



UDS Service Chatbot

Upload UDS service `.txt` files



Drag and drop files here

Limit 200MB per file • TXT

Browse files



TesterPresent.txt 8.3KB



LinkControl.txt 22.4KB



ECUReset.txt 14.1KB



Files uploaded! Click 'Initialize Services' to build agents.

Initialize Services



Compare ECU Reset, Tester Present, Link control services and explain how they are related



◆ TopAgent (Multi-agent reasoning):

Comparison of ECU Reset, Tester Present, and Link Control Services

This document provides a detailed comparison of three essential UDS (Unified Diagnostic Services) services: ECU Reset (0x11), Tester Present (0x3E), and Link Control (0x87), and explains how they relate to one another during a diagnostic session.

Service Overviews

1. ECU Reset (Service 0x11)

- **Purpose:** To request a reset of an Electronic Control Unit (ECU). This is equivalent to rebooting a computer.
- **Functionality:** It supports different types of resets, such as a `hardReset` (simulating a power cycle), `softReset` (restarting the application), and `keyOffOnReset` (simulating an ignition cycle).
- **Effect:** After a reset, the ECU typically returns to the `defaultSession`, terminating any active diagnostic states.

2. Tester Present (Service 0x3E)

- **Purpose:** To keep a diagnostic session active.
- **Functionality:** A diagnostic tool (the "tester" or client) periodically sends this message to the ECU (the server) to signal that it is still connected.
- **Effect:** It prevents the ECU from timing out and reverting to the `defaultSession`. This is crucial for maintaining non-default sessions (like `programmingSession` or `extendedDiagnosticSession`) where advanced diagnostic tasks are performed.

3. Link Control (Service 0x87)

- **Purpose:** To manage the communication link's parameters, most commonly to change the baud rate.
- **Functionality:** This is used to increase communication speed, which is vital for time-sensitive operations like ECU software flashing (programming). It is a two-step process: first verify the new parameters, then transition to them.
- **Effect:** It temporarily alters the communication protocol settings. This service can only be used in a non-default session, which must be kept alive by the Tester Present service.

Comparison Table

Feature	ECU Reset (0x11)	Tester Present (0x3E)	Link Control (0x87)
Primary Goal	Reboot the ECU.	Maintain an active diagnostic session.	Modify communication link parameters (e.g., baud rate).
Service ID	<code>0x11</code>	<code>0x3E</code>	<code>0x87</code>

Feature	ECU Reset (0x11)	Tester Present (0x3E)	Link Control (0x87)
(SID)			
Typical Use Case	Applying new software, clearing faults, or returning an ECU to a known state.	During any extended diagnostic task to prevent session timeout.	Before ECU programming to speed up data transfer.
Session Impact	Terminates the current session and returns the ECU to the <code>defaultSession</code> .	Prevents the current session from timing out and reverting to the <code>defaultSession</code> .	Requires a non-default session to be active. The modified link settings are lost if the session times out or an ECU Reset occurs.
Response	Sends a positive response <i>before</i> executing the reset.	Can be configured to suppress the positive response to reduce bus traffic.	Sends a positive response during the verification step but typically not during the final transition step.
Sub-functions	Yes (e.g., <code>hardReset</code> , <code>softReset</code>).	Yes, but primarily uses <code>zeroSubFunction</code> (0x00).	Yes (e.g., <code>verifyModeTransition</code> , <code>transitionMode</code>).

How the Services are Related

The relationship between these three services is hierarchical and sequential, especially during advanced diagnostic procedures like ECU programming.

- Session Maintenance is Foundational (Tester Present):** Before you can perform any advanced diagnostics, you must move the ECU from the `defaultSession` to a non-default session (e.g., `extendedDiagnosticSession`). The **Tester Present** service is the heartbeat that keeps this session alive. Without it, the ECU would time out and revert to the default state, interrupting any ongoing process.

2. **Preparing the Communication Link (Link Control):** For tasks that involve transferring large amounts of data, like flashing new firmware onto an ECU, the standard communication speed may be too slow. The **Link Control** service is used to increase the baud rate.
 - **Dependency:** Link Control can only be initiated while in a non-default session, which is being maintained by **Tester Present**.
3. **Finalizing the Process (ECU Reset):** After a task like programming is complete, the new software often requires a system reboot to become active. The **ECU Reset** service is used to perform this reboot.
 - **Dependency and Impact:** Executing an ECU Reset will terminate the active diagnostic session (including the one kept alive by **Tester Present**) and revert any changes made by **Link Control** (the baud rate will return to its default). The ECU will restart in the `defaultSession` .

Example Workflow: ECU Programming

Here is a typical sequence showing how the services work together:

1. **Start Session:** The client requests to enter the `programmingSession` .
2. **Maintain Session:** The client starts sending **Tester Present** messages periodically to keep the `programmingSession` active.
3. **Increase Speed:** The client uses **Link Control** to verify and then transition the communication link to a higher baud rate.
4. **Perform Programming:** The client transfers the new software to the ECU at the higher speed. **Tester Present** continues to be sent in the background.
5. **Revert Speed:** The client uses **Link Control** again to return the baud rate to the normal speed.
6. **Reboot ECU:** The client sends an **ECU Reset** (`hardReset`) request. The ECU sends a positive response and then reboots.
7. **End of Process:** The diagnostic session is terminated. The ECU starts up with the new software in the `defaultSession` . The diagnostic tool can now end the connection or start a new session.



◆ Baseline RAG:

Based on the provided information, here is a comparison of the ECU Reset and Tester Present

services and their relationship. The context does not contain information about a "Link control" service.

Service Comparison

ECUReset (Service ID: 0x11)

- **Purpose:** This service is used by a client to request that a server performs a reset.
- **Functionality:** It instructs the server to execute a reset based on a specified `resetType`. Available reset types include:
 - `hardReset` : Simulates disconnecting and reconnecting the power supply, potentially re-initializing both volatile and non-volatile memory.
 - `keyOffOnReset` : Simulates turning the ignition key off and on.
 - `softReset` : Restarts the application program, typically without reinitializing learned data.
 - `enableRapidPowerShutDown` : For battery-powered ECUs, this initiates a quick transition to sleep mode when the ignition is turned off.
- **Outcome:** After a successful reset, the server activates the `defaultSession`. A positive response message, if requested, is sent before the reset is executed.

TesterPresent (Service ID: 0x3E)

- **Purpose:** This service is used to signal to a server that a client is still connected.
- **Functionality:** Its primary function is to keep the server in a diagnostic session other than the `defaultSession`. By sending this request periodically, the client prevents the server from automatically timing out and returning to the `defaultSession`.
- **Outcome:** The server remains in its current, non-default diagnostic session.

Relationship Between ECUReset and TesterPresent

The relationship between ECUReset and TesterPresent centers on the management of the server's diagnostic session state.

- The **ECUReset** service is a command that forces the server to end its current operations, perform a reset, and return to the foundational `defaultSession`.
- Conversely, the **TesterPresent** service acts as a "keep-alive" signal, specifically used to *prevent* the server from returning to the `defaultSession` and to maintain an active, non-

default diagnostic session.

In essence, they have opposing effects on the server's session state: ECUReset terminates the current state and reverts to default, while TesterPresent sustains the current non-default state.



What are the NRC mentioned ECU Reset



◆ **TopAgent (Multi-agent reasoning):**

The Negative Response Codes (NRCs) that can be sent by an ECU in response to an ECU Reset (0x11) request are:

- **0x12 (sub-functionNotSupported):** This code is sent if the ECU does not support the specific type of reset requested (e.g., hardReset, softReset).
- **0x13 (incorrectMessageLengthOrInvalidFormat):** This indicates that the request message sent by the client has an incorrect length or format.
- **0x22 (conditionsNotCorrect):** This code is used when the ECU is not in the correct state to perform the reset. For example, certain operational conditions might prevent a reset from being safely executed.
- **0x33 (securityAccessDenied):** This response is sent if the requested reset type requires security access to be unlocked, and the client has not yet gained the necessary security privileges.



◆ Baseline RAG:

The Negative Response Codes (NRC) for an ECU Reset are as follows:

- **0x12 (SFNS):** `sub-functionNotSupported` - This code is sent if the sub-function parameter is not supported.
- **0x13 (IMLOIF):** `incorrectMessageLengthOrInvalidFormat` - This code is sent if the message length is incorrect.
- **0x22 (CNC):** `conditionsNotCorrect` - This code is returned if the criteria for the ECUReset request are not met.
- **0x33 (SAD):** `securityAccessDenied` - This code is sent if the requested reset is secured and the server is not in an unlocked state.



In a scenario where the ECU is reset frequently, how should Tester Present messages be scheduled?



◆ TopAgent (Multi-agent reasoning):

When an ECU is reset, it typically terminates the current diagnostic session and reverts to the default session. The purpose of the Tester Present (0x3E) message is to keep a non-default session (like the extended session) active.

Therefore, the scheduling should be handled as follows:

1. **Stop Tester Present:** Before sending the `ECUReset` (0x11) command, you should stop the periodic transmission of Tester Present messages.
2. **Perform ECU Reset:** Send the `ECUReset` command.
3. **Wait for Reboot:** Allow the ECU sufficient time to complete its reset and reboot sequence.
4. **Re-establish Session:** After the ECU is back online, you must start a new diagnostic session (e.g., using `DiagnosticSessionControl`, 0x10) if you need to perform actions outside the default session.
5. **Resume Tester Present:** Once the new non-default session is successfully established, you must resume sending Tester Present messages periodically to keep this new session active.

In short, sending Tester Present messages *during* or *immediately after* a reset is ineffective because the reset terminates the very session that the Tester Present message is designed to

maintain. You must always re-establish the session first after a reset.



◆ **Baseline RAG:**

Based on the provided information, the ECUReset service is used to request a server reset. After a successful reset, the server is required to activate the `defaultSession`.

The TesterPresent service is used to keep a server in a diagnostic session other than the `defaultSession`. Since an ECUReset forces the server back to the `defaultSession`, any active non-default session maintained by TesterPresent would be terminated.

Furthermore, it is recommended that from the time the server sends a positive response to an ECUReset request until the reset is complete, the ECU should not accept any request messages. Therefore, sending a TesterPresent message during this period would be ineffective.



Ask about UDS services...

