Remote Vehicle Interface (RVI) Core Services

This document specifies the core services of the Remote Vehicle Interface (RVI) framework.

Remote Vehicle Interface (RVI) is a universal framework for devices to interact with other devices. The term *device* losely refers to vehicles, mobile devices, cloud servers etc. which all can transparently connect to and exchange data with each other over a variety of networks.

RVI Core represents the core of the RVI framework. Any node looking to exchange data with other nodes over RVI must implement RVI Core. Details on RVI Core and the RVI specification in general can be found at RVI.

RVI Core itself is agnostic to the data that is sent. It provides, however, an addressing and routing scheme that allows the indentification of a service on any device in a global namespace. The syntax of a service name is

```
[organization]/[path]
```

- [organization] A domain name describing the organization that hosts a sub-section of the root. All service paths under the domain name are managed by the given organization.
- [path] A path to access a given service. Path segments are separated by `/`.

Technically, the organization is just a part of the path. However, it is separated because of the responsibility and ability of the organization to manage all subpaths below itself.

A typical service name for a vehicle that specifies a service to lock a door:

```
message:jaguarlandrover.com/vin/1HGCM82633A004352/services/body/lock
```

Services can freely be defined and implemented by anyone for any purpose as long as the global name space is not violated (hence the organization). RVI defines a couple of core services that are specified by this document.

RVI Service Messages

RVI service messages are encoded as JavaScript Object Notation (JSON) Remote Procedure Calls (RPC) (JSON-RPC).

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/logging/subscribe",
    "timeout" : 5000,
    "params": {
        "channels":, ["location", "odometer", "speed"],
        "reporting_interval": 5000
     }
}
```

- jsonrpc - JSON-RPC version number.
- id - Request ID, used to match the response to the request.
- method - Method to be invoked. For RVI the method is `message`.
- service - The full service name.
- timeout - Timeout in seconds since UNIX Epoch UTC (January 1, 1970 UTC).
- params - Array of objects passed to `method` as parameters.

RVI Core Services

Device Management

Services for configuring the device after inital bootstrapping.

Certificates

Services to manage certificates.

Provision Signed Key

Provision a private-root-key-signed version of the device public key.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/dm/key_provision",
    "timeout" : 5000,
    "params": {
        "keyid" : "xyzzy123",
        "key": "[JWT-encoded key]"
     }
}
```

The parameters are:

- keyid - Unique key ID.
- key - JWT encoded key, signed by the private root key, where the payload is a JWK-formatted JSON object.

After receiving a key the device will typically store it in its key store.

Provision Certificate

Provision a certificate from a server to a client.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/dm/cert_provision",
    "timeout" : 5000,
    "params": {
        "certid" : "cx2343",
        "checksum": <md5_checksum>,
        "certificate": "<certificate>",
     }
}
```

The parameters are:

- certid - Unique certificate ID.
- checksum - The md5 checksum calculated over the entire certificate
- certificate - The certificate encoded as base64.

After receiving a certificate the device will typically store it in its certificate store.

Erase Certificate

Erase a certificate from the device's certificate store.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/dm/cert_erase",
    "timeout" : 5000,
    "params": {
        "certid" : "cx2343",
     }
}
```

The parameters are:

- certid - Unique certificate ID.

After receiving an erase certificate request the device must remove it from its certificate store.

Clear Certificates

Erase all certificates from the device's certificate store.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/dm/cert_clear",
    "timeout" : 5000,
    "params": {
        "dummy" : 0
     }
}
```

The parameters are:

- dummy - Placeholder parameter

After receiving an erase certificate request the device must remove all certificates from its certificate store.

Revoke Certificate

Revoke a certificate. This message is sent to all devices who may accept the certificate to blacklist that certificate and all certificates that are derived from it.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/dm/cert_revoke",
    "timeout" : 5000,
    "params": {
        "certid" : "cx2343",
     }
}
```

The parameters are:

- certid - Unique certificate ID.

After receiving a revoke certificate request the device must add it to its certificate blacklist. When the device is presented with a blacklisted certificate or a certificate that has been derived from a blacklisted certificate it must reject that certificate as invalid.

Configuration

Services to manage device configuration.

Read Configuration Variables

Retrieve the value of one or more configuration variables from the device.

The server sends to the device:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/dm/var_read",
    "timeout" : 5000,
    "params": {
        "variable" : ["varname1","varname2"]
     }
}
```

The parameters are:

- variable - The name of the variable.

The device responds with:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/backend/dm/var_value",
    "timeout" : 5000,
    "params": {
        "vin": "1234",
        "variables" : [
            { "name" : "varname1", "value" : "Hello World" },
            { "name" : "varname2", "value" : "Hello Universe" },
        ]
     }
}
```

The parameters are:

- vin - The VIN of the device reporting the variable.
- variables - An array of dictionaries with variable names and values.
- value - The value of the variable.

Write Configuration Variables

Set the value of one or more configuration variable in the device.

The server sends to the device:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/dm/var_write",
    "timeout" : 5000,
    "params": {
        "variables" : [
            { "name" : "varname1", "value" : "Hello World" },
            { "name" : "varname2", "value" : "Hello Universe" },
        ]
     }
}
```

The parameters are:

- variables - An array of dictionaries with variable names and values.
- value - The value of the variable.

Over-the-Air Update

RVI can transmit any type of file from one device to another. The most typical application is software over the air updates (SOTA) where a server sends software packages to a device for the device to install. RVI implements file transmission through a series of services that are executed in a particular sequence of events. For the following we use the term

- server - for a device withing to transmit a file to another device
- client - for a device receiving a file transmission

Sequence of events:

1. The server first sends the `notify` message to the client with the description of the download and a unique ID to identify the file transmission transaction. The client can use the description to inform a user of the pending download.
2. The client can then either
    - accept the file transmission by sending `initiate_download` to the server, or
    - reject the file transmission by sending `cancel_download` to the server
3. If the server receives an `initiate_download` response it initiates the file download by sinding `start` to the client together with the `file size` and a `chunk size` indicating the total size of the download and the size of the download portion in each message.
4. The server then immediately starts sending file portions, one per message, with the `chunk` message.
5. After the server has sent the last chunk it sends `finish` to the client.
6. Once the client receives the `finish` message and has assembled and verified the download it sends `download_complete` to the server with a status indicator.

Notify

Inform client of pending file transfer.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/sota/notify",
    "timeout" : 5000,
    "params": {
        "package": "Hello World Application Version 2",
        "retry": 1
     }
}
```

The parameters are:

- package - Package/file description that the client may use to display a user

```
        message.
```

- retry - Unique numeric ID, used by the identify the download transaction.

Initiate Download

Tell server to start the download.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/backend/sota/initiate_download",
    "timeout" : 5000,
    "params": {
        "package": "HelloWorld-2.0.0.rpm",
        "destination" : "12345",
        "retry": 1
     }
}
```

The parameters are:

- package - Package/file name.
- destination - VIN number of vehicle.
- retry - Unique numeric ID, used by the identify the download transaction. Must

```
      match the ID from the `notify` message this message is sent in response
      to.
```

Start Download

Initiate the download from the server to the client.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/12345/sota/start",
    "timeout" : 5000,
    "params": {
        "package": "HelloWorld-2.0.0.rpm",
        "chunk_size" : 65535,
        "total_size": 23234334
     }
}
```

The parameters are:

- package - Package/file name.
- chunk_size - Size of a file chunk in bytes.
- total_size - File size in bytes.

Transmit Chunk

Transmit a file chunk from the server to the client.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/12345/sota/chunk",
    "timeout" : 5000,
    "params": {
        "index": 1,
        "msg" : <file chunk>
     }
}
```

The parameters are:

- index - Chunk index for the client to assemble the chunks correctly. Chunks

  ```
  may not arrive in order.
  ```

- msg - File chunk encoded with base64.

Finish Transmission

Indication by the server to the client that the last chunk has been sent.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/12345/sota/finish",
    "timeout" : 5000,
    "params": {
        "dummy": 0
     }
}
```

The parameters are:

- dummy - Placeholder parameter.

Download Complete

Message from the client to the server to indicate that the client has received the file.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/backend/sota/download_complete",
    "timeout" : 5000,
    "params": {
        "status": 0,
        "retry" : 1
     }
}
```

The parameters are:

- status - `= 0` on success, `!= 0` failure
- retry - Unique numeric ID, used by the identify the download transaction. Must

    ```
    match the ID from the `notify` message this message is sent in response
    to.
    ```

Cancel Download

Tell server to cancel the download.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/backend/sota/cancel_download",
    "timeout" : 5000,
    "params": {
        "retry": 1
     }
}
```

The parameters are:

- retry - Unique numeric ID, used by the identify the download transaction. Must

  ```
  match the ID from the `notify` message this message is sent in response
  to.
  ```

Information Reporting

Services to observe data channels and/or to report them in regular intervals.

Subscribe

Subscribe to one or more data channels. This message is typically implemented by a client device.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/logging/subscribe",
    "timeout" : 5000,
    "params": {
        "channels":, ["location", "odometer", "speed"],
        "reporting_interval": 5000
     }
}
```

The parameters are:

- channels - An array with the data channels to subscribe to.
- reporting_interval - The reporting interval in milliseconds [ms].

Unsubscribe

Unsubscribe from one of more data channels. This message is typically implemented by a client device.

```
{
    "jsonrpc": "2.0",
    "id": 2,
    "method": "message",
    "service": "jlr.com/vin/123456/logging/unsubscribe",
    "timeout" : 5000,
    "params": {
        "channels":, ["location", "speed"],
    }
}
```

The parameters are:

- channels - An array with the data channels from which to unsubscribe from.

Report

Report data channels. This message is typically implemented by a server device receiving data.

```
{
    "jsonrpc": "2.0",
    "id": 3,
    "method": "message",
    "service": "jlr.com/backend/logging/report",
    "timeout" : 5000,
    "params": {
        "vin": "1234",
        "timestamp": 1415143459110,
        "data": [
            { "channel": "location",
              "value": { "lat": 39.0313, "lon": 125.3787, "alt”: 345.6 } },
            { "channel": "speed", "value": 15 },
            { "channel”: "odometer", "value": 10455 },
        ]
    }
}
```

The parameters are:

- vin - The VIN of the device reporting the data.
- timestamp - The timestamp of the data record in UTC formatted as YYYY-MM-DDThh:mm:ss.SSSZ for example: 2014-11-16T08:43:02.000Z
- data - The reported data. Data is always reported as a list of dictionaries, that is `key:value` pairs: `{"channel" : "channel_name", "value" : channel_value }`. The key `channel` is a string representing the name or identification of the channel as used by the `subscribe` and `unsubscribe` services. The `value` can be any JSON data type. In particular `value` can be a dictionary in itself, as it is with the `location` channel.

Currently defined channels:

Channel	Description	Value Data Type
speed	vehicle speed	uint16
rpm	engine rpm	uint16
ctemp	engine coolant temperature	uint16
maf	mass air flow	uint16
ait	air intake temperature	int
eload	engine load	uint16
tpos	throttle position	uint16
bvolt	battery voltage	double
atemp	ambient air temperature	int
otemp	engine oil temperature	int
etime	engine running time	uint32
location	vehicle location	dict { “lat” : double , “lon” : double , “alt” : double , “dir” : uint16 }

Remote Vehicle Control

Services to lock, unlock, etc, a vehicle

Lock / Unlock doors, trunks, etc

Lock one or more doors, trunks, and other hatches.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/control/lock",
    "timeout" : 5000,
    "params": {
        "action": "lock",
        "locks": [ "r1_lt", "r2_rt", "trunk" ]
    }
}
```

The parameters are:

- action - Specifies if the doors should be locked or unlocked `lock` lock the doors `unlock` unlock the doors.
- locks - An array specifying which doors to be locked
  `r1_lt` and `r1_rt` are row one (front) doors.
  `r2_lt` and `r2_rt` are row two (rear) doors.
  `trunk` is the rear trunk.
  `hood` is the rear hood.

Start / Stop Engine

Start or stop the engine.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/control/engine",
    "params": {
        "action": "start"
    }
}
```

The parameters are:

- action - Specifies if the engine should be started or stopped `start` start the engine.
  `stop` stop the engine.

Trunk Open/Close

Open or close a motorized trunk. Optionally, the trunk can be opened/popped but not closed. In this case "close" is not supported.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/control/trunk",
    "timeout" : 5000,
    "params": {
        "action": "open"
    }
}
```

The parameters are:

- action - Specifies if trunk should be opened or closed `open` open the trunk.
  `close` close the trunk.

Horn

Activate the horn.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/control/horn",
    "timeout" : 5000,
    "params": {
        "duration": 1500
    }
}
```

The parameters are:

- duration - The duration, in milliseconds, that the horn should be activated

Lights

Activate the horn.

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/control/lights",
    "timeout" : 5000,
    "params": {
        "lights": [ "low", "interior"  ],
        "duration": 60000
    }
}
```

The parameters are:

- duration - The duration, in milliseconds, that the lights should be on for

Windows

Open/close windows and other hatches

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "message",
    "service": "jlr.com/vin/123456/control/windows",
    "timeout" : 5000,
    "params": {
        "windows": [ "r1_lt", "r2_rt", "sunroof" ]
        "position": 75
    }
}
```

The parameters are:

- windows - The windows to operate on `r1_lt` and `r1_rt` are row one (front) windows.
  `r2_lt` and `r2_rt` are row two (rear) windows.
  `sunroof` is the sunroof.
  `rear` is the rear window.
- position - The posistion to set the specified windows to. 0 = fully closed
  100 = fully open

Capabilities

Sent from vehicle to connecting device to inform the device which capabilities the vehicle has

```
{
    "jsonrpc": "2.0",
    "id": 2,
    "method": "message",
    "service": "jlr.com/mobile/46701231234/control/capabilities",
    "timeout" : 5000,
    "params": {
        "windows": [ "r1_lt", "r1_rt", "r2_lt", "r2_rt" ],
        "locks: [ "r1_lt", "r1_rt", "r2_lt", "r2_rt", "trunk" ],
        "trunk":, ["open"],
        "engine":, "true",
        "lights":, [ "interior", "high", "low", "running", "rear" ]
        "horn":, "false"
    }
}
```

The parameters are:

- windows - List all windows available for open and closing.
  `r1_lt` and `r1_rt` are row one (front) left and right windows.
  `r2_lt` and `r2_rt` are row two (rear) left and right windows.
  `r3_lt` and `r3_rt` are row two (rear) left and right windows.
  Etc.
  `sunroof` is the sunroof.
  `rear` is the rear window.
  Set to "false" if not supported.
- locks - List all doors available for locking and closing.
  `r1_lt` and `r1_rt` are row one (front) left and right doors.
  `r2_lt` and `r2_rt` are row two (rear) left and right doors.
  Etc.
  `trunk` is the rear trunk.
  `hood` is the rear hood.
  Set to "false" if not supported.
- trunk - List operations available for trunk.
  `open` The car can open or pop the trunk.
  `close` The car can close the trunk.
  Set to "false" if not supported.
  Please note that opening and closing the trunk is different from locking and unlocking the trunk.

- lights - Specifies what lights can be turned on and off. `low` - low beam.
  `high` - high beam.
  `running` - running lights.
  `rear` - rear lights.
  `interior` - interior lighs.
  `side` - side lights (or instep lights next to doors).
  `rear_fog` - rear fog light.
  `front_fog` - front fog light.
  `left_blinkers` - left_blinkers.
  `right_blinkers` - left_blinkers.
  Set to "false" if not supported.
- engine - Specifies engine start/stop is supported. `true` - Engine can be started and stopped.
  `false` - Engine cannot be started and stopped.
- horn - Specifies if horn activation is supported. `true` - Horn can be activated.
  `false` - Horn cannot be activated.

Status

Sent from vehicle to connecting device to inform the device of the current status of the vehicle. Only command and parameters reported by a previous `capabilities` call will be listed.

```
{
    "jsonrpc": "2.0",
    "id": 2,
    "method": "message",
    "service": "jlr.com/mobile/46701231234/control/status",
    "timeout" : 5000,
    "params": {
        "windows": {
            "r1_lt": 70,
            "r1_rt": 0,
            "r2_lt": 0,
            "r2_rt": 0
        },
        "locks": {
            "r1_lt": "unlocked",
            "r1_rt": "locked",
            "r2_lt": "locked",,
            "r2_rt": "locked",
            "trunk": "locked"
        },
        "trunk": "closed",
        "horn": 0,
        "lights": {
            "high": 30000,
            "low": 0,
            "interior": 0,
            "side": 15000
        },
        "engine": "on",
    }
}
```

The parameters are:

- windows - List each window with its current position.
- locks - List each lock and its current status.
- trunk - Status of the trunk.
- lights - List each light and its remaining time on.
  0 = Off.
  The time remaining for the light to be on is given in milliseconds.
- engine - Status of the engine.
- horn - Milliseconds remaining on horn active.
  0 = Horn not active.