**Export: Descriptions.tdltx**

### X.Y.1 Descriptions

Table X.Y.1-1: Test Purpose TPD\_MDD\_01

|  |  |
| --- | --- |
| **TP Id** | TPD\_MDD\_01 |
| **Test Objective** | Issue a request via Postman to all domain controllers, to check the overall viability of the test network. |
| **Reference** | ETSI Plugtests Test Plan V1.0 (2020-11), 8.3.2 |
| **Configuration** | TestLanArchitecture |
| **PICS Selection** | (MW\_8040 or MW\_8345) |
| Initial Conditions | |
| //etc. with {  //initial textual  perform action: "Postman has been correctly initialized earlier, by executing  TD\_POSTMAN\_INIT"  perform action: "All the Domain Controller instances up and running normally"  perform action: "All the devices are upgraded to correct versions"  perform action: "All basic configurations are completed (e.g., NE\_id, OSPF, PCEP, etc.)"  //then gradually formalised, e.g.  execute DomainControllersInitialisation  execute DevicesUpgrade  execute BasicConfigurationsSetup  //... } | |
| **Expected Behaviour** | |
| ensure that {  when {  //initial textual  perform action: "Send GET request via Postman to all domain controllers by  executing Collection TD\_MDD\_01"  //refine to first structure  //TODO: what happens to all the parentheses?!?! -> make filter optional (or remove?)  pm::mpi sends rGET() to X::mpi  //refine to specific uri inline  pm::mpi sends rGET(  uri = "{{TD\_SDN\_RESTCONF\_PATH}}/data/ietf-yang-library:modules-state"  ) to X::mpi  //...or define elsewhere  pm::mpi sends rGET(  uri = modules\_state\_uri  ) to X::mpi  }  then {  //initial textual  perform action: "Check the response body of each request and confirm if all the  Restconf servers are serviceable."  perform action: "The response body of the request should contain a list of all  YANG modules and submodules used by the Restconf server along with  information about name and revision for each module."  perform action: "The response body of each query should contain the specified YANG  module along with its name and revision."  //refinements  // Test 1: Check the response body of the above request and confirm if the Restconf  // server is serviceable.  pm::mpi receives OK() from X::mpi  // Step 2: The response body of the request should contain a list of all YANG modules  //and submodules used by the Restconf server along with information about name and revision  //for each module.  //---  //asserts can be used but probably simpler to use a specification of the expected data  //optionally: explicit objective that can be checked  Objective: ResponseContainsAllYanModules  pm::mpi receives OK(  body = JSON\_instance\_modules\_state(  modules\_state = ?,  modules\_state = ?,  //TODO: do we need a pattern ? / \* for collections? Otherwise we need a loop instead  //TODO: as a side note, is checking for the presence of a property  // equivalent to that property having any value?  modules[\*] = (name = ?, revision = ?, namespace = ?)  //...  )) from X::mpi  //asserts can provide fine granular diagnostics though, for individual problems  response = pm::mpi receives OK() from X::mpi  //optionally: explicit objective that can be checked  Objective: ResponseContainsAllYanModules  //optionally: annotations can be used for inline information  @Failure: "The response body of the request should contain a list of all YANG modules"  //TODO: property access for subclasses?  //TODO: do we have matching expressions in assertions as well  //TODO: do we have checking of types  assert ( pm::response.body == JSON\_instance\_modules\_state(  modules\_state = ?,  //TODO: do we need a pattern ? / \* for collections? Otherwise we need a loop instead  //TODO: as a side note, is checking for the presence of a property  // equivalent to that property having any value?  modules[\*] = (name = ?, revision = ?, namespace = ?)  //  ))  on pm //may be skipped at first  with {  timeLabel=now  }  //alternatively with a loop  //TODO: do we need counters? integrated in loops? e.g. with state to address collections?  pm::i = 0  repeat 5 times on pm  //[MW\_8040]  {  //DONE: fixed constraint with bounded loop behaviour, still not quite there  assert ( pm::response.body == JSON\_instance\_modules\_state(  modules\_state = ?,  //TODO: as a side note, is checking for the presence of a property  // equivalent to that property having any value?  modules[pm::i] = (name = ?, revision = ?, namespace = ?)  //  ))  on pm //may be skipped at first  //DONE: handle undefined return types -> a bit of a hack, assuming they are the same  pm::i = ( pm::i + 1 )  }  } } | |

Table X.Y.1-2: Test Purpose TPD\_MSP\_01

|  |  |
| --- | --- |
| **TP Id** | TPD\_MSP\_01 |
| **Test Objective** | Create the specified L2 data services over all available domains. The TGA confirms that data start flowing. The traffic on the inter-domain links is classified based on S- VLAN only. |
| **Reference** | ETSI Plugtests Test Plan V1.0 (2020-11), 8.5.2 |
| **Configuration** | TestLanArchitecture |
| **PICS Selection** | (MW\_8040 or MW\_8345) |
| Initial Conditions | |
| //etc. with {  //initial textual  perform action: "Postman has been correctly initialized earlier, by executing TD\_POSTMAN\_INIT"  perform action: "The Domain Controller instance is up and running normally"  perform action: "All the devices are upgraded to correct versions"  perform action: "All basic configurations are completed (e.g., NE\_id, OSPF, PCEP, etc.)"  perform action: "All Restconf servers are serviceable."  perform action: "If TD\_SSP\_01 has been run before TD\_ MSP\_01, all Domain Controllers and  microwave units should be reset to the state they were before executing  TD\_SSP\_01"  //then gradually formalised, e.g.  execute DomainControllerInitialisation  execute DomainControllerInitialisation  execute DevicesUpgrade  execute BasicConfigurationsSetup  if [TD\_SSP\_01\_Completed] {  execute ResetDomainControllers  execute ResetMicrowaveUnits  }  //... } | |
| **Expected Behaviour** | |
| ensure that {  when {  //initial textual  perform action: "Send POST request via Postman to all domain controllers  by executing Collection TD\_ SSP\_01" // //refine to first structure // pm::mpi sends rPOST() to X::mpi // //refine to specific body (or define inline) // pm::mpi sends rPOST(body = JSON\_instance) to X::mpi // // //provide additional parameter overrides // pm::mpi sends rPOST( // body = JSON\_instance ( // services[0].adminStatus = "up", // //.. // //TODO: check constraints // services[0].ports[1].accessNodeId = node1 // //.. // ) // ) // to X::mpi //....  }  then {  //initial textual  perform action: "Check the TGA if the data start flowing properly." // pm::mpi receives OK() from X::mpi // pm::mpi receives OK(body = JSON\_instance) from X::mpi // pm::mpi receives OK( // body = JSON\_instance( // //.. // services[2] = omit, // //... // services[3].ports[1] = omit // ) // ) // from X::mpi // //first response definition, then refined, also with overrides // //!TODO: Why are comments before receive message not OK? // //... refine further  } } | |

Table X.Y.1-3: Test Purpose TPD\_MSP\_04

|  |  |
| --- | --- |
| **TP Id** | TPD\_MSP\_04 |
| **Test Objective** | The Ethernet service information is requested from all the DCs under test in TD\_MSP\_03. The received information is checked to correctly not list the newly deleted services anymore. |
| **Reference** | ETSI Plugtests Test Plan V1.0 (2020-11), 8.5.2 |
| **Configuration** | TestLanArchitecture |
| **PICS Selection** | (MW\_8040 or MW\_8345) |
| Initial Conditions | |
| //etc. with {  //initial textual  perform action: "Postman has been correctly initialized earlier, by executing TD\_POSTMAN\_INIT"  perform action: "All the Domain Controller instances are up and running normally"  perform action: "All the devices are upgraded to correct versions"  perform action: "All basic configurations are completed (e.g., NE\_id, OSPF, PCEP, etc.)"  perform action: "All Restconf servers are serviceable."  perform action: "All L2 data services are successfully created."  //then gradually formalised, e.g.  execute DomainControllersInitialisation  execute DevicesUpgrade  execute BasicConfigurationsSetup  execute L2DataServicesCreation  //... } | |
| **Expected Behaviour** | |
| ensure that {  when {  //initial textual  perform action: "Send GET request via Postman to  all domain controllers by executing  Collection TD\_MSP\_04"  //refine to first structure  pm::mpi sends rGET() to X::mpi  //refine to specific body (or define inline)  pm::mpi sends rGET(body = JSON\_instance) to X::mpi  //provide additional parameter overrides  pm::mpi sends rGET(  body = JSON\_instance (  services[0].adminStatus = "up" {JSON\_String},  //..  //DONE: check constraints  services[0].ports[1].accessNodeId = node1  //..  )  )  to X::mpi //....  }  then {  //initial textual  perform action: "The response body should no longer contain  information about the L2 service deleted in TD\_MSP\_03"  //basic response definition  pm::mpi receives OK() from X::mpi  //refined with reference to body definition  pm::mpi receives OK(body = JSON\_instance) from X::mpi  //refined with inline overrides  pm::mpi receives OK(  body = JSON\_instance(  //..  //TODO: does an optional collection member imply entire collection is ommitted?  // or also individual items?  //TODO: does special value use conform to everything?  services[2] = omit,  //...  services[3].ports[1] = omit  )  )  from X::mpi  //... refine further  } } | |