. |
| Trigger Condition | Acceleration above threshold. |
| Device Functionality | Monitors accelerometer. |
| Severity Level | Warning |
| Regulatory Impact | Driver behavior |
| Applicable Device | eFuel |

*Activity ID: 012 – Internal Battery Low*

| | |
|---|---|
| Activity Name | Internal Battery Low |
| Activity Category | Alert |
| Description | Internal battery below warning level. |
| Trigger Condition | Voltage below warning threshold. (battery < 30%) |
| Device Functionality | Monitors battery. |
| Severity Level | Warning |

| | |
|---|---|
| Regulatory Impact | Device availability |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 013 – Internal Battery Critical Low*

| | |
|---|---|
| Activity Name | Internal Battery Critical Low |
| Activity Category | Alert |
| Description | Internal battery critically low. |
| Trigger Condition | Voltage below critical threshold (battery < 15%). |
| Device Functionality | Enters power saving for Eseal/Elock. |
| Severity Level | Critical |
| Regulatory Impact | Device continuity |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 014 – Battery Level Excessively High*

| | |
|---|---|
| Activity Name | Battery Level Excessively High |
| Activity Category | Alert |
| Description | Battery voltage abnormally high. |
| Trigger Condition | Voltage exceeds max threshold. |
| Device Functionality | Detects abnormal voltage. |
| Severity Level | Warning |
| Regulatory Impact | Battery safety |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 015 – Internal Battery Disconnected*

| | |
|---|---|
| Activity Name | Internal Battery Disconnected |
| Activity Category | Alert |
| Description | Internal battery disconnected. |
| Trigger Condition | Battery circuit open. |
| Device Functionality | Detects disconnection. |
| Severity Level | Critical |
| Regulatory Impact | Device tamper |
| Applicable Device | eFuel |

*Activity ID: 016 – External Power Supply Low*

| | |
|---|---|
| Activity Name | External Power Supply Low |
| Activity Category | Alert |
| Description | External power supply voltage low (<9V). |
| Trigger Condition | External voltage below threshold. |
| Device Functionality | Monitors power input. |
| Severity Level | Warning |
| Regulatory Impact | Power stability |
| Applicable Device | eFuel |

*Activity ID: 017 – External Power Disconnected*

| | |
|---|---|
| Activity Name | External Power Disconnected |
| Activity Category | Alert |
| Description | External power disconnected from device. |
| Trigger Condition | Loss of external power. |
| Device Functionality | Detects power loss. |

| | |
|---|---|
| Severity Level | Warning |
| Regulatory Impact | Power continuity |
| Applicable Device | eFuel |

*Activity ID: 018 – Accident / Rollover*

| | |
|---|---|
| Activity Name | Accident / Rollover |
| Activity Category | Alert |
| Description | Rollover or severe impact detected. |
| Trigger Condition | Acceleration exceeds impact threshold. |
| Device Functionality | Detects via accelerometer. |
| Severity Level | Critical |
| Regulatory Impact | Safety incident |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 019 – Device Tampering*

| | |
|---|---|
| Activity Name | Device Tampering |
| Activity Category | Alert |
| Description | Unauthorized device tampering detected. |
| Trigger Condition | Casing or sensor tamper switch triggered. |
| Device Functionality | Detects tamper. |
| Severity Level | Critical |
| Regulatory Impact | Security breach |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 020 – Off Route*

| | |
|---|---|
| Activity Name | Off Route |
| Activity Category | Alert |
| Description | Device deviates from planned route. |
| Trigger Condition | GPS outside allowed corridor. |
| Device Functionality | Monitors route compliance. |
| Severity Level | Warning |
| Regulatory Impact | Transit compliance |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 021 – Invalid NFC Scanned*

| | |
|---|---|
| Activity Name | Invalid NFC Scanned |
| Activity Category | Alert |
| Description | Invalid NFC tag scanned. |
| Trigger Condition | NFC UID not authorized. |
| Device Functionality | Validates NFC. |
| Severity Level | Warning |
| Regulatory Impact | Access control |
| Applicable Device | eSeal,eLock,eFuel |

*Activity ID: 022 – RFID Card Not Registered*

| | |
|---|---|
| Activity Name | RFID Card Not Registered |
| Activity Category | Alert |
| Description | Unregistered RFID card used. |
| Trigger Condition | RFID UID not in whitelist. |

| | |
|---|---|
| Device Functionality | Validates RFID. |
| Severity Level | Warning |
| Regulatory Impact | Access control |
| Applicable Device | eSeal,eLock |

*Activity ID: 023 – Seal Broken – Procedure Error*

| | |
|---|---|
| Activity Name | Seal Broken – Procedure Error |
| Activity Category | Alert |
| Description | Seal cable/lock disconnected before proper deactivation. |
| Trigger Condition | Seal opened without procedure. |
| Device Functionality | Detects seal open. |
| Severity Level | Critical |
| Regulatory Impact | Customs violation |
| Applicable Device | eSeal |

*Activity ID: 024 – Seal Broken – Unauthorized Zone*

| | |
|---|---|
| Activity Name | Seal Broken – Unauthorized Zone |
| Activity Category | Alert |
| Description | Seal cable/lock opened outside authorized zone. |
| Trigger Condition | Seal open outside geofence. |
| Device Functionality | Detects seal open. |
| Severity Level | Critical |
| Regulatory Impact | Customs violation |
| Applicable Device | eSeal |

*Activity ID: 025 – Seal Cut Alert*

| | |
|---|---|
| Activity Name | Seal Cut Alert |
| Activity Category | Alert |
| Description | Seal cable cut detected. |
| Trigger Condition | Cable continuity lost. |
| Device Functionality | Immediate tamper alert. |
| Severity Level | Critical |
| Regulatory Impact | Cargo integrity |
| Applicable Device | eSeal |

*Activity ID: 026 – Seal Lock Error*

| | |
|---|---|
| Activity Name | Seal Lock Error |
| Activity Category | Alert |
| Description | Seal unable to lock properly. |
| Trigger Condition | Lock mechanism failure. |
| Device Functionality | Detects lock error. |
| Severity Level | Warning |
| Regulatory Impact | Operational reliability |
| Applicable Device | eSeal,eLock |

*Activity ID: 027 – Seal Unlock Error*

| | |
|---|---|
| Activity Name | Seal Unlock Error |
| Activity Category | Alert |

| | |
|---|---|
| Description | Seal cannot unlock. |
| Trigger Condition | Unlock mechanism failure. |
| Device Functionality | Detects unlock error. |
| Severity Level | Warning |
| Regulatory Impact | Operational reliability |
| Applicable Device | eSeal,eLock |

*Activity ID: 028 – Seal Detached*

| | |
|---|---|
| Activity Name | Seal Detached |
| Activity Category | Alert |
| Description | Seal detached from container. |
| Trigger Condition | Mounting sensor triggered. |
| Device Functionality | Detects detachment. |
| Severity Level | Critical |
| Regulatory Impact | Cargo security |
| Applicable Device | eSeal,eLock |

*Activity ID: 029 – Illegal Seal Activation*

| | |
|---|---|
| Activity Name | Illegal Seal Activation |
| Activity Category | Alert |
| Description | Seal activated without valid NFC. |
| Trigger Condition | Activation without authorization. |
| Device Functionality | Validates activation. |
| Severity Level | Critical |
| Regulatory Impact | Security breach |
| Applicable Device | eSeal,eLock |

*Activity ID: 030 – Illegal Seal Deactivation*

| | |
|---|---|
| Activity Name | Illegal Seal Deactivation |
| Activity Category | Alert |
| Description | Seal deactivated illegally. |
| Trigger Condition | Deactivation without authorization (or valid NFC scan). |
| Device Functionality | Validates deactivation. |
| Severity Level | Critical |
| Regulatory Impact | Security breach |
| Applicable Device | eSeal,eLock |

*Activity ID: 031 – Cross Country Alert*

| | |
|---|---|
| Activity Name | Cross Country Alert |
| Activity Category | Alert |
| Description | Device crosses national border unexpectedly. |
| Trigger Condition | Country code change detected. |
| Device Functionality | Monitors MCC/GPS. |
| Severity Level | Critical |
| Regulatory Impact | Customs enforcement |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 032 – Command Failure*

| | |
|---|---|
| Activity Name | Command Failure |
| Activity Category | Alert |
| Description | Failure executing OTA command. |
| Trigger Condition | Command validation/execution error. |
| Device Functionality | Aborts command. |
| Severity Level | Warning |
| Regulatory Impact | Audit trail |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 033 – Exceed Transit Duration*

| | |
|---|---|
| Activity Name | Exceed Transit Duration |
| Activity Category | Alert |
| Description | Transit duration exceeded. |
| Trigger Condition | Trip active beyond expected exit time. (Defined as trip_expected_time in smart contract) |
| Device Functionality | Monitors trip timer. |
| Severity Level | Warning |
| Regulatory Impact | Transit SLA |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 034 – Exceed Stop Duration*

| | |
|---|---|
| Activity Name | Exceed Stop Duration |
| Activity Category | Alert |
| Description | Vehicle stopped longer than allowed. |
| Trigger Condition | Stop duration exceeds threshold. |
| Device Functionality | Monitors speed/time. |
| Severity Level | Warning |
| Regulatory Impact | Operational compliance |
| Applicable Device | eSeal,eFuel,eLock |

*Activity ID: 035 – eFuel Casing Open*

| | |
|---|---|
| Activity Name | eFuel Casing Open |
| Activity Category | Alert |
| Description | eFuel casing opened. |
| Trigger Condition | Open the casing of the eFuel device. |
| Device Functionality | Detects tamper. |
| Severity Level | Critical |
| Regulatory Impact | Fuel security |
| Applicable Device | eFuel |

*Activity ID: 036 – eSeal Casing Open*

| | |
|---|---|
| Activity Name | eSeal Casing Open |
| Activity Category | Alert |
| Description | eSeal casing opened. |
| Trigger Condition | Open the casing of the eSeal device. |
| Device Functionality | Detects tamper. |
| Severity Level | Critical |

| | |
|---|---|
| Regulatory Impact | Security breach |
| Applicable Device | eSeal |

*Activity ID: 037 – eSeal Fully Charged*

| | |
|---|---|
| Activity Name | eSeal Fully Charged |
| Activity Category | Event |
| Description | eSeal battery fully charged. |
| Trigger Condition | Battery reaches full charge. |
| Device Functionality | Monitors charging. |
| Severity Level | Informational |
| Regulatory Impact | Maintenance |
| Applicable Device | eSeal |

*Activity ID: 038 – Fuel Sensor Hardware Error*

| | |
|---|---|
| Activity Name | Fuel Sensor Hardware Error |
| Activity Category | Alert |
| Description | Invalid or unsupported fuel sensor detected. |
| Trigger Condition | Plug unauthorized fuel sensor / Wrong Fuel sensor hardware detected. |
| Device Functionality | Validates sensor. |
| Severity Level | Critical |
| Regulatory Impact | Measurement accuracy |
| Applicable Device | eFuel |

*Activity ID: 039 – Fuel Sensor Detached*

| | |
|---|---|
| Activity Name | Fuel Sensor Detached |
| Activity Category | Alert |
| Description | Fuel sensor detached from tank. |
| Trigger Condition | Sensor connection lost. |
| Device Functionality | Detects disconnection. |
| Severity Level | Critical |
| Regulatory Impact | Fuel integrity |
| Applicable Device | eFuel |

*Activity ID: 040 – Fuel Sensor Disconnected*

| | |
|---|---|
| Activity Name | Fuel Sensor Disconnected |
| Activity Category | Alert |
| Description | Fuel sensor disconnected from eFuel. |
| Trigger Condition | Fuel sensor wiring cut/unplugged from eFuel. |
| Device Functionality | Detects signal loss. |
| Severity Level | Critical |
| Regulatory Impact | Fuel monitoring |
| Applicable Device | eFuel |

*Activity ID: 041 – Fuel Offloading Alert*

| | |
|---|---|
| Activity Name | Fuel Offloading Alert |
| Activity Category | Alert |
| Description | Abnormal fuel offloading detected. |
| Trigger Condition | Rapid fuel decrease. |

| | |
|---|---|
| Device Functionality | Analyzes trend. |
| Severity Level | Critical |
| Regulatory Impact | Fuel theft detection |
| Applicable Device | eFuel |

*Activity ID: 042 – Fuel Sensor Power Loss*

| | |
|---|---|
| Activity Name | Fuel Sensor Power Loss |
| Activity Category | Alert |
| Description | Fuel sensor power failure. |
| Trigger Condition | eFuel unable to power the fuel sensor (eg. shorted the power supply) |
| Device Functionality | Detects power failure. |
| Severity Level | Critical |
| Regulatory Impact | Fuel monitoring |
| Applicable Device | eFuel |

*Activity ID: 043 – Fuel Density Change Alert*

| | |
|---|---|
| Activity Name | Fuel Density Change Alert |
| Activity Category | Alert |
| Description | Fuel density abnormal change detected. |
| Trigger Condition | Density outside allowed range. |
| Device Functionality | Analyzes density. |
| Severity Level | Warning |
| Regulatory Impact | Fuel quality |
| Applicable Device | eFuel |

*Activity ID: 044 – 11-Hour Driving Limit*

| | |
|---|---|
| Activity Name | 11-Hour Driving Limit |
| Activity Category | Alert |
| Description | Driving limit as per FMCSA regulation, that driving time for property-carrying vehicles can not exceed 11hrs after 10 hrs off duty. |
| Trigger Condition | Driving duration exceeds limit 11-hours |
| Device Functionality | Tracks driving time. |
| Severity Level | Warning |
| Regulatory Impact | Driver compliance (FMCSA) |
| Applicable Device | eFuel |

*Activity ID: 045 – 60 Hour / 7 Day Limit*

| | |
|---|---|
| Activity Name | 60 Hour / 7 Day Limit |
| Activity Category | Alert |
| Description | Driving limit as per FMCSA regulation, that driving time for property-carrying vehicles can not exceed 60hrs within 7 consecutive days. |
| Trigger Condition | Cumulative driving exceeds limit. |
| Device Functionality | Tracks driving hours. |
| Severity Level | Warning |
| Regulatory Impact | Driver compliance (FMCSA) |
| Applicable Device | eFuel |

*Activity ID: 046 – 70 Hour / 8 Day Limit*

| | |
|---|---|
| Activity Name | 70 Hour / 8 Day Limit |
| Activity Category | Alert |
| Description | Driving limit as per FMCSA regulation, that driving time for property-carrying vehicles can not exceed 70hrs within 8 consecutive days. |
| Trigger Condition | Cumulative driving exceeds limit. |
| Device Functionality | Tracks driving hours. |
| Severity Level | Warning |
| Regulatory Impact | Driver compliance (FMCSA) |
| Applicable Device | eFuel |

Addon data model

*Power data model*

Mandatory data for power status activity (activity 6/7):

| **Object** | **Description** | **Type** |
|---|---|---|
| ext_power_voltage | External Power Voltage | Number |
| int_battery_voltage | Device Internal Battery Voltage | Number |

Example:

```
{
    “ext_power_voltage”:”24.22”,
    “int_battery_voltage”:”3.89”
}
```

*Trip data model*

Mandatory data for deactivation activity (activity 27):

| **Object** | **Description** | **Type** |
|---|---|---|
| distance_travelled | Distance travelled in km format | Number |
| trip_duration | Trip duration in minutes | Number |
| avgSpeed | Average speed (KM/h) | Number |
| maxSpeed | Max speed (KM/h) | Number |

Example:

```
{
    “distance_travelled”:”505.2”,
    “trip_duration”:”400”,
    “avgSpeed”: “60”,
    “maxSpeed”: “85”
}
```

*ESEAL Activation model*

| Object | Description | Type |
|---|---|---|
| seal_lock_left_status | The left side of seal lock status (Locked/Unlocked) | String |
| seal_lock_right_status | The right side of seal lock status (Locked/Unlocked) | String |
| seal_cable_status | Seal Cable status (Secured/Broken/Cut) | String |
| seal_lock_left_id | The detected tag ID for the left side lock header | String |
| seal_lock_right_id | The detected tag ID for the right side lock header | String |

Mandatory data for deactivation activity (activity 16/17/18/19/25):

Example:

```
{
    “seal_lock_left_status”:”Locked”,
    “seal_lock_right_status”:”Locked”,
    “seal_cable_status”: “Secured”,
    “seal_lock_left_id”:”11223344”,
    “seal_lock_right_id”:”44332211”
}
```

FuelInfo data model

| Object | Description | Type |
|---|---|---|
| validFlag | Data valid flag (0 for valid, other value to indicate non-valid fuel data) | Number |
| signalLevel | Received Signal Sensitivity of the fuel sensor (0-99) | Number |
| softStatus | Software status code (0 for normal) | Number |
| hardFault | Hardware fault code (0 for normal, see table 2 for details) | Number |
| fuelLevel | Fuel level (smoothed) in mm  (Integer) | Number |
| rtFuelLevel | Real time fuel level in mm (Integer) | Number |
| tankTemp | Tank temperature in Celsius * (integer value with original temperaturex10, example: For 23.5 Celsius, 235 should be input in API) | Number |
| channel | Fuel tank compartment (integer value with default should be 1) | Number |

Example:

```
{
  "validFlag": "0",
  "signalLevel": "75",
  "softStatus": "0",
  "hardFault": "0",
  "fuelLevel": "2100",
  "rtFuelLevel": "2134",
  "tankTemp": "312",
  "channel": "1"
}
```

| Object | Description | Type |
|---|---|---|
| txHash | The command hash send by system | String |

Command Failure data model (For activity 28)

Example:

```
{
    "txHash": “XXXXXXXXXXXXXXXXXXXXXXXXXXXX”
}
```

## Device Downlink commands with blockchain for enhanced security

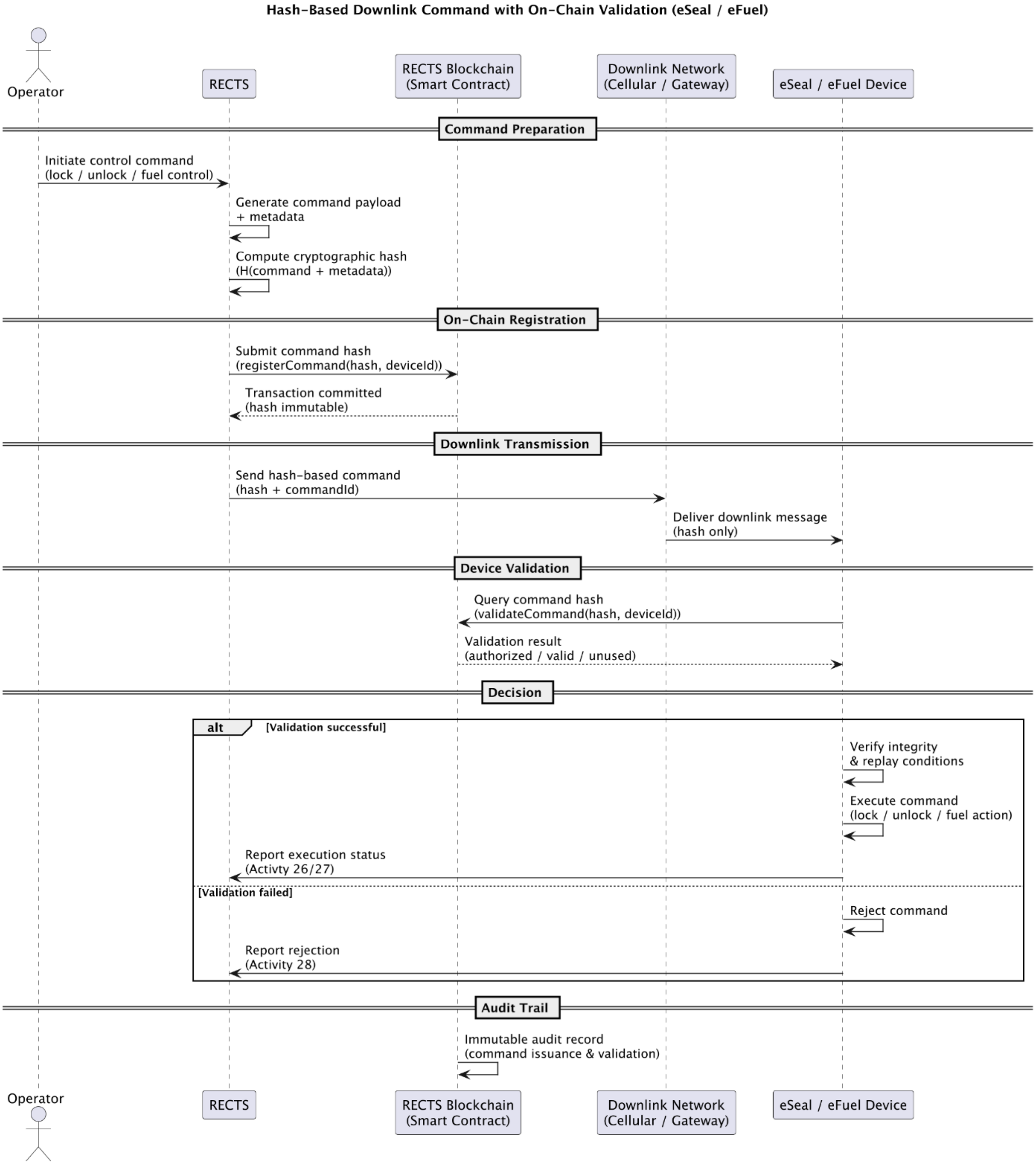
### Overview

The system implements a secure command control mechanism for eSEAL and eFUEL devices using hash-based downlink commands combined with on-chain validation. Instead of transmitting full command payloads to devices, the backend generates a cryptographic hash representing the authorized command and registers it on a blockchain smart contract. The blockchain acts as a distributed and immutable source of truth for command authorization and device trust.

Only the compact hash value is transmitted to the device over the downlink channel, significantly reducing bandwidth usage and improving reliability in low-connectivity environments. Upon receiving the hash, the device (or an associated gateway) independently validates the command against the on-chain record, ensuring that the command is authorized, untampered, and has not been previously executed or revoked.

This approach enforces device-side trust validation, preventing unauthorized or forged commands even if backend systems or communication channels are compromised. The combination of off-chain command generation and on-chain authorization provides strong security guarantees, including integrity protection, replay attack prevention, decentralized trust, and full auditability. The architecture is well suited for regulated, multi-authority environments such as customs control, cross-border transit, and bonded logistics operations.

# Downlink Flow Overview


Hash-Based Downlink Command with On-Chain Validation (eSeal / eFuel)
Operator
RECTS
RECTS Blockchain (Smart Contract)
Downlink Network (Cellular / Gateway)
eSeal / eFuel Device
Command Preparation
Initiate control command (lock / unlock / fuel control)
Generate command payload + metadata
Compute cryptographic hash (H(command + metadata))
On-Chain Registration
Submit command hash (registerCommand(hash, deviceId))
Transaction committed (hash immutable)
Downlink Transmission
Send hash-based command (hash + commandId)
Deliver downlink message (hash only)
Device Validation
Query command hash (validateCommand(hash, deviceId))
Validation result (authorized / valid / unused)
Decision
alt
[Validation successful]
Verify integrity & replay conditions
Execute command (lock / unlock / fuel action)
Report execution status (Activty 26/27)
[Validation failed]
Reject command
Report rejection (Activity 28)
Audit Trail
Immutable audit record (command issuance & validation)
Operator
RECTS
RECTS Blockchain (Smart Contract)
Downlink Network (Cellular / Gateway)
eSeal / eFuel Device

Step-by-Step Execution Flow (eSEAL / eFUEL)

**Step 1 Command Creation (Off-Chain Backend)**

- The backend system generates a control command (e.g. lock, unlock, fuel enable, fuel disable) together with contextual metadata such as device ID, timestamp, and command sequence.
- A cryptographic hash of the command payload and metadata is calculated.
- The full command payload remains off-chain and is not transmitted to the device.

**Step 2 On-Chain Command Registration**

- The generated command hash is submitted to the blockchain smart contract.
- The smart contract records the hash together with command metadata and authorization rules.
- Once committed, the hash becomes an immutable, authoritative reference for command validation.

**Step 3 Downlink Transmission to Device**

- The backend sends only the command hash (and minimal identifiers such as command ID) to the eSEAL / eFUEL device via the downlink channel.
- This minimizes data size and improves reliability in low-bandwidth or unstable networks.

**Step 4 Device Receives Hash Command**

- The device receives the hash-based downlink command.
- No sensitive command logic or business rules are exposed on the device at this stage.

**Step 5 On-Chain Validation by Device (or Gateway)**

- The device (or a trusted gateway acting on its behalf) queries the blockchain to verify:
- The hash exists on-chain
- The hash is associated with the correct device
- The command is authorized and not revoked
- The command has not been previously executed or expired

**Step 6 Integrity and Replay Check**

- The device ensures the received hash matches the on-chain record exactly.
- Sequence numbers, timestamps, or one-time-use flags are validated to prevent replay attacks.

**Step 7 Command Authorization Decision**

- If on-chain validation succeeds, the command is marked as authorized for execution.
- If validation fails, the command is rejected and logged without execution.

**Step 8 Secure Command Execution**

- The device executes the corresponding local action (e.g. seal lock/unlock, fuel control).
- Execution occurs only after successful on-chain validation.

**Step 9 Execution Result Reporting**

- The device reports execution status (success/failure) back to the backend.
- A confirmation hash or status update may also be recorded on-chain for audit purposes.

**Step 10 Audit and Traceability**

- The blockchain maintains an immutable record linking:
- Command issuance
- Authorization
- Validation
- Execution outcome
- This supports dispute resolution, regulatory audits, and cross-agency trust.

Example code of getting verification from blockchain

```
#!/usr/bin/env python3

import grpc
import json
from fabric_gateway import Gateway, Wallets

# =========================
# CONFIGURATION
# =========================
CHANNEL = "mychannel"
CHAINCODE = "fuel-config"
PEER_ENDPOINT = "<blockchainserverIP>:<port>"

TLS_CERT_PATH = "tls/ca.crt"
WALLET_PATH = "wallet"
IDENTITY_LABEL = "deviceClient"

DEVICE_ID = "EFUEL_001"
LOCAL_STATE_FILE = "fuel_state.json"

ALLOWED_FUEL_TYPES = {
    "WATER",
    "GASOLINE",
    "DIESEL",
    "KEROSENE",
    "ETHANOL"
}

# =========================
# FABRIC CONNECTION
# =========================
def get_contract():
    with open(TLS_CERT_PATH, "rb") as f:
        tls_cert = f.read()

    channel = grpc.secure_channel(
        PEER_ENDPOINT,
        grpc.ssl_channel_credentials(tls_cert)
    )

    wallet = Wallets.new_file_system_wallet(WALLET_PATH)
    identity = wallet.get(IDENTITY_LABEL)
    if not identity:
        raise Exception("Device identity not found in wallet")

    gateway = Gateway()
    gateway.connect(identity, channel, discovery=True)

    network = gateway.get_network(CHANNEL)
    return network.get_contract(CHAINCODE)

# ========================
# LOCAL STATE (REPLAY PROTECTION)
# ========================
def load_local_state():
    try:
        with open(LOCAL_STATE_FILE, "r") as f:
            return json.load(f)
    except:
        return {"version": 0, "fuel_type1": None}

def save_local_state(version, fuel_type):
    with open(LOCAL_STATE_FILE, "w") as f:
        json.dump({
            "version": version,
            "fuel_type1": fuel_type1
        }, f)

# ========================
# CHAIN QUERY
# ========================
def get_onchain_fuel_command(device_id):
    contract = get_contract()
    result = contract.evaluate_transaction(
        "GetFuelCommand",
        device_id
    )
    return json.loads(result)

# ========================
# SENSOR CONFIGURATION
# ========================
def apply_fuel_type_to_sensor(fuel_type):
    if fuel_type not in ALLOWED_FUEL_TYPES:
        raise Exception("INVALID_FUEL_TYPE")

    # === HARDWARE-SPECIFIC LOGIC GOES HERE ===
    # Example: select calibration table, density curve, etc.
    print(f"[DEVICE] Sensor configured for fuel type: {fuel_type}")

# ========================
# CORE ON-CHAIN VALIDATION
# ========================
def perform_onchain_fuel_type_check():
    print("[DEVICE] Checking on-chain fuel type configuration")

    local_state = load_local_state()
    local_version = int(local_state.get("version", 0))
```

```
        onchain = get_onchain_fuel_command(DEVICE_ID)

        onchain_version = int(onchain["version"])
        onchain_fuel = onchain["fuel_type1"]

        if onchain_fuel not in ALLOWED_FUEL_TYPES:
            raise Exception("ONCHAIN_FUEL_TYPE_NOT_ALLOWED")

        if onchain_version <= local_version:
            print(
                f"[DEVICE] No update needed "
                f"(local={local_version}, onchain={onchain_version})"
            )
            return

        print(
            f"[DEVICE] Fuel type change detected: "
            f"{local_state.get('fuel_type1')} → {onchain_fuel} "
            f"(v{onchain_version})"
        )

        apply_fuel_type_to_sensor(onchain_fuel)
        save_local_state(onchain_version, onchain_fuel)

        print("[DEVICE] Fuel type update applied successfully")

    # ========================
    # DOWNLINK HANDLER
    # ========================
    def handle_downlink_command(downlink):
        # Downlink is only a trigger, never authoritative
        if downlink.get("command") == "SYNC_FUEL_CONFIG":
            perform_onchain_fuel_type_check()
        else:
            print("[DEVICE] Unknown downlink command ignored")

    # ========================
    # MAIN ENTRY
    # ========================
    if __name__ == "__main__":
        # Simulated downlink trigger
        downlink_message = {
            "command": "SYNC_FUEL_CONFIG"
        }

        try:
            handle_downlink_command(downlink_message)
        except Exception as e:
print("[DEVICE] ERROR:", str(e))
```

Blockchain Access For Device Command Integration
Please engage with RECTS support group to get blockchain instance access

**Smart Contract Data Model – eSeal**

| Attribute Name | Data Type | Description |
|---|---|---|
| device_id | String | Unique identifier of the eSeal device registered on the blockchain. |
| device_type | String | Device classification value identifying the device as an eSeal. |
| rects_comm_server | String | <domain>:<port> setting for RECTS comm server |
| registration_timestamp | Timestamp | Time at which the device was registered on-chain. |
| seal_status | Enum | Current seal state (e.g. Activated, Deactivated, Locked, Unlocked). |
| gps_reporting_interval | Integer | Regular reporting intervals for GPS tracking in seconds |
| trip_id | String | Identifier of the active trip associated with the seal, if applicable. |
| trip_route_points | String | Comma seperated <lat>:<lon>:<type> list for route detection. Possible type: plot/checkpoint/destination |
| trip_expected_time | String | The expected trip total time in minutes |
| state_version | Integer | Monotonically increasing version number representing the current seal state. |
| last_state_change_time | Timestamp | Time of the most recent seal state transition. |
| command_hash | Hash | Cryptographic hash representing an authorized control command. |
| command_type | Enum | Type of authorized command (e.g. Activate, Deactivate, Unlock). |
| command_status | Enum | Command lifecycle status (Authorized, Executed, Rejected, Expired). |
| command_valid_from | Timestamp | Start of command validity period. |
| command_valid_until | Timestamp | End of command validity period. |
| security_event_type | Enum | Reference to critical security events (e.g. Seal Cut, Illegal Activation). |
| security_event_time | Timestamp | Time at which the security event was recorded. |
| tx_id | String | Blockchain transaction identifier for traceability and audit. |

Smart Contract Data Model – eFuel

eFuel Smart Contract Data Model (Multi-Compartment)

| Attribute Name | Data Type | Description |
|---|---|---|
| device_id | String | Unique identifier of the eFuel master device. |
| device_type | String | Device classification identifying the device as eFuel. |
| rects_comm_server | String | <domain>:<port> setting for RECTS comm server |
| registration_timestamp | Timestamp | Time at which the eFuel device was registered on-chain. |
| gps_reporting_interval | Integer | Regular reporting intervals for GPS tracking in seconds |
| trip_id | String | Identifier of the active trip associated with the seal, if applicable. |
| trip_route_points | String | Comma seperated <lat>:<lon>:<type> list for route detection. Possible type: plot/checkpoint/destination |
| compartment_count | Integer | Number of active fuel compartments supported by the device (maximum 9). |
| compartments | Array | Collection of compartment configuration objects (see Table X-3). |
| command_hash | Hash | Cryptographic hash of an authorized configuration or control command. |
| command_category | Enum | Category of command (Configuration or Control). |
| command_status | Enum | Status of the command (Authorized, Executed, Rejected, Expired). |
| command_valid_from | Timestamp | Start time of command validity. |
| command_valid_until | Timestamp | End time of command validity. |
| integrity_event_type | Enum | Reference to integrity or security events related to sensors or casing. |
| integrity_event_time | Timestamp | Time at which the integrity event occurred. |
| tx_id | String | Blockchain transaction identifier for audit and traceability. |

Fuel Compartment Configuration Object

| **Attribute Name** | **Data Type** | **Description** |
|---|---|---|
| compartment_id | Integer | Logical compartment index (1–9). |
| sensor_id | String | Unique identifier of the fuel sensor attached to the compartment. |
| fuel_type | Enum | Authorized fuel type for the compartment (e.g. Diesel, Gasoline, Water). |
| config_version | Integer | Monotonically increasing configuration version for the compartment. |
| last_updated_at | Timestamp | Time when the compartment configuration was last updated. |
| updated_by | String | Identity of the authority that authorized the configuration change. |
| status | Enum | Compartment status (Active, Disabled, Fault). |
| tx_id | String | Blockchain transaction identifier for compartment configuration update. |