1 Appendix A: File Formats

Details on how to format files uploaded to the portal follow.

1.1 Software packages

A software package upload consists of a zip file containing the actual software package file, a manifest file, and a signature file. All files within the zip file are located in the top-level directory:

package.zip
\manifest.json
\package
\signature

1.1.1 Manifest file

The manifest file must be named manifest.json. It is a well-formed JSON document that describes the intrinsic properties of the software upload contents.

The properties of the manifest file are as follows:

Property	Type	Mandatory	Description
companyName	String (limit 255)	Yes	Name of the company
subsystem	String (limit 100)	Yes	Name of the subsystem to which the software package applies. The subsystem is converted to lower case when stored. Example: 'HEAD UNIT' is stored as 'head unit'. Search is case insensitive.
softwareName	String (limit 255)	Yes	A unique name to identify software package
buildDate	ISO 8601 Date-Time	Yes	Date the package is built
filetype	String	Yes	Package type {DIFFERENTIAL, FULL}
filename	String (limit 255)	Yes	The filename for the software update package. Must match the exact name of the software package file.

Property	Type	Mandatory	Description
price	Big Decimal	No	The price of the package if the package is purchasable
imagesFileName	String	No	The zip file name if the package contains any zip of images.
resultConfig	VehicleElementConfig	Yes	The resulting VehicleElementConfig supplied by this update.
startConfig	VehicleElementConfig []	Yes	An array of starting VehicleElementConfig for which this update applies.
dependencies	VehicleElementConfig []	No	An array of required VehicleElementConfig dependencies in order for this update to apply.
description	String (limit 1000)	Yes	A short description of the update that will be visible to a system administrator, but not the consumer.
independent	boolean	No	Determines whether the package is independent of other packages. Default is false
updateType	Enum	No	{UPGRADE/DOWNGRADE}

VehicleElementConfig properties:

Property	Type	Mandatory	Description
id	Long	No	Unique identifier assigned by the server for a software package. Not present in request to add a software package.
partNumber	String (limit 100)	Yes	Unique part number for the vehicle element.

hwVersion	String (limit 100)	No	Hardware version for the vehicle element.
fwVersion	String (limit 100)	No	Firmware version for the vehicle element.
swVersion	String (limit 100)	Yes	Software version for the vehicle element.
elements	VehicleElementConfig []	No	If a VehicleElement has sub-elements (i.e. sub-parts) that are updatable, they are included as well.

For example software update package **infobin.zip** containing three files:

infobin.zip
\manifest.json
\infobin (binary file)
\signature

manifest.json contents

```
{
    "companyName": "Example Inc",
    "buildDate": "2024-08-14T08:00:00Z ",
    "subsystem": "Infotainment",
    "softwareName": "Info NAFTA Advanced",
    "independent": false,
    "updateType": "UPGRADE",
    "fileName": "infobin",
    "fileType": "DIFFERENTIAL",
    "resultConfig": {
        "partNumber": "Info_8",
        "swVersion": "13.00.10.00"
    },
    "startConfig": [{
            "partNumber": "Info_8",
            "swVersion": "13.00.00.00"
        },
        {
            "partNumber": "Info_8",
            "swVersion": "12.90.10.00"
        }
```

```
    ],
    "description": "This is an Infotainment upgrade for latest and
greatest."
}
```

1.1.2 Package file

The package file itself does not have a fixed name, but is defined in the manifest file. The contents of the package file are not parsed by the server in any fashion.

1.1.3 Signature file

The signature properties file contains the digital signatures for the manifest and software package files, one per line. The signatures are Base64 encoded values.

manifest.json=Base64SignatureForManifest
package=Base64SignatureForPackage