RobotFramework_UDS

v. 0.1.0

TODO

23.08.2024

Contents

Chapter 1

Introduction

1.1 TODO

TODO

Chapter 2

Description

2.1 TODO

TODO

Chapter 3

DiagnosticServices.py

3.1 Class: DiagnosticServices

Imported by:

```
from RobotFramework_UDS.DiagnosticServices import DiagnosticServices
```

3.1.1 Method: read_data_by_name

3.1.2 Method: get_encoded_request_message

Chapter 4

UDSKeywords.py

4.1 Class: UDSKeywords

Imported by:

```
from RobotFramework_UDS.UDSKeywords import UDSKeywords
```

4.1.1 Method: connect_uds_connector

4.1.2 Method: load_pdx

Description: Load PDX

Parameters:

- param `pdx_file`: pdx file path
- type `pdx_file`: str
- param `variant`:
- type `variant`: str

4.1.3 Method: build_payload

4.1.4 Method: send_request

4.1.5 Method: interpret_response_data

4.1.6 Method: validate_content_response

Validates the content of a UDS response.

param response The UDS response object to validate.

param expected_service The expected service ID of the response.

param expected_data The expected data (optional) to be matched within the response.

return True if the response is valid, False otherwise.

4.1.7 Method: create_config

Description: Create a config for UDS connector

Parameters: Will be update later

4.1.8 Method: set_config

Description: Set UDS config Using create_configure to create a new config for UDS connector. If not, the default config will be use.

4.1.9 Method: connect

Description: Open uds connection

4.1.10 Method: disconnect

Description: Close uds connection

4.1.11 Method: access_timing_parameter

Description: Sends a generic request for AccessTimingParameter service

Parameters:

- param `access_type` (required): The service subfunction
  - readExtendedTimingParameterSet = 1
  - setTimingParametersToDefaultValues = 2
  - readCurrentlyActiveTimingParameters = 3
  - setTimingParametersToGivenValues = 4
- type `access_type` int
- param `timing_param_record` (optional): The parameters data. Specific to each ECU.
- type `timing_param_record` bytes

4.1.12 Method: clear_dianostic_infomation

Description: Requests the server to clear its active Diagnostic Trouble Codes.

Parameters:

- param `group`: The group of DTCs to clear. It may refer to Powertrain DTCs, Chassis DTCs, etc. Values are defined by the ECU manufacturer except for two specific values
  - `0x000000` : Emissions-related systems
  - `0xFFFFFF` : All DTCs
- type `group`: int
- **param memory_selection: MemorySelection byte (0-0xFF). This value is user defined and introduced** Only added to the request payload when different from None. Default : None
- type `memory_selection`: int

4.1.13 Method: communication_control

Description: Switches the transmission or reception of certain messages on/off with CommunicationControl service.

Parameter:

- param `control_type` (required): The action to request such as enabling or disabling some messages. This value can also be ECU manufacturer-specific
  - enableRxAndTx = 0
  - enableRxAndDisableTx = 1
  - disableRxAndEnableTx = 2
  - disableRxAndTx = 3

  - enableRxAndDisableTxWithEnhancedAddressInformation = 4
  - enableRxAndTxWithEnhancedAddressInformation = 5
- type `control_type`: int
- param `communication_type` (required): Indicates what section of the network and the type of message that should be affected by the command. Refer to `CommunicationType<CommunicationType>` for more details. If an integer or a bytes is given, the value will be decoded to create the required `CommunicationType<CommunicationType>` object
- type communication_type: `CommunicationType<CommunicationType>`, bytes, int

* param `node_id` (optional): DTC memory identifier (nodeIdentificationNumber). This value is user defined and introduced in 2013 version of ISO-14229-1. Possible only when control type is `enableRxAndDisableTxWithEnhancedAddressInformation` or `enableRxAndTxWithEnhancedAddressInfo` Only added to the request payload when different from None. Default : None * type `node_id`: int

4.1.14 Method: control_dtc_setting

Description: Controls some settings related to the Diagnostic Trouble Codes by sending a ControlDTCSetting service request. It can enable/disable some DTCs or perform some ECU specific configuration.

Paramters:

- **param `setting_type` (required): Allowed values are from 0 to 0x7F.**
  - on = 1
  - off = 2
  - vehicleManufacturerSpecific = (0x40, 0x5F) # To be able to print textual name for logging only.
  - systemSupplierSpecific = (0x60, 0x7E) # To be able to print textual name for logging only.
- type `setting_type`: int
- param `data` (optional): Optional additional data sent with the request called DTCSettingControlOptionRecord
- type `data`: bytes

4.1.15 Method: diagnostic_session_control

Description: Requests the server to change the diagnostic session with a DiagnosticSessionControl service request.

Parameters:

- **param `newsession` (required): The session to try to switch.**
  - defaultSession = 1
  - programmingSession = 2
  - extendedDiagnosticSession = 3
  - safetySystemDiagnosticSession = 4
- type `newsession`: int

4.1.16 Method: dynamically_define_did

Description: Defines a dynamically defined DID.

Parameters:

- param `did`: The data identifier to define.
- type `did`: int
- **param `did_definition`: The definition of the DID. Can be defined by source DID or memory addre**
  If a `MemoryLocation<MemoryLocation>` object is given, definition will automatically be by memory address
- type `did_definition`: `DynamicDidDefinition<DynamicDidDefinition>` or `MemoryLocation<Memory`

4.1.17 Method: ecu_reset

Requests the server to execute a reset sequence through the ECUReset service.

- **param `reset_type` (required): The type of reset to perform.**
  - hardReset = 1
  - keyOffOnReset = 2
  - softReset = 3
  - enableRapidPowerShutDown = 4
  - disableRapidPowerShutDown = 5
- type reset_type: int

4.1.18 Method: io_control

Description: Substitutes the value of an input signal or overrides the state of an output by sending a InputOutputControlByIdentifier service request.

Parameters:

- param `did` (required): Data identifier to represent the IO
- type "did': int
- **param `control_param` (optional):**
  - returnControlToECU = 0
  - resetToDefault = 1
  - freezeCurrentState = 2
  - shortTermAdjustment = 3
- type `control_param`: int
- **param `values` (optional): Optional values to send to the server. This parameter will be given to Dic**
  - A list for positional arguments
  - A dict for named arguments
  - An instance of IOValues<IOValues> for mixed arguments
- type `values`: list, dict, IOValues<IOValues>
- **param masks: Optional mask record for composite values. The mask definition must be included in**
  - A list naming the bit mask to set
  - A dict with the mask name as a key and a boolean setting or clearing the mask as the value
  - An instance of IOMask<IOMask
  - A boolean value to set all masks to the same value.
- type masks: list, dict, IOMask<IOMask>, bool

4.1.19 Method: link_control

Description: Controls the communication baudrate by sending a LinkControl service request.

Parameters:

- **param `control_type` (required): Allowed values are from 0 to 0xFF.**
  - verifyBaudrateTransitionWithFixedBaudrate = 1
  - verifyBaudrateTransitionWithSpecificBaudrate = 2
  - transitionBaudrate = 3
- type `control_type`: int
- param `baudrate` (required): Required baudrate value when `control_type` is either `verifyBaudrateTransitio` (1) or `verifyBaudrateTransitionWithSpecificBaudrate` (2)
- type `baudrate`: Baudrate <Baudrate>

4.1.20 Method: read_data_by_identifier

Description: Requests a value associated with a data identifier (DID) through the ReadDataByIdentifier<ReadDataByI service.

Parameters:

See an example<reading_a_did> about how to read a DID

- param data_id_list: The list of DID to be read

* type data_id_list: int | list[int] robotframework-uds-udskeywords-udskeywords-read-dtc-information ----------

Update later robotframework-uds-udskeywords-udskeywords-read-memory-by-address ----------

Description: Reads a block of memory from the server by sending a ReadMemoryByAddress service request.

Parameters:

- param memory_location (required): The address and the size of the memory block to read.

* type memory_location: MemoryLocation <MemoryLocation> robotframework-uds-udskeywords-udskeywords-request-download ----------

Description: Informs the server that the client wants to initiate a download from the client to the server by sending a RequestDownload service request.

Effective configuration exception_on_<type>_response server_address_format server_memorysize_format

Parameters:

- param memory_location (required): The address and size of the memory block to be written.
- type memory_location: MemoryLocation <MemoryLocation>
- **param dfi (optional): Optional defining the compression and encryption scheme of the data.** If not specified, the default value of 00 will be used, specifying no encryption and no compression
- type dfi: DataFormatIdentifier <DataFormatIdentifier>

4.1.21 Method: request_transfer_exit

Description: Informs the server that the client wants to stop the data transfer by sending a RequestTransferExit service request.

Effective configuration exception_on_<type>_response

Parameters:

- param data (optional): Optional additional data to send to the server
- type data: bytes

4.1.22 Method: request_upload

Description: Informs the server that the client wants to initiate an upload from the server to the client by sending a RequestUpload<RequestUpload> service request.

Effective configuration exception_on_<type>_response server_address_format server_memorysize_format

Parameters:

- param memory_location (required): The address and size of the memory block to be written.
- type memory_location: MemoryLocation <MemoryLocation>

*** param dfi (optional): Optional defining the compression and encryption scheme of the data.** If not specified, the default value of 00 will be used, specifying no encryption and no compression

*type dfi: DataFormatIdentifier <DataFormatIdentifier>

4.1.23 Method: routine_control

Description: Sends a generic request for the RoutineControl service

Parameters:

- param `routine_id` (required): The 16-bit numerical ID of the routine
- type `routine_id` int
- param `control_type` (required): The service subfunction
- type `control_type` int
- **valid `control_type`**
  - startRoutine = 1
  - stopRoutine = 2
  - requestRoutineResults = 3
- param `data` (optional): Optional additional data to give to the server
- type `data` bytes

4.1.24 Method: security_access

Description: Successively calls request_seed and send_key to unlock a security level with the SecurityAccess service. The key computation is done by calling config['security_algo']

Effective configuration `exception_on_<type>_response` `security_algo` `security_algo_params`

Parameters:

- param `level` (required): The level to unlock. Can be the odd or even variant of it.
- type `level`: int
- param `seed_params` (optional): Optional data to attach to the RequestSeed request (securityAccessDataRecord).
- type `seed_params`: bytes

4.1.25 Method: tester_present

Description: Sends a TesterPresent request to keep the session active.

Effective configuration `exception_on_<type>_response`

4.1.26 Method: transfer_data

Description: Transfer a block of data to/from the client to/from the server by sending a TransferData service request and returning the server response.

Effective configuration `exception_on_<type>_response`

Parameters:

- **param `sequence_number` (required): Corresponds to an 8bit counter that should increment for each** Allowed values are from 0 to 0xFF
- type `sequence_number`: int
- param `data` (optional): Optional additional data to send to the server
- type `data`: bytes

4.1.27 Method: write_data_by_identifier

Description: Requests to write a value associated with a data identifier (DID) through the WriteDataByIdentifier service.

Effective configuration `exception_on_<type>_response` `data_identifiers`

Parameters:

- param did: The DID to write its value
- type did: int
- param value: Value given to the DidCodec.encode method. The payload returned by the codec will be sent to the server.
- type value: int

4.1.28 Method: write_memory_by_address

Description: Writes a block of memory in the server by sending a WriteMemoryByAddress service request.

Effective configuration `exception_on_<type>_response` `server_address_format` `server_memorysize_format`

Parameters:

- param `memory_location` (required): The address and the size of the memory block to read.
- type `memory_location`: MemoryLocation <MemoryLocation>
- param `data` (required): The data to write into memory.
- type `data`: bytes

4.1.29 Method: request_file_transfer

Parameters:

- **param `moop` (required): Mode operate**
  - AddFile = 1
  - DeleteFile = 2
  - ReplaceFile = 3
  - ReadFile = 4
  - ReadDir = 5
  - ResumeFile = 6
- type `moop`: int
- param `path` (required):
- type `path`: str
- **param `dfi`: DataFormatIdentifier defining the compression and encryption scheme of the data.** If not specified, the default value of 00 will be used, specifying no encryption and no compression. Use for `moop`: - AddFile = 1 - ReplaceFile = 3 - ReadFile = 4 - ResumeFile = 6
- type `dfi`: DataFormatIdentifier

* param `filesize` (optional): The filesize of the file to write. If filesize is an object of type Filesize<Filesize>, the uncompressed size and compressed size will be encoded on the minimum amount of bytes necessary, unless a `width` is explicitly defined. If no compressed size is given or filesize is an `int`, then the compressed size will be set equal to the uncompressed size or the integer value given as specified by ISO-14229 Use for `moop`: - AddFile = 1 - ReplaceFile = 3 - ResumeFile = 6

- type `filesize`: int | Filesize

4.1.30 Method: authentication

Description: Sends an Authentication request introduced in 2020 version of ISO-14229-1. You can also use the helper functions to send each authentication task (sub function).

Effective configuration `exception_on_<type>_response`

Parameters:

- **param `authentication_task` (required): The authenticationTask (subfunction) to use.**
  - deAuthenticate = 0
  - verifyCertificateUnidirectional = 1
  - verifyCertificateBidirectional = 2
  - proofOfOwnership = 3
  - transmitCertificate = 4
  - requestChallengeForAuthentication = 5
  - verifyProofOfOwnershipUnidirectional = 6
  - verifyProofOfOwnershipBidirectional = 7
  - authenticationConfiguration = 8
- type `authentication_task`: int
- **param `communication_configuration` (optional): Optional Configuration information about how to** Allowed values are from 0 to 255.
- type `communication_configuration`: int
- param `certificate_client` (optional): Optional The Certificate to verify.
- type `certificate_client`: bytes
- param `challenge_client` (optional): Optional The challenge contains vehicle manufacturer specific formatted client data (likely containing randomized information) or is a random number.
- type `challenge_client`: bytes
- **param `algorithm_indicator` (optional): Optional Indicates the algorithm used in the generating an** which further determines the parameters used in the algorithm and possibly the session key creation mode. This field is a 16 byte value containing the BER encoded OID value of the algorithm used. The value is left aligned and right padded with zero up to 16 bytes.
- type `algorithm_indicator`: bytes
- **param `certificate_evaluation_id`: Optional unique ID to identify the evaluation type of the trans** The value of this parameter is vehicle manufacturer specific. Subsequent diagnostic requests with the same evaluationTypeId will overwrite the certificate data of the previous requests. Allowed values are from 0 to 0xFFFF.
- type certificate_evaluation_id: int
- param `certificate_data` (optional): Optional The Certificate to verify.
- type `certificate_data`: bytes
- param `proof_of_ownership_client` (optional): Optional Proof of Ownership of the previous given challenge to be verified by the server.
- type `proof_of_ownership_client`: bytes
- param `ephemeral_public_key_client` (optional): Optional Ephemeral public key generated by the client for Diffie-Hellman key agreement.
- type `ephemeral_public_key_client`: bytes
- param `additional_parameter` (optional): Optional additional parameter is provided to the server if the server indicates as neededAdditionalParameter.
- type `additional_parameter`: bytes

4.1.31 Method: routine_control_by_name

Description: Sends a request for the RoutineControl service by routine name

Parameters:

- param `routine_name` (required): Name of routine
- type `routine_name`: str
- param `control_type` (required): The service subfunction
- type `control_type` int
- **valid `control_type`**
  - startRoutine = 1
  - stopRoutine = 2
  - requestRoutineResults = 3
- param `data` (optional): Optional additional data to give to the server
- type `data` bytes

4.1.32 Method: read_data_by_name

Description: Get diagnostic service list by list of service name

Parameters:

- param `service_name_list`: list of service name
- type `service_name_list`: list[str]

4.1.33 Method: get_encoded_request_message

Description: Get diagnostic service encoded request list (hex value)

Parameters:

- param `diag_service_list`: Diagnostic service list
- type `diag_service_list`: []
- param `parameters`: parameter list
- type `parameters`: list[]

Chapter 5

__init__.py

5.1 Class: RobotFramework_UDS

Imported by:

```
from RobotFramework_UDS.__init__ import RobotFramework_UDS
```

RobotFramework_UDS is a Robot Framework library aimed to provide UDP client to handle request/response.

Chapter 6

Appendix

About this package:

Table 6.1: Package setup

Setup parameter	Value
Name	RobotFramework_UDS
Version	0.1.0
Date	23.08.2024
Description	TODO
Package URL	robotframework-uds
Author	TODO
Email	TODO
Language	Programming Language :: Python :: 3
License	License :: OSI Approved :: Apache Software License
OS	Operating System :: OS Independent
Python required	>=3.0
Development status	Development Status :: 4 - Beta
Intended audience	Intended Audience :: Developers
Topic	Topic :: Software Development

Chapter 7

History

0.1.0	09/2024
Initial version	

RobotFramework_UDS.pdf
Created at 27.08.2024 - 17:59:50
by GenPackageDoc v. 0.41.1