Infotainment Systems Product Development

A51s Supplier Feed Specification

SYNC Generation 4

Version 2.14

Version Date: February 12, 2018

UNCONTROLLED COPY IF PRINTED

FORD CONFIDENTIAL

Revision History

Date	Version	Created/Modified By	Notes
3/30/2008	0.10	Michael Westra/mwestra	Final version 3.0 from Gen1 carried forward as baseline draft.
2/9/2009	0.20	Michael Westra/mwestra	Initial draft in new format, minor updates to add new feed elements.
3/12/2009	0.30	Michael Westra/mwestra	Added requested compression support feature.
6/16/2009	0.40	Michael Westra/mwestra	Added clarifying sections requested by Ford team and supplier teams (mostly related to where the data is generated and presented to the system).
7/13/2009	0.41	Michael Westra/mwestra	Closed several open items.
10/17/2011	1.00	Michael Westra/mwestra	Update Continental version.
3/11/2013	1.10	Michael Westra/mwestra	Updated for ESN prefixes and BT MAC address handling.
5/13/2013	2.00	Michael Westra/mwestra	Update for Gen2.5/PCA new hardware/software release.
5/14/2013	2.01	Michael Westra/mwestra	Minor update to remove display type, explicitly call-out ordering for the data, and data retention.
8/22/2013	2.02	Michael Westra/mwestra	Added prefixes for more SYNC programs.
9/12/2013	2.03	Michael Westra/mwestra	Added prefixes for more SYNC programs.
4/8/2014	2.04	Michael Westra/mwestra	Added prefixes for more SYNC programs.
12/2/2014	2.05	Michael Westra/mwestra	Added prefixes for more SYNC programs and TCU programs.
3/18/2015	2.06	Michael Westra/mwestra	Added prefixes for more SYNC programs.
03/16/2016	2.07	Dave Erkkila/derkkila	Added explicit statement specifying the use of only uppercase for alpha characters in the ESN. Removed reference to B36 formatting (confusing). Added prefixes for SYNC, TCU and Flashing Tool programs
9/29/2016	2.08	Dave Erkkila/derkkila	Added new ESN prefixes. Changed GECHub IP Address to Domain Names.
11/22/2016	2.09	Dave Erkkila/derkkila	Added ESN prefixes for LCIS program
2/27/2017	2.10	Dave Erkkila/derkkila	Removed ambiguity and contradictory statement regarding checksums in sections 2.3.3 and 2.3.4
1/25/2018	2.11	Dave Erkkila/derkkila	Added ESN prefix for Panasonic5 Added ESN prefix for PaaK program
01/30/2018	2.12	Mustafa Tambawala/mtambawa	Multiple changes to accommodate SYNC 4 specs for FNV2.
02/03/2018	2.13	Mustafa Tambawala/mtambawa	Updated schema to allow specific value for module type in metadata. Changed ESN to FESN in section 2.2. Updated validation example in section 2.2. Updated Package ID description in section 2.4.4
02/06/2018	2.14	Mustafa Tambawala/mtambawa	Changes made to this version is highlighted in yellow 1. Spec name changed from A51 to A51s 2. Section 2.2 response file validation note 3. Section 2.4.1 updated Ford GecHub mailbox. Added text for QA and Prod GecHub environments. 4. Section 2.4.4 updated manufacturing facility code description and example. Changed Sender/Receiver example to

			uppercase. Updated Bluetooth description. 5. Section 3.3 added 411 error code. 6. Bluetooth spell correction in XML example and schema. DID correction in XML example and schema. 7. Section 2.4.6 reworded some of the text. 8. Spell correction ran on whole document.

Table of Contents

1 Introduction

1.1 Executive Summary

Module supplier feed is an extremely critical part of Connected Vehicle life cycle, affecting various stages such as manufacturing at plant to customer's utilization of Connected Vehicle Features.

The SYNC Infotainment system, containing a hardware module coupled with an Infotainment operating system, will deliver integrated hands-free phone usage and peripheral device audio streaming. The system offers ensured connectivity to most popular electronic devices.

These advanced features of the SYNC module require development of systems and security infrastructure underneath SYNC and those systems to manage many of the complexities involved with software installation, personalization, digital "copyright" signature processes and online consumer purchase interface.

These advanced features offered by the SYNC requires necessary security infrastructure to protect the information exchanged between the SYNC and the cloud based ecosystem.

Supplier feed will be divided into two separate feeds (Refer to data flow diagram in appendix section).

1. Ford to Module manufacturer.
    - Ford will provide list of serial numbers for the modules FESN, previously referred to as ESN, associated with Security Package ID. The detailed design of this feed will be designed and specified by Ford Security team. (IVSS)

2. Module manufacturer to Ford
    - Manufacturer, after receiving first part of the data and flashing the security data on the module, will send a feed file to Ford IT system along with other module specific information. This feed file will be similar to existing supplier feed process with some changes illustrated in this specs.

1.2 Purpose of Document

The purpose of this document is to document details for the data feed to receive, process, and validate SYNC and SYNC-like module-specific details from the supplier. The document will focus on the following areas:

- Flow of data through the process
- GecHub interaction details
- The use of encryption and signing
- Format of the data, including an XML Schema
- Return Receipt details

1.3 References

This section contains references to documents which affect the requirements presented in this requirement specification.

Reference Title	Document Location
GEC Hub Reference	http://www.gsec.ford.com/
GEC Hub FAQ	http://www.gsec.ford.com/GEC/faqs.htm,
GEC Hub FTP-VPN	http://www.gsec.ford.com/GEC/doc/FTP-VPN.pdf
Key Packaging	https://team.extsp.ford.com/sites/FordConnectedServices/tcu/Features/Supplier%20Feed/IVSS/Key%20Packaging%20Spec_V0.71.pdf
FESN Generation	https://team.extsp.ford.com/sites/FordConnectedServices/tcu/Features/Supplier%20Feed/IVSS/FESN%20Generation%20Spec_V0.6.pdf

1.4 Terminology and Abbreviations

Term	Description
GecHub	Ford EDI infrastructure that provides for inter-company transfer of information via a mailbox architecture (via an FTP interface).
OpenSSL	Open source implementation of the SSL/TLS protocols, this tool also provides

Term	Description
	general CA, SMIME, etc. See OpenSSL man page for additional details.
SMIME	Secure/Multipurpose Internet Mail Extensions is a standard public key encryption and signing of payloads encapsulated in MIME. See RFC 3852 for additional information.
PKCS#7	Defined in RFC 2315 and used to sign/encrypt messages within a PKI infrastructure and formed the basis for SMIME security.
ZLIB	Zlib is a software library used for data compression and leverages the deflate algorithm used in gzip.
FESN	Ford Electronic Serial Number is a unique per module ASCII identifier.
GIVIS	Global In-Vehicle Information System
GVMS	Global Vehicle Management System
AES	Advanced Encryption Standard (Symmetric key encryption)
FNV2	Fully Networked Vehicle 2
DER	Distinguished Encoding Rule
PEM	Privacy Enhanced Mail
ECG	Enhanced Central Gateway

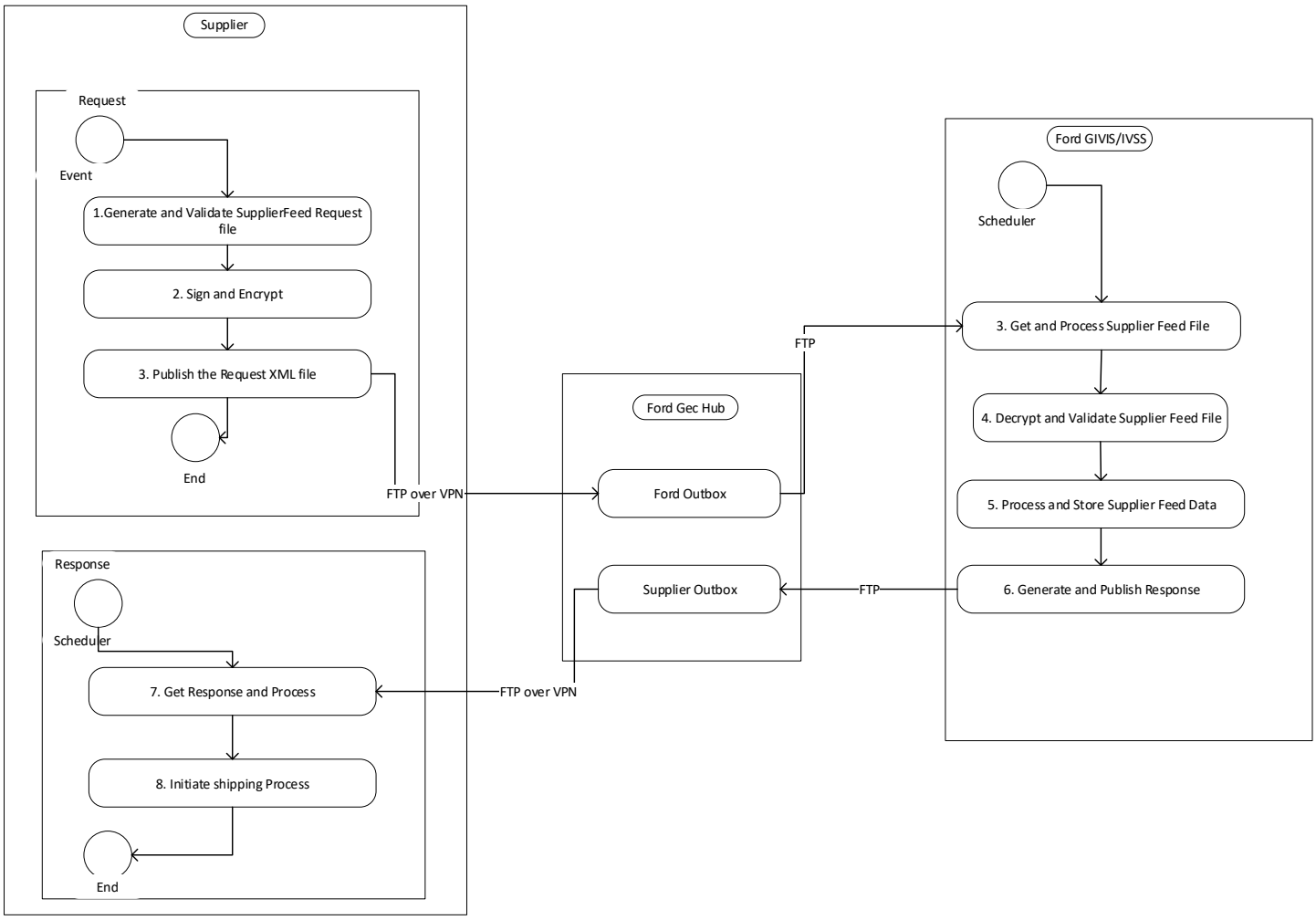
1.5 Schedule and Dependencies

This section will be identified by the supplier in accordance with the module production volume on a day-to-day/week basis.

- Supplier will be allowed to send one or multiple files on a daily basis.
- Each file volume cannot exceed a threshold of 3000 modules. If supplier has more than 3000 modules to be sent for processing, these modules should be scattered over multiple files.
- Test feeds required earlier during prototyping phases may be delivered via secure email, but later prototype builds should exercise the system.

2 Feature Design Requirements

2.1 Architecture



2.2 SYNC Supplier(s)

The high-level flow for the data feed is as follows:

- The supplier will create and process the data for SYNC metadata feed
  - Supplier will receive the FESN and Package ID attributes as a part of security package from Ford System. Please refer to Key Packaging specification for more details on this in section 1.4 and high level dataflow diagram in appendix section.
  - The supplier will generate module-specific data as part of the production process.
  - This data is stored in a local plant database.
  - On a periodic basis a batch process will be executed to transfer this data to Ford.
  - The data is extracted from the database, encoded in the XML format (described below), encrypted/signed (described below).
    Note: The supplier feed file should be validated to comply with XML schema definitions prior sending to Ford. Please refer to schema and other validation rules defined in other sections of this document.
  - The XML payload is signed by the supplier using a private cert generated and managed by the supplier.
  - The XML payload is then encrypted using the public cert provided by Ford

- The supplier transfers the data file to the appropriate GecHub mailbox location (details below).

- Supplier should set the Supplier Feed's Message Id to "*APRF*" attribute of the Supplier Feed File. Supplier should also set the supplier feed's file name to "SNRF" attribute of the supplier feed file. Refer Section 1.4 for GecHub commands documentation locations. .
- GIVIS will periodically process the mailbox and check for new mail (and return receipts to process).

- Upon receiving a new message, GIVIS will send a request to the Security Services to process a specific request file (as it cannot decrypt the original message).
- The Security Service will validate signature on file and decrypt the incoming data file.
- Security Services will send decrypted XML file to GIVIS.
- GIVIS will process the incoming response
  - GIVIS persists the data
  - GIVIS will perform validation checks (e.g. FESN formatting, FESN uniqueness, FESN and Package ID relation, BT Address, etc.).
  - GIVIS will generate a return receipt for the supplier and include the status for each module (see return receipt detailed section below).
    Note: The supplier response XML *file shall be validated to comply with XML schema definitions prior sending to Supplier GecHub Outbox.* Please refer to schema and other validation rules defined in other sections of this document.
- A return receipt is placed in the mailbox for the supplier to process (see GecHub detail and return receipt section below)
  - The Supplier will read the response XML from GecHub Outbox data. The response data shall be stored at supplier's Manufacturing plant database. Refer other sections of this document for Response file archival requirements.
  - Supplier shall get the Supplier Feed's Correlation Id from "*APRF*" attribute of the Response XML. Refer Section 1.4 for GecHub commands documentation locations.

Note: The Supplier shall not ship the modules which have one more errors in the Response file.

2.3 Goals

Goals include:

- Define high level design for the feed processing sent by the manufacturer and received by the Ford IT systems.
- Establish a general approach for supplier feed exchange for SYNC modules to reduce development efforts for both Ford and suppliers.
- This document details the format, interaction, and details of requirements for this feed of data in general.

2.4 Design Dependencies

2.4.1 GecHub Interaction Details

GecHub is the Global Electronic Commerce Hub used by both SYNC Supplier and Ford Systems to exchange the Supplier Feed and Response files.

In general it can parse and route data using the EDI file formatting standard. For this project, given the high CIA (Confidentiality-Integrity-Availability) rating of the data (3-2-1), EDI format cannot be used. This CIA rating requires that data in flight be digitally encrypted (Confidentiality) and signed (Integrity).

In this mode, GecHub appears much like a traditional FTP service – providing a set of mailboxes where files can be deposited and picked up. Supplier will connect to GecHub thru secure PIVPN connection. A PIVPN tunnel will be established between Ford network and supplier network. GecHub main value comes in that it does provide the infrastructure, especially with ANX/VAN connections to many suppliers. It also provides clean-up of out-of-date data, meaning that data will be deleted after 5 days from out mail box and after 15 days from archive.

Some details required to setup/use GecHub include:

- The filename shall be presented in the following format: **<MessageID>__<DATE>[__status].[PEM|DER]**
- Where:
  - **MessageID**: The unique sequence number for the feed.
  - **Date**: Date that the feed was presented (e.g. 20061207_0700).
  - **__status**: included if this is a response feed.
  - **[PEM|DER]**: file extension based on the file type (DER is preferred).
- The file naming convention is to ensure that file names do not collide.

Sending the file:

- Supplier when sending file to Ford GecHub must use command "%< DestinationMAILBOX ID>%MesssageID%<FileName>%b"
- Where:
    - DestinationMailbox: This parameter indicates that file is destined for which mailbox. When supplier is sending the file, destination mailbox will be Ford mailbox and when Ford is sending receipt for the file, destination mailbox will be supplier mailbox. On Ford side, this destination mailbox is owned by GIVIS team.
    - MessageId: The unique sequence number for the feed generated by the supplier – To be defined only by the supplier.
    - Filename: Described above <MessageID>__<DATE>[__status].[PEM|DER]

Receiving the file:

- When supplier sends the file to Ford mailbox, it is received as following format in Ford mailbox. with "%< SourceMAILBOX>%< DestinationMAILBOX>%MessageID%<FileName>%YYYYMMDDhhmmssss
- Where
    - SourceMailbox : This parameter indicates which source mailbox the file came from. File will be received in the out folder of the Destination mailbox. This is true for both supplier feed request file (Supplier to Ford) as well as response file/ receipt file.(Ford to Supplier)
- Data shall be poled at a minimum daily and optionally more frequently.
- The supplier shall place data on the hub when a shipping event is processed, which is dependent upon shipping volumes (generally 1-2 per week, no more than 4-5 per day).
- The supplier's GSDB supplier mailbox code is supplier defined
- The Ford Internal IT Team's mailbox code is FTPFNVSC (userid FTPFNVSC).
- GecHub have two separate environments QA for test builds and Production for production builds, GecHub hostnames for these environments are provided in GecHub documents refer to section 1.4.
- GiVIS/IVSS Supplier Feed scheduler service will periodically process the GecHub mailbox and check for new feed files. GiVIS/IVSS shall get the Supplier Feed's file name and MessageID from GecHub file name describe above. Refer Section 1.4 for GecHub commands documentation locations for more details.
- Each Supplier shall follow Ford GecHub defined process to connect to GecHub and exchange the files securely. Ford Team will work with individual Suppliers to setup the VPN, GecHub accounts and mailboxes during implementation of Supplier Feed process.

2.4.2 Archival and Retention requirements the Feed and Response XML files:

After processing the feed files GiVIS/IVSS shall archive and retain a copy of Supplier Feed and Response files of all Suppliers for 2 years.
The module supplier shall store a copy of all feed XML files that were sent to Ford for a period of 2 years. Upon Ford request, the supplier shall provide all Feed XML files that were delivered to Ford during the past 2 year period.

2.4.3 Message Encryption and Signing

The supplier and Ford shall support encryption and digital signing to support message Confidentiality and Integrity appropriate for CIA=3-2-1. Specifically the following details describe the high-level requirements:

- Messages shall be formatted in a PKCS#7 message either as binary (DER format) or base-64 encoded (PEM format). See http://en.wikipedia.org/wiki/PKCS for more details and definitions.
    - The binary DER format shall be used over the base-64 PEM version or the SMIME attachment output version.
    - The supplier is expected at its discretion to use the OpenSSL command-line tool to perform this function.
    - Ford will perform the same function using OpenSSL API integrated with Windows libraries included as part of the base Visual Studio environment (which integrates with the HSM's Cryptographic Service Provider).
- X.509 certificates shall be used, with a dedicated key pair for encryption and signing of messages (meaning that a dedicated certificate shall be used for each function). See http://en.wikipedia.org/wiki/X.509 for more details and definitions.
    - The supplier may purchase certificates from a public certificate authority, generate them from an internal PKI, or generate self-signed certificates using openssl at their discretion.
    - Ford will generate certificates from its internal PKI. Additionally, a certificate bundle will be included with certificates for the intermediate and root Certificate Authorities (CA). This is so that the supplier may validate the signature of any messages signed by Ford.

  - The CN, Email, and fields listed in Appendix A (example use of certificate generation) should be filled in on the certificate at minimum.
- Certificates shall be exchanged out-of-band and be replaced on a reasonable cadence.
  - The current plan is to exchange certificates (public keys) via email.
  - Certificates will be generated with a valid life-span of not more than 5 years.

See the Appendix for openssl and Windows examples on how to generate certificates and create encrypted/signed messages.

2.4.4 Data and Attributes of Supplier Feed

The following data values will be retrieved from the supplier:

Field	Size/Type	Description/Source/Location	Example
FESN (Electronic Serial Number) (Same as PSN)	String (8) 64-bit*	Electronic Serial Number value. This is a unique serial number for a given module. See the FESN formatting section for more details on what data is embedded within the number is and how it is formatted. Note: The FESN value shall not be repeated within a given feed – and not repeated in general unless that FESN is replacing one from a previous feed (See resendFlag field).	**1SNF001M**
Package ID	String (40)	SHA-1 hash (Hex ASCII) of unencrypted data package also referred to as SPID.	18132b4a69c29a4d55e4c963c8fd9588557f37a3
ECU Hardware Part Number	String 24-bytes*	Ford part number for the SYNC module hardware. The part number is generated by Ford and given to the supplier.	ABCD-14D203-AAA
VMCU Strategy Software Part Number	String 24-bytes*	Ford part number for the VMCU strategy software. The part number is generated by Ford and given to the supplier.	ABCD-14D205-AAA
CCPU "Image" Software Part Number	String 24-bytes*	Ford part number for the CCPU image software. The part number is generated by Ford and given to the CCPU Image builder.	ABCD-14D206-AAA
ECU Assembly Part Number	Hex/SED/BCD (4-2-4 bytes split) 10-bytes*	Ford part number for the SYNC assembly. An assembly represents a unique collection of parts for a given ECU. An assembly's components parts may include hardware, strategy, calibration, primary bootloader and image. These parts are considered electrical parts and any revision to an electrical part implies a revision to parent assemblies. The part number is generated by Ford and given to the supplier.	**ABCD-14D544-AAA <base-64 encoded> See PartII Spec for details on contents/ RDBMS storage (e.g. for GIVIS)**
Intended Plant Ship Date	Datetime	Intended date and timestamp of when the SYNC module is shipped to plant, in UTC format	2018-07-16T19:20:30.45+01:00
Manufacturing Facility Code	String 10 bytes	A unique code provided by the supplier and signed off by Ford, for each one of their SYNC manufacturing facilities. This code will help isolate post launch issues that may be isolated to the SYNC manufactured at a given manufacturing facility of the supplier	**FLEXDRB1**

Bluetooth MAC Address	String (17) 48-bit	Unique 48-bit Media Access Control address for Bluetooth Chipset of the SYNC modules.	`00:16:6f:1f:c4:e2`
WiFi MAC Address	String (17) 48-bit	Unique 48-bit Media Access Control address for Wi-Fi Chipset of the SYNC modules.	`00:16:6f:1f:c4:e2`
Resend Flag	Boolean	A Boolean value to indicate if a particular SYNC module's metadata is sent again by the supplier as a replacement/re-order part. The backend systems shall use this indicator to determine if a new record has to be created or an existing SYNC module metadata shall be updated with new metadata	`True/False`

Feed metadata shall have the following data elements

Message ID	Positive integer 16-bit	A unique number for the feed generated by both sender and receiver to uniquely identify feed request and response. This is a mandatory attribute that MUST be present in both request and response XML messages.	possible values range from 1 to 65,535
Correlation ID	Positive integer 16-bit	To be used only in the feed response and should be left blank by the sender. The receiver shall always populate this attribute with the MessageId received in the feed request from the sender.	possible values range from 1 to 65,535
Sender	String 15 bytes	Describes the originator of the XML message for both request and response.	FORD FLEXTRONICS
Receiver	String 15 bytes	Describes the recipient of the XML message for both request and response.	FORD FLEXTRONICS
Feed Sent DateTime	datetime	An XML standard date/time stamp when the feed was sent in UTC format.	2018-07-16T19:20:30.45+01:00
Total Records Sent	Positive integer 16-bit	Number of module records sent by the sender	possible values range from 1 to 65,535
Module Type	String 8 bytes	String representing the module type for which feed is generated	Possible values 1 to 8 character string values between [A-Z, 0-9] example SYNC4

Feed Summary should have the following data elements:

Processed DateTime	datetime	An XML standard date/time stamp when the feed was sent in UTC format.	2018-07-16T19:20:30.45+01:00
Total Records Processed	Positive integer 16-bit	Number of module records processed by the receiver	possible values range from 1 to 65,535
Success Count	Positive integer 16-bit	Number of module records processed successfully	possible values range from 1 to 65,535

		(without any errors) by the receiver	
Failure Count	Positive integer 16-bit	Number of module records processed with errors by the receiver	possible values range from 1 to 65,535

- For all data in the feed, leading and trailing white space and nulls will be stripped as the space allocated on the module may be larger than the width of the data.

2.4.5 Supplier Feed Request Example

See the Appendix for the XML Schema document.

The following example describes the format and data to be included in the supplier's feed to Ford:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:SupplierFeedRequest xmlns:tns="urn:ford/Vehicle/SYNC/SupplierFeed/v1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ford/Vehicle/SYNC/SupplierFeed/v1.0 Sync4_1.xsd ">
  <tns:FeedMetadata Receiver="FORD" Sender="FLEXTRONICS" correlationId="1" feedSentDatetime="2001-12-31T12:00:00" messageId="1"
totalRecordsSent="1" ModuleType="SYNC4"/>
  <tns:Module resendFlag="true">
    <tns:SYNCAssemblyPartNumber DID="F113" xsi:type="tns:FordPartNumberType">FA1T-14G087-AB</tns:SYNCAssemblyPartNumber>
    <tns:SYNCHardwarePartNumber DID="F111" xsi:type="tns:FordPartNumberType">FA1T-14G145-AA</tns:SYNCHardwarePartNumber>
    <tns:VMCUProcessor index="0" type="ProcessorType">
      <tns:SoftwarePartNumber DID="F188" type="VMCU Strategy">EJ7T-14G139-AC</tns:SoftwarePartNumber>
    </tns:VMCUProcessor>
    <tns:CCPUProcessor index="0" type="ProcessorType">
      <tns:SoftwarePartNumber DID="8033" type="CCPU Image">EJ7C-14G139-AC</tns:SoftwarePartNumber>
    </tns:CCPUProcessor>
    <tns:FordElectronicSerialNumber>1SN00001</tns:FordElectronicSerialNumber>
    <tns:PackageID index="0">18132b4a69c29a4d55e4c963c8fd9588557f37a3</tns:PackageID>
    <tns:BlueToothMACAddress>00:16:6f:1f:c4:e2</tns:BlueToothMACAddress>
    <tns:WiFiMACAddress>00:16:6f:1f:c4:e2</tns:WiFiMACAddress>
    <tns: IntendedPlantShipDate>2001-12-31T12:00:00</tns: IntendedPlantShipDate>
    <tns:ManufacturingFacilityCode>FLEXDRB1</tns:ManufacturingFacilityCode>
    <tns:ErrorState>
      <tns:ErrorCode>tns:ErrorCode</tns:ErrorCode>
      <tns:Description>tns:Description</tns:Description>
      <tns:Reference>tns:Reference</tns:Reference>
    </tns:ErrorState>
  </tns:Module>
</tns:SupplierFeedRequest>
```

2.4.6 Supplier Feed Return Receipt Example

The following example describes the format and data to be included in Ford's returned response feed to the supplier (Note the status fields may be included when processing the message and may be mixed/matched as needed with the actual data elements):

- All data elements will be included in the output message (if possible – see below).
- In some instances it will not be possible to return any information (e.g. message does not decrypt/sign properly, XML does not parse properly or conform to the schema). In these cases, the MessageId will be extracted from the filename and the appropriate failure status messages will be attached.
- The ErrorState XML node can be attached at two levels: the FeedSummary and Module levels.
  - It is encouraged that as many errors are generated as possible within a given run - to report all issues with data in one pass. To this end, multiple errors may appear in several modules and at both levels.
  - If no errors are found, then a status code of success (see appendix) shall be generated and inserted at both the Module and FeedSummary levels.
- It is encouraged that data be processed (if reasonable), stored, and flagged. If errors are encountered, replacement feeds, using the next appropriate MessageId would contain replacement data, which would have the resend flag field set to true and would clear and overwrite the existing data. The MessageId is never repeated for a new or replacement feed.
  - If any issues exist within a given feed, the supplier will regenerate the entire contents of the original feed (this is at the supplier's request).
- If an issue exists with a single module excluding an XML parsing or schema validation issue, it is required that the system support processing other modules that do not have issues.

- If an issue exists at the overall feed file, the system may refuse to process any modules at its discretion.
- If the feed file fails to decrypt, the signature does not match, the certificates have expired, the XML data does not parse, or conform to the schema, then none of the feed data will be reported back. Only the MessageId (extracted from or containing the filename—see GecHub Section) will be filled in with the appropriate error message.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:SupplierFeedResponse xmlns:tns="urn:ford/Vehicle/SYNC/SupplierFeed/v1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ford/Vehicle/SYNC/SupplierFeed/v1.0 Sync4_1.xsd ">
   <tns:FeedMetadata Receiver="FLEXTRONICS" Sender="FORD" correlationId="1" feedSentDatetime="2001-12-31T12:00:00" messageId="1"
totalRecordsSent="0" ModuleType="SYNC4"/>
   <tns:FeedSummary failureCount="0" processedDatetime="2001-12-31T12:00:00" successCount="0" totalRecordsProcessed="0">
      <tns:ErrorState>
         <tns:ErrorCode>tns:ErrorCode</tns:ErrorCode>
         <tns:Description>tns:Description</tns:Description>
         <tns:Reference>tns:Reference</tns:Reference>
      </tns:ErrorState>
   </tns:FeedSummary>
   <tns:Module resendFlag="true">
      <tns:SYNCAssemblyPartNumber DID="F113" xsi:type="tns:FordPartNumberType">FA1T-14G087-AB</tns:SYNCAssemblyPartNumber>
      <tns:SYNCHardwarePartNumber DID="F111" xsi:type="tns:FordPartNumberType">FA1T-14G087-AC</tns:SYNCHardwarePartNumber>
      <tns:VMCUProcessor index="0" type="ProcessorType">
         <tns:SoftwarePartNumber DID="F188" type="VMCU Strategy">EJ7T-14G139-AC</tns:SoftwarePartNumber>
      </tns:VMCUProcessor>
      <tns:CCPUProcessor index="0" type="ProcessorType">
         <tns:SoftwarePartNumber DID="8033" type=" CCPU Image">EJ7T-14G139-AB</tns:SoftwarePartNumber>
      </tns:CCPUProcessor>
      <tns:FordElectronicSerialNumber>1SN00001</tns:FordElectronicSerialNumber>
      <tns:PackageID index="0">18132b4a69c29a4d55e4c963c8fd9588557f37a3</tns:PackageID>
      <tns:BlueToothMACAddress>00:16:6f:1f:c4:e2</tns:BlueToothMACAddress>
      <tns:WiFiMACAddress>00:16:6f:1f:c4:e2</tns:WiFiMACAddress>
      <tns: IntendedPlantShipDate>2001-12-31T12:00:00</tns: IntendedPlantShipDate>
      <tns:ManufacturingFacilityCode>FLEXDRB1</tns:ManufacturingFacilityCode>
      <tns:ErrorState>
         <tns:ErrorCode>tns:ErrorCode</tns:ErrorCode>
         <tns:Description>tns:Description</tns:Description>
         <tns:Reference>tns:Reference</tns:Reference>
      </tns:ErrorState>
   </tns:Module>
</tns:SupplierFeedResponse>
```

2.5 Security Requirements

PKCS#7 encoding is used to encrypt and sign the feed to support security requirements.
See the Appendix for command line examples.

3 Appendix

3.1 Openssl Examples

The following openssl operations are for example purposes only. They provide a rough guide on how to perform several of the common activities described in the previous section using the openssl command.

- Covert a common certificate generated by Microsoft makecert (.cer) into a PEM encoded certificate:
  `openssl x509 -in cert.cer -inform der -out cert.pem`
  This command invokes the openssl command with the x509 set of options to manipulate/convert certificates.

- Create a self-signed certificate:
  `openssl req -x509 -nodes -newkey rsa:2048 -keyout cert2.pem -out cert2_req.pem -config cert2.config -outform pem -days 300 -batch –verbose`

  This command invokes the certificate request set of options, creates a new certificate, cert2.pem, a certificate request file, cert2_req.pem (used to have a CA sign the certificate). It also takes the certificate configuration file, cert2.config (provided as an example only):

```
[ req ]
default_bits           = 2048
default_keyfile        = privkey.pem
distinguished_name     = req_distinguished_name
attributes             = req_attributes
x509_extensions        = v3_ca
dirstring_type = nobmp

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = AU
countryName_min                 = 2
countryName_max                 = 2
localityName                    = Locality Name (eg, city)
organizationalUnitName          = Organizational Unit Name (eg, section)
commonName                      = Common Name (eg, YOUR name)
commonName_max                  = 64
emailAddress                    = Email Address
emailAddress_max                = 40

[ req_attributes ]
challengePassword               = A challenge password
challengePassword_min           = 4
challengePassword_max           = 20

[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true
```

- Encrypt a file using PKCS#7 encoding (DER|PEM) format:
  `openssl smime -encrypt –binary -in test_message.txt -out test_out.der -outform der cert.pem`
  This command invokes the S/MIME message set of options to encrypt the input file, test_message.txt, in PKCS#7 binary (DER) format using the key specified in cert.pem.

- Decrypt a file:
  `openssl smime -decrypt -in test_out.pem -recip cert.pem -inform der`
  This command invokes the S/MIME message set of options to decrypt the input file, test_out.pem, which is encrypted in PKCS#7 base-64 encoded (PEM) format using the key specified in cert.pem.

- Sign a file using PKCS#7 encoding (DER|PEM) format:

```
openssl smime -sign -binary -in test_message.txt -out test_out.pem -signer
cert4.pem -outform der -nodetach
```

This command invokes the S/MIME message set of options to sign the input file, test_message.txt, which includes the signature and entire message in the base-64 encoded output file, test_out.pem (via –nodetach option) and signed using cert4.pem.

- Verify a message:

```
openssl smime -verify -in test_out.pem -inform der -certfile cert4.pem -CAfile
cert4_root.pem
```

This command invokes the S/MIME message set of options to verify the signature on the input file, test_out.pem. The certificate, cert4.pem, is provided along with any intermediate and root certificates specified in cert4_root.pem.

- Encrypt and Sign a message:

```
openssl smime -sign -binary -outform pem -in test_message.txt -signer cert4.pem -
text | openssl smime -encrypt -binary -out test_out.pem -outform pem cert.pem
```

This command invokes the S/MIME message set of options to sign and encrypt the input file, test_message.txt, using the certificate cer4.pem. The message is then encrypted using cert.pem.

Notice that the message is first signed, then encrypted. This is done to provide additional security (especially necessary if the same certificate is used for both functions). In this example, two different certificates are used – one to sign the message, and one to perform encryption.

3.2 Data Feed XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:ford/Vehicle/SYNC/SupplierFeed/v1.0" xmlns:tns="urn:ford/Vehicle/SYNC/SupplierFeed/v1.0"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="SupplierFeedRequest" type="tns:SupplierFeedRequestType" />
  <xs:element name="SupplierFeedResponse" type="tns:SupplierFeedResponseType" />
  <xs:complexType name="SupplierFeedRequestType">
    <xs:annotation>
      <xs:documentation>Supplier Feed Request element of module metadata
        for Ford FNV2.0 Modules
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="FeedMetadata" type="tns:FeedMetadataType"
        minOccurs="1" maxOccurs="1" />
      <xs:element name="Module" type="tns:ModuleType" maxOccurs="unbounded"
        minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SupplierFeedResponseType">
    <xs:annotation>
      <xs:documentation>Supplier Feed Response element that Ford responds
        to supplier
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="FeedMetadata" type="tns:FeedMetadataType"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="FeedSummary" type="tns:FeedSummaryType"
        minOccurs="1" maxOccurs="1" />
      <xs:element name="Module" type="tns:ModuleType" maxOccurs="unbounded"
        minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
<xs:complexType name="ModuleType">
  <xs:annotation>
    <xs:documentation>
      This section details the information about the
      metadata that Ford shall receive for each SYNC.
      SYNCAssemblyPartNumber – SYNC Assembly Part Number for the Module (ie. DID Value F113 - ex: DM5T-14G087-AA)
      SYNCHardwarePartNumber – SYNC Hardware PartNumber for the Module (ie. DID Value F111 - ex: ???)
      SoftwarePartNumber - Ford Software part number for following softwares on the two ECU.
```

```
        Ford Electronic Serial Number – This is the unique serial number for
        See FESN Generation specification for details on how this value is generated.
        IntendedPlantShipDate – Datetime Module is intended to be shipped to Ford in UTC format
        Manufacturer Code – A unique code given by Ford to the module supplier.
        ResendFlag - Indicates that this module is a resend of a previous feed and should overwrite the
        previously sent module data.
        ManufacturingFacilityCode - A unique code provided by the SYNC supplier that determines which supplier
        facility the part is manufactured at.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <!-- Assembly Part number (DID value = F113 - SYNC Assembly) -->
      <xs:element name="SYNCAssemblyPartNumber" type="tns:FordPartNumberType"
        minOccurs="1" maxOccurs="1" />
      <!-- Hardware Part number (DID value = F111 - SYNC Hardware) -->
      <xs:element name="SYNCHardwarePartNumber"
        type="tns:FordPartNumberType" minOccurs="1" maxOccurs="1" />
      <!-- Strategy Part number (DID value = F188 - Main Micro - also referred
as "firmware_version_teseo2") -->
      <!-- Secondary micro Part number (DID value = F120 - Cellular Micro -
also referred as "firmware_version_he920") -->
      <xs:element name="VMCUProcessor" type="tns:ProcessorType"
        minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="CCPUProcessor" type="tns:ProcessorType"
        minOccurs="1" maxOccurs="unbounded" />
      <xs:element name="FordElectronicSerialNumber" type="tns:FESNType"
        minOccurs="1" maxOccurs="1" />
      <xs:element name="PackageID" type="tns:PackageIDType"
        minOccurs="1" maxOccurs="1" />
      <xs:element name="BlueToothMACAddress" type="tns:MACAddressType"
        minOccurs="1" maxOccurs="1" />
      <xs:element name="WiFiMACAddress" type="tns:MACAddressType"
        minOccurs="1" maxOccurs="1" />
      <xs:element name="IntendedPlantShipDate" type="xs:dateTime"
        minOccurs="1" maxOccurs="1" />
      <xs:element name="ManufacturingFacilityCode"
        type="tns:ManufacturingFacilityCodeType" minOccurs="1"
        maxOccurs="1" />
      <xs:element name="ErrorState" type="tns:ErrorStateType"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="resendFlag" type="xs:boolean" use="required" />
</xs:complexType>
<xs:complexType name="FeedMetadataType">
    <xs:annotation>
      <xs:documentation>
        This section contains details for the overall feed
        file:
        Sender: Describes the originator of the XML message for both request and response.
        Receiver: Describes the recipient of the XML message for both request and response.
        FeedSentDatetime: An XML standard
        date/time stamp when the feed was sent in UTC format.
        MessageId: A unique number for the feed generated by both sender and receiver to
        uniquely identify feed request and response
        CorrelationId: To be used only in the
        feed response and should be left blank by the sender.
        The receiver shall always populate this attribute with the MessageId
        received in the feed request from the sender.
        Total Records Sent: Number of records sent in this instance of feed.
      </xs:documentation>
    </xs:annotation>
    <xs:attribute name="Sender" type="tns:SenderCodeType"
      use="required"></xs:attribute>
    <xs:attribute name="Receiver" type="tns:ReceiverCodeType"
      use="required"></xs:attribute>
<xs:attribute name="ModuleType" type="tns:ModuleCatType"
      use="required"></xs:attribute>
    <xs:attribute name="feedSentDatetime" type="xs:dateTime"
      use="required"></xs:attribute>
    <xs:attribute name="messageId" type="xs:positiveInteger" use="required"></xs:attribute>
    <xs:attribute name="correlationId" type="xs:positiveInteger" use="optional"></xs:attribute>
    <xs:attribute name="totalRecordsSent" type="xs:int" use="required"></xs:attribute>
</xs:complexType>
    <xs:complexType name="SoftwarePartNumberType">
      <xs:simpleContent>
```

```
      <xs:extension base="tns:FordPartNumberType">
        <xs:attribute name="type" type="tns:SoftwarePartType" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="SoftwarePartType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="VMCU Strategy"></xs:enumeration>
      <xs:enumeration value="VMCU Primary Bootloader"></xs:enumeration>
      <xs:enumeration value="VMCU Recovery Load"></xs:enumeration>
      <xs:enumeration value="CCPU Image"></xs:enumeration>
      <xs:enumeration value="CCPU Primary Bootloader"></xs:enumeration>
      <xs:enumeration value="CCPU Recovery Load"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
  <!-- F124 is for calibration - Default Config -->
  <xs:simpleType name="DIDType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="F188"></xs:enumeration>
      <xs:enumeration value="F120"></xs:enumeration>
      <xs:enumeration value="F111"></xs:enumeration>
      <xs:enumeration value="F113"></xs:enumeration>
      <xs:enumeration value="F10A"></xs:enumeration>
      <xs:enumeration value="8033"></xs:enumeration>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="FordPartNumberType">
    <xs:simpleContent>
      <xs:extension base="tns:PartNumberType">
        <xs:attribute name="DID" type="tns:DIDType" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="PartNumberType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z0-9-]{14,24}" />
      <xs:whiteSpace value="collapse"></xs:whiteSpace>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="ProcessorType">
    <xs:sequence>
      <xs:element name="SoftwarePartNumber" type="tns:SoftwarePartNumberType"
        minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="type" type="xs:string" use="required" />
    <xs:attribute name="index" type="xs:nonNegativeInteger" use="required" />
  </xs:complexType>
  <xs:simpleType name="FESNType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z0-9]{8}" />
      <xs:whiteSpace value="collapse"></xs:whiteSpace>
    </xs:restriction>
  </xs:simpleType>
<xs:simpleType name="ModuleCatType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="SYNC4"></xs:enumeration>
      <xs:whiteSpace value="collapse"></xs:whiteSpace>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="PackageIDType">
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:attribute name="index" type="xs:nonNegativeInteger"
          use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="MACAddressType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9a-fA-F]{2}:[0-9a-fA-F]{2}:[0-9a-fA-F]{2}:[0-9a-fA-F]{2}:[0-9a-fA-F]{2}:[0-9a-fA-F]{2}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="SenderCodeType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z0-9]{1,15}" />
```

```
      <xs:whiteSpace value="collapse"></xs:whiteSpace>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ReceiverCodeType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z0-9]{1,15}" />
      <xs:whiteSpace value="collapse"></xs:whiteSpace>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ManufacturingFacilityCodeType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z0-9]{1,10}" />
      <xs:whiteSpace value="collapse"></xs:whiteSpace>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="FeedSummaryType">
    <xs:annotation>
      <xs:documentation>
        This section contains details on feed processing
        summary and will be only set in the feed response from Ford to
        supplier.
        Processed Datetime: An XML standard date/time stamp when
        the feed was processed by Ford.
        Total Records Processed: Number of
        module records processed in this feed from the supplier.
        SuccessCount: Count of records that have been successfully processed
        by
        Ford.
        FailureCount: Count of records that have not been
        successfully
        processed by Ford.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ErrorState" type="tns:ErrorStateType"
        minOccurs="1" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="failureCount" type="xs:int"></xs:attribute>
    <xs:attribute name="successCount" type="xs:int"></xs:attribute>
    <xs:attribute name="totalRecordsProcessed" type="xs:int"></xs:attribute>
    <xs:attribute name="processedDatetime" type="xs:dateTime"></xs:attribute>
  </xs:complexType>
  <xs:complexType name="ErrorStateType">
    <xs:annotation>
      <xs:documentation>
        This element may be attached at the feed summary
        level or at individual module
        level. An ErrorState describes a
        specific error state / condition during processing of the feed.
        ErrorCode: A specific code
        for the given error (see the appendix
        section of
        the Supplier Feed
        Spec (A51s) for a list of error
        conditions).
        Description: A generic
        description for the given error
        code.
        Reference: A specific output
        diagnostic detail for the error.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ErrorCode" type="xs:string" minOccurs="1"
        maxOccurs="1"></xs:element>
      <xs:element name="Description" type="xs:string"></xs:element>
      <xs:element name="Reference" type="xs:string"></xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

3.3 Data Feed Error Codes

- Errors are bundled into groups of common errors

- The groups begin with a general error for that category (e.g. 100 for data feed)
- These general errors should not be present, in general, unless an undefined error occurred

- Errors may occur at the feed or individual module level (depending on the type)
- The use for the fields provided are intended for:
  - ErrorCode – indented to give a general error that a machine could process to take automatic action upon as appropriate.
  - Description – a general description for the error type, which will always directly match the error code.
  - Ref – a more specific, freeform of the information. It should provide additional dynamic information, which would aid a human in diagnosing the problem.

Error Code	Description	Reference Information/ Comment	Result
0	**Success**	**General Success Message.**	Error code added as warning to receipt and file processed normally.
100	**General Feed Error**	**Series of errors over entire data feed file.**	
101	Out of Sequence Feed	More a warning, which should not stop processing of the current data. If sequence numbers are sequential, then this may indicate data is missing. Display the missing sequence range or value expected.	Error code added to receipt and file processed normally.
102	Module Number Mismatch Error	Reported number of modules found and actual number found in the file. Display both value given and value expected. It shouldn't stop the feed from processing.	Error code added to receipt and file processed normally.
103	MessageId mismatch	MessageId in filename does not match messageId in file contents	Error added to receipt; file is not processed further. Fatal Error.
104	Cannot identify Sender	Manufacturer Code (Sender) length not >=1 and <=15.	Error added to receipt; file is not processed further. Fatal Error.
200	**General Message Security Error**	**Series of errors over the entire data feed file related to security formatting.**	
201	PKCS7 Message Error	Message is not in a valid PKCS#7 format. Nothing additional to display.	Error added to receipt; file is not processed further. Fatal Error.
202	Encryption Error	Unable to decrypt the data using key. Display key ID(s) used to attempt decryption.	Error added to receipt; file is not processed further. Fatal Error.
203	Signature Validation Error	Message Signature did not match. Display key ID(s) used to attempt decryption.	Error added to receipt; file is not processed further. Fatal Error.
204	Certificate Expired Error	Encryption or Signing signature has expired. Display valid dates for the certificates.	Error added to receipt; file is not processed further. Fatal Error.
300	**General XML Parsing Error**	**Series of errors dealing with XML parsing.**	
301	XML General Parsing Error	Unable to parse XML data. Display parsing errors from the parsing engine.	Error added to receipt; file is not processed further. Fatal Error.
302	XML Schema Error	Unable to parse XML data against schema. Display parsing errors from the parsing engine.	Error added to receipt; file is not processed further. Fatal Error.
400	**General field validation Error**	**Series of errors dealing with field validation.**	

401	<Field> Length Error	<Field> is a variable which should be replaced by the field name that encountered the error. Length error can occur on Part Number, FESN, Package ID, MAC addresses and Manufacturing Facility Code Length of <field> exceeds expected length of characters Display expected length, actual length and content.	Error added to receipt but file is processed further. Order of priority of validation processing - 1
402	<Field> Format Error	<Field> is a variable which should be replaced by the field name that encountered the error. Format error can occur on Date fields. <Field> is not formatted according to specification. Display specific content issue (e.g. invalid date)	Error added to receipt but file is processed further. Order of priority of validation processing - 2
403	<Field> Invalid Characters Error	<Field> is a variable which should be replaced by the field name that encountered the error. This error can occur on Date fields, FESN, Package ID, Manufacturing Facility Code and Part Number. <Field> contains invalid characters. Display offending characters.	Error added to receipt but file is processed further. Order of priority of validation processing - 3
404	<Field> Duplicate Error	<Field> is a variable that can be replaced by field name that is duplicated. Duplicate error can occur for FESN, Package ID <Field> is duplicated within the database and the resend flag is not set. Display offending FESN.	Error added to receipt but file is processed further. Order of priority of validation processing - 5
411	Invalid FESN value	Display an error if FESN, Package ID record could not be found on GIVIS.	Error added to receipt but file is processed further.
500	**Other General Errors**	**General Error Conditions that don't fit into other categories**	Unable to save information for the module. Error added to receipt but file is processed further.
600	**Custom Defined Error Range**	**Extension point for additional errors**	
700	**Custom Defined Warning Range**	**Extension point for warning state**	No modules found. Warning added to receipt and file processed normally.
<100	**Custom Defined Information Range**	**Extension point for informational messages**	

3.4 Supplier Feed High Level Data Flow