NBS 7.11 Data Ingestion API Testing

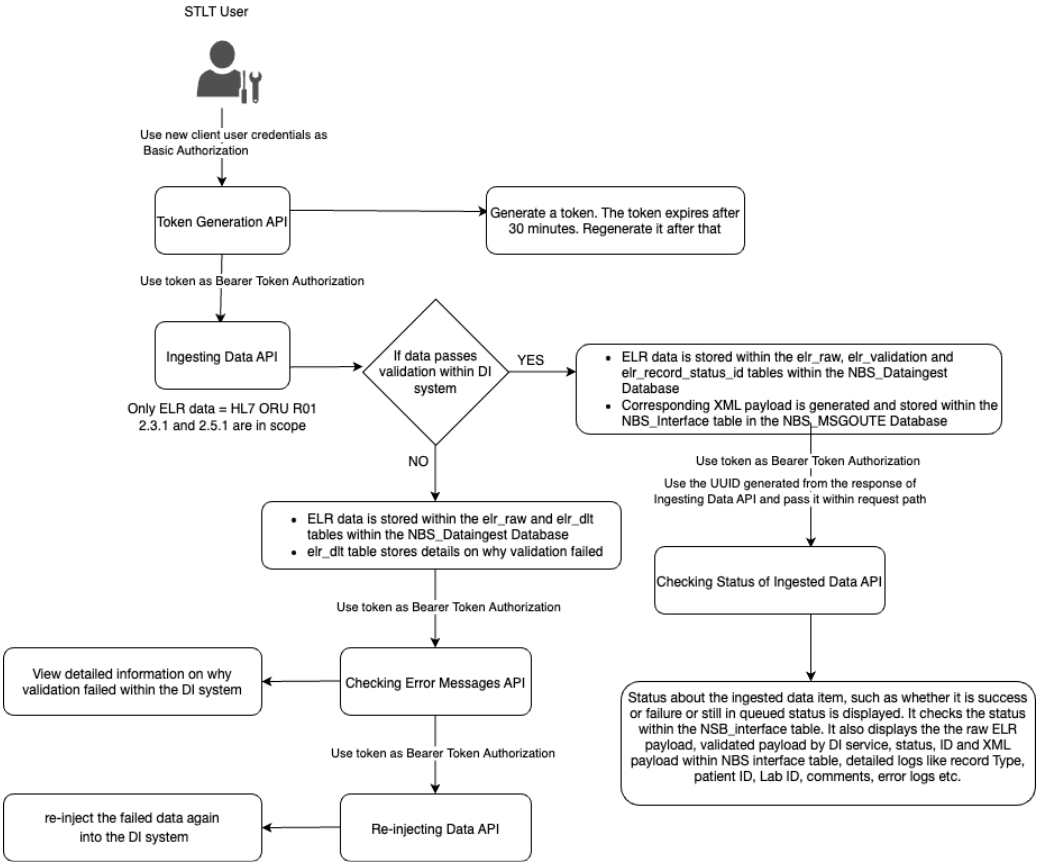
INTRODUCTION

This guide is designed to assist clients who are working with the new Data Ingestion system that ingests HL7 messages into a database. These APIs include Token Generation, Ingesting Data, Checking the detailed status of Ingested Data, Checking the basic status of Ingested Data, Checking Error Message for a specific ID, Checking all the Error Messages, Validation of HL7 message, and Re-injecting Data. If you encounter issues while using these APIs, this guide will help you diagnose and resolve common problems.

SCOPE:

1. Only ELR Data that is HL7 messages with ORU RO1 Data type are in scope for 7.5 release.
2. Only HL7 messages with versions 2.3.1 and 2.5.1 are in scope for 7.5 release. When user tries to post an HL7 message with version 2.3.1, the Data Ingestion system will automatically convert it to 2.5.1 and post it.

FLOW DIAGRAM



SUMMARY TABLE OF THE API ENDPOINTS

SNO	Purpose	Endpoint	Method	Description
1	Token Generation API	https://dataingestion.dts1.nbspreview.com/api/auth/token	POST	Before labs and providers can leverage the Data Ingestion APIs, they must first acquire an authentication token. Registered labs and providers can obtain this token by providing

				their client username and secret. This is implemented using Keycloak
2	Ingesting Data API	https://dataingestion.dts1.nbspreview.com/api/elrs	POST	Armed with the token acquired in the previous step, users can effortlessly ingest HL7 documents using the Data Ingestion system. This process enables users to securely transmit documents stored in HL7 format from their local machines, ensuring data integrity and reliability in the NBS system.
3	Check detail Status of Ingested Data API	https://dataingestion.dts1.nbspreview.com/api/elrs/status-details/{ID}	GET	This endpoint is intended to retrieve the processing status of ingested data in the Classic MSGOUTE Database. By specifying a unique data identifier in the request path, the API should return the raw ELR

				payload, validated payload by DI service, status ,Id and XML payload within NBS interface table, detailed logs like record Type, patient ID, Lab ID, comments, error logs etc.
4	Check basic Status of Ingested Data API	https://dataingestion.dts1.nbspreview.com/api/elrs/status/{ID}		This endpoint is similar to above API but just displays the basic status (success, failure, queued, not in the system) of the ingested API within the NBS interface table.
5	Check specific Error Message API	https://dataingestion.dts1.nbspreview.com/api/elrs/error-messages/{ID}	GET	Data Ingestion system is designed to ensure that every message sent through it is tracked, ensuring data integrity and system reliability. When a message is successfully posted, it gets stored in the main database. However, in the event of an error, the message

				is stored in a separate error table to facilitate troubleshooting and system monitoring. To access these error messages, a dedicated Error Messages API has been provided.
6	Re-injecting Data API	https://dataingestion.dts1.nbspreview.com/api/elrs/{dlt-id}	POST	This endpoint allows user to re-inject the failed data again into the new system
7	Check All Error Messages API	https://dataingestion.dts1.nbspreview.com/api/elrs/error-messages	GET	This endpoint retrieves all the failed messages within the DI system
8	Validate HL7 message API	https://dataingestion.dts1.nbspreview.com/api/elrs/validate	POST	This endpoint is to verify whether a payload is a valid HL7 or not

SAMPLE HL7 MESSAGE

MSH|^~\&|HL7 Generator^^|ANDALUSIA HEALTH^34D3444114^CLIA|ALDOH^OID^ISO|AL^OID^ISO|202408071210||ORU^R01^ORU_R01|20240807121060|P|2.5.1PID|1|205719915^^^ANDALUSIA HEALTH&34D3444114&CLIA|205719915^^^Social Security Administration&SSA&CLIA^SS||Butler^Ronnie^Sarah^Sr^Dr^MED|Benjamin|197103010000|F|kavita|2028-9^Asian^CDCREC|694 Barbara Mills^unit 49292^South Patricia^HI^30342^USA||^^^LauraPruitt60@gmail.com^^732^7205584^4552|^^^RonnieButler60

@yahoo.com^^732^6127393^4552|ENG|T^^^^^|||139-84-0087|||2135-2^Hispanic or Latino^CDCREC|Port Billy|Y|3|||USA|202407171210|YORC|RE||97888856802^Thornton, Jackson and Moore^46D4442442^CLIA||N|||||||||||||||Chang-Bishop|697 Lee Meadows^unit 200^Port Elizabeth^AK^59008^USA|^^^^^^979^5814824^6272|138 Rebecca Pines^unit 29055^Floydhaven^HI^48990^USAOBR|1|60^Test One^111^|503202020x^ANDALUSIA HEALTH^34D3444114^CLIA|77190-7^Hepatitis B virus core and surface Ab and surface Ag panel - Serum^LN^60^TestData^L|S||202408071210|202408121210||||BLD|No people myself.|202408071210||3290103^Adams^James^III^Dr^MD|3651924515^^^DonnaGreen60@icloud.com|||||202408121210|||P|||5467384^Shaw^Phillip^ESQ^Mrs^MPH|||88995^Nonspecific abnormal results of liver funct^56449|15100&Adams&Alejandro&&V&Mr&APN^202408071210^202408121210OBX|1|NM|77190-7^Hepatitis B virus core and surface Ab and surface Ag panel - Serum^LN|1|100^1^:^1|mL|10-100|H|||F|||||||202408071210SPM|1|2391204&Test One&111&^4813037&ANDALUSIA HEALTH&34D3444114&CLIA||GEN^Genital^HL70487^DIAF^Dialysis fluid^SCT^2.5.1^Genital|||DIAF^DIAF^TG|335525591^seat^SNM||||60^ML||Realize avoid sign western whom message.|||202408071210^202408121210|202408071210|||

TOKEN GENERATION API

Obtaining Tokens for Labs and Providers:

The Token Generation endpoint is designed to authenticate users and provide a token for authorized access to secured endpoints.

Endpoint: https://dataingestion.dts1.nbspreview.com/api/auth/token

Pre-Requisite: The clientid and clientsecret has been created by the STLT Administrator

HTTP Method: POST

Authorization Type: NONE

Custom Headers: Enter 2 new headers called clientid and clientsecret along with their values.

Note: Tokens expires after 1 hour. There is no way to refresh tokens at this point. The tokens must be regenerated again after an hour to ingest ELR data. It's essential to keep this token regeneration requirement in mind to ensure uninterrupted access to the Data Ingestion system.

SNO	Scenario	Custom Headers	HTTP Method	Response Code	Response message
1	Allow client users to generate a new token required to ingest the HL7 messages.	Add 2 Request headers with the key/value pairs as seen below: clientid: clientsecret: Enter their values acordingly .	POST	200	Token string is displayed
2	Do not allow users to create a new token when clientid header info is missing	Add only 1 Request headers with the key/value pairs as seen below: clientsecret:somevalue .	POST	400	{ "type": "about:blank", "title": "Bad Request", "status": 400, "detail": "Required header 'clientid' is not present.",

					"instance": "/token" }

3	Do not allow users to create a new token when clientsecret is missing	Add only 1 Request headers with the key/value pairs as seen below:
		clientid:somevalue
			POST	400	{
					"type": "about:blank",
					"title": "Bad Request",
					"status": 400,
					"detail": "Required header 'clientsecret' is not present.",
					"instance": "/token"
					}

4	Do not allow users to create new token when both clientid and clientsecret header key/values are not added	NONE	POST		{ "type": "about:blank", "title": "Bad Request", "status": 400, "detail": "Required header

					'clientid' is not present.", "instance": "/token" }
5	Do not let users create a token when user added both mandatory custom headers but did not enter their values	Add 2 Request headers with the key/value pairs as seen below: clientid: none clientsecret: none Do not enter any values for these headers	POST	401	401 Unauthorized: " {"error":"invalid_client","error_description":"Invalid client or Invalid client credentials"}"
6	Do not let users create a token when custom headers are added but wrong values are entered	Add 2 Request headers with the key/value pairs as seen below: clientid:	POST	401	401 Unauthorized: " {"error":"invalid_client","error_description":"Invalid client or Invalid client credentials"}"

		clientsecret: none Enter dummy values			

INGESTING DATA API

Ingesting ELR(HL7) Documents: Armed with the token, users can effortlessly ingest HL7 documents using the Data Ingestion system. This process enables users to securely transmit documents stored in HL7 format from their local machines, ensuring data integrity and reliability in the NBS system.

Pre-Requisite: A significant prerequisite for utilizing this endpoint is to have the token ready.

HL7 Data format: Only HL7 ORU R01 2.3.1 and HL7 ORU R01 2.5.1 versions are in scope for this release. Other formats of HL7 are out of scope. When we try to post an HL7 ORU R01 2.3.1 message, the Data ingestion service with automatically convert it to 2.5.1 and post it.

Endpoint: https://dataingestion.dts1.nbspreview.com/api/elrs

HTTP Method: POST

Request Headers: Within the header section of the API, please add the following 3 key/value pairs. These are mandatory for the API to work.

- msgType: HL7
- clientid:
- clientsecret:

Authorization Type: Bearer Token

Token: Enter the token generated via Token API.

Body: Select Raw and text options and enter a valid ELR(HL7) data within the body section of the API.

Note: One of the validations within the Data Ingestion service is the Duplicate check. If we post the exact same HL7 message more than once, the validation will fail in the Data Ingestion

system.

Scenario	Request Headers	Authorization Type	Response Code	Response message	Database Validation
Post a valid formatted HL7 message with correct header keys and authorization selected	msgType: HL7 clientid: clientsecret:	Bearer Token and enter the token generated	200	UUID is displayed. Ex: F8CFC9D0-93C5-479D-963A-00DEBEBD1771	Select * from elr_raw where id= ‘UUID’ Select * from elr_validated where raw_message_id=’UUID’

Do not let users post a valid formatted HL7 message when incorrect token	msgType: HL7
	clientid:
	clientsecret:
		Bearer Token and enter dummy token	401	{
				"statusCode": 401,
				"message": "Unauthorized",
				"details": "Provided token isn't active"
				}
					None

Do not let users post a valid formatted HL7 message with expired token	msgType: HL7
	clientid:
	clientsecret:
		Bearer Token and enter an expired token. Usually, token expires after 30 min.	401	{
				"statusCode": 401,
				"message": "Unauthorized",
				"details": "Provided token isn't active"
				}
					None

Do not let users post a valid formatted HL7 message with incorrect formatted token	msgType: HL7 msgType: HL7 clientid: clientsecret:	Bearer Token and enter incorrect format token. Ex: - token = acbdef123	401	{ "statusCode": 401, "message": "Unauthorized", "details": "An error occurred while attempting to decode the Jwt: Invalid	None

				JWT serialization: Missing dot delimiter(s)" }	
Post an invalid formatted HL7 message with correct header keys and authorization selected. Enther the hl7 message within the body section as “testmessage”	msgType: HL7 clientid: clientsecret:	Bearer Token and enter the token generated	200	UUID is displayed. Ex: F8CFC9D0-93C5-479D-963A-00DEBEBD1771	Select * from elr_raw where id= ‘UUID’ Select * from elr_validated where raw_message_id=’UUID’ Select * from elr_dlt where raw_message_id =’UUID’. It fails validation and hence goes to elr_dlt table. This table has detail logs.

					No XML record will be generated.
Verify if the validation fails within the DI system when user posts an HL7 ORU R01 message with version other than 2.3.1 or 2.5.1	msgType: HL7 clientid: clientsecret:	Bearer Token and enter the token generated	200	UUID is displayed. Ex: F8CFC9D0-93C5-479D-963A-00DEBEBD1771	Select * from elr_raw where id= ‘UUID’ Select * from elr_validated where raw_message_id=’UUID’ Select * from elr_dlt where raw_message_id =’UUID’. It fails validation and hence goes to elr_dlt table. This table has detail logs. No XML record will be generated.

Verify if the validation fails within the DI system when user posts an HL7 message with data type other than ORU R01	msgType: HL7 clientid: clientsecret:	Bearer Token and enter the token generated	200	UUID is displayed. Ex: F8CFC9D0-93C5-479D-963A-00DEBEBD1771	Select * from elr_raw where id= ‘UUID’ Select * from elr_validated where raw_message_id=’UUID’ Select * from elr_dlt where raw_message_id =’UUID’. It fails validation and hence goes to elr_dlt table. This table has detail logs. No XML record will be generated

Verify if the validation fails within the DI system when user posts the exact same HL7 message more than once	msgType: HL7 msgType: HL7 clientid: clientsecret:	Bearer Token and enter the token generated	200	UUID is displayed. Ex: F8CFC9D0-93C5-479D-963A-00DEBEBD1771	Select * from elr_raw where id= ‘UUID’ Select * from elr_validated where raw_message_id=’UUID’ Select * from elr_dlt where raw_message_id =’UUID’. It fails validation and hence goes to elr_dlt table. This table has detail logs. No XML record will be generated.

CHECK BASIC STATUS OF INGESTED DATA API

Verify the outcome of the ingested data: This endpoint lets you determine the status of ingested data. By using the UUID provided in the response from the Ingested Data API in the request path, the API will return the status and pertinent details of the ingested data, indicating if it has been processed successfully, is still pending, or has experienced an error.

Endpoint: https://dataingestion.dts1.nbspreview.com/api/elrs/status/{ID}

Pre-Requisite:

- The HL7 data has been posted via Ingested Data API.
- The classic NBS wildfly scheduler and batch job are up and running.

HTTP Method: GET

Request Headers: Within the header section of the API, please add the following 2 key/value pairs. These are mandatory for the API to work.

- clientid:
- clientsecret:

Authorization Type: Bearer Token

Token: Enter the token generated as part of Token API

SNO	Scenario	Authorization Type	Headers	Response Code	Response message	Comments
1	Once user posts a valid formatted HL7 message, check the status of the message immediately	Bearer Token and enter the token	clientid: clientsecret:	200	{"id":"{UUID}","status": “Queued"}	Once user posts HL7 message, the data persists in the data ingestion Database. It also creates

						an XML record within the NBS interface table. By default, the status of this xml record goes to queued. It stays here for 2 minutes or so until the wildfly scheduler picks it up and batch job processes it.
2	Once user posts valid HL7 message, check the status of the message after 2	Bearer Token and enter the token	clientid: clientsecret:	200	{"id":"{UUID}","status”: “Success"}	The wildfly scheduler picks up the queued records and process

	minutes or so					them, If the generated xml record passes the validation, it converts it to success status. Eventually, we can see this within the Classic NBS UI portal for that particular patient.
3	Once user posts an invalid HL7 message that might fail XML validation, check the status of the message after 2	Bearer Token and enter the token	clientid: clientsecret: clientid: clientsecret:	200	{"id":"{UUID}","status”: “Failure"}	The wildfly scheduler picks up the queued records and process them. In this

	minutes or so					case, we post a HL7 message that generates an XML record in NBS DB but fails the validation.
4	Once user posts an invalid HL7 message that fails validation within the data ingestion system, check the status of the message after few seconds	Bearer Token and enter the token	clientid: clientsecret:	200	{"error_message":"Provided UUID is not present in the database. Either provided an invalid UUID or the injected message failed validation.","id":"4c658e34-bd8c-4dd5-8564-	Ex: Missing some mandatory segments in HL7 or incorrect formatted HL7 message. In this case, it will not even generate an XML record in NBS system.

					69b99981ea19"}	The validation fails within the data ingestion system.
5	Check the status of the message by entering a dummy ID within the API	Bearer Token and enter the token	clientid: clientsecret:	400	{"error_message":"Provided UUID is not present in the database. Either provided an invalid UUID or the injected message failed validation.","id":"a7887e4b-d693-48d3-8e3b-2db229709c99"}	Enter some dummy ID within the API and make sure that correct error message is displayed

6	Check the status of the message by entering an incorrect format ID within the API	Bearer Token and enter the token	clientid: clientsecret:	400	Invalid 'UUID' parameter provided.	Enter some dummy ID like “123” within the API and make sure that correct error message is displayed
7	Check the status of the message by appending some random text to the ID section of the API	Bearer Token and enter the token	clientid: clientsecret:	400	Invalid 'UUID' parameter provided.	Append some random text at the end of the ID part within the API. Ex: https://dataingestion.data team-cdc-nbs.eqsandbox.com/report-status/F8CFC9

						D0-93C5-479D-963A-00DEBEBD1771dummytextaddedaftertheIDsection

CHECK DETAIL STATUS OF INGESTED DATA API

Verify the outcome of the ingested data: This endpoint lets you determine the detailed status of ingested data. By using the UUID provided in the response from the Ingested Data API in the request path, the API will return the status and pertinent details of the ingested data, ELR payload, validated payload by DI service, status, Id and XML payload within NBS interface table, detailed logs like record Type, patient ID, Lab ID, comments, error logs etc.

Endpoint: https://dataingestion.dts1.nbspreview.com/api/elrs/status-details/{ID}

Pre-Requisite:

- The HL7 data has been posted via Ingested Data API.
- The classic NBS wildfly scheduler and batch job are up and running.

HTTP Method: GET

Request Headers: Within the header section of the API, please add the following 2 key/value pairs. These are mandatory for the API to work.

- clientid:
- clientsecret:

Authorization Type: Bearer Token

Token: Enter the token generated as part of Token API

CHECK SPECIFIC ERROR MESSAGE API

Checking the error message: Data Ingestion system is designed to ensure that every message sent through it is tracked, ensuring data integrity and system reliability. When a message is successfully posted, it gets stored in the main database. However, in the event of an error, the message is stored in a separate error table to facilitate troubleshooting and system monitoring. To access these error messages, a dedicated Error Messages API has been provided.

Endpoint: https://dataingestion.dts1.nbspreview.com/api/elrs/error-messages/{ID}

Pre-Requisite: An invalid HL7 message has been posted via API as reviewed in previous steps.

HTTP Method: GET

Request Headers: Within the header section of the API, please add the following 2 key/value pairs. These are mandatory for the API to work.

- clientid:
- clientsecret:

Authorization Type: Bearer Token

Token: Enter the token generated earlier.

Scenario: Once user posts the HL7 message that would fail validation within the data ingestion system, check the error message on why it failed. Below are some of the sample examples.

Scenario	Authorization Type	Headers	Response Code	Response message	Comments
Post the same HL7 message more than once, the duplication check fails the validation and verify the error	Bearer Token and enter the token	clientid: clientsecret:	200	"errorMessageId": "5B62EE5E-6EEA-415D-8661-70EEAEA1F0C0", "errorMessag	Within the Data ingestion system, we have additional logic implemented to check for duplicate

message via API				eSource": "elr_raw", "message": "{displays the posted HL7 message}", "errorStackTrace": “{displays the exact code where the issue is}”, "errorStackTraceShort": "DuplicateHL7FileFoundException: HL7 document already exists in the database. Please check elr_raw table for the failed document. Record Id: DB2A07BD-6365-4A3D-8137-724FBCA8E77E", "dltOccurrence": 1,	messages. When end user posts the same HL7 message more than once, it flags it as a duplicate for the second time and then fails the validation. The error response gives us detailed information on what the error is and what caused it.

				"dltStatus": "ERROR", "createdOn": "2023-10-18T16:57:25.180+00:00", "updatedOn": null, "createdBy": "elr_raw_dlt", "updatedBy": "elr_raw_dlt" }	
Post an HL7 message with incorrect HL7 message version like 2.2 and verify the error message via API	Bearer Token and enter the token	clientid: clientsecret:	200	"errorMessageId": "5B62EE5E-6EEA-415D-8661-70EEAEA1F0C0", "errorMessageSource": "elr_raw", "message": "{displays the posted HL7 message}",	The error response gives us detailed information on what the error is and what caused it.

				"errorStackTrace": “{displays the exact code where the issue is}”, "errorStackTraceShort": "DiHL7Exception: Unsupported HL7 Version, please only specify either 2.3.1 or 2.5.1. Provided version is: \t\t2.2" ", "dltOccurrence": 1, "dltStatus": "ERROR", "createdOn": "2023-10-18T16:57:25.180+00:00", "updatedOn": null,	

				"createdBy": "elr_raw_dlt", "updatedBy": "elr_raw_dlt" }	

Check the status with dummy UUID and validate the error message	Bearer Token and enter the token	clientid:
		clientsecret:
			500	{
				
				"statusCode": 500,
				"message": "An internal server error occurred.",
				"details": "The Record does not exist in elr_dlt. Please try with a different ID"
				}	

Check the status with incorrect formatted UUID and validate the error message.	Bearer Token and enter the token	clientid: clientsecret:	500	{ "statusCode": 500, "message": "An internal server error occurred.",	

Ex: - UUID = abcdef				"details": "abcdef is an Invalid Unique Id, please provide the correct id." }	
User cannot view the error logs when authorization is missing	No Auth		401	{ "statusCode": 401, "message": "Unauthorized", "details": "Full authentication is required to access this resource" }	

RE-INJECTING DATA API

Checking the error message: This endpoint allows user to re-inject the HL7 messages that failed validation within the Data Ingestion system again into the DI system again.

Endpoint: https://dataingestion.dts1.nbspreview.com/api/elrs/{dlt-id}

Pre-Requisite: HL7 validation within DI system passed, however during the transformation phase that in the process of generating the XML record, it failed.

HTTP Method: POST

Request Headers: Within the header section of the API, please add the following 2 key/value pairs. These are mandatory for the API to work.

- clientid:
- clientsecret:

Authorization Type: Bearer Token

Token: Enter the token generated earlier

SNO	Scenario	Auth Type	Response Code	Headers	Response message	Comments
1	Determine if the HL7 failed validation within the Data ingestion system itself using “CHECK STATUS OF INGESTED DATA API”	Bearer Token and enter the token	400	clientid: clientsecret:	{"error_message":"Provided UUID is not present in the database. Either provided an invalid UUID or the injected message failed validation.","id":"4c658e34-	This is the first step to determine that the posted HL7 message has failed within the Data ingestion system itself. It could fail due to 2 reasons; one is

					bd8c-4dd5-8564-69b99981ea19"}	the data validation and other is it passed the validation but failed during the transformation phase of generating an XML record. The next step would determine the phase where it failed.
2	Determine if the HL7 message failed validation during the transformation phase within the Data	Bearer Token and enter the token	200	clientid: clientsecret:	{ "errorMessageId": "935026fe-c848-41ee-	Within the response, if “errorMessageSource": "xml_prep", then it means

	ingestion system itself using “CHECK THE ERROR MESSAGES API”.				b47d-98597b62a593", "errorMessageSource": "xml_prep", "message": "{HL7 message is displayed here}", "errorStackTrace": "{detailed code is displayed where error occurred}", "errorStackTraceShort": "time.format.DateTimePa	it passed the validation with the DI system passed but failed during the transformation phase that is in the process of generation XML record. By default, the DltOccurrence is 1 **Make sure you copy the HL7 messag**

					rseException: Text '19000000000000' could not be parsed: Invalid value for MonthOfYear (valid values 1 - 12): 0", "dltOccurrence": 1, "dltStatus": "ERROR", "createdOn": "2023-11-08T11:21:02.170+00:00", "update	**e from the message section within the response. We need this for re-injection purpose. It needs to be in this format. If we use regular format of HL7 message, reinjection will fail.** **Note**: If the response was instead, “errorMe

					dOn": **null**, "createdBy": "xml_prep_dlt", "updatedBy": "xml_prep_dlt" }	ssageSource": "elr_raw", it means the validation itself failed within the DI system.
3	Re-inject the same HL7 message with no corrections made to it and then after few seconds check the Error Messages API	Bearer Token and enter the token	200	clientid: clientsecret:	{ "errorMessageId": "935026fe-c848-41ee-b47d-98597b62a593", "errorMessageSource": "xml_prep", "messag	If the ingestion fails again, this time, for the same record, the response will be same just that the DltOccurrence is updated to 2.

						e": "{HL7 message is displayed here}", "errorStackTrace": "{detailed code is displayed where error occurred}", "errorStackTraceShort": "time.format.DateTimeParseException: Text '19000000000000' could not be parsed: Invalid value for MonthOfYear	Not all failed HL7 messages can be injected back successfully. If the failed message is due to bad formatting issues like having version 2.2 etc., even if we inject back the message, it stays as failed within the elr_dlt table with status as

					(valid values 1 - 12): 0", "dltOccurrence": 2, "dltStatus": "ERROR", "createdOn": "2023-11-08T11:21:02.170+00:00", "updatedOn": **null**, "createdBy": "xml_prep_dlt", "updatedBy": "xml_prep_dlt" }	ERROR. However, the dltOccurence count changes from 1 to 2.

4	Re-inject the same HL7 message after making corrections to HL7 message. After few seconds check the Error Messages API again.	Bearer Token and enter the token	200		{ "errorMessageId": "935026fe-c848-41ee-b47d-98597b62a593", "errorMessageSource": "xml_prep", "message": "{HL7 message is displayed here}", "errorStackTrace": "{detailed code is displaye	The dltstatus is updated to REINJECTED within the response. If you see this that means the HL7 data is re-ingested successfully. Also, updated On field displays the updated timestamp. If the dltstatus stays as REINJECTED, it creates

						d where error occurred}", "errorStackTraceShort": "time.format.DateTimeParseException: Text '19000000000000' could not be parsed: Invalid value for MonthOfYear (valid values 1 - 12): 0", "dltOccurrence": 2, "dltStatus": "REINJECTED",	a XML record within the NBS_Interface table within the MSGOUTE Database.

					"createdOn": "2023-11-08T11:21:02.170+00:00", "updatedOn": 2023-11-08T11:26:02.170+00:00, "createdBy": "xml_prep_dlt", "updatedBy": "xml_prep_dlt" }	
5	Once HL7 message is re-injected successfully, check the status of the XML record	Bearer Token and enter the token	200	clientid: clientsecret:	{"id":"{UUID}", "status”: “Success"}	The wildfly scheduler picks up the queued records

	within the NBS_Interface table					and process them, If the generated xml record passes the validation, it converts it to success status. Eventually, we can see this within the Classic NBS UI portal for that patient.
6	Once the dltstatus is REINJECTED, users cannot reinject or make changes to that record	Bearer Token and enter the token	200	clientid: clientsecret:	{ "statusCode": 500, "message": "An internal server	Once the record is in REINJECTED state, users cannot make

					error occurred.", "details" : "Selected record is in REINJECTED state. Please either wait for the ERROR state to occur or select a different record." }	changes to it.

GET ALL ERROR MESSAGES

Checking the error message: This endpoint allows user to view all the HL7 messages that failed validation within the Data Ingestion

Endpoint:https://dataingestion.dts1.nbspreview.com/api/elrs/error-messages

HTTP Method: GET

Request Headers: Within the header section of the API, please add the following 2 key/value pairs. These are mandatory for the API to work.

- clientid:
- clientsecret:

Authorization Type: Bearer Token

Token: Enter the token generated earlier.

VALIDATE HL7 MESSAGE API

Checking the error message: This endpoint is to verify whether a payload is a valid HL7 or not within the Data Ingestion

Endpoint: https://dataingestion.dts1.nbspreview.com/api/elrs/validate

HTTP Method: POST

Request Headers: Within the header section of the API, please add the following 2 key/value pairs. These are mandatory for the API to work.

- clientid:
- clientsecret:

Authorization Type: Bearer Token

Token: Enter the token generated earlier.

DATA VALIDATION WITHIN DI SYSTEM

SNO	Database Name	Table Name	Query	Validation	Potential errors
1	NBS_DataIngest	elr_raw	SELECT * FROM [NBS_DataIngest].[dbo].[elr_raw] where id = {UUID captured from DI API response}	We can validate the posted hl7 message under payload column. By Default, all the	If API is up and running and tables with right schema exist in the NBS_Data ingest

				messages go to this table irrespective of it being a good or bad formatted message.	database, no issues should be observed in this table.
2	NBS_DataIngest	elr_validated	SELECT * FROM [NBS_DataIngest].[dbo].[elr_validated] where raw_message_id ={ UUID captured from API response}	The message should be under Validated_message column.	If there are any issues with the HL7 message like in if it's missing some info etc., validation would fail and you will not see any record here. Instead, it will be present within the elr_dlt table. Also, no

					xml record will be generated within the NBS_Interface table
4	NBS_DataIngest	elr_record_status_id	SELECT * FROM [NBS_DataIngest].[dbo].[elr_record_status_id] where raw_message_id ={ UUID captured from API response}	Once the validation passes, it creates an xml record in the NBS_interface table. For the ingested record, this gives the link between classic NBS_interface table and the new DI system based NBS_ingestion table. Capture the nbs_interface_id from this table	None

5	NBS_DataIngest	elr_dlt	SELECT * FROM [NBS_DataIngest].[dbo].[elr_dlt] where error_message_id ={{Id captured from API response}}	If the inserted record does not even exist in elr_validated table due to bad formatting issues like missing mandatory elements etc. it can be found here along with the reason on why it failed validation.	The only time, you see a record in elr_raw table and not in elr_validated and elr_dlt table is when kafka is running but has incorrect configurations like incorrect broker numbers etc. Minimum of 2 brokers are required.

6	NBS_DataIngest	elr_dlt	SELECT * FROM [NBS_DataIngest].[dbo].[elr_dlt] where error_message_id ={Id captured from elr_validated table}	Sometimes, HL7 validation within DI system will be passed, however during the transformation phase that in the process of generating the XML record, it fails. It can be found here along with the reason on why it failed validation	None
7	NBS_MSGOUTE	nbs_interface	Select * from nbs_interface where id= {nbs_interface_id captured in step 4}	We can validate the status of the xml record here. The status by default goes to queued status. From there is either changed to	If the Wildfly scheduler is down, or not working, the status remains in queued status. If the validation of XML

				success or failure. Here, we are checking the status if the message in NBS system which is the last step in process. If the status is success, that means the data we posted is successfully transmitted.	fails, status goes to failure.