Function Specification (FncS)

IVSU_OTA Cloud Interface Specification

()

Document ID	546616	
Document Location	VSEM Rich Client, VSEM Thin Client	
Document Owner	Gill, Balwinder (bgill51)	
Document Version	C	
Document Status	Released	
Date Issued	16-Jul-2019 14:43	
Date Revised	30-Oct-2019 17:44	
Document Classification	GIS1 Item Number:	
	GIS2 Classification:	

Document Approval			
Person	Role	Email Confirmation	Date

Printed Copies are Uncontrolled

DISCLAIMER

This document contains Ford Motor Company Confidential information. Disclosure of the information contained in any portion of this document is not permitted without the expressed, written consent of a duly authorized representative of Ford Motor Company, Dearborn, Michigan, U.S.A.

This document contains information developed and accumulated by and for FORD MOTOR COMPANY. As such, it is a proprietary document, which, if disseminated to unauthorized persons, would provide others with restricted information, data, or procedures not otherwise available, exposing the FORD MOTOR COMPANY to potential harm.

Employees and suppliers having custody of this specification or authorized to use it must be cognizant of its proprietary nature and ensure that the information herein is not made available to unauthorized persons.

FORD MOTOR COMPANY reserves the right to protect this work as an unpublished copyrighted work in the event of an inadvertent or deliberate unauthorized publication. FORD MOTOR COMPANY also reserves its rights under copyright laws to protect this work as a published work.

This document or portions thereof shall not be distributed outside FORD MOTOR COMPANY without prior written consent.

Refer all questions concerning disclosure to the author(s) or any duly authorized representative of Ford Motor Company.

Copyright, © 2016 Ford Motor Company

Content

List of Figures

List of Tables

No table of figures entries found.

1 Title page

OTA SYSTEM INTERFACE SPECIFICATIONOTA SYSTEM INTERFACE SPECIFICATION
Version No. 1.0.0

Copyright © 2001-2020 Ford Motor Company
(U.S. and international notice, and original material were added in each indicated year.)

Project Document Revision history

Version #	Date	Revision Author	Description
1.0.0	**10/11/2017**	**Brunilda Caushi Ali Suleiman Mohamad Nasser Vijay Jayaraman**	**Document Structure and initial requirements**
1.0.4	**1/3/2018**	**Ali Suleiman**	**Added Legacy IVSU client module functionalities. Added CTR/Generic ECU information. Updated NFR section to apply to legacy and new clients.**
	1/3/2018	**Ali Suleiman**	**Updated NFR to 30s, the rational being that there should be an improvement in performance over the previous cloud interface.**
1.0.5	**1/4/2018**	**Ali Suleiman**	**Removed Legacy IVSU client module functionalities. Removed CTR/Generic ECU information. Advised we will keep the old interface separated from the new, and this document will be scoped for FNV2.**
1.0.5	**1/4/2018**	**Ali Suleiman**	**Added Cloud API, intended for Service, and Consumer Site support for usb.**
1.0.12	**7/31/2018**	**Ali Suleiman**	**OTA Cloud Spec v1.0.12 Removed Campaign Expiration Date. (Already present in campaign Manager) Added TTL to manifest Definition. Removed Audience from Manifest. OPL add to VADR requirements for Meta data. Campaign ID, this is the actual campaign ID and is different from the Campaign Correlation ID. This only used for tracking purposes, ECG doesn't currently use this. Need confirmation status manager will use ID from trigger and not manifest. ticket #227. Unbreakable manifest time, this item still needs further Review. Jira #228 Replace CAGID with Group ID. Inhale Exhale moved to update procedure. Added Security JSON translation of MDX structure. Activation methods changed to: Instantaneous, vehicleInhibitRequired, and IgnitionCycleRequired Renamed Permenantinhibit field to "ResponseOnFailedActivationWithInhibit" Added definitions Permanent Inhibit, De-Inhibit, Reduced Function Added CANFD variants to programmingMethod Field. Added RSASig to support E/R use case. Added UDSProgramTime. Removed "Update" option from action field for APP updates. Added “ActivationDateTime” as an optional field under Coordination as sibling to group ID.**
1.0.13			**Network data moved to the node level in the ODL, added new sample “ODL_ORFIN v1.0.json”**
1.0.16			**Added Pre and Post installation scripts to manifest. Added Direct Configuration details to manifest. Added Requirements References for UMT.**
1.0.17			**Added additional limp home with MIL.**

2 Cloud Interface Specification Sign OffNameDept.RoleDateSignatureCommentsBrunilda CaushiCVSIn Vehicle Software Update (OTA and US

2.1 Table of ContentsCloud Interface Specification Sign Off31Introduction61.1Purpose61.2Scope61.3Glossary and Acronyms61.4Assumption

2.2 Table of Figures

3 Introduction

3.1 Purpose

The purpose of this document is to define an interface specification between Client device module and Ford back end OTA cloud infrastructure.

3.2 Scope

Currently, there are three types of Client device (product variants) modules

1. CHR+ (a SYNC variant for low cost vehicles)
2. ECG (the smart gateway module in the FNV2 architecture)
3. Connected Service Tool

This specification provides detail characteristics of Ford back end server, cloud endpoints, message formats, Payload formats, transport protocols, and security of communication.

The interface specification shall support different communication protocols for vehicle software update such as Wi-Fi, Bluetooth, 3G, 4G-LTE, etc.

The acronym 'IVSU' is used throughout this document to represent the 'In Vehicle Software Update' feature developed by Ford Motor Company Mobility Organization. IVSU is a feature provided to Ford vehicle owners client device modules in their vehicles. This feature would enable automatic OTA software updates to the vehicle or USB updates for a subset of vehicle components.

3.3 Glossary and Acronyms

In summary, IVSU consists of following steps to perform successful software updates

Glossary/Acronym	Description
DID	Diagnostic Data Identifier
ODL JSON	Optimized DID List, which consists of minimum required list of DIDs needed by Ford backend server. Ford back end server sends this in JSON document format. JSON is included to distinguish the format from the prior ODL format used.
Default ODL	Client module stores the ODL received from the cloud. Client module stores the ODL in JSON document format. Initial Default ODL (which may be different for different client modules) is provided to supplier.
Vehicle Interrogator Log	Vehicle Interrogator Log, which consists data elements mentioned in Appendix. Client module creates the Vehicle Interrogator Log based on ODL and what the OTA trigger is requesting.
ST	Service Type
CT	Command Type
MST/MT	Message Status Type or Message Type

3.4 Assumption

3.5 References

The following references table consists of References name, link and copy of document.

Ref. No.	Name	Owner Contact information	Description/comments
1	S13b SyncP Services Assignment		

4 System Overview

Following infrastructure diagram depicts the different components of IVSU infrastructure.

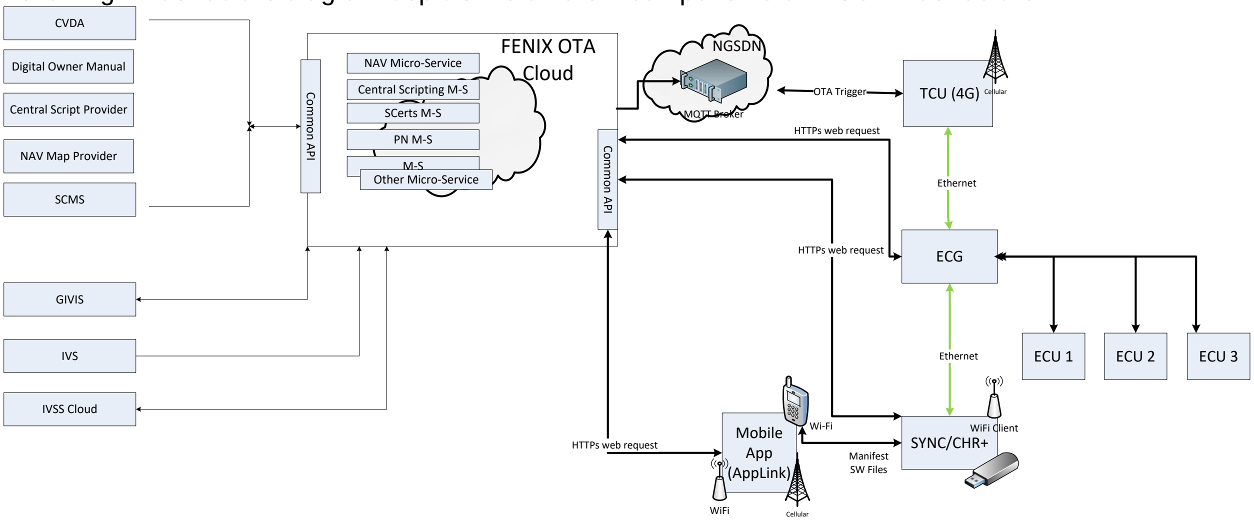


Figure 2-1 IVSU System Overview

In summary, IVSU consists of following steps to perform successful software updates:

- OTA command trigger to update software
- Generate the interrogation file based on the trigger
- Post Vehicle interrogator log to ford backend cloud server
- Cloud backend server receives VIL, parses and processes.
- Cloud backend server sends update information to the module.
- Client module receives and Parses manifest and update procedure.
- Client module downloads the binary files from URLs provided in manifest.
- Client module installs or streams to another module the downloaded binary files
- Target modules install the downloaded files
- Client module send progress status to Ford back end server during the full update process

5 IVSU MMOTA Client module functionalities

5.1 Web service end points and HTTP format

5.1.1 Web Service Endpoints

Service	Env	Module	Endpoint
FENIX Cloud PROD Endpoint	PROD	OTA	TBD
		OTA Policy	
FENIX Cloud Performance Testing Endpoint	PRF	OTA	TBD
		OTA Policy	
FENIX Cloud Integration Testing Endpoint	QA	OTA	TBD
		OTA Policy	
FENIX Cloud Development Endpoint	DEV	OTA	TBD
		OTA Policy	
FENIX Cloud TestHarness Endpoint	Test Harness	OTA	
		OTA Policy	

Standards:

Protocol:	HTTPS v1.1
Security:	TLS v1.2
Method:	POST
Message Format:	JSON
Architecture Pattern:	REST Web Service
Encryption:	Message level symmetric key encryption
Authentication:	Message level signature validation

5.1.2 Transport Layer Security requirements

The Transport Layer Security (TLS) standard shall be used to provide authentication of senders and recipients, and integrity and confidentiality of the data delivered between the client module and FENIX over TCP/IP during HTTPs POST vehicle interrogator log.

The minimum version of the TLS protocol implemented shall be TLS 1.2. The client module and FENIX shall never allow negotiation to a lower TLS version or usage of cipher suites other than those documented and agreed upon.

TLS 1.2 necessitates that both the client module and server implement digital certificates provided by the Ford private In-Vehicle Issuing Certificate Authority (CA). Each client module (as part of a given Product Development program) shall contain an identical certificate, which may be referred to as the Vehicle Public Certificate. This certificate will be provided to the given module supplier through an out of band process (e.g. encrypted email). Each server endpoint (i.e. unique Domain Name or DN) shall also contain a certificate, which may be referred to as the Server Certificate. Client modules may contain more than one Vehicle Public Certificate if multiple endpoint Domain Names are needed for different processes, features or functions, but each TLS session must be validated to match the certificate chain for the intended domain/purpose.

For all TLS connections, the client module shall only trust certificates chaining up to the appropriate Issuing CA. Certificates from other CAs shall be rejected.

5.1.2.1 Ford Production Certificate Hierarchy

The production certificate hierarchy will be issued by Ford's internal PKI and rooted with the Ford Internal Root and Policy Certificate Authorities (CA). A unique certificate will be created for each server endpoint Domain Name. Ford will provide the client module supplier with the trusted Ford public certificate chain (p7b format) which will be built into the module’s certificate store and used to validate the server's identity and negotiate TLS parameters as needed. In addition to be provide to the module provider, it will also be provided to Ford Pass and Lincoln Way implementation team for including inside of the mobile application.

- TLS certificates are X.509 format, 2048-bit, SHA256-RSA Signature Algorithm.
- TLS certificates are issued/signed for each Domain Name by the appropriate **Ford private In-Vehicle Issuing CA**. They will be valid for 3 years and shall be updated accordingly, in coordination with the In-Vehicle Security Services team.
- The client module shall utilize partial chain validation for verifying authenticity of a signed update. It shall validate the certificate chain up to the **appropriate Issuing CA**.
- Certificates above the Issuing CA shall not be installed on modules. These certificates (i.e. Root or Policy CAs) should not validate signed files.
- The Server Certificate Extended Key Usage must be validated as being set to Server Authentication (1.3.6.1.5.5.5.7.3.1) i.e. `–purpose serverAuth`
- The client module shall have the capability to ignore the expiration time stamp during certificate validation process.

5.1.2.2 Development Certificates

Ford will create a Development/Test certificate hierarchy which is structurally identical to the production hierarchy described above. The public Ford Internal Development Issuing CA certificate shall be installed in pre-production modules instead of the Production certificate. *Development/Test certificates must be removed from the module image for all production-intent software release builds – only Production certificates shall be contained in the client module's certificate store*.

5.1.2.3 Certificate Storage

The client module shall not expose the public certificate on any public interface (especially diagnostics). There are no other requirements for storage of the public certificate within the implementing module.

5.1.2.4 Certificate Revocation / Expiration

There is currently no plan for implementing "real-time" handling of Ford certification revocation or expiration. In the unlikely event of a breach, revocation would be achieved by creating a delta software update (firmware file) between the current production software image (which contains the certificates to be removed) and a subsequent image which contains new certificates generated by Ford.

The same basic process would be used in the case where a certificate's expiration date is approaching. This also will be very rare as the Ford Internal Issuing CA Chain will be valid for at least 30 years. In any event, when required a new set of certificates would be generated by Ford and added to those already contained in the current production software image. A software update would be used to deliver the new certificates to the module.

5.1.2.5 Cipher List

The client module and GIVIS shall utilize the following cipher suites for TLS communication, listed in order of priority:

- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P256

Suppliers shall confirm with Ford if the specific strings listed above are not valid (references to particular cipher suites in OpenSSL may be different from that of Windows SChannel or other TLS implementations).

The client module shall only support the suites listed above. GIVIS may support additional suites if necessary, but shall default to these suites as the highest priority. A complete list of enabled suites and priority shall be documented, approved by Ford and maintained.

5.1.3 HTTPs Header fields

IVSU Client module shall set following HTTP header field values for HTTP POST request.

- Content-Type:application/json
- Accept:application/json

IVSU Client module shall receive and process following HTTP header values for HTTP POST response.

- Content-Type:application/json

- Accept:application/json
- Content-Length: (number of bytes received)

IVSU Client module shall receive and process following HTTP header values for HTTP POST response for Status Messages from the IVSU client to the cloud.

- Content-Disposition: filename="00001123453_1521211908"
- Content-Disposition: filename="1FTFW1RG7HFA48540_1521211908"

5.1.4 HTTPs Content Format

Getsoftwareupdate web service endpoints HTTPs POST request and response content in JSON format.

JSON format consists of attribute and value pairs.

{attribute:[array of values]}

Attribute format is string. It is “data”

Value format is string. Array of Values shall be of single or multiple elements.

Each string element is base64 encoded SynP packet.

{"data":["SyncP1"] }

For example,

{"data":["DwMYAAAAAUZON0ExMTk0deeVhZOvwxfQkyKAM4TlIjdaDLAEMdENpqG6JNSm8zK\/"] }

The JSON array structure "data" can hold multiple SyncP messages as well.

{"data":["SyncP1",”SyncP2”,”SyncP3”,.. “SyncPn”] }

For example,

{"data":["GHMYAAAAAUZON0ExMTk0deeVhZOvwxfQkyKAM4TlIjdaDLAEMdENpqG6JNSm8zHD==",
"DwMYAAAAAUZON0ExMTk0deeVhZOvwxfQkyKAM4TlIjdaDLAEMdENpqG6JNSm8zK\/"]] }

For real world examples with actual SyncP and Payload, please refer to Client devices sections.

5.2 IVSU SyncP Message Format

SyncP is a Ford Motor Company's telematics proprietary message format designed to exchange data between Client modules and Ford back end cloud and enterprise systems. The structure of SyncP message is composed of several data elements as defined below.

Note: Please refer to S13e_SyncP_Network_Installation_(TBD) specification for detail information on SyncP message formats.

Please refer to S13b_SyncP_Services_Assignment_(TBD) specification for security related fields (Signed and Encrypted bits) in SyncP message formats.

5.2.1 SyncP Header

Octet\Bit	0	1	2	3	4	5	6	7
1	Protocol version			Response Required	High Bandwidth	Signed	Encrypted	Has ESN
2	Service Type							
3	Version/Command Type				CPU Destination	Encryption Key Index		
4..7	Payload Size							
8..15	[ESN]							
16..19	Module Message ID							
20..23	Server Message ID							
24	Message Status							
25..40	[IV]							
41..n	Payload							
n+1..n+16	[Signature Tag]							

5.2.2 SyncP header IVSU usage description

Sync P Data Structure			
	Byte position	**Bits position**	**IVSU Usage Description**
Protocol version	1	0 to 2	Refer to S13a for appropriate version.
Response Required	1	3	Response Required value should be 1
High Bandwidth	1	4	Default, shall be set to High Bandwidth mode.
Signed	1	5	Shall be set based on IVSU Service Type and Command Type (Refer section 4.3)
Encrypted	1	6	Shall be set based on IVSU Service Type and Command Type (Refer section 4.3)
Has ESN	1	7	Shall be set to 1. If SyncP header has ESN.
Service Type	2	0 to 7	Expected Service Type values that are listed in table in Section 4.3
Version/Command Type	3	0 to 3	Expected Command Type values that are listed in table in Section 4.3
CPU Destination	3	4	-
Encryption Key Index	3	5 to 7	S13a
Payload Size	4 to 7		Payload size in number of bytes
[ESN]	8 to 15		ESN value 8 characters in length
Module Message ID	16 to 19		Module ID sent from the module to IVSU Cloud which should always be greater than IVSU Database value. If value is less or equal to previously recorded value a replay message would be sent from IVSU Cloud back to module.
Server Message ID	20 to 23		Server ID sent from the module to IVSU which should be always less than or equal to IVSU

			Database value if not replay attack message would be sent from IVSU back to module.
Message Status (0)	24		Message status should be 0
[IV]	25 to 40		Only present if encryption is used.
Payload	41 to n		
[Signature Tag]	n+1 to n+16		Included if signing is enabled.

5.2.3 IVSU Service Type/Command Types

The table below is an overview of how the SyncP fields are populated. Look to the referenced sections for additional details.

SyncP Command	Message Status/ Message Type	Message Size Limit	Message Type	Originator	Destination	Payload	Meaning
Following Table was created based on information from document S13b_SyncP_Services_Assignment_TBD							
SERVICE TYPE = 44 MODULE to CLOUD							
0x0	NA	5MB	Sign/Enc/ID ModID= ModID + 1 SrvrID= Unch	Module	Cloud	Current Vehicle Interrogator Log (Check for Update)	VIL reported when checking for a sofware update. Software Update Request
0x1	NA	5MB	Sign/Enc/ID ModID= ModID + 1 SrvrID= Unch	Module	Cloud	Current Vehicle Interrogator Log (Report Vehicle Status)	VIL sent with the intent of reporting current state. Reporting VIL
0x2	NA	5MB	Sign/Enc/ID ModID= ModID + 1 SrvrID= Unch	Module	Cloud	Current Vehicle Interrogator Log (Post Update)	VIL sent after a successful activation. Post Activation VIL
0x4	NA	5MB	Sign/Enc/ID ModID= ModID + 1 SrvrID= Unch	Module	Cloud	Status type for Installation progress (to backend) -- see UC/spreadsheet.	Send Status to Cloud
0x8	NA	5MB	Sign/Enc/ID ModID= ModID + 1 SrvrID= Unch	Module	Cloud	AUTH Command Request	OVTP Client Authorization Command Request)

0xA	NA	5MB	Sign/Enc/ID ModID= ModID + 1 SrvrID= Unch	Module	Cloud	Customer Consent	Send Customer Consent SyncP Message Lay Out
0xC	NA	5MB	Sign/Enc/ID ModID= ModID + 1 SrvrID= Unch	Module	Cloud	Request Policy Table	Request Policy Table
0xD	NA	5MB	Sign/Enc/ID ModID= ModID + 1 SrvrID= Unch	Module	Cloud	Request ODL	Request ODL
0xF	NA	5MB	Sign/Enc/ID ModID= ModID + 1 SrvrID= Unch	Module	Cloud	Critical Status Messages	

SyncP Command	**Message Status/ Message Type**	**Message Size Limit**	**Message Type**	**Originator**	**Destination**	**Payload**	**Meaning**
SERVICE TYPE = 44 CLOUD to MODULE							
0x3	*NA*	5M limit	Sign/Enc/ID ModID=Unchanged ServerID + 1	Cloud	Module	Optimized DID List	Sent to the vehicle from the cloud. ODL is used to determine what needs to be queried on the vehicle and included in the DIL. Respond ODL to module
0x5	*NA*	5M limit	Sign/Enc/ID ModID=Unchanged ServerID + 1	Cloud	Module	Update_Procedure	OTA Rules Response
0x6	*NA*	5M limit	Sign/Enc/ID ModID=Unchanged ServerID + 1	Cloud	Module	Manifest	Update available scenario
0x7	NA	5M limit	Sign/Enc/ID ModID=Unchanged ServerID + 1	Cloud	Module	No Update	No Update available scenario
0x9	*NA*	5M limit	Sign/Enc/ID ModID=Unchanged ServerID + 1	Cloud	Module	AUTH Command Response	Get OVTP Authorization

0xB	*NA*	5M limit	Sign/Enc/ID ModID=Unchanged ServerID + 1	Cloud	Module	Policy Table	Update OTA Policy Table
0xE	*NA*	5M limit	Sign/Enc/ID ModID=Unchanged ServerID + 1	Cloud	Module	Manifest for CST	
Following Table was created based on information from document S13i_SyncP_Applink_QLite_Specification_104							
SERVICE TYPE = 1 - MODULE to CLOUD							
0x00	0x01	---	----	Module	Cloud	None	**Invalid Security Type** Record and Return 200
0x00	0x04	4k	SyncP-Encoded Signed Non-Encrypted ModId= ModId+1 SrvrID= Unch	Module	Cloud	None	**Invalid Security Type** Record and Return 200
0x00	0x06	4k	SyncP-Encoded Signed **Non**-Encrypted ModId= ModId+1 SrvrID= Unch	Module	Cloud	None	**Message Size Out-of-Bounds** Record and Return 200
0x00	0x09	4k	SyncP-Encoded Signed **Non**-Encrypted ModId= ModId+1 SrvrID= Unch	Module	Cloud	None	**Service Error** Record and Return 200
0x00	0x0B	4k	SyncP-Encoded Signed **Non**-Encrypted ModId= ModId+1 SrvrID= Unch	Module	Cloud	None	**Invalid Data** Record and Return 200
SERVICE TYPE = 1 - Cloud to Module							
0x02	0x05	4k	SyncP-Encoded Signed Encrypted ModId= 0 SrvrID=0	Cloud	Module	None	**Out-of-Sequence Message** **Record and Trigger ST =1, CT=2, Status=5 – this will have a payload, signed and encrypted**

5.3 OTA Command trigger to update software

OTA command trigger is occurrence of an event in the vehicle or cloud that initiates an OTA software update. The event shall be of different methods based on the OTA policies.

5.4 Generate Vehicle Interrogator Log

The Vehicle Interrogator Log is a JSON document format, which contains current state of Vehicle, sent from Client module to Ford backend cloud server.
The Vehicle Interrogator Log consists of the mandatory and optional data elements and attributes. The VIL is used by the cloud to determine if any software updates are available for any specific module in the vehicle.
The VIL is generated based on ODL. The ODL is unique to each vehicle program.
ODL is an Optimized DID list in JSON document format. ODL Example is embedded here.

5.4.1 VIL Fields

VIL schema is embedded below.
The table below describes all the fields of the interrogator file (VIL) and IVSU cloud usage
See VIL Definitions

5.5 Post Vehicle Interrogator Log

The POST Vehicle Interrogator Log is HTTPS POST of Vehicle Interrogator Log to the Ford web service end-points.
The service end points are defined in the Web service end points and message format section.
When the Ford backend cloud server receives the Vehicle Interrogator log, it does the following:

- Detects any replay attack requests and filter them from processing and hitting Ford back end servers.
- Authenticate a request by validating request signature composed by module and decrypt the payload for further processing
- Forward uploaded interrogator files to Ford back end systems that maintain vehicle update history.

Any of the following could be the response from Ford back end server from Vehicle Interrogator log post:

- No update available
- Update available
- HTTP error response
- SyncP Service Type 1 error messages (like Message ID out of sequence, etc.)

5.5.1 Client Modules embedded in the vehicle

Client modules such as TCU and SYNC that are part of vehicle architecture will be built with a base ODL saved in its protective memory.
Once the module receives an ODL from the cloud, that ODL should be considered by the module as the latest schema information required by the cloud. Therefore, it shall be saved in its protective memory and shall be used to generate the DIL that will be posted to the cloud. Every time the module receives an ODL, the latest file shall be saved and replace the previous received file.
In cases of corruption or bad received ODL, then the module should revert back to the base ODL that was built with.

5.5.2 Client Modules not embedded in the vehicle

Client Modules that are not embedded in the vehicle will first request and ODL, prior to generating a VIL for the vehicle. See Get ODL for more details.

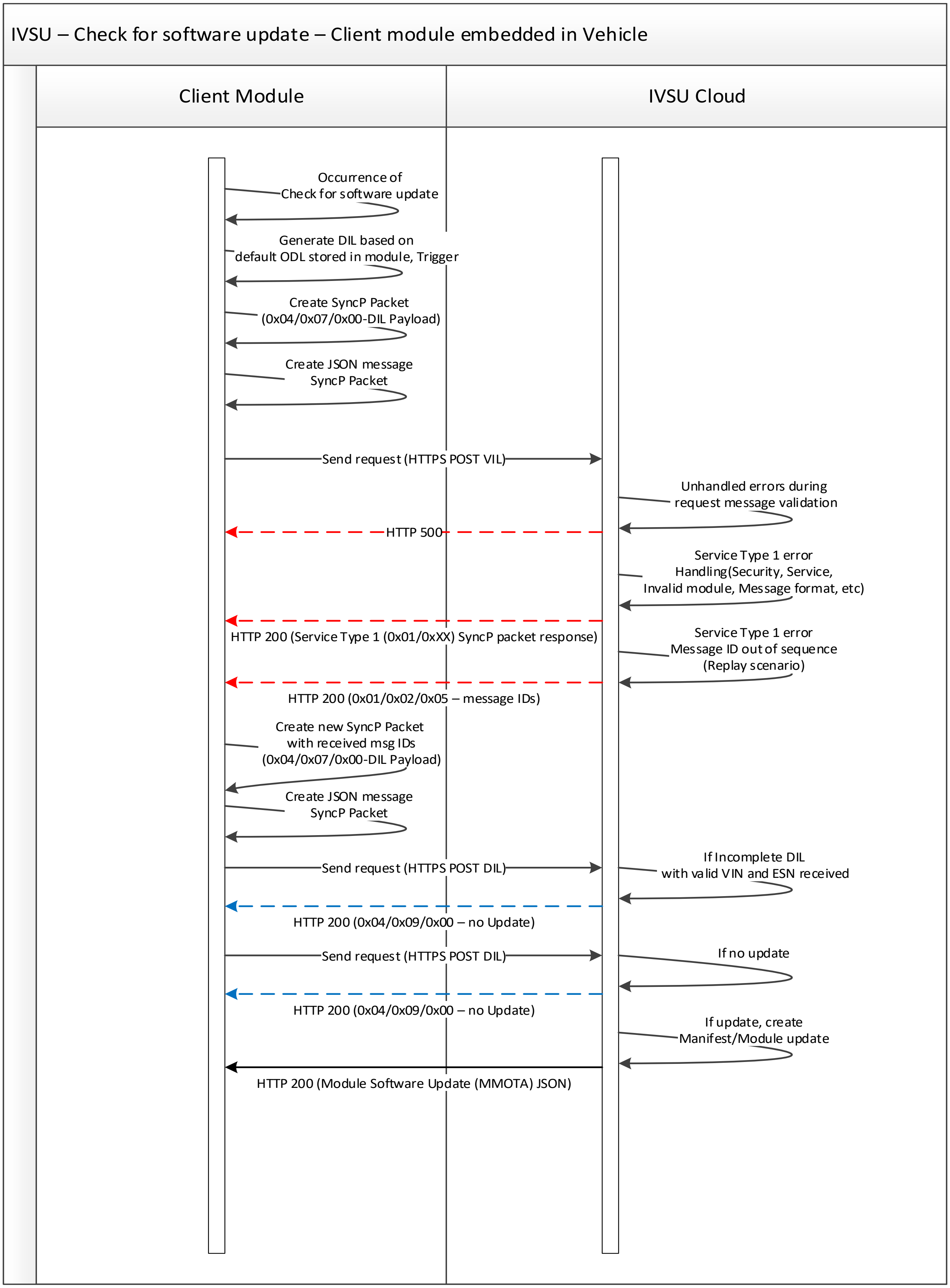

IVSU – Check for software update – Client module embedded in Vehicle
Client Module
IVSU Cloud
Occurrence of
Check for software update
Generate DIL based on
default ODL stored in module, Trigger
Create SyncP Packet
(0x04/0x07/0x00-DIL Payload)
Create JSON message
SyncP Packet
Send request (HTTPS POST VIL)
Unhandled errors during
request message validation
HTTP 500
Service Type 1 error
Handling(Security, Service,
Invalid module, Message format, etc)
HTTP 200 (Service Type 1 (0x01/0xXX) SyncP packet response)
Service Type 1 error
Message ID out of sequence
(Replay scenario)
HTTP 200 (0x01/0x02/0x05 – message IDs)
Create new SyncP Packet
with received msg IDs
(0x04/0x07/0x00-DIL Payload)
Create JSON message
SyncP Packet
Send request (HTTPS POST DIL)
If Incomplete DIL
with valid VIN and ESN received
HTTP 200 (0x04/0x09/0x00 – no Update)
Send request (HTTPS POST DIL)
If no update
HTTP 200 (0x04/0x09/0x00 – no Update)
If update, create
Manifest/Module update
HTTP 200 (Module Software Update (MMOTA) JSON)

Figure 3.3.3.1-1 Software Update check for module embedded in vehicle - sequence diagram

5.5.3 Reporting VIL

When the customer switches automatic updates to off the module sends the VIL with the purpose field set to “Report”. Along with a Customer Consent Message to the cloud.

5.5.4 Post Activation VIL

When the vehicle completes a successful activation, it sends a VIL with the purpose field set to “PostUpdate” to the cloud.

5.5.5 Software Update Request

5.5.5.1 Software Update Request SyncP Sample Layout (0x44/0x00)

The following SyncP message layout is valid for inbound messages to IVSU cloud API to request available software for MOTA.

Octet\Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	Protocol version 0			Response Required **0**	High Bandwidth **1**	Signed **1**	Encrypted **0**	Has ESN **1**
2	Service Type **(44)**							
3	Version/Command Type **(00)**				CPU Destination **0/1**	Encryption Key Index **Any value in range (0-7)**		
4..7	Payload Size (**Size of the payload**)							
8..15	[ESN] (**XN3W2HRR**)							
16..19	Module Message ID (**512**)							
20..23	Server Message ID (**600**)							
24	Message Status (0x00)							
25..40	[IV] (**15 byte IV used for encryption**)							
41..n	Payload (**XML Interrogator File**) *see sample in component’s section* *Note: The payload will be encrypted using IV/one of module keys and encoded in Base64 string to preserve integrity of the bytes.*							
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*							

5.5.5.2 VIL Definitions

Element Name	Description	Mandatory	Occurrence	Data Type	IVSU Cloud Usage
Version	Version to describe or identify its schema version	N	0 or 1	Date	Cloud is not currently using this element. The objective of this element is in case if there is a schema change. Only the new module might be programmed or installed to use this new schema. The modules that are already fitted in the vehicle which are used by consumer might still have older schema. New modules would populate this field in the DIL for cloud usage.
Campaign ID		Y	1	String	Copied from the Trigger Message.
Trigger Type		Y	1	String	Copied from the Trigger Message. Maybe hold one of the following values: “Software Update Trigger” “Add Application” “Remove Application” “Post Activation”
~~purpose~~		~~Y~~	~~1~~	~~String~~	~~May be set to:~~ ~~updateCheck~~ ~~postUpdate~~ ~~report~~
VIN	Vehicle Identification Number	Y	1	String	VIN number of the vehicle where the module is installed and this request is originating from.
reportingNodeAddress	Module Name for specifying the ECU acronym	Y	1	String	Identifies the Node that is contacting the cloud.
ModuleName	Module Name for specifying the ECU acronym	Y	1	Enumeration	Enumeration values - ECU, APIM. No specific business rules as of today. It is just a pass through value to cloud and to Core.
Request Role					
Role	Role attribute identify the service caller role	Y	1	Enumeration	Enumeration Values - "CONSUMER","DEALER","AFTERMARKET","MOD CENTER","EOL","ECATS","VHR","MFR","ENGINEER","PLANT TECHNICIAN","TESTCONSUMER","OTA","TESTOTA". Although there are several values listed in the enumeration as of today Sync Gen3, TCU 4G and TCU 4G uses "CONSUMER" in its request.
RoleSource	RoleSource attribute identify the service caller role source	Y	1	Enumeration	Enumeration Values - "PTS","IDS","ETIS","SMR","CKS","LCS","ECATS","EOL","VHR","FCS","OTA". This attributes major identifies the application/system that is calling this service. In case of TUC or Sync Gen 3 module since it is not a system it uses the role source "OTA" to identify itself.
BroadcastDTCType					

DTC	Diagnostic Trouble Code	N	1,∞	String	IVSU cloud has no usage or business rule for the values if in case sent by the modules (Sync Gen3 / TCU). Basically IVSU ignore the values sent and also does not pass these value to GiVIS core.
Node			1,∞		Consists of address, ECUAcronym, specificationCategory, ODLNetwork, and DIDs.
Specification Category	Specification Category of the Hardware of that module	N	0	String	Possible values are GGDS and GDS. No usage or business rules for IVSU but in GIVIS Analyze log service would validate for the presence of this value in this element since this value is mandatory in the GetCurrent schema.
Address	Node address assigned for that module	Y	1	String	Hex value denoting the specific node address for that module. For Sync it is 7D0 and for TCU it is 754. This will be replaced with a more specific ID for FNV2. To be determined at a later date.
ECUAcronym					
name	ECUAcronym name assigned for that specific Node address	N	0,1	String	ECUAcronym values such as APIM, TCU, BCM, etc.. Pass through values stored in GiVIS core.
DID		Y	1,∞		
didvalue	DID value	Y	1	String	Address of the DID, primary identifier of each DID. (Unique within each gateway of a node?) e.g., F188, E21A,E217, E219. etc.
didType	Did Type	Y	1	String	Type description of each DID and its response. e.g., Strategy Software is the DID type for DID value F188.
didFormat	DID Format	Y	1	String	Indicates the format of DID response in the module. Regardless, format received by IVSU is expected as ASCII unless the Type is configuration.
responseLength	Response length of the DID	Y	1	String	Indicates the length of DID response e.g. 3 means 3 bytes.
response	DID response for the DID value	Y	1	String	Actual DID response for example 9L3T-14D212-AA. For 8060 see AppDescriptor
AppDescriptor(s)	Applications associated with DID.	N	1,∞	JSON Object Array	JSON Object that contains information that describes an ALM application. Used for determining if an Update is available for the application, and reporting ALM state to the cloud. This object will be embedded in an ascii response for in 8060+ range DID. Example: [{\"AppID\":\"Harman_02392\",\"AppName\":\"HArman Navigation\",\"AppVersion\":\"1.2.3\",\"AppType\":\" Navigation\",\"AppDescription\":\"Some description of sorts...\"}] Note: The escaped quote represents a single byte they are escaped so that they do not conflict with the VIL JSON Schema.

					Total number of Apps that can be reported in each 806X DID will be determined by the Part 2 for the module, and with an upper limit for 4KB.
AppID	**Unique Application Identifier**	**Y**	**1**	**String**	**See AppDescriptor.**
AppName	**Human readable name of the application.**	**Y**	**1**	**String**	**See AppDescriptor.**
AppVersion	**Application Version**	**Y**	**1**	**String**	**See AppDescriptor.**
AppType	**Simple description of Application category.**	**Y**	**11**	**String**	**See AppDescriptor.**
AppDescription	**Human readable description of Application**	**Y**	**1**	**String**	**See AppDescriptor.**
isConfig	**This attribute represents whether that DID is a config or not**	**N**	**1**	**Boolean**	**Set to True if DID is a configuration DID.**
isPrivateNetwork	**Not Used**	**N**	**0**	**Boolean**	**Not used. No need to populate values. If populated IVSU would ignore.**
isVinSpecific	**Not Used**	**N**	**0**	**Boolean**	**Not used. No need to populate values. If populated IVSU would ignore.**
ComplainceDID	**Indicates that deviation exists and value is the deviation DID.**	**N**	**0 to Many**		**Element populated in case there is a deviation.**
didvalue	**Compliance DID value**	**N**	**1**	**String**	**Address of the Compliance DID, primary identifier of each DID.**
response	**Compliance DID response**	**N**	**1**	**String**	**Actual Compliance DID response in Hex format for example 394c33542d3134443231322d4141 which is hex conversion of 9L3T-14D212-AA**

Software	**Not Used**	**N**	**0**		**Not used. No need to populate values. If populated IVSU would ignore.**
ESNMetadata	**Not Used**	**N**	**0**		**Not used. No need to populate values. If populated IVSU would ignore.**
DTC	**Not Used**	**N**	**0,1**		**Not used. No need to populate values. If populated IVSU would ignore.Should remain for FCSD use.**

5.5.5.2.1 Example VIL

ProtoVILCombined7D0_716Schema_r3.json

VilCloud_r6.json

5.5.5.3 Consent

In addition to the VIL being sent a User Consent Sync Packet maybe included. The cloud will update consent prior to attempting to determine if an update is available.
 See Send Customer Consent.

5.5.5.4 Sample HTTPS Payload

Sending VIL along with Customer Consent
{“data”:[“base64(SyncP with VIL)”, “base64(SyncP with Customer Consent)”]}
Sending VIL along without Customer Consent
{“data”:[“base64(SyncP with VIL)”]}

5.5.6 No Update available scenario

When a client modules receives no update response packet. It will receive the following packet with no payload.

5.5.6.1 No Update Response (0x44/0x07/0x00)

Octet\Bit	0	1	2	3	4	5	6	7
1	Protocol version **0**			Response Required **0**	High Bandwidth **1**	Signed **1**	Encrypted **1**	Has ESN **1**
2	Service Type **(44)**							
3	Version/Command Type **(07)**				CPU Destination **0/1**	Encryption Key Index **Any value in range (0-7)**		
4..7	Payload Size (**Size of the payload**)							
8..15	[ESN] (**XN3W2HRR**)							
16..19	Module Message ID (**513**)							
20..23	Server Message ID (**601**)							
24	Message Status (0x00)							
25..40	[IV] (**15 byte IV used for encryption**)							
41..n	None							
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*							

5.5.7 Update available scenario

When an Update is available, the module will receive at least one vehicle update manifest JSON file, encoded in a SyncP array. If an Ethernet connected module has a software update it will receive its own manifest file which shall only include items for itself to process. This shall be forwarded to the Ethernet connected ECU so that it may be processed.

The module, which receives the manifest, will download and install each of the items in the JSON Manifest.
If the manifest is transmitted over an unsecure medium, it shall be signed by Ford so that it may be verified at its final destination.

5.5.7.1 Software Update Response (0x44/0x06)

Octet\Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	Protocol version **0**			Response Required **0**	High Bandwidth **1**	Signed **1**	Encrypted **1**	Has ESN **1**
2	Service Type **(44)**							
3	Version/Command Type **(06)**				CPU Destination **0/1**	Encryption Key Index **Any value in range (0-7)**		
4..7	Payload Size (**Size of the payload**)							
8..15	[ESN] (**XN3W2HRR**)							
16..19	Module Message ID (**512**)							
20..23	Server Message ID (**601**)							
24	Message Status (0x00)							
25..40	[IV] (**15 byte IV used for encryption**)							
41..n	Payload (**Update Manifest File**) *see sample in component's section* *Note: The payload will be encrypted using IV/one of module keys and encoded in Base64 string to preserve integrity of the bytes.*							
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*							

The following SyncP message is returned to the Client module if an update is available for that module.

If IVSU cloud determines if an update is available, the response to ECG module is format as below.
Example HTTP Response Payloads:
Response including two manifests.
{“data”:[“base64(Encrypt(SyncP with Manifest1)), base64(Encrypt(SyncP with Manifest2))]}
Response including a single manifest and a rules file.
{“data”:[“base64(Encrypt(SyncP with Manifest1)), base64(Encrypt(SyncP with Rules)))]}
Response including a single Manifest Alone
{“data”:[“base64(Encrypt(SyncP with Manifest)]}

5.5.8 OTA Rules (Update Procedure)

5.5.8.1 OTA Rules Response (0x44/0x05) Sample SyncP Header

Octet\Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	Protocol version **0**			Response Required **0**	High Bandwidth **1**	Signed **1**	Encrypted **1**	Has ESN **1**
2	Service Type **(44)**							

3	Version/Command Type **(05)**	CPU Destination **0/1**	Encryption Key Index **Any value in range (0-7)**
4..7	Payload Size (**Size of the payload**)		
8..15	[ESN] (**XN3W2HRR**)		
16..19	Module Message ID (**512**)		
20..23	Server Message ID (**601**)		
24	Message Status (0x00)		
25..40	[IV] (**15 byte IV used for encryption**)		
41..n	Payload (**OTA Rules File**) *see sample in component's section* *Note: The payload will be encrypted using IV/one of module keys and encoded in Base64 string to preserve integrity of the bytes.*		
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*		

5.5.8.2 (Update Procedure)Definitions

Field Name	Multiplicity	Description	Type
campaignID	1	Campaign ID from the trigger for tracking.	String
VIN	1	Vehicle Identification Number	String
Phase(s)	1, ∞	Consists of a "phaseNode", "phaseName", and "Actions"	JSON Object Array
phaseName	1	Download, Install, Activate.	String
phaseNode	1, ∞	Node Address for the phase.	String
Action(s)	1, ∞	Always appears along with a Target FESN	String
Action Name	1	Pause, Resume, Cancel, Begin, Notify Cloud.	String
Rule(s)	1, ∞	Each rule consists of a "ruleName", "expectedValue", and type.	JSON Object Array
ruleName	1	Item to be read, example "7D0", or PowerMode.	String
expectedValue	1	Value to test for in order to perform the action.	String
Type	1	DID, CanSignal, Local Variable.	String
nodeAddress	1,0	Node Address where the value is read.	String

5.5.8.3 Sample Manifest:

RulesBreakManifestV12.json

5.5.9 Error messages for POST Dealer Interrogator Log

Ford back end server shall send following type of error messages to Client module.

1. HTTP ERROR codes
2. SyncP Service Type 1 error messages

5.5.9.1 HTTP ERROR codes for POST Dealer Interrogator Log

IVSU Cloud would respond with following HTTP error codes for HTTPS POST DIL.
Client module shall receive generic HTTP error codes (3xx, 4xx and 5xx) from HTTP server and retry based on strategy defined in SPSS.

Sl.No	Error Condition	Client module action	Message Type

1	Unhandled Error IVSU Cloud application	Client module shall retry as per retry strategy specified in SPSS. Retry shall not exceed configurable number of times.	HTTP Status Code 500

5.5.9.2 SyncP Service Type 1 Error Status Codes

	SyncP Service	SyncP Command	Message Status/ Message Type	Message Size Limit	Message Type	Originator	Destination	Payload	Meaning	Comments
Following Table was created based on information from document S13b_SyncP_Services_Assignment_136										
SERVICE TYPE = 1 CLOUD to MODULE										
1	0x01	0x00	0x02	4k	SyncP - Encoded **Not-**Signed **Not-**Encrypted ModId =**0** SrvrID =**0**	Cloud	Module	No	**Service Busy** The module may treat this as equivalent to Service Down (and wait until next ignition cycle) initially. The backend will initially only send Service Down messages, but this protects for eventual rollout of more complex infrastructure components. **Service Down** Backend service is unavailable, module should wait to retry message not initiated directly by a user. **Unhandled** Any unhandled errors encountered when processing the request from the module ***Note: Uses Message ID values of zero***	**Business agreed that for the following three scenario's: Service Busy/Service Down/Unhandled, Ford backend will return ST=1, CT=0, MST=2**
2	0x01	0x00	0x04	4k	SyncP - Encoded Signed **Not-**Encrypted ModId = Unch SrvrID = Server ID+1	Cloud	Module	No	**Invalid Security Type** Service requires signing/encryption/both, but required security was not present in the message. ***Note: It will not increment its own MessageID***	**Business agreed that for all Service Type 1 packets going back from Cloud to Module, Sync P packet need not have to be Encrypted unless there is a payload** ***Note: Spec needs to be updated***

	SyncP Service	SyncP Command	Message Status/ Message Type	Message Size Limit	Message Type	Originator	Destination	Payload	Meaning	Comments
3	0x01	0x02	0x05	4k	SyncP-Encoded Signed Not-Encrypted ModId = **0** SrvrID =**0**	Cloud	Module	**Yes**	**Out-of-Sequence Message** The Message ID presented from the other side was not a valid value. ***Note: Uses Message ID values of zero***	**Replay Attack Scenario**
4	0x01	0x00	0x06	4k	SyncP-Encoded Signed **Not-**Encrypted ModId = Unch SrvrID = Server ID+1	Cloud	Module	No	**Message Size Out-of-Bounds** The message did not pass normal sanity checks for the message size	**Business agreed that for all Service Type 1 packets going back from Cloud to Module, Sync P packet need not have to be Encrypted unless there is a payload** ***Note: Spec needs to be updated*** **Testing is deferred until Ray's utility tool can accept Message Size > 1M**
5	0x01	0x00	0x09	4k	SyncP-Encoded Signed **Not-**Encrypted	Cloud	Module	No	**Service Error** The specified service/sub-service is not registered to a currently valid service	**Business agreed that for all Service Type 1 packets going back from Cloud to Module, Sync P packet need not**

	SyncP Service	SyncP Command	Message Status/ Message Type	Message Size Limit	Message Type	Originator	Destination	Payload	Meaning	Comments
					ModId = Unch SrvrID = Server ID+1					**have to be Encrypted unless there is a payload** ***Note: Spec needs to be updated***
6	0x01	0x00	0x0A	4k	SyncP-Encoded **Not-**Signed **Not-**Encrypted ModId = **0** SrvrID =**0**	Cloud	Module	No	**Invalid Module** The specific module was not found in backend systems. ***Note: Uses Message ID values of zero***	**Cannot Sign as ESN is not found**
7	0x01	0x00	0x0B	4k	SyncP-Encoded Signed **Not-**Encrypted ModId = Unch SrvrID = Server ID+1	Cloud	Module	No	**Invalid Data** Attempt to decrypt or check signature failed. This can be caused by a damaged payload in transit or an error in the keys.	**Business agreed that for all Service Type 1 packets going back from Cloud to Module, SyncP packet need not have to be Encrypted unless there is a payload**

5.5.9.2.1 Replay Attack/Message ID Out of Sequence (0x01/0x02/0x05) scenario

The following SyncP message format is used to send replay message id message from IVSU cloud.

The inbound request to module from takes form as below.

Octet\Bit	0	1	2	3	4	5	6	7
1..p	SyncP Packet Header* (**Standard Header**)							
p+1	Number of Message IDs in payload (1 byte) (**This number depends on number of service type a specific module supports**)							
p+2	Service Type (1 byte) (**Service Type associated with module id’s defined below**)							
p+3..p+6	Current Module Message ID (4 bytes) (**Value of Current Module Message ID from IVSU database**)							
p+7..p+10	Current Backend Message ID (4 bytes) (**Value of Current Module Message ID from IVSU database**)							
p..n	Repeating sections of Service Type, Message IDs (Number x 9 bytes)							
n+1..n+16	[Signature Tag]							

{“data”:[“base64(above SyncP Message”]}

Client module shall parse Server ID and Module ID from above response from server, shall retry HTTP POST DIL with new Server ID and Module ID + 1.

5.6 Download Binaries

Client module binary files download functionalities are as follows:

- Client module shall parse Manifest JSON received from the server.
- MOTA update JSON schema is included in Update available scenario.
- If the MOTA update JSON is empty/invalid format, Client module shall send status update to Ford backend server.
- With a Valid MOTA Update, the Client module shall make HTTP HEAD request to check validity of URL and file size.
- Client module shall send progress / error status messages to Cloud in status update JSON in SyncP (0x44/0x06/0x00)
- Client module shall download binary files via HTTP GET request, based on communication manager strategy (Partial download – Pause/Resume, Refer Client Module SPSS)
- Standards:
  - Protocol: HTTP v1.1
  - Method: GET

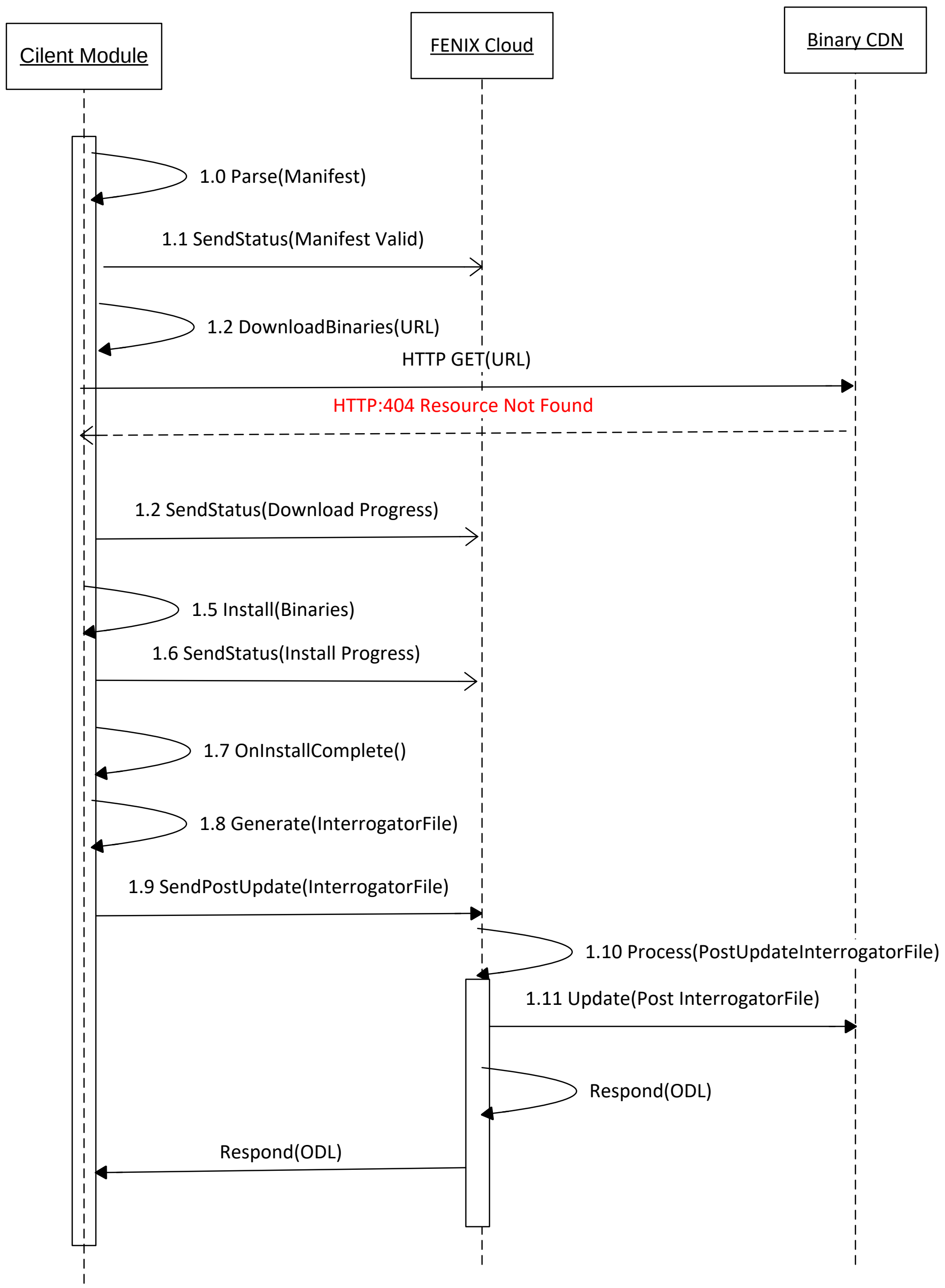

Cilent Module
FENIX Cloud
Binary CDN
1.0 Parse(Manifest)
1.1 SendStatus(Manifest Valid)
1.2 DownloadBinaries(URL)
HTTP GET(URL)
HTTP:404 Resource Not Found
1.2 SendStatus(Download Progress)
1.5 Install(Binaries)
1.6 SendStatus(Install Progress)
1.7 OnInstallComplete()
1.8 Generate(InterrogatorFile)
1.9 SendPostUpdate(InterrogatorFile)
1.10 Process(PostUpdateInterrogatorFile)
1.11 Update(Post InterrogatorFile)
Respond(ODL)
Respond(ODL)

Figure 3.4-1 Download Software Update download/install Sequence diagram

The above system interaction diagram demonstrates at high level sequence of requests to process by Client module and FENIX cloud components for software Download process function.

5.7 Send Regular Status to Cloud

5.7.1 Send status update messages (0x44/0x04/0x00)

Below is a sample SyncP Header for a status message sent from the vehicle to the cloud.

Octet\Bit	0	1	2	3	4	5	6	7
1	Protocol version 0			Response Required	High Bandwidth	Signed 1	Encrypted 1	Has ESN

		0	1			1
2	Service Type **(44)**					
3	Version/Command Type **(04)**		CPU Destination **0/1**	Encryption Key Index **Any value in range (0-7)**		
4..7	Payload Size (**Size of the payload**)					
8..15	[ESN] (**XN3W2HRR**)					
16..19	Module Message ID (**513**)					
20..23	Server Message ID (**601**)					
24	Message Status (0x00)					
25..40	[IV] (**15 byte IV used for encryption**)					
41..n	Status JSON payload					
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*					

5.7.2 Sample HTTP request Payload.

{“data”:[“base64(above SyncP Message”]}

5.7.3 Status update JSON schema is attached below.

statusmessageschema.json

5.7.4 Status update sample json is attached below

sampleStatusMessage.json

ErrorCode in Status Update JSON schema shall be following format:
Size - Unsigned long
First 3 digits of error code shall depict categories of error. Ford Cloud shall use these 3 digits for scheduling related use cases.
Remaining digits are up to supplier to implement with as many as sub type needed These items will be expressed as attributes inside of the look up codes.
For example, the download manager reports an error code 100, this would result in the following look up code in status message:
"lookupCode": " DM_E100”
Alternatively, if additional attributes need to be included for example a file name:
"lookupCode": " DM_E200_GB5T-14G381-AA.tar.gz”
Error code Categories:

100	Low memory
200	Content Integrity errors(Checksum failure)
300	Security Errors(syncP, Signing)
400	Internal errors (Memory failure, overflow)
500	Connection errors (HTTP errors, CAN errors)
503`	Service Busy/Unavailable
600	Configuration
700	Differential
800	Operation condition

Figure 3.3.2.4.4-1 Status message error code categories

5.8 Send Critical Status to Cloud

5.8.1 Send status update messages (0x44/0x04/0x00)

Below is a sample SyncP Header for a status message sent from the vehicle to the cloud.

Octet\Bit	0	1	2	3	4	5	6	7
1	Protocol version **0**			Response Required **0**	High Bandwidth **1**	Signed **1**	Encrypted **1**	Has ESN **1**
2	Service Type **(44)**							
3	Version/Command Type **(0F)**				CPU Destination **0/1**	Encryption Key Index **Any value in range (0-7)**		
4..7	Payload Size (**Size of the payload**)							
8..15	[ESN] (**XN3W2HRR**)							
16..19	Module Message ID (**513**)							
20..23	Server Message ID (**601**)							
24	Message Status (0x00)							
25..40	[IV] (**15 byte IV used for encryption**)							
41..n	Status JSON payload							
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*							

5.8.2 Sample HTTP request Payload.

{“data”:[“base64(above SyncP Message”]}

5.8.3 Status update JSON schema is attached below.

priorityStatusCacheSchema.json

5.8.4 Status update sample json is attached below

PriorityStatusCache.json

ErrorCode in Status Update JSON schema shall be following format:

Size - Unsigned long

First 3 digits of error code shall depict categories of error. Ford Cloud shall use these 3 digits for scheduling related use cases.

Remaining digits are up to supplier to implement with as many as sub type needed These items will be expressed as attributes inside of the look up codes.

For example, the download manager reports an error code 100, this would result in the following look up code in status message:

"lookupCode": " DM_E100”

Alternatively, if additional attributes need to be included for example a file name:

"lookupCode": " DM_E200_GB5T-14G381-AA.tar.gz”

Error code Categories:

100	Low memory
200	Content Integrity errors(Checksum failure)
300	Security Errors(syncP, Signing)

400	Internal errors (Memory failure, overflow)
500	Connection errors (HTTP errors, CAN errors)
503`	Service Busy/Unavailable
600	Configuration
700	Differential
800	Operation condition

Figure 3.3.2.4.4-1 Status message error code categories

5.9 Install Binaries

During the install/program of the software of the files, the client module shall report progress to the FENIX cloud via the status manager.

This shall include:

Successful activation

5.10 Get OVTP Authorization

Once the content in the manifest is downloaded to the vehicle, three separate authorizations requests are made to the cloud. All requests follow the SyncP Header Layout in the following section. The three request consist of the following:

- getAuthEraseProgramDiff
  - authorizeEraseMemory (0x12) - SUCounter X
  - authorizeDownload (0x14) - SUCounter X+1
  - initiateForceSyncCounter (0x1E) - SUCounter X+2

If file is Diff VBF file:

  - diffUpdate(0x18) - SUCounter X+3
  - initiateForceSyncCounter (0x1E) - SUCounter X+4
- getAuthPrepareActRoll
  - prepareActivation (0x1A), - SUCounter X + 5
  - initiateForceSyncCounter (0x1E) - SUCounter X+6
  - authorizeActivation (0x1B) - SUCounter X+7
  - initiateForceSyncCounter (0x1E) - SUCounter X+8
  - initiateRollBack (0x1D) - SUCounter X+9
  - initiateForceSyncCounter (0x1E) - SUCounter X+10
- getAuthVehCtrl
  - Vehicle De-Inhibit (0x00) - CCCounter X + 1
  - Vehicle Inhibit (0x01) - CCCounter X+2

Keep in mind all OVTP Authorization messages are accompanied by Force Software Update counter messages.

~~The Authorization is provided based on Client FESN and Target FESN relationship. This is intended to only allow a vehicles client's credential to update its own modules.~~ (Will address in OVTP Authorization Cloud function.)

The Get OVTP Authorization request may trigger replay scenarios that are handled the same way as checking for a software update as described in "Post Vehicle Interrogator Log".

5.10.1 SyncP Header Layout

5.10.1.1 OVTP Client Authorization Command Request (0x44/0x08/0x00)

Octet\Bit	0	1	2	3	4	5	6	7
1	Protocol version 0			Response Required 0	High Bandwidth 1	Signed 1	Encrypted 1	Has ESN 1
2	Service Type (44)							
3	Version/Command Type (08)				CPU Destination 0/1	Encryption Key Index Any value in range (0-7)		
4..7	Payload Size (Size of the payload)							

8..15	[ESN] (**XN3W2HRR**)
16..19	Module Message ID (**513**)
20..23	Server Message ID (**601**)
24	Message Status (0x00)
25..40	[IV] (**15 byte IV used for encryption**)
41..n	Payload (Gzip compressed Status update XML)
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*

5.10.1.2 OVTP Client Authorization Command Response (0x44/0x09/0x00)

Octet\Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	Protocol version **0**			Response Required **0**	High Bandwidth **1**	Signed **1**	Encrypted **1**	Has ESN **1**
2	Service Type **(44)**							
3	Version/Command Type **(09)**				CPU Destination **0/1**	Encryption Key Index **Any value in range (0-7)**		
4..7	Payload Size (**Size of the payload**)							
8..15	[ESN] (**XN3W2HRR**)							
16..19	Module Message ID (**513**)							
20..23	Server Message ID (**601**)							
24	Message Status (0x00)							
25..40	[IV] (**15 byte IV used for encryption**)							
41..n	Payload (Client Authorization Response JSON)							
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*							

5.10.2 OVTP Authorization Request

Request:

A sample request, "getAuthEraseProgramDiff" is embedded below, and each field is described in the following table:

Field Name	Multiplicity	Description	Type
ecgFesn	1	Unique electronic serial for the module.	String
VIN	1	Include VIN for checking FESN association	String
Version	1		
CampaignID	1	Campaign ID from the trigger the initiated the process.	String
TargetECUs	1	List of target ECU objects which consist of ecuNodeId, ecuAcronym, targetEcuFesn, ecuLogicalAddress, and software.	Array of JSON Objects
functionNames	1, ∞	Set to (getAuthEraseProgramDiff, getAuthPrepareActRoll)	String
ecuNodeId	1	One Node ID is present per Target ECU.	String
ecuAcronym	1	The GMRDB ECU Acronym.	String
targetEcuFesn	1, ∞	Unique serial that identifies the target module.	String

ecuLogicalAddress	1	GMRDB ECU Logical Address.	String
software	1, ∞	Software element is a JSON Array of JSON objects containing sourcePartnumber, destinationPartnumber, sourceFileName, and destinationFileName.	JSON Array of JSON Objects.
sourcePartnumber	1	The Part Number of the software on a module prior to software activation.	String
destinationPartnumber	1	The Part Number of the software on a module after a successful software activation. This information will appear in the update assembly in the manifest.	String
sourceFileName	1	Filename associated with Software Part number prior to software activation. This information will appear in the manifest.	String
destinationFileName	1	Filename associated with Software Part number after a successful software activation. This information will appear in the update assembly in manifest.	String
Cavc	1	JSON object consisting of ecuNodeId, FunctionNames, ecuAcronym, targetEcuFesn, and ecuLogicalAddress. All multiplicities for these elements apply in the same way they do for the TargetEcus object. However it is a single JSON Object rather than an array of JSON objects.	JSON Object

Example requests:

OVTP Authorization Response:

Field Name	Multiplicity	Description	Type
ecgFesn	1	Unique electronic serial for the module.	String
campaignId	1	Campaign ID from the trigger the initiated the process.	String
targetEcuSignedCommands	1, ∞	Each Target ECU consists of a TargetAddress, Targer_FESN, and a list of Commands.	JSON Array, of JSON Objects.
ecuNodeId	1	List of target ECU FESN’s long with the filename the authorization is requested for.	String
ecuLogicalAddress	1	Unique serial the identifies the target module.	String
targetEcuFesn	1, ∞	Ordered List of commands, which each consists of a FID, and a CommandBase64.	JSON Array, of JSON Objects. (“Commands”)

functions	1	List of function which each contain a functionName and software element.	JSON Array of JSON Objects
functionName	1	Short plain description of the function.	String
software	1	List of software objects containing a softwarePartNumber, and Commands.	JSON Array
softwarePartNumber		SoftwarePartNumber associated with the list of commands.	String
Commands		List of commands which consist of sequence, key, and value.	JSON Array
Sequence		Indicates the order in which the command will be used.	String
key		Simple ID of the type of command being used. This is not unique.	String
Value		Base64 encoded command. See OVTP function definition, and OTA Manager specification for details.	String
cavcSignedCommands	1	Signed CAVC commands.	JSON Object
error		Consists of code and a message.	JSON Object
code	1	Identifies the error.	String
message	1	Human readable description of the error response.	String

Sample Response:

getAuthEraseProgramDiff_Response.js

getAuthPrepareActRoll_Response.json getAuthEraseProgram_Response.json

5.10.3 Get Vehicle Inhibit Authorization

Request:

Field Name	Multiplicity	Description	Type
functionNames	1	Set to “getAuthVehStartInhibit”	String
ECG_FESN	1	Electronic Serial Number for the ECG, or Client Module making the request.	String
BCM_FESN	1	Electronic Serial of the Module that executes the inhibit commands.	String
CampaignID	1	Campaign ID ties to the trigger that set the current actions into motion.	String

InhibitSampleRequest.json

Response:

Field Name	Multiplicity	Description	Type
Name	1	Set to “getAuthVehStartInhibit”	String
ECG_FESN	1	Electronic Serial Number for the ECG, or Client Module making the request.	
BCM_FESN	1	Electronic Serial of the Module that executes the inhibit commands.	
CampaignID	1	Campaign ID ties to the trigger that set the current actions into motion.	
Command	1, ∞	Each Command contains a Command Name, and the actual command embedded as a Base64.	

VehicleInhibitResponse.json

5.11 Send Customer Consent

When Consent is required, it is send to the interface for recording. This event is triggered when a customer responds to a dialogue, and the client sends the recorded response.

5.11.1 Send Customer Consent SyncP Message Lay Out (0x44/0x0A/0x00)

Octet\Bit	0	1	2	3	4	5	6	7
1	Protocol version **0**			Response Required **0**	High Bandwidth **1**	Signed **1**	Encrypted **0**	Has ESN **1**
2	Service Type **(44)**							
3	Version/Command Type **(0A)**				CPU Destination **0/1**	Encryption Key Index **Any value in range (0-7)**		
4..7	Payload Size (**Size of the payload**)							
8..15	[ESN] (**XN3W2HRR**)							
16..19	Module Message ID (**513**)							
20..23	Server Message ID (**601**)							
24	Message Status (0x00)							
25..40	[IV] (**15 byte IV used for encryption**)							
41..n	Payload							
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*							

5.11.2 Customer Consent Notification Description

Field Name	Multiplicity	Description	Type
Name	1	Always set to "Customer Consent"	
VIN	1	Vehicle Identification Number	
Campaign	1	Campaign ID, from the trigger.	

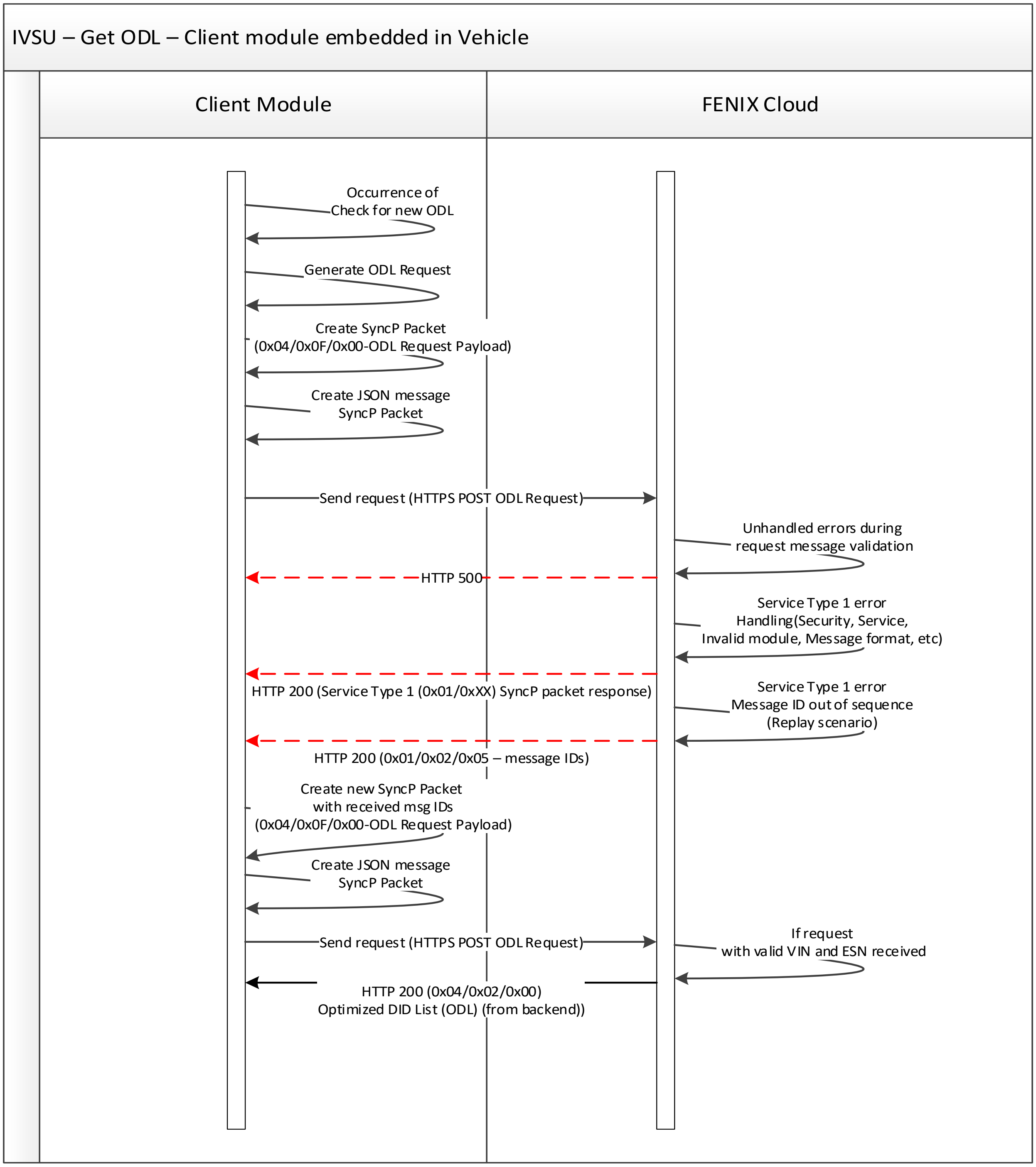
AuthorizationLevel	1	Indicates the level of permission required from the customer.	
Customer Response	1	Recorded response from the customer. This maybe any String as required, but current recommendation, is a Yes or No.	
Date	1	Date and time stamp in UTC. With the following **format string:** YYYY-MM-DDThh:mm:ss+00:00 **Example:** 2017-10-13T19:36:55+00:00	

Sample Customer Consent Notification：

CustomerConsentNotification.json

5.12 Get ODL

When the vehicle requests a new ODL, the Vehicle sends a GetODL request along with the VIN, and campaignID. The Ford backend then responds with the latest ODL for the vehicle program.


IVSU – Get ODL – Client module embedded in Vehicle
Client Module
FENIX Cloud
Occurrence of Check for new ODL
Generate ODL Request
Create SyncP Packet (0x04/0x0F/0x00-ODL Request Payload)
Create JSON message SyncP Packet
Send request (HTTPS POST ODL Request)
Unhandled errors during request message validation
HTTP 500
Service Type 1 error Handling(Security, Service, Invalid module, Message format, etc)
HTTP 200 (Service Type 1 (0x01/0xXX) SyncP packet response)
Service Type 1 error Message ID out of sequence (Replay scenario)
HTTP 200 (0x01/0x02/0x05 – message IDs)
Create new SyncP Packet with received msg IDs (0x04/0x0F/0x00-ODL Request Payload)
Create JSON message SyncP Packet
Send request (HTTPS POST ODL Request)
If request with valid VIN and ESN received
HTTP 200 (0x04/0x02/0x00) Optimized DID List (ODL) (from backend))

5.12.1 Request ODL (0x44/0xD)

Octet\Bit	0	1	2	3	4	5	6	7
1	Protocol version 0			Response Required 0	High Bandwidth 1	Signed 1	Encrypted 0	Has ESN 1
2	Service Type (44)							
3	Version/Command Type (D)				CPU Destination 0/1	Encryption Key Index Any value in range (0-7)		
4..7	Payload Size (Size of the payload)							

8..15	[ESN] (**XN3W2HRR**)
16..19	Module Message ID (**512**)
20..23	Server Message ID (**601**)
24	Message Status (0x00)
25..40	[IV] (**15 byte IV used for encryption**)
41..n	Payload (**VIN**)
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*

Example Request Payload:

```
{
	"TriggerType":"GetODL",
	"VIN":"5LMTJ4DH4GUJ2391X",
	"CampaignID":"DF000000202",
	“ODLSchemaVersion”:”1.3.4”
}
```

5.12.2 Respond ODL to module (0x44/0x02)

The
Ford

Octet\Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	Protocol version **0**			Response Required **0**	High Bandwidth **1**	Signed **1**	Encrypted **0**	Has ESN **1**
2	Service Type **(04)**							
3	Version/Command Type **(02)**				CPU Destination **0/1**	Encryption Key Index **Any value in range (0-7)**		
4..7	Payload Size (**Size of the payload**)							
8..15	[ESN] (**XN3W2HRR**)							
16..19	Module Message ID (**512**)							
20..23	Server Message ID (**601**)							
24	Message Status (0x00)							
25..40	[IV] (**15 byte IV used for encryption**)							
41..n	Payload (**ODL in JSON**)							
n+1..n+16	[Signature Tag] (**Signature hash**) *Computed by signing SyncP header and payload included.*							

Backend will respond to the request with latest ODL associated with the program.

Response to Client module from IVSU cloud shall be in following format.
{“data”:”[“base64(above SyncP message”]”}
Example ODL:

SampleODL_r2.json

ODL Schema:

See OptimizedDIDListType in VehicleModuleInfo_V4.0:

VehicleModuleInfo_V4.0.xsd

6 Cloud API

The purpose of the Cloud API is to allow access to the IVSU feature functionality to other cloud based apps.
This portion of the spec was initially developed for USB but could also be used for testing, and standing up additional features in the near future.

6.1 Get Available

Get Available Software Updates for a provided VIL, without updating the current state.
Requires Login restricted to owners of VIN.
Payload includes a VIL.
See VIL Definitions

6.2 Report State

Requires Login restricted to owners VIN

6.3 Get Current State

Requires Login restricted to owners VIN.

6.4 FUR-REQ-365954/A-VSDN to accept and digest vehicle mode change message

VSDN shall accept vehicle mode updates from vehicles, so that downstream systems can discover VIN's based on vehicle mode.
e.g: Anytime vehicle change mode from factory to transport mode, Transport to Normal, Normal to Factory..etc

6.5 FUR-REQ-365955/A-Software update scheduling in vehicle Transport Mode

VSDN shall be able to send a schedule to the vehicle that determines when the vehicle should perform activation of downloaded software updates while the vehicle is Transport Mode, so that the Governance Board can set this remotely.

6.6 FUR-REQ-365956/A-VSDN to ensure that OTA SMS type are sent only when vehicle is in transport mode

VSDN to send OTA SMS (when there is a pending software update) to the vehicle and to ensure that OTA SMS Type messages that are used to wake up the ECG are only allowed when the Vehicle Mode is set to Transport mode.

6.7 FUR-REQ-365961/A-VSDN to send vehicle mode to VSS and update when it is changed

VSDN shall interface with VSS to provide vehicle mode and to update as the vehicle mode changes. The first mode change shall occur when the vehicle changes from factory to transport mode.

7 Non-Functional Requirements

7.1 Assumptions

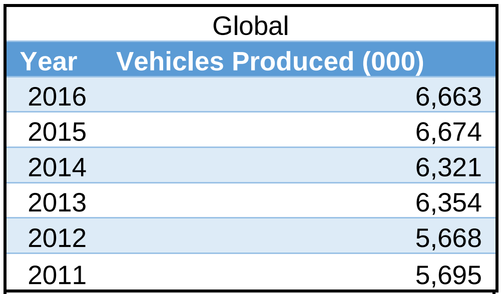
The Non-functional requirements are made with the following assumptions in mind:

1. North America accounts for approximately 50% global vehicle sales.
2. North America's EST and CST time zones account for 80% of the US population.
3. The entirety of China is in a single Time Zone represents the next largest market for Ford vehicles.
4. The Chinese morning rush hour occurs during EST and CST rush hour.
5. This means that th
6. The rush hour time window is approximately 4 hours in the morning and 4 hours in the evening, and 3 hours of each of these overlap between EST and CST time zones.
7. The average age of vehicles on the road is 11.6 years.
8. The average annual number of vehicles produced by Ford over the last 6 years in North America rounds up to 3 million per year.
9. The average annual number of global vehicle produced averages to 6.2 million per year over the last 6 years.
10. vb

Based on these assumptions, we can calculate the maximum number of requests per second
Max number of requests in North America alone is equal to total number of Ford vehicles on the road.
11.6 x 3e6 = 34,800,000
The expected number of requests (R_{avg}) per second based and rush hour(R) window is
R = Number of Vehicles/
FENIX Cloud shall have following Non-functional requirements implemented with Vehicle Client module shall use these parameters for Retry and Timeout.

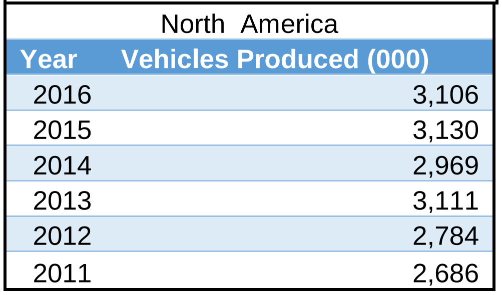
Sl.No	Requirement	Value
1	Roundtrip response time module – cloud	30 seconds
2	Module timeout value	30 seconds
3	Concurrent transactions per second - Normal	625
4	Concurrent transactions per second - Peak	1250
5	Roundtrip response time Application to Cloud API	15 seconds
6	Application to Cloud API timeout value	15 seconds

8 Appendix



Global

Year	Vehicles Produced (000)
2016	6,663
2015	6,674
2014	6,321
2013	6,354
2012	5,668
2011	5,695



North America

Year	Vehicles Produced (000)
2016	3,106
2015	3,130
2014	2,969
2013	3,111
2012	2,784
2011	2,686