Verification Router Service

Request and Response
Messaging Standard

Version:	1.0
Date:	27-APR-2018

Table of Contents

Revision History

Version	Date	Revision Description
0.1	15-FEB-2018	Initial draft assembled from 2016 pilot content
0.2	20-FEB-2018	Updated version from initial peer review
0.3	23-FEB-2018	Update from VRS technical team meeting
0.4	2-MAR-2018	Updates during VRS technical sub-group meeting
0.5	5-MAR-2018	Additional updates from 2-Mar-2018 meeting
0.6	7-MAR-2018	Follow-up revisions from 6-Mar-2018 meeting
0.7	9-MAR-2018	Content placed into HDA template
0.8	19-MAR-2018	Updates from 16-Mar-2018 meeting
0.9	20-MAR-2018	Updates from technical review
1.0	27-APR-2018	Released as v1.0

1. Verification Router Service: Request and Response Messaging Standard

1.1. Background on VRS

Healthcare Distribution Alliance (HDA) formed the Traceability Pilots Work Group in 2016 to focus on a pilot study of nine (9) potential methods or solutions to meet the 2019 Saleable Returns DSCSA Requirements. Through the process of evaluating nine scenarios, the Work Group acknowledged no single solution for the supply chain existed, and put forward two preferred options, keeping in mind solution cost, implementation effort, process execution, exception handling, and other advantages and disadvantages.

One of the options studied in the pilot was a Verification Router Service (VRS). A proof of concept was successfully built and utilized during a live pilot, but it was a temporary system designed solely for the purposes of the pilot. (For the full pilots report, see https://healthcaredistribution.org/resources/hda-saleable-returns-pilots-report.) At the conclusion of the pilot study, the Work Group concluded that the Verification Router Service was a verification method that should be pursued.

HDA subsequently formed a task force consisting of industry members and later expanded the team to include solution providers in order to mobilize efforts for defining the business requirements, high-level functionality, solution components, component interactions, minimal required data and recommended communications protocols. With assistance from KPMG LLP, HDA and the task force members conducted virtual meetings and in-person workshops throughout 2017. The resulting output is consolidated into two documents:

1. VRS Business Requirements Document (BRD)
2. Solution Architecture Reference Document (SARD)

Additional technical specifications are required to support the framework provided in the BRD and SARD documents.

One of these specification is to define the format and content for the Request and Response messages that support interactions 4-6 (requesting processes) and 7-9 (responding processes) from the SARD.

It is anticipated that this document will be used by solution providers to support their build and testing activities in 2018. Once approved, frequent updates to this document are not anticipated.

1.2. Request Messaging Types and Samples

The request message is in support of SARD interactions 4 through 6. This includes the initiation of a Verification Request (VR) via Requestor (*e.g.* Distributor) using either their own internal system interfaced to the Requestor's VRS or via Requestor using their VRS Provider's system to direct the request to the Responder (*e.g.* Manufacturer) system in order to provide a response.

1.2.1. Message Format

POST request:

Resource Path:
{version}/asset/gtin/{gtin}/sn/{sn}/{requestType}

- EPC_IDENTIFIER – The identifier consisting of the GTIN and serial number (sn) for the Asset whose validation is being requested. requestType – enumerated request type. 000 and 001 will be used for connectivity testing and saleable returns verifications. 002 and 003 have been listed to indicate that additional request type values can be added in the future as needed.
    - 000 – Connectivity/Communication Testing
    - 001 – Saleable Return Verification Request
    - 002 – Placeholder for additional requestType
    - 003 – Placeholder for additional requestType

Request Body: JSON keys

- correlationGuid – unique identifier generated by the distributor
- requestorGln – Global location number for requestor that is making request - 13 characters long
- data (unique to requestType and the EPC_IDENTIFIER):
    - lotNum – Lot number for the item to be looked up
    - expiry – Date of expiry for the item to be looked up in format YYMMDD

1.2.2. Sample Request using Request Type = “001” (Saleable Return Verification Request)

```
POST /v1/asset/gtin/00361414567894/sn/400806/001
HTTP/1.1
Host: 3and9.org
Accept: application/json
Content-Type: application/json

{
   "correlationGuid": "21EC2020-3AEA-4069-A2DD-
08002B30309D",
   "requestorGln": "0321012345676",
   "data" : {
     "lotNum" : "1908642E",
      "expiry" : "170728"
   }
}
```

1.2.3. Sample Request using Request Type = “000” (Connectivity / Communication Testing)

```
POST /v1/asset/gtin/00361414567894/sn/400806/000
HTTP/1.1
Host: 3and9.org
Accept: application/json
Content-Type: application/json



{
   "correlationGuid": "998CDC77-6860-4351-9277-
6F3E6F870AC6",
   "requestorGln": "0321012345676",
}
```

1.3. Response Types

The response message is in support of SARD interactions 7 through 9. This includes the formulation of the response from the Responder (*e.g.* Manufacturer) system and handling of any errors.

1.3.1. Response Types

There are five possible response types as listed in the following table:

#	Response Type	Description
1	No response	Due to some failures there might not be any response to the HTTP request. In this case, the HTTP client (either distributor system in Distributor-to-VRS or VRS system in VRS-to-Manufacturer communication) will reach a connection timeout and will handle the error according to their business logic. The initial timeout by distributor should be set to 15s and will be adjusted after collecting real-time end-to-end application performance.
2	HTTP error response	In Requestor-to-VRS communication, if the VRS system/application encounters certain errors and/or is not able to send the request to manufacture the VRS system will return the HTTP error response with specific error code in response headers; in VRS-to-Manufacturer communication, if the Manufacturer system/application encounters errors it will return the HTTP error response with specific error code in response headers.
3	Application error response	In Distributor-to-VRS communication, if the VRS application receives errors from manufacturer, VRS application will return HTTP 200 response, and will include JSON formatted error message in response body.
4	Failed lookup response	If the VRS failed to verify the return item (the data sent for lookup did not match the data available), VRS will return HTTP 200 response, and will include JSON formatted message indicating the failed lookup in response body.

#	Response Type	Description
5	Successful lookup response	If the VRS or system successfully verified the return item PI data, VRS or Manufacturer will return HTTP 200 response, and will include JSON formatted message indicating the successful lookup in response body.

1.4. Response Codes

The VRS solution may respond with any of the HTTP codes listed in the following table:

Code	Description
200	A response code of 200 means the request was successful and details about the response can be found in the body of the response. Only a 200 response will issue a JSON payload.
400	The request was not formatted properly. Please verify the request conforms to this specification, and re-issue the request in the correct format.
401	The request was not allowed because the request did not pass authentication.
404	The requested resource does not exist.
408	The server timed out waiting for the request.
500	System failed to process the request because of an error inside the system.
502	Bad Gateway – The server was acting as a gateway or proxy and received an invalid response from the upstream server. Indicates that one server tried to use another VRS system and that system was down.
503	System is undergoing maintenance or is otherwise temporarily unavailable for API queries.
504	The server, while acting as a gateway or proxy, performed multiple retries but did not receive a timely response from the upstream server specified by the URI (e.g. HTTP, FTP, LDAP) or some other auxiliary server (e.g. DNS) it needed to access in attempting to complete the request.

1.5. Response Entities

POST methods respond with the JSON. Entities returned in response:

- responderGln – Global location number for the responder that is making request - 13 characters long
- data: (Unique to request type)
    - verified – true or false. If ALL 4 elements of the PI were matched – Always use true, else provide "false" with optional "verifiedInfo".
    - verifiedInfo – optional: Only used if "verified" is "false" to represent which PI element(s) did not match **Instead of showing empty string "" or null, do NOT include unless there's a value.**
        - 001 – no matching sGTIN
        - 002 – no matching lotNum
        - 003 – no matching expiry
        - 004 – no matching lotNum & expiry
- additionalInfo – optional: will be used if response is "true" to provide additional information of the state of the sGTIN, *e.g.* recalled. These codes can range from 001 to 099 with various statuses. **Instead of showing empty string "" or null, do NOT include unless there is a value.**
    - 001 – do not re-sell; recalled
    - 002 – placeholder for future use
    - 003 – placeholder for future use

- errorMessage - Optional If needed (e.g. not the Responder for PI). **Instead of showing empty string "" or null, do NOT include unless there's value.**
- correlationGuid – unique identifier generated by the distributor

1.6. Response Sample: Successful Lookup (with additionalInfo included)

```
This is an example of a successful lookup response to a
saleable return verification request whose
correlationGuid is 21EC2020-3AEA-4069-A2DD-08002B30309D.

HTTP 1.1 200 OK
Cache-Control: private, no-cache
Content-Type: application/json

{
  "responderGln": "0312231245676",
  "data" : {
     "verified": true,
     "additionalInfo": "001"
  },
  "correlationGuid": "21EC2020-3AEA-4069-A2DD-
08002B30309D"
}
```

1.7. Response Sample: Failed Lookup (with verifiedInfo included)

This is an example of a failed lookup response to a saleable return verification request whose correlationGuid is 21EC2020-3AEA-4069-A2DD-08002B30309D.

```
HTTP 1.1 200 OK
Cache-Control: private, no-cache
Content-Type: application/json

{
 "responderGln": "0312231245676",
  "data" : {
     "verified": false,
     "verifiedInfo": "001"
  },
  "correlationGuid": "21EC2020-3AEA-4069-A2DD-
08002B30309D"
}
```

1.8. Response Sample: Successful Connectivity Testing

This is an example of a successful response to a connectivity testing request whose correlationGuid is 998CDC77-6860-4351-9277-6F3E6F870AC6.

```
     HTTP 1.1 200 OK
     Cache-Control: private, no-cache
     Content-Type: application/json

     {
       "responderGln": "0312231245676",
       "correlationGuid": "998CDC77-6860-4351-9277-
     6F3E6F870AC6"
}
```

1.9. Response Sample: Authentication Failure (HTTP error response)

```
HTTP 1.1 401 Forbidden
Cache-Control: private, no-cache
Content-Type: application/json
```

1.10. Response Sample: VRS Server Error Response (HTTP error response)

```
HTTP 1.1 500 Internal Server Error
Cache-Control: private, no-cache
Content-Type: application/json
```

1.11. Response Sample: Bad Gateway Error for VRS to VRS communication (HTTP error response)

```
HTTP 1.1 502 Server Tried To Invoke Another VRS System
That Is Down
Cache-Control: private, no-cache
Content-Type: application/json
```

1.12. Response Sample: VRS Temporarily Unavailable (HTTP error response)

```
HTTP 1.1 503 Service Temporarily Unavailable
Cache-Control: private, no-cache
Content-Type: application/json
```

1.13. Response Sample: Gateway Timeout (HTTP error response)

```
HTTP 1.1 504 Gateway Timeout After Multiple Retries
Cache-Control: private, no-cache
Content-Type: application/json
```

1.14. Response Sample: Manufacturer Failure (Application error response)

In the event the manufacturer is experiencing an internal server error, the VRS will issue the following:

```
HTTP 1.1 500 Manufacturer Internal Server Error
Cache-Control: private, must-revalidate
Content-Type: application/json
```

1.15. **Response Sample: Manufacturer Timeout (Application error response)**

In the event the manufacturer is not responding within the allotted timeframe, the VRS will issue the following:

```
HTTP 1.1 408 Manufacturer Timeout Response
Cache-Control: private, no-cache
Content-Type: application/json
```

2. JSON Schema

```
{
  "openapi": "3.0.0",
  "info": {
    "version": "1.0.0",
    "title": "VRS P2P Messaging Standard",
    "contact": {
      "email": "apiteam@verificationrouterservice.com"
    },
    "description": "This the API specification for peer-to-peer communication between
Verification Router Services or VRS"
  },
  "servers": [
    {
      "url": "https://apis.tracelink.com/vrs"
    },
    {
      "url": "https://anothervrs.com/gateway/verificationrouterservice"
    }
  ],
  "paths": {
    "/v1/asset/gtin/{gtin}/sn/{sn}/000": {
      "post": {
        "tags": [
          "Test"
        ],
        "description": "Test VRS endpoints",
        "parameters": [
          {
            "name": "gtin",
            "in": "path",
            "description": "Global Trade Item Number",
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "sn",
            "in": "path",
            "description": "Serial Number",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "requestBody": {
          "required": true,
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Request000"
              }
            }
          }
        },
        "responses": {
          "200": {
            "description": "A response code of 200 means the request was successful and
details about the response can be found in the body of the response. Only a 200
response will issue a JSON payload.",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Response000"
                }
              }
            }
          },
          "400": {
            "description": "The request was not formatted properly. Please verify the request
conforms to this specification, and re-issue the request in the correct format. "
          },
          "401": {
            "description": "The request was not allowed because the request did not pass
authentication."
          },
          "404": {
            "description": "The requested resource does not exist."
          },
          "408": {
            "description": "The server timed out waiting for the request."
          },
          "500": {
            "description": "System failed to process the request because of an error inside
the system."
          },
          "502": {
            "description": "The server was acting as a gateway or proxy and received an invalid response from the upstream server.  Indicates that one server tried to use another VRS system and that system was down."
          },
          "503": {
            "description": "System is undergoing maintenance or is otherwise temporarily unavailable for API queries."
          },
          "504": {
            "description": "The server, while acting as a gateway or proxy, performed multiple retries but did not receive a timely response from the upstream server specified by the URI (e.g. HTTP, FTP, LDAP) or some other auxiliary server (e.g. DNS) it needed to access in attempting to complete the request."
          }
        }
      }
    },
    "/v1/asset/gtin/{gtin}/sn/{sn}/001": {
      "post": {
        "tags": [
          "Verification"
        ],
        "description": "Verify a saleable return",
        "parameters": [
          {
            "name": "gtin",
            "in": "path",
            "description": "Global Trade Item Number",
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "sn",
            "in": "path",
            "description": "Serial Number",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "requestBody": {
          "required": true,
          "content": {
```

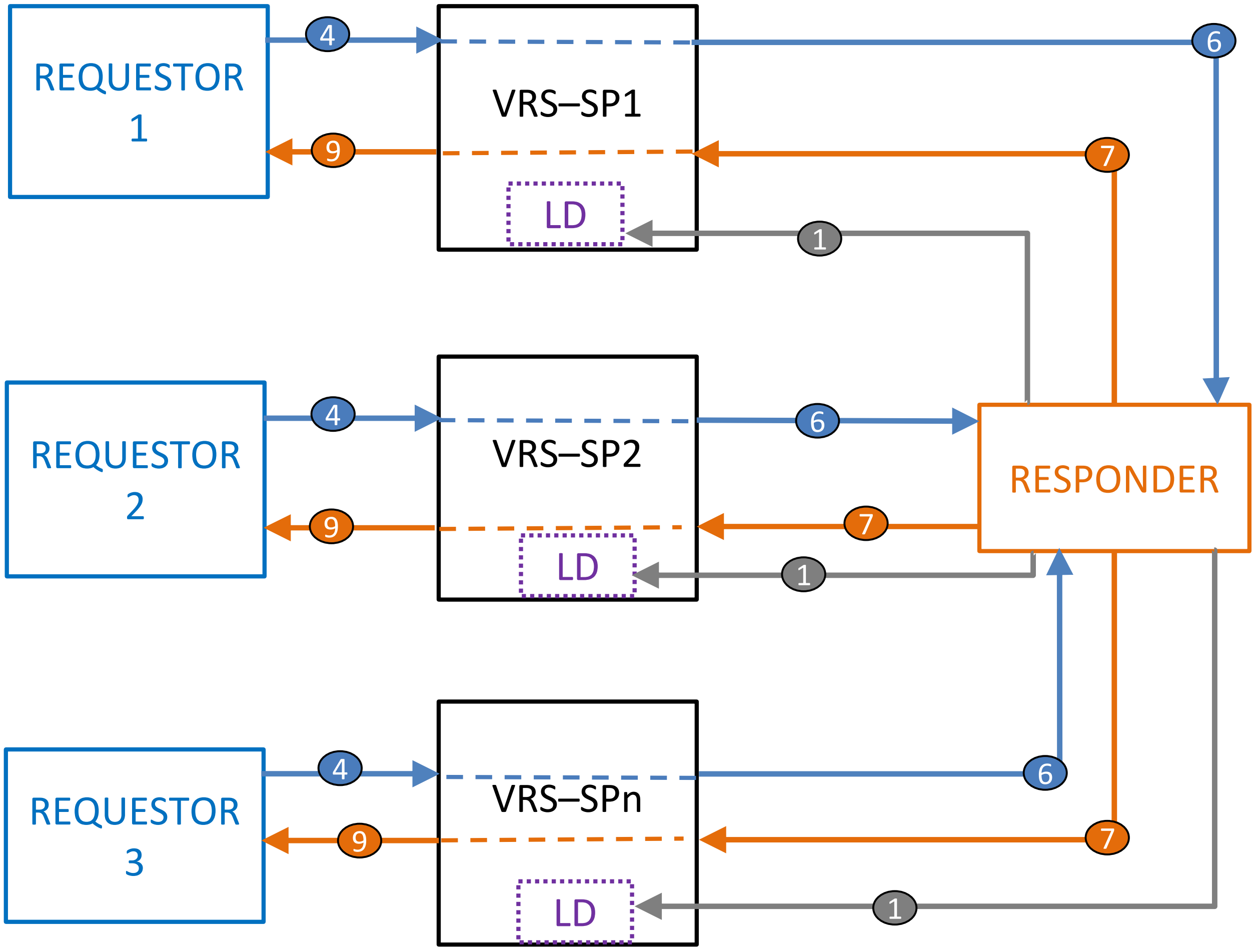
```
      "application/json": {
       "schema": {
        "$ref": "#/components/schemas/Request001"
       }
      }
     }
    },
    "responses": {
     "200": {
      "description": "A response code of 200 means the request was successful and
details about the response can be found in the body of the response. Only a 200
response will issue a JSON payload.",
      "content": {
       "application/json": {
        "schema": {
         "$ref": "#/components/schemas/Response001"
        }
       }
      }
     },
     "400": {
      "description": "The request was not formatted properly. Please verify the request
conforms to this specification, and re-issue the request in the correct format. "
     },
     "401": {
      "description": "The request was not allowed because the request did not pass
authentication."
     },
     "404": {
      "description": "The requested resource does not exist."
     },
     "408": {
      "description": "The server timed out waiting for the request."
     },
     "500": {
      "description": "System failed to process the request because of an error inside
the system."
     },
     "502": {
      "description": "The server was acting as a gateway or proxy and received an
invalid response from the upstream server.  Indicates that one server tried to use
another VRS system and that system was down."
     },
     "503": {
      "description": "System is undergoing maintenance or is otherwise temporarily
unavailable for API queries."
```

```
          },
          "504": {
            "description": "The server, while acting as a gateway or proxy, performed multiple retries but did not receive a timely response from the upstream server specified by the URI (e.g. HTTP, FTP, LDAP) or some other auxiliary server (e.g. DNS) it needed to access in attempting to complete the request."
          }
        }
      }
    }
  },
  "components": {
    "schemas": {
      "Request000": {
        "required": [
          "correlationGuid",
          "requestorGLN"
        ],
        "properties": {
          "correlationGuid": {
            "type": "string",
            "example": "59bc5c88-15f7-49a7-9687-73b05d2c50a4",
            "pattern": "^[a-f\\d]{8}-[a-f\\d]{4}-4[a-f\\d]{3}-[89ab][a-f\\d]{3}-[a-f\\d]{12}$"
          },
          "requestorGLN": {
            "type": "string",
            "minLength": 13,
            "maxLength": 13,
            "example": "9071404000003"
          }
        }
      },
      "Response000": {
        "required": [
          "correlationGuid",
          "responderGLN"
        ],
        "properties": {
          "correlationGuid": {
            "type": "string",
            "example": "59bc5c88-15f7-49a7-9687-73b05d2c50a4",
            "pattern": "^[a-f\\d]{8}-[a-f\\d]{4}-4[a-f\\d]{3}-[89ab][a-f\\d]{3}-[a-f\\d]{12}$"
          },
          "responderGLN": {
            "type": "string",
            "minLength": 13,
            "maxLength": 13,
            "example": "9071404000002"
          }
        }
      },
      "Request001": {
        "required": [
          "correlationGuid",
          "requestorGLN",
          "data"
        ],
        "properties": {
          "correlationGuid": {
            "type": "string",
            "example": "59bc5c88-15f7-49a7-9687-73b05d2c50a4",
            "pattern": "^[a-f\\d]{8}-[a-f\\d]{4}-4[a-f\\d]{3}-[89ab][a-f\\d]{3}-[a-f\\d]{12}$"
          },
          "requestorGLN": {
            "type": "string",
            "minLength": 13,
            "maxLength": 13,
            "example": "9071404000003"
          },
          "data": {
            "type": "object",
            "required": [
              "lotNum",
              "expiry"
            ],
            "properties": {
              "lotNum": {
                "type": "string",
                "description": "Lot number for the asset to be verified",
                "example": "LZ109B15"
              },
              "expiry": {
                "type": "string",
                "description": "Date of expiry for the item to be looked up in format YYMMDD",
                "minLength": 6,
                "maxLength": 6,
                "example": "221203"
              }
            }
          }
        }
      },
      "Response001": {
        "required": [
          "correlationGuid",
          "responderGLN",
          "data"
        ],
        "properties": {
          "correlationGuid": {
            "type": "string",
            "example": "59bc5c88-15f7-49a7-9687-73b05d2c50a4",
            "pattern": "^[a-f\\d]{8}-[a-f\\d]{4}-4[a-f\\d]{3}-[89ab][a-f\\d]{3}-[a-f\\d]{12}$"
          },
          "responderGLN": {
            "type": "string",
            "minLength": 13,
            "maxLength": 13,
            "example": "9071404000002"
          },
          "data": {
            "type": "object",
            "required": [
              "verified"
            ],
            "properties": {
              "verified": {
                "type": "boolean",
                "description": "If ALL 4 elements of the PI were matched, always use 'true'. Else, provide 'false' with optional 'verifiedInfo'.",
                "example": false
              },
              "verifiedInfo": {
                "type": "string",
                "description": "Optional. Only used if 'verified' is 'false' to represent which PI element(s) did not match  Instead of showing empty string or null, do NOT include unless there’s value. Values: 001 – no matching sGTIN, 002 – no matching LOT, 003 – no matching Exp Date, 004 – no matching LOT & Exp date",
                "enum": [
                  "001",
                  "002",
                  "003",
                  "004"
                ],
                "example": "002"
              },
              "additionalInfo": {
                "type": "string",
                "description": "Optional. Will be used if response is 'true' to provide additional information of the state of the sGTIN, like shipped, recalled, in-stock, decommissioned etc. These codes can be ranged from 001 to 099 with various statuses.  Instead of showing empty string or null, do NOT include unless there’s value. Values: 001 – do not re-sell; recalled, 002 – placeholder for future use, 003 – placeholder for future use",
                "enum": [
                  "001",
                  "002",
                  "003"
                ]
              },
              "errorMessage": {
                "type": "string",
                "description": "Optional if needed (e.g. not the Responder for PI).  Instead of showing empty string or null, do NOT include unless there’s value."
              }
            }
          }
        }
      }
    }
  }
}
```

3. Appendix – Solution Architecture Illustrations

The following illustrations depict the VRS Solution Architecture variations for two (2) scenarios:

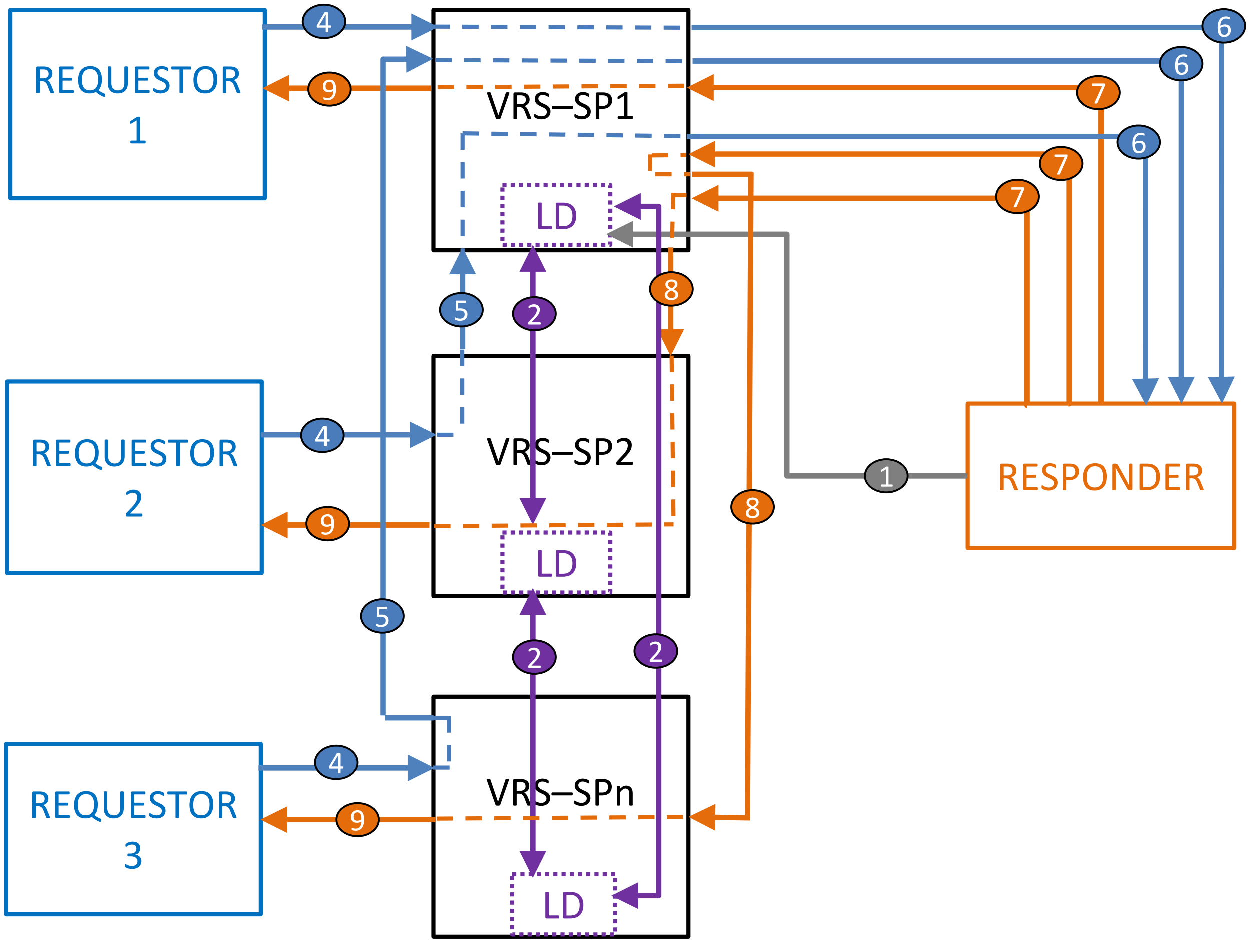
Scenario 1: VRs are routed through the VRS Provider of the Requestor



In this scenario the Responder provides their Connectivity Information (CI) directly to each VRS solution provider as part of the Responder's on-boarding / security access procedure. Interaction #2 is therefore not in scope for this scenario.

🟦 Request
🟧 Response
⬛ Responder GS1 GCP/GTIN Repository Info

Scenario 2: VRs are routed through the Responder's VRS Provider



In this scenario the Responder's VRS solution provider Connectivity Information is shared across authorized VRS providers. Interaction #1 is therefore between the Responder and their selected VRS Provider.

🟪 Responder VRS Provider ID at GS1 GCP / GTIN level

🟦 Request

🟧 Response

◼ Responder GS1 GCP/GTIN Repository Info