

Electronic Receipt and Invoice System,
and Electronic Sales Reporting system (PC)

EIS e-invoice API

Development Guide for PILOT

Ver. 2.0



Revision history

Ver	Date	Reason ¹	Detail ²	Writer	Approver
1.0	2021.10.20	New		Lee Yongmin	Jung Kwangok
1.1	2021.11.10	Modification	Format modification, Section 9 removed	Lee Yongmin	Jung Kwangok
1.2	2021.12.01	Modification	- API request parameter, Error codes added - 5.2 Example changed - 5.3 Sample JSON changed	Lee Yongmin	Jung Kwangok
2.0	2022.02.22	Modification	- JSON format changed - Sample JSON/Code changed	Lee Yongmin	Jung Kwangok

¹ Reason: Select from initial preparation/approval/ addition/modification/ deletion for the changes made

² Detail: Write what changes have made in detail (Describe the chapters, sections of the change)

INDEX

1. Overview.....	1
2. Type of EIS e-invoice API.....	1
3. Interlink Process	2
3.1 API Call Sequence.....	2
3.1.1 CASE 01: POLLING.....	2
3.1.2 CASE 02: PUSH.....	2
3.2 Authentication API process	4
3.3 Invoice Issuance API Process	5
3.4 Inquiry Result API Process.....	6
3.5 Invoice Result Callback process.....	7
4. e-invoice Format.....	8
4.1 CAS Issued e-invoice JSON	8
4.2 CRM/POS Issued e-invoice JSON	11
5. Separate issuance of mixed invoices.....	14
5.1 Summary	14
5.2 Examples	14
5.3 Sample JSON	17
6. Security.....	20
6.1 Summary	20
6.2 7steps of API security safeguard	20
6.3 invoice security	20
6.3.1 Main standards and Algorithms.....	21
6.4 JWS digital signature.....	21
6.5 Transmission Data Encryption	22
6.5.1 Overview.....	22
6.5.2 Encryption algorithm.....	22
6.5.3 Target data to be encrypted.....	22
6.5.4 Asymmetric encryption sample code.....	23
6.5.5 Symmetric encryption/decryption sample code.....	24

6.6	Transmission security	26
6.6.1	HMAC Signature.....	26
6.6.2	HMAC Signature Rule.....	26
6.6.3	Sample Signature, Request Header transmitted by the taxpayer	27
6.6.4	HMAC Signature generation + API request sample	27
7.	<i>API Specification.....</i>	28
7.1	Authentication.....	28
7.1.1	Path.....	28
7.1.2	Request Parameters.....	28
7.1.3	Response Message.....	30
7.2	Invoice Issuance.....	32
7.2.1	Path.....	32
7.2.2	Request Parameters.....	32
7.2.3	Response Message.....	34
7.3	Inquiry Result	36
7.3.1	Path.....	36
7.3.2	Request Parameters.....	36
7.3.3	Response Message.....	37
7.4	Invoice Result Callback.....	41
7.4.1	Path.....	41
7.4.2	Request Parameters.....	41
7.4.3	Response Message.....	43
8.	<i>API interface example</i>	45
8.1	Authentication API Call	45
8.1.1	Temporary Secret Key Generation.....	45
8.1.2	Application authentication information asymmetric key encryption	45
8.1.3	Request Authentication API.....	46
8.2	Sending invoices by the invoice issuance API.....	47
8.2.1	Generate issued invoice with e-invoice JWS.....	47
8.2.2	e-invoice JSON digital signature	52
8.2.3	Encrypt generated JWS using Session Secret Key.....	53
8.2.4	Request invoice issuance API.....	53
8.3	Checking the registration result through "Invoice Issuance Result Check API"	56
8.3.1	Request Inquiry Result API.....	56

1. Overview

EIS e-invoice API is an API that can transmit the invoice issued from the taxpayers' system (CAS/CRM/POS) to the BIR.

For those taxpayers who will interlink his/her system (CAS/CRM/POS) with the EIS using e-invoice API, the taxpayer does not need to issue an invoice from the EIS web portal. E-invoice API will be provided in RESTful Web Service format to be interlinked with various system and development languages easily.

2. Type of EIS e-invoice API

EIS e-invoice API provides 4 types of API to receive, acknowledge the results of verification and registration (enrollment) of invoices received.

API Name	API Description
Authentication	API that receives Authentication Token for externally linked API calls For invoice Issuance, Inquiry Result API call, the API should be called to get an authentication token. The issued token is valid for 6 hours, therefore, there is no need to call this API again unless it expires.
Invoice Issuance	API to transmit/send invoices issued by taxpayers' system (CAS/CRM/POS) to the EIS system for single up to 100 cases. Return acknowledgment message (success/failure) immediately in JSON data for the call.
Inquiry Result	The API that returns the processing result document for the invoice issuance with Submit ID The response of the API is When the requested Submit ID is still on the processing of registration and verification: return the response 'processing'. When the requested Submit ID is completed with the registration and verification: return the JSON data.
Invoice Result Callback (Optional)	The API to deliver results to the taxpayer for the Invoice issuance process per-Submit ID. It is a callback REST API that is configured in the Taxpayer server if the taxpayer wants to receive the JSON data of the processing result from the EIS as soon as the invoice registration and verification processing is completed in the Taxpayer system.

In the case of Invoice Result Callback API, it is not API provided by the EIS e-invoice API but build and provided from the taxpayers' system.

In general, the taxpayer needs to send an invoice using the Invoice Issuance API, check the processing results of the periodically sent invoices, check the success/failure of registration of each invoice using the Inquiry Result API, and apply the results in the system.

If the taxpayer's system needs to receive the processing results as soon as the verification and registration of invoices in the EIS are completed without checking the results periodically, the taxpayer must build and provide the Invoice Result Callback API in the system.

3. Interlink Process

3.1 API Call Sequence

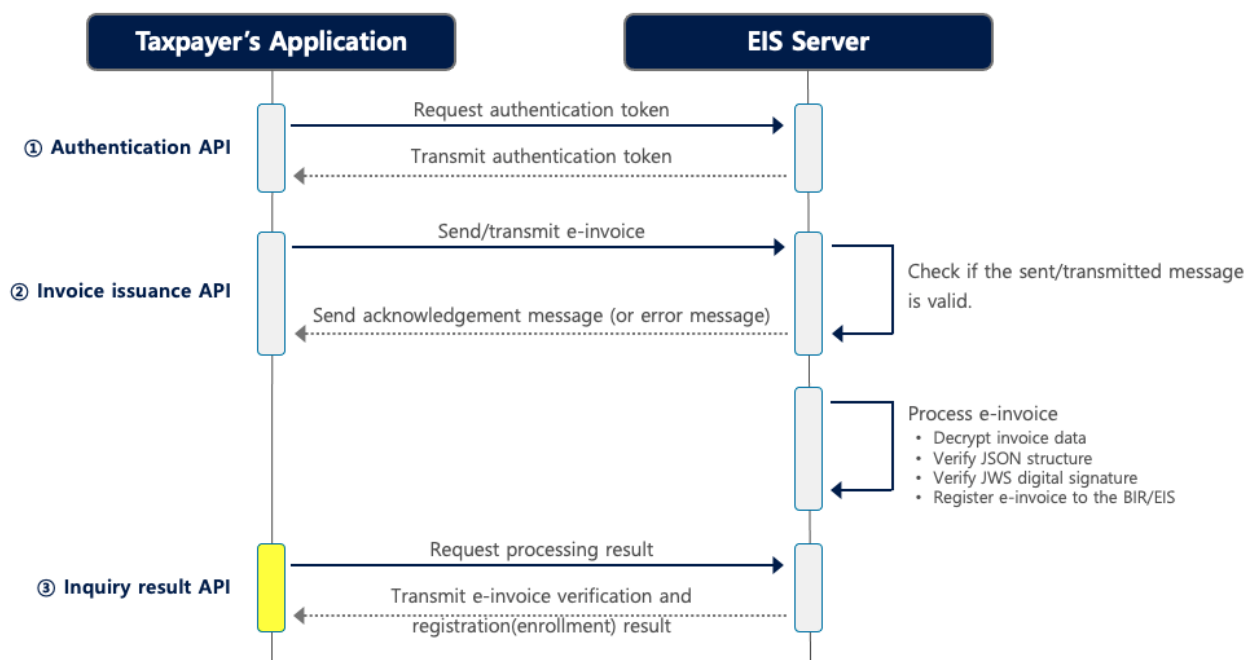
In order to send invoices issued from the taxpayer's system (CAS/CRM/POS) to the EIS, the first Authentication token should be issued. After the Authentication token is issued, Invoice issuance, Inquiry result..etc API could be called.

The authentication token can be issued through Authentication API, and when calling the Authentication API, the credentials corresponding to each user and registered application on the EIS Accreditation website must be transmitted.

The detailed usage of the Authentication API can be checked or found in the API guide.

3.1.1 CASE 01: POLLING

This process is a case of sending an invoice issued from the taxpayer's system to the EIS, and after a certain time, the taxpayer's system requests and confirms the processing result of the invoice. If the transmission is still being processed, the taxpayer's system should request the processing result by calling the Inquiry result API periodically until the processing is complete.

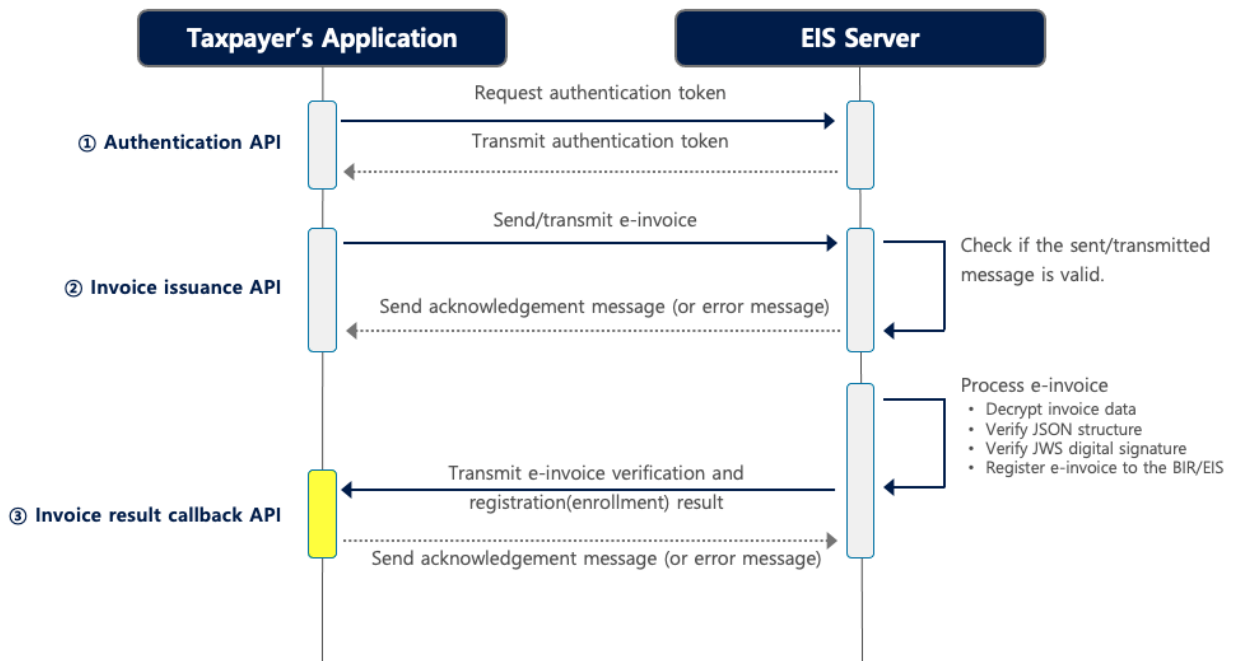


3.1.2 CASE 02: PUSH

This process is a case of sending the invoice issued by the taxpayer's system to the EIS, and then sending the processing result to the taxpayer's system as soon as the verification & registration processing of the invoice is completed.

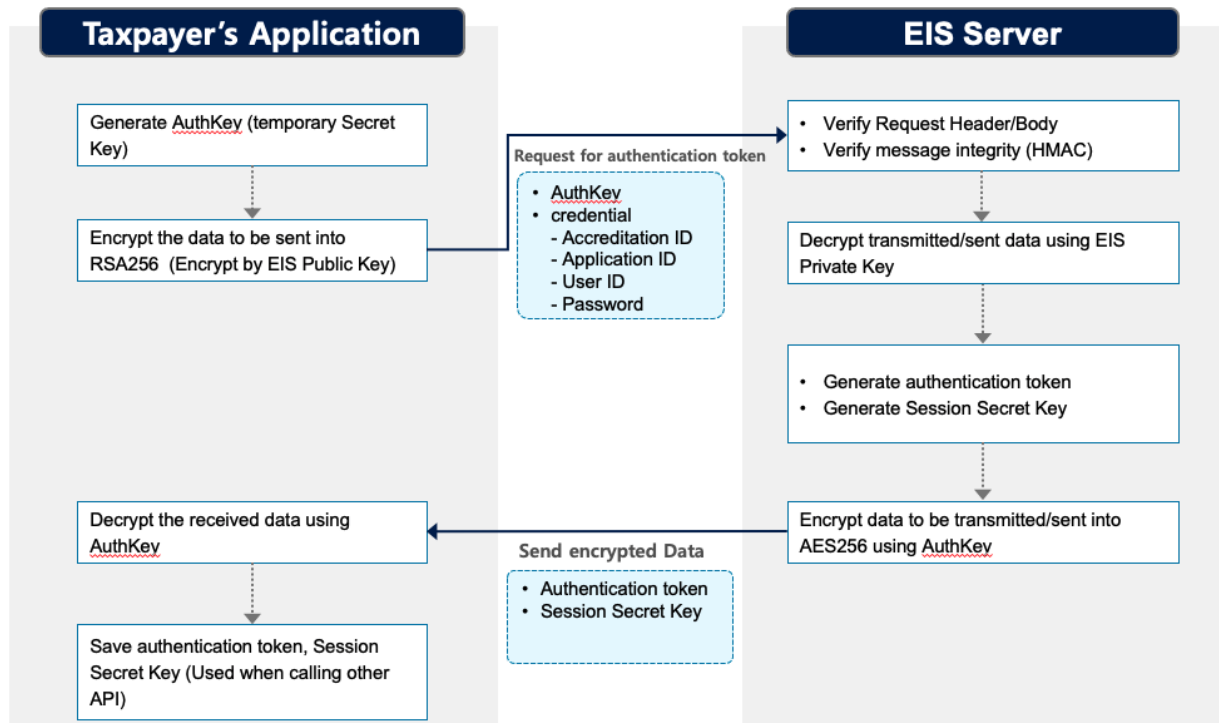
This has an advantage which taxpayer do not need to periodically request the results of processing, and will be able to receive results immediately after processing.

In order to utilize this process, a callback API to receive information/data from the taxpayer's system must be developed, and a Web API server must be configured and managed for the API to be serviced



3.2 Authentication API process

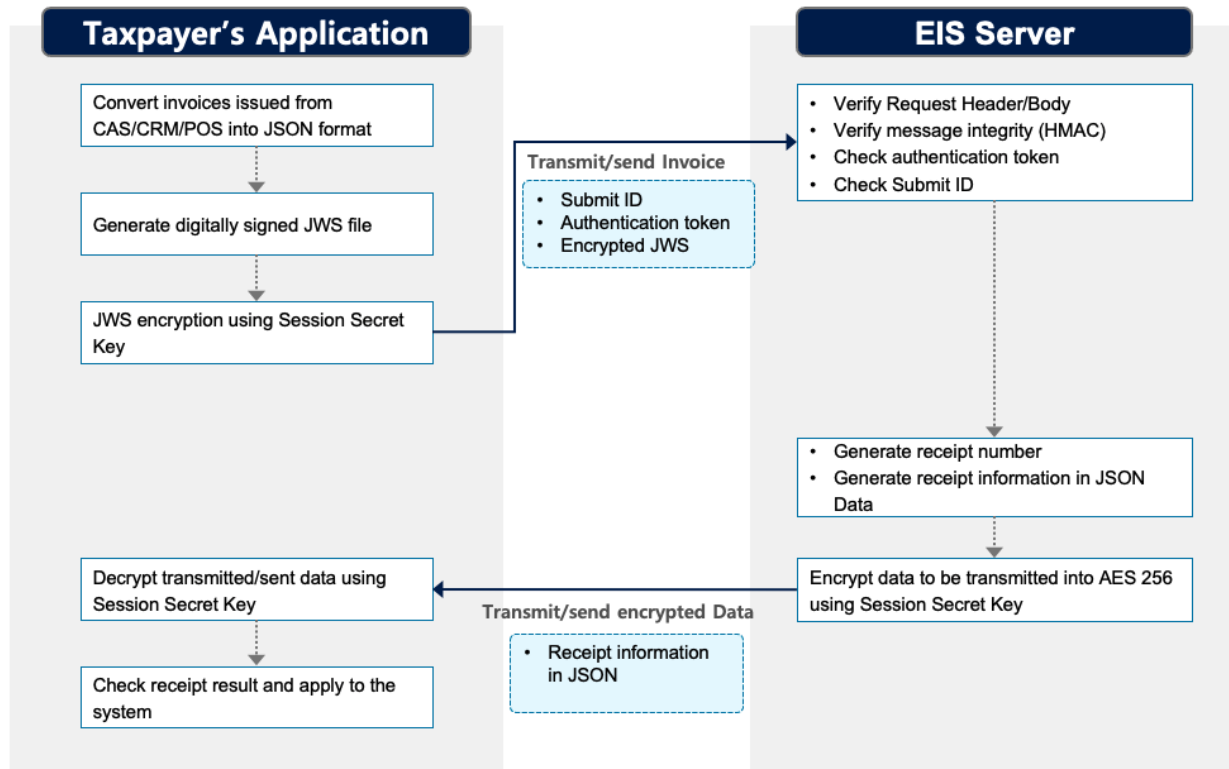
The Authentication API process to receive an authentication token for the security of API calls is as follows.



1. Generate temporary Secret Key
2. Encrypt data to transmit in RSA using the EIS Public Key registered in the EIS Accreditation
3. Submit temporary Secret Key and credentials (Accreditation ID, Application ID, User ID, Password)
4. Verify Request Header/Body, verify message integrity, verify credential
5. Issue Authentication token
6. Generate Session Secret Key
7. Encrypted with the temporary Secret Key that received the authentication token and the Session Secret Key
8. Transmit encrypted Data
9. Decrypt with temporary Secret Key
10. Save the Authentication token and use it when calling other APIs

3.3 Invoice Issuance API Process

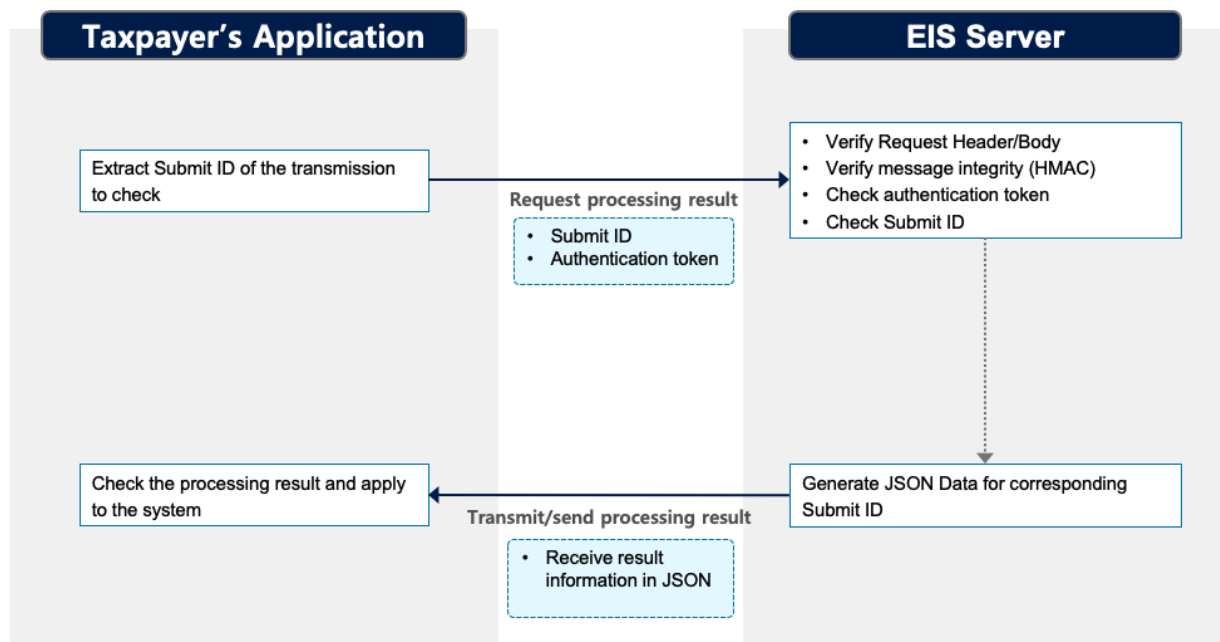
The invoice issuance API process for issuing invoice is as follows.



1. Convert invoices issued from CAS/CRM/POS into JSON format
2. Generate digitally signed JWS file
3. JWS encryption using Session Secret Key
4. Transmit issued Authentication token and encrypted data
5. Verify Request Header/Body, verify message integrity (HMAC), check authentication token, check Submit ID
6. Generate receipt number and generate receipt information in JSON Data
7. Encrypt data to be transmitted into AES 256 using Session Secret Key
8. Return encrypted receipt result data

3.4 Inquiry Result API Process

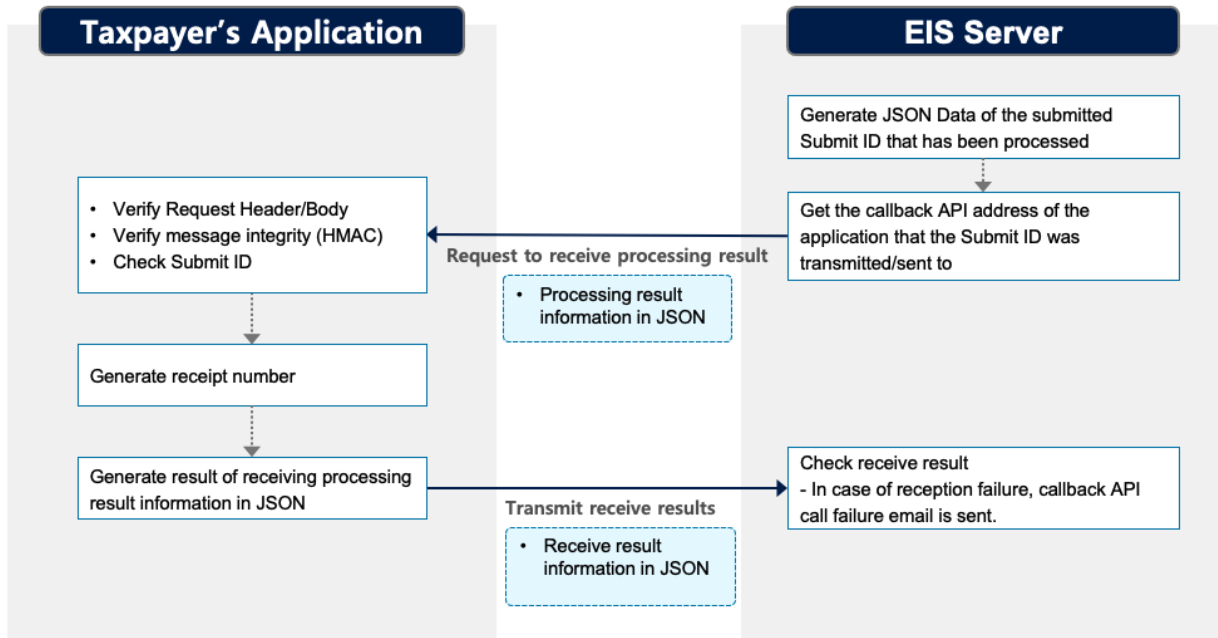
The inquiry result API process that returns the processing status of a sent invoice is as follows.



1. Transmit issued Authentication token and transmit Submit ID used when transmitting an invoice
2. Verify Request Header/Body, verify message integrity, Check Authentication token and Submit ID
3. Generate and return JSON Data of corresponding Submit ID

3.5 Invoice Result Callback process

The Invoice result callback API process notify the taxpayer of the result immediately after the verification & registration of the sent invoice is complete.



1. Generate JSON Data of processed Submit ID result
2. Transmit the result to Callback API endpoint of corresponding Application in JSON Data
3. Verify Request Header/Body, verify message integrity, check Submit ID
4. Generate receipt number and generate a result of receiving processing result information in JSON
5. Return receipt result (success/fail)
6. Check to receive a result. In case of reception failure, a callback API call failure email is sent.

4. e-invoice Format

E-invoice is using JSON format which is lighter than XML format for the convenient data exchange between different/various/heterogeneous systems/types.

4.1 CAS Issued e-invoice JSON

JSON format description on the invoice issued from CAS.

(Refer to attachment excel file – 02.8.2022_EIS e-invoice JSON File format v2.0.xlsx Sheet ‘1.e-invoice format-CAS_v2.0’)

No	Item Name		Definition	Field Name	Data Type	Length (maximum)	Occurs	Default	(Note1) If the data type is 'Number', the field has 2 decimal places and allows negative numbers including '-'. (Note2) If there is no value in the field whose data type is 'String', enter 'null' or leave the field blank.
	category 1	category 2							
1	SI/OR/SB/DM/CM Management Information		Company SI/OR/SB/DM/CM No.	CompInvoiceId	String	50	1..1 (Mandatory)		Control number of documents issued and maintained by the taxpayer system (CAS, invoicing system).
2			SI/OR/SB/DM/CM Issuance Date	IssueDtm	String	8	1..1 (Mandatory)		Documents : There are 5 document types, to be sent to EIS: (a) Sales of goods : 3 documents / Sales Invoice(SI), Debit Memo(DM), Credit Memo(CM) (b) Sales of Services : 2 documents / Service Billing(SB), Official Receipt(OR) - Service Billing is a 'Statement of Account' or a 'Billing Statement' issued under sale of service.
3	E-Invoice Basic information	e-invoice Unique ID	BIR e-invoice Unique ID	EisUniqueld	String	24	1..1 (Mandatory)		Date the document (SI/DM/CM/SB/OR) was issued by seller Format: YYYYMMDD A unique value that identifies the e-invoice in the EIS, generated by the taxpayer/SW system consisting of 24 digits, as shown below. Issuance Date(8 digits) + EIS-Certification ID(8 digits) + Control Values(8 characters) (a) Issuance Date(YYYYMMDD) Invoice issue date, reflected in the system-generated document (b) EIS-Certification ID EIS generated ID for taxpayers/Software (SW) providers to transmit sales data from their CAS, invoicing system, etc. to EIS, consisting of 8 digits as shown below: Source code(2 digits) + XXXXXX(6 digits) (c) Control Values(8 characters) Eight (8) characters which may be random combination of numeric, alpha (uppercase), or alpha-numeric.
4		Invoice Classification	Document Type	DocType	String	2	1..1 (Mandatory)		01: Sales Invoice (SI) 02: Debit Memo (DM) / Debit Note (DN) 03: Credit Memo (CM) / Credit Note (CN) 04: Service Billing (SB, Statement of Account / Billing Statement) 05: Official Receipt (OR)
5		Transaction Classification	TransClass	String	2	1..1 (Mandatory)			Classify whether transaction data is VATable, zero-rated, or exempt sales. For mixed transaction data in a single document, they must be transmitted individually per classification, as follows: 01: VATable 02: Zero-Rated 03: Exempt
6		Invoice Correction	Correction Yes or Not	CorrYN	String	1	1..1 (Mandatory)		The value is "Y" for corrected/adjusted/modified previously transmitted data; otherwise, the value is "N"
7		e-invoice correction code	CorrectionCd	String	2	0..1 * (Conditional)			Mandatory field if the value in "CorrYN" is "Y" The reasons for the modification/correction/adjustment of e-invoice are classified as follows. Single previously issued e-invoice 01: Error - In any mandatory field such as: a. Seller Information (TIN & Branch Code, Registered Name, VAT Type) b. Buyer Information (TIN & Branch Code, Registered Name) c. Details of Sales, Amount, etc. - Total Net Sales After Discount - VAT Amount - Issuance Date - Details of Sold Items d. Classification of Sales as Taxable, Exempt, or Zero-rated. 02: Duplication - Sales data transmitted more than once regardless of transaction period. 03: Addition/reduction - Addition/reduction of contracted volume or amount 04: Cancellation - For sales data previously transmitted but transaction was subsequently cancelled or rescinded by buyer/seller for failure to consummate. Full return of goods without replacement is considered cancellation. 05: Return - For original goods supplied and returned for replacement. Ex: Damaged, defective, wrong specifications, change item, or for any other reason 09: Others
8		E-Invoice Unique ID of the document to be corrected	PrevUniqueld	String	24	0..1 * (Conditional)			This is the 'e-invoice unique ID' of the previously transmitted sales data to EIS to be modified/corrected/adjusted. If multiple documents are issued in one adjustment document, 'e-invoice correction code' value is '02', enter the 'e-invoice unique ID' of one document among multiple documents in this field, and the 'e-invoice unique ID' of the remaining documents in 'Remarks1'.
9		Remarks1	Remarks1	Rmk1	String	500	0..1 (Optional)		Explanation for the correction made
	Seller Information		Seller Information	SellerInfo			1..1 (Mandatory)		In accordance or consistent with the BIR registration information
10		Seller TIN	Tin	String	9	1..1 (Mandatory)			9-digit TIN of seller; Only numbers must be entered, example, 123456789(○), 123-456-7(X)
11		Branch Code	BranchCd	String	5	1..1 (Mandatory)	00000		5-digit code that identifies a headquarters or branch office. Branch code uses 3 digits, add '00' to the first two digits; example, branch code is '101', it must be converted to '00101'.
12		Seller Type	Type	String	1	1..1 (Mandatory)			"VAT" or "Non-VAT" Registered Seller Type 0: VAT registered 1: Non-VAT registered
13		Registered Name	RegNm	String	200	1..1 (Mandatory)			In accordance or consistent with BIR registration information.
14		Business Name/Trade Name	BusinessNm	String	200	1..1 (Mandatory)			In accordance or consistent with BIR registration information. If not, enter the same as the 'Registered Name'.
15		Email address	Email	String	100	0..1 (Optional)			Company email address or personal email address of the authorized representative
16		Registered Address	RegAddr	String	300	1..1 (Mandatory)			In accordance or consistent with BIR registration information.
	Buyer Information		Buyer Information	BuyerInfo			1..1 (Mandatory)		In accordance or consistent with BIR registration information This is buyer/client information who purchase goods and services. However, for export transactions, buyer may not be registered with the BIR.
17		Buyer TIN	Tin	String	9	1..1 (Mandatory)			9-digit TIN of the buyer Buyer is not registered with BIR, enter the TIN as '000000000(9 digits)'.
18		Branch Code	BranchCd	String	5	1..1 (Mandatory)	00000		5-digit code that identifies a headquarters or branch office. Branch code uses 3 digits, add '00' to the first two digits. Example, branch code is '101', it must be converted to '00101'.
19		Registered Name	RegNm	String	200	1..1 (Mandatory)			In accordance or consistent with BIR registration information.
20		Business Name/Trade Name	BusinessNm	String	200	1..1 (Mandatory)			In accordance or consistent with BIR registration information. If there is no Business Name/Trade Name, enter the same as the Registered Name.
21		Email address	Email	String	100	0..1 (Optional)			Buyer company email address or the email address of the accounting representative within the company.
22		Registered Address	RegAddr	String	300	0..1 (Optional)			Business address of buyer registered with BIR or as provided by the buyer to the seller
23		Delivery Address	DevAddr	String	300	0..1 (Optional)			Address where goods/service will be delivered/rendered
24		Airway Bill Number	AirNum	String	50	0..1 (Optional)			Applicable only to exporters
25		Airway Bill Number Date	AirNumDt	String	8	0..1 (Optional)			Format: YYYYMMDD
26		Bill of Lading Number	LadNum	String	50	0..1 (Optional)			Applicable only to exporters
27		Bill of Lading Number Date	LadNumDt	String	8	0..1 (Optional)			Format: YYYYMMDD

Line Items Information	Line Items	Item List				1,1000 (Mandatory)	Up to 1,000 line items. The monetary unit of all amounts on e-Invoices is Philippine peso (PHP).
28		Item Name	Nm	String	100	1,1 (Mandatory)	Name/brand of product
29		Item Description/Service	Desc	String	100	0,1 (Optional)	Description of item sold. Example: Size, color, nature of service ex.-Rental, food service
30		Item Quantity	Qty	Number	12	1,1 (Mandatory)	Number of item/s, if the quantity is fractional, indicate numerical equivalent. eg. 1/2 = 0.50.
31		Item Unit of measure	Unit	String	50	0,1 (Optional)	pieces, box, pack, etc
32		Unit Cost	UnitCost	Number	30	1,1 (Mandatory)	Price per unit of measure, net of VAT
33		Item Sales Amount	SalesAmt	Number	30	1,1 (Mandatory)	Net of VAT if item is VATable. This format will be prescribed for purposes of transmission to EIS. Formula: $\text{Item Sales Amount} = \text{Item Qty} \times \text{Unit Cost}$
34		Regular Item Discount Amount	RegDiscontAmt	Number	30	1,1 (Mandatory)	Granted per item that can be deducted for both VAT and income tax purposes (outright discounts).
35		Special Item Discount Amount	SpeDiscontAmt	Number	30	1,1 (Mandatory)	Granted per item which can be deducted for income tax purposes only (ex. Conditional discounts)
36		Net of Item Sales	NetSales	Number	30	1,1 (Mandatory)	Formula: $\text{Net of Item Sales} = \text{Item Sales Amount} - (\text{Regular Discount Amount} + \text{Special Discount Amount})$
37	Item Sales Summary	Total of Net of Item Sales	TotNetItemSales	Number	30	1,1 (Mandatory)	For VATable sales, special discount amount is not considered. Total of all "Net of Item Sales," exclusive of VAT for items tagged as VATable.
	Sales Information	Discount Information	Discount			1,1 (Mandatory)	Discount granted based on the Total of Net of Item Sales in addition to the discount per line item.
38		Senior Citizen Discount Amount	ScAmt	Number	30	1,1 (Mandatory)	Applicable to transactions with Senior Citizen (SC)
39		PWD Discount Amount	PwdAmt	Number	30	1,1 (Mandatory)	Applicable to transactions with Person With Disability (PWD)
40		Regular Discount Amount	RegAmt	Number	30	1,1 (Mandatory)	Granted based on the Total Net Sales Amount that can be deducted for both VAT and income tax purposes (outright discounts)
41		Special Discount Amount	SpeAmt	Number	30	1,1 (Mandatory)	Granted based on the Total Net Sales Amount that can be deducted for income tax purposes only (ex. Conditional discounts)
42		Remarks2	Rmk2	String	500	0,1 (Optional)	Brief explanation of discount, if needed
43		Other taxable revenue	OtherTaxRev	Number	30	1,1 (Mandatory)	Other taxable revenues not part of the primary revenue-generating items (e.g. miscellaneous income, taxable service charge).
44		Total Net Sales After Discounts	TotNetSalesAtDisct	Number	30	1,1 (Mandatory)	Total of Net of Item Sales + Other Taxable Revenue - Discounts (SC/PWD/Regular/Special)
45		VAT Amount	VATAmt	Number	30	1,1 (Mandatory)	VAT Amount reflected in the document
46		Withholding Tax-Income Tax	WithholdIncome	Number	30	1,1 (Mandatory)	Amount withheld by buyer for purposes of income tax.
47		Withholding Tax-Business VAT	WithholdBusVAT	Number	30	1,1 (Mandatory)	Amount withheld by buyer for purposes of VAT.
48		Withholding Tax-Business Percentage	WithholdBusPT	Number	30	1,1 (Mandatory)	Amount withheld by buyer on transactions subject to percentage tax.
49		Other Non-taxable charges	OtherNonTaxCharge	Number	30	1,1 (Mandatory)	Any non-taxable charges of seller to buyer (e.g. local tax, documentary stamp tax, non-taxable service charge, etc.)
50		Net Amount Payable	NetAmtPay	Number	30	1,1 (Mandatory)	Amount to be paid by the buyer.
	Foreign Currency Information	Foreign Currency	ForCur			0,1 (Optional)	Lawful money issued by the government of any country other than the Philippine Peso (PHP).
51		Currency	Currency	String	3	1,1 (Mandatory)	Currency used in the transaction using the 3 letter codes (e.g.: JPY, USD, SGD).
52		Conversion Rate	ConvRate	Number	30	1,1 (Mandatory)	Forex conversion rate as shown in the document
53		Currency Amount	ForexAmt	Number	30	1,1 (Mandatory)	Value in foreign currency as indicated in the document
54	PTU Information	PTU Number/Acknowledgment Certificate Control Number	PluNum	String	50	1,1 (Mandatory)	Number issued by the BIR allowing/registering taxpayer's use of Computerized Accounting System (CAS)

Example of JSON)

```
{
  "CompInvoiceId": "0000000205",
  "IssueDtm": "20220217",
  "EisUniqueId": "2022021720146IT400000205",
  "DocType": "01",
  "TransClass": "01",
  "CorrYN": "N",
  "CorrectionCd": "",
  "PrevUniqueId": "",
  "Rmk1": "test e-invoice : 2022021720146IT400000205",
  "SellerInfo": {
    "Tin": "999999999",
    "BranchCd": "00000",
    "Type": "0",
    "RegNm": "DOUZONE",
    "BusinessNm": "DOUZONE",
    "Email": "tester@douzone.com",
    "RegAddr": "130, Beodeul 1-gil, Namsan-myeon, Chuncheon-si, Gangwon-do, Republic of Korea"
  },
  "BuyerInfo": {
    "Tin": "225125979",
    "BranchCd": "00000",
    "RegNm": "DONG-A PHARMA PHILS. INC.",
    "BusinessNm": "DONG-A PHARMA PHILS. INC.",
    "Email": "donga_admin@donga.com",
    "RegAddr": "UNIT 2803 ATLANTA CENTRE ILOILO 1500 PHILIPPINES",
    "DevAddr": "UNIT 2803 ATLANTA CENTRE ILOILO 1500 PHILIPPINES",
    "AirNum": "",
    "AirNumDt": "",
    "LadNum": "",
    "LadNumDt": ""
  },
  "ItemList": [
    {
      "Nm": "Wehago",
      "Desc": "Wehago",

```

```

    "Qty": 2,
    "Unit": "EA",
    "UnitCost": 44130,
    "SalesAmt": 88260,
    "RegDscntAmt": 440,
    "SpeDscntAmt": 40,
    "NetSales": 87780
  },
  {
    "Nm": "Nahago",
    "Desc": "Nahago",
    "Qty": 9,
    "Unit": "EA",
    "UnitCost": 77840,
    "SalesAmt": 700560,
    "RegDscntAmt": 550,
    "SpeDscntAmt": 650,
    "NetSales": 699360
  },
  {
    "Nm": "ERP10",
    "Desc": "ERP10",
    "Qty": 9,
    "Unit": "EA",
    "UnitCost": 56070,
    "SalesAmt": 504630,
    "RegDscntAmt": 80,
    "SpeDscntAmt": 390,
    "NetSales": 504160
  },
  {
    "Nm": "GroupWare",
    "Desc": "GroupWare",
    "Qty": 4,
    "Unit": "EA",
    "UnitCost": 54810,
    "SalesAmt": 219240,
    "RegDscntAmt": 840,
    "SpeDscntAmt": 660,
    "NetSales": 217740
  },
  {
    "Nm": "A Item",
    "Desc": "A Item",
    "Qty": 4,
    "Unit": "EA",
    "UnitCost": 88700,
    "SalesAmt": 354800,
    "RegDscntAmt": 240,
    "SpeDscntAmt": 580,
    "NetSales": 353980
  },
  {
    "Nm": "B Item",
    "Desc": "B Item",
    "Qty": 5,
    "Unit": "EA",
    "UnitCost": 97670,
    "SalesAmt": 488350,
    "RegDscntAmt": 200,
    "SpeDscntAmt": 340,
    "NetSales": 487810
  },
  {
    "Nm": "C Item",
    "Desc": "C Item",
    "Qty": 2,
    "Unit": "EA",
    "UnitCost": 81020,
    "SalesAmt": 162040,
    "RegDscntAmt": 430,
    "SpeDscntAmt": 790,

```

```

    "NetSales": 160820
  }
},
"TotNetItemSales": 2511650,
"Discount": {
  "ScAmt": 360,
  "PwdAmt": 230,
  "RegAmt": 740,
  "SpeAmt": 550,
  "Rmk2": ""
},
"OtherTaxRev": 530,
"TotNetSalesAftDisct": 2510300,
"VATAmt": 301398.0,
"WithholdIncome": 420,
"WithholdBusVAT": 990,
"WithholdBusPT": 93,
"OtherNonTaxCharge": 97,
"NetAmtPay": 2810385.0,
"ForCur": { "Currency": "USD", "ForexAmt": 380, "ConvRate": 60 },
"PtuNum": "20146IT4-4"
}

```

4.2 CRM/POS Issued e-invoice JSON

The main difference is that the e-invoice JSON issued by CRM/POS does not contain item information and buyer information that exists in the e-invoice JSON issued from CAS, and it supports mixed tax invoices.

JSON format description on the invoice issued from CRM/POS.

(Refer to attachment excel file – 02.8.2022_EIS e-invoice JSON File format v2.0.xlsx Sheet ‘2.e-invoice format-POS_v2.0)

No	Item Name		Definition	Field Name	Data Type	Length (maximum)	Occurs	Default	Description
	category 1	category 2							(Note 1) If the data type is 'Number', the field has 2 decimal places and allows negative numbers including '-'. (Note 2) If there is no value in the field whose data type is 'String', enter 'null' or leave the field blank.
1	SI/OR Management Information		Company SI /OR/Sales Adjustment Documents No	Complnvoiceld	String	50	1..1 (Mandatory)		Control number of documents issued and maintained by the taxpayer system (POS, CRM). Documents : There are 3 document types to be sent to EIS. Sales of goods and services : 3 documents - Sales Invoice(SI), Official Receipt(OR), Sales Adjustment Documents to adjust sales data. (e.g. void, return, or cancellation of sold goods and services)
2			SI/OR/Adjustment slip Issuance Date	IssueDtm	String	8 44	1..1 (Mandatory)		Date the document (SI/OR/adjustment slip) was issued by seller. Generally, 'Issuance Date' and 'Transaction Date' are the same. Format: YYYYMMDD
3	E-Invoice Basic Information	BIR e-invoice Unique ID	BIR e-Invoice Unique ID	EisUniqeId	String	24	1..1 (Mandatory)		A unique value that identifies the e-invoice in the EIS, generated by the taxpayer/software (SW) system consisting of 24 digits, as shown below: Issuance Date (8 digits) + EIS-Certification ID (8 digits) + Control Value (8 digits) (a) Issuance Date(YYYYMMDD) Invoice issue date reflected in the system-generated invoice. (b) EIS-Certification ID EIS generated ID for taxpayers/SW providers to transmit sales data from their POS, CRM to EIS, consisting of 8 digits as shown below: Source code (2 digits) + XXXXXX (6 digits) (c) Control Values (8 digits) Eight (8) characters which may be random combination of numeric, alpha (uppercase), or alpha-numeric.
4		Invoice Classification	Document Type	DocType	String	2	1..1 (Mandatory)		01 : Sales Invoice (SI) 05 : Official Receipt (OR) 06 : Adjustment Document (e.g VoidReturn/Cancel Slip, etc)
5		Invoice Correction	Correction Yes or Not	CorrYN	String	1	1..1 (Mandatory)		The value is "Y" for corrected/adjusted/modified previously transmitted data; otherwise, the value is "N".
6			e-Invoice correction code	CorrectionCd	String	2	0..1* (Conditional)		Mandatory field if the value in "CorrYN" is "Y" The reasons for the modification/correction/adjustment of e-invoice are classified as follows. Single previously issued e-Invoice 01: Error - In any mandatory field such as: a. Seller Information (TIN & Branch Code, Registered Name, VAT Type) b. Buyer Information (TIN & Branch Code, Registered Name) c. Details of Sales, Amount, etc. - Total Net Sales After Discount - VAT Amount - Issuance Date - Details of Sold Items d. Classification of Sales as Taxable, Exempt, or Zero-rated. 02: Duplication - Sales data transmitted more than once regardless of transaction period. 03: Addition/reduction - Addition/reduction of contracted volume or amount 04: Cancellation - For sales data previously transmitted but transaction was subsequently cancelled or rescinded by buyer/seller for failure to consummate. Full return of goods without replacement is considered cancellation. 05: Return - For original goods supplied and returned for replacement. Ex: Damaged, defective, wrong specifications, change item, or for any other reason 09: Others This is the 'e-invoice unique ID' of the previously transmitted sales data to EIS to be modified/corrected/adjusted. If multiple documents are issued in one adjustment document, 'e-Invoice correction code' value is '09', enter the 'e-Invoice unique ID' of one document among multiple documents in this field, and the 'e-Invoice unique ID' of the remaining documents in 'Remarks1'.
7			E-Invoice Unique ID of the document to be corrected	PrevUniqeId	String	24	0..1* (Conditional)		
8		Remarks	Remarks1	Rmk1	String	500	0..1 (Optional)		Explanation for the correction made

	Seller Information	Seller Information	Seller Information	SellerInfo			1.1 (Mandatory)	In accordance or consistent with the BIR registration information.
9			Seller TIN	Tin	String	9	1.1 (Mandatory)	9-digit TIN of seller Only numbers must be entered, example, 123456789(), 123-456-7(X)
10			Branch Code	BranchCd	String	5	1.1 (Mandatory)	5-digit code that identifies a headquarters or branch office. Branch code uses 3 digits, add '00' to the first two digits. For example, branch code is '101', it must be converted to '00101'.
12			Seller Type	Type	String	1	1.1 (Mandatory)	~VAT" or "Non-VAT" Registered Seller Type 0 : VAT registered 1 : Non-VAT registered
13			Registered Name	RegNm	String	200	1.1 (Mandatory)	In accordance or consistent with BIR registration information.
14			Business Name/Trade Name	BusinessNm	String	200	1.1 (Mandatory)	In accordance or consistent with BIR registration information if not, enter the same as the "Registered Name".
15			Email address	Email	String	100	0.1 (Optional)	Company email address or personal email address of the authorized representative
16			Registered Address	RegAddr	String	300	0.1 (Optional)	In accordance or consistent with BIR registration information
	Buyer Information	Buyer Information	TaxPayer Buyer Information	BuyerInfo			0.1 (Optional)	In accordance or consistent with BIR registration information. This is optional. In case the buyer will be claiming for input VAT or will be deducted from cost/expense for ITR.
17			Buyer TIN	Tin	String	9	0.1 (Optional)	9-digit TIN of the buyer Buyer is not registered with the BIR (i.e. he does not have a TIN), enter the TIN as '00000000'(9 digits).
18			Branch Code	BranchCd	String	5	0.1 (Optional)	5-digit code that identifies a headquarters or branch office. Branch code uses 3 digits, add '00' to the first two digits. Example, branch code is '101', it must be converted to '00101'.
19			Registered Name	RegNm	String	200	0.1 (Optional)	In accordance or consistent with BIR registration information.
20			Business Name/Trade Name	BusinessNm	String	200	0.1 (Optional)	In accordance or consistent with BIR registration information. If there is no Business Name/Trade Name, enter the same as the Registered Name.
21			Email address	Email	String	100	0.1 (Optional)	Buyer company email or the email of the accounting representative within the company.
22			Registered Address	RegAddr	String	300	0.1 (Optional)	In accordance or consistent with BIR registration information. and indicated on the SI/OR adjustment slips
23	Sales Information	Sales Summary	VATable Sales	VATSales	Number	30	1.1 (Mandatory)	0.00 Sales subject to VAT at 12%
24			Other taxable revenue	OtherTaxRev	Number	30	1.1 (Mandatory)	0.00 Other taxable revenues not part of the primary revenue-generating items (e.g. miscellaneous income, taxable service charge).
25			VAT Exempt Sales	ExemptSales	Number	30	1.1 (Mandatory)	0.00 Sales amount exempt from VAT.
26			Zero Rated Sales	ZeroSales	Number	30	1.1 (Mandatory)	0.00 Sales amount subject to VAT at zero percent (0%) .
27			Total Sales Amount	TotSalesAmt	Number	30	1.1 (Mandatory)	0.00 Sum of all sales, exclusive of VAT + other taxable revenues.
		Discount Information	Discount	Discount			1.1 (Mandatory)	Discount based on the Total Sales Amount
28			Senior Citizen Discount Amount	ScAmt	Number	30	1.1 (Mandatory)	0.00 Applicable to transactions with Senior Citizen (SC)
29			PWD Discount Amount	PwdAmt	Number	30	1.1 (Mandatory)	0.00 Applicable to transactions with Person With Disability (PWD)
30			Regular Discount Amount	RegAmt	Number	30	1.1 (Mandatory)	0.00 Granted based on the Total Sales Amount that can be deducted for both VAT and Income tax purposes (outright discounts)
31			Special Discount Amount	SpeAmt	Number	30	1.1 (Mandatory)	0.00 Granted based on the Total Sales Amount that can be deducted for income tax purposes only (ex. Conditional discounts)
32			Remarks2	Rmk2	String	500	0.1 (Optional)	Brief explanation of discount, if needed
33		Total Net Sales after discount	Total Net Sales After Discounts	TotNetSalesAftDisct	Number	30	1.1 (Mandatory)	0.00 Total Sales Amount + Other Taxable Revenue - Discount on Total Net Sales
34		Tax Information	VAT Amount	VATAmt	Number	30	1.1 (Mandatory)	0.00 VAT Amount reflected in the document
35			Withholding Tax-Income Tax	WithholdIncome	Number	30	1.1 (Mandatory)	0.00 Amount withheld by buyer for purposes of income tax.
36			Withholding Tax-Business VAT	WithholdBusVAT	Number	30	1.1 (Mandatory)	0.00 Amount Withheld by buyer for purposes of VAT
37			Withholding Tax-Business Percentage	WithholdBusPT	Number	30	1.1 (Mandatory)	0.00 Amount Withheld by buyer on transactions subject to percentage tax
38		Non-Taxable	Other Non-Taxable Charges	OtherNonTaxCharge	Number	30	1.1 (Mandatory)	0.00 Any non-taxable charges of seller to buyer (e.g. local tax, documentary stamp tax, non-taxable service charge, etc.)
39		Net Amount Payable	Net Amount Payable	NetAmtPay	Number	30	1.1 (Mandatory)	0.00 Amount to be paid by the buyer.
40	PTU Information	CRM/POS	Permit To Use (PTU) Number	PtUNum	String	50	1.1 (Mandatory)	Number issued by the BIR allowing/registering taxpayer's use of POS/CRM
41			Machine Identification Number(MIN)	Min	String	50	1.1 (Mandatory)	Unique number generated by the eAccReg system for every POS/CRM.
42			Machine Serial Number (MSN)	MsN	String	50	1.1 (Mandatory)	MSN of machine that generated sales data

Example of JSON)

```
{
  "CompInvoiceId": "0000000004",
  "IssueDtm": "20220217",
  "EisUniqueId": "2022021731FL1P0700000004",
  "DocType": "01",
  "CorrYN": "N",
  "CorrectionCd": "",
  "PrevUniqueId": "",
  "Rmk1": "test e-invoice : 2022021731FL1P0700000004",
  "SellerInfo": {
    "Tin": "999999999",
    "BranchCd": "00000",
    "Type": "0",
    "RegNm": "DOUZONE",
    "BusinessNm": "DOUZONE",
    "Email": "tester@douzone.com",
    "RegAddr": "130, Beodeul 1-gil, Namsan-myeon, Chuncheon-si, Gangwon-do, Republic of Korea"
  },
  "BuyerInfo": {
    "Tin": "",
    "BranchCd": "",
    "Type": "",
    "RegNm": "",
    "BusinessNm": "",
    "Email": "",
    "RegAddr": ""
  },
  "VatSales": 980,
}
```



```
"OtherTaxRev": 57,  
"ExemptSales": 970,  
"ZeroSales": 680,  
"TotSalesAmt": 2630,  
"Discount": {  
  "ScAmt": 320,  
  "PwdAmt": 620,  
  "RegAmt": 620,  
  "SpeAmt": 360,  
  "Rmk2": ""  
},  
"TotNetSalesAftDisct": 767,  
"VATAmt": 117.6,  
"WithholdIncome": 32,  
"WithholdBusVAT": 19,  
"WithholdBusPT": 48,  
"OtherNonTaxCharge": 5,  
"NetAmtPay": 838.6,  
"PtuNum": "00004",  
"Msn": "test-msn-00004",  
"Min": "test-min-00004"  
}
```

5. Separate issuance of mixed invoices

5.1 Summary

Invoice issued by CAS must be issued separately for VATable, VAT Exempt, and Zero Rated. It is impossible to issue a mix of VATable, VAT Exempt, and Zero Rated on one invoice item. Therefore if VATable, VAT Exempt, and Zero Rated are mixed in one invoice issued by CAS, VATable, VAT Exempt, and Zero tax must be issued separately as separate invoices.

5.2 Examples

Example of invoice with mixed VATable, Zero Rated, and VAT Exempt sales issued by CAS.

Pharmaceutical Company							
***** Manalao Avenue 1882 Taguig City Distributor TIN : 004-000-000-00000				***** 23rd Flr., Citibank Tower 5741 Paseo de Roxas, Salcedo Village 1227 Telephone : 552-5570500 Fax : 552-5491471 VAT REG TIN : 000-000-000-00000 Name : CITIBANK N.A Bank account number : ***** SWIFT cod : CITIPHMM			
Invoice/Receipt No Sales Invoice : 5551000028				Sales Invoice Date : Nov-10-2020			
Customer No :		2282014	Our Order No :		1550000139		
Purchase Order No :		test	Order Date :		Nov-10-2020		
Terms of payment :		Up to 08-19-2018. Within go days Due net	Our Delivery No :		3550000067		
Terms of delivery :		DDP Manila	Date of Supply :		Nov-10-2020		
Material	Description	Quantity	Tax Code	Unit Price	Discount	Net Unit Price	Net Amount
301357	MICARDIS.TABS/30/40MG BLIBR Batch : 706534 Exp.Date : 30-Sep-2021	5EA	A1	100.00	30.00	70.00	350.00
301357	MICARDIS.TABS/30/40MG BLIBR Batch : 706534 Exp.Date : 30-Sep-2021	5EA	A1	100.00	0.00	100.00	500.00
396357	Micardis 40mg PS 1's (own test do not us Batch : PH10002BIX Exp.Date : 17-Apr-2019	1EA	A2	1,000.00	0.00	1,000.00	1,000.00
116676	Giyxambi Tablet 10mg/5mg 30's Batch : 706669 Exp.Date : 30-Sep-2020	1EA	A3	50.00	0.00	50.00	50.00
Total Amount Due : VATable Sales : 758.93 VAT Amount : 91.07 Zero Rated Sales : 1,000.00 VAT Exempt Sales 50.00							19,000.00
							A1 : VATable Amount 12% A2 : Zero Rated VAT 0% A3 : VAT Exempt

When sending the above invoice to EIS, it must be sent separately into 3 invoices for VATable Sales, Zero Rated Sales, and VAT Exempt Sales.

Separately sent invoices have the same Invoice Number in EIS and the EIS Unique Invoice ID is displayed as 3 different invoices. #1, #2, and #3 below are examples of separate invoices in EIS.

e-invoice on EIS (VATable Sales) #1

Unique Invoice ID 20201110-20000001-00000001						Invoice No. or B/S No. or SOA No. 55510000028																																												
SELLER	TIN	000-000-111	Branch Code	00000	VAT or Non-VAT	VAT	TIN	444-444-444	Branch Code	00000																																								
	Name	Pharmaceutical Company					Name	AAA Corporation																																										
	Business Name	Pharmaceutical Company					Business Name																																											
	Address	Manalao Avenue 1632 Taguig City					Address	23rd Flr, Cisbank Tower 5741 Paseo de Roxas, Saicedo Village 1227																																										
	E-mail Address	Pharmaceutical@gmail.com					E-mail Address	purchase@aaacorp.co.ph																																										
							Delivery Address																																											
	Reason for correction	Return					Airway bill number		Airway bill number date																																									
						Bill of lading number		Bill of lading number date																																										
Issuance Date		2020.11.10		Note																																														
Item Name	Description of Item	Quantity	Unit	Unit Cost	Sales Amount	Regular Discount Amount	Special Discount Amount	Net of Item Sales																																										
301357	MICARDIS.TABS... BLIBR	5	Set	89.29	446.43	133.95	0.00	312.50																																										
301357	MICARDIS.TABS... BLIBR	5	Set	89.29	446.43	0.00	0.00	446.43																																										
<table border="1"> <tr> <td rowspan="5">Discount Amount</td> <td>Senior Citizen</td> <td>0.00</td> <td rowspan="4">Deduct : Creditable Withholding Tax</td> <td>Income Tax</td> <td>0.00</td> <td rowspan="5">Total Net Sales Amount</td> <td>758.93</td> </tr> <tr> <td>PWD</td> <td>0.00</td> <td>Business VAT</td> <td>0.00</td> <td>Currency</td> <td>0.00</td> </tr> <tr> <td>Regular</td> <td>0.00</td> <td>Business Percentage</td> <td>0%</td> <td>Currency Amount</td> <td>0.00</td> </tr> <tr> <td>Special</td> <td>0.00</td> <td>Service Charge</td> <td>0.00</td> <td>Total Net Sales After Discount</td> <td>758.93</td> </tr> <tr> <td>Remark</td> <td>-</td> <td>Local Tax</td> <td>0.00</td> <td>VAT Amount</td> <td>91.07</td> </tr> <tr> <td colspan="6"></td> <td>Net Amount Payable</td> <td>850.00</td> </tr> </table>											Discount Amount	Senior Citizen	0.00	Deduct : Creditable Withholding Tax	Income Tax	0.00	Total Net Sales Amount	758.93	PWD	0.00	Business VAT	0.00	Currency	0.00	Regular	0.00	Business Percentage	0%	Currency Amount	0.00	Special	0.00	Service Charge	0.00	Total Net Sales After Discount	758.93	Remark	-	Local Tax	0.00	VAT Amount	91.07							Net Amount Payable	850.00
Discount Amount	Senior Citizen	0.00	Deduct : Creditable Withholding Tax	Income Tax	0.00	Total Net Sales Amount	758.93																																											
	PWD	0.00		Business VAT	0.00		Currency	0.00																																										
	Regular	0.00		Business Percentage	0%		Currency Amount	0.00																																										
	Special	0.00		Service Charge	0.00		Total Net Sales After Discount	758.93																																										
	Remark	-	Local Tax	0.00	VAT Amount		91.07																																											
						Net Amount Payable	850.00																																											
Correction Issue		Copy Issue				JWS Download		Print																																										

e-invoice on EIS (Zero Rated Sales) #2

■ Details of e-Tax Invoice

Unique Invoice ID		20201110-20000001-00000002				Invoice No. or B/S No. or SOA No.		55510000028	
-------------------	--	----------------------------	--	--	--	-----------------------------------	--	-------------	--

SELLER	TIN	000-000-111	Branch Code	00000	VAT or Non-VAT	VAT
	Name	Pharmaceutical Company				
	Business Name	Pharmaceutical Company				
	Address	Manalao Avenue 1632 Taguig City				
	E-mail Address	Pharmaceutical@gmail.com				
	Reason for correction	05				
	Issuance Date	2020.11.10		Note		

BUYER	TIN	444-444-444	Branch Code	00000
	Name	AAA Corporation		
	Business Name			
	Address	23rd Plr. CIsbank Tower 5741 Paseo de Roxax. Saicedo Village 1227		
	E-mail Address	purchase@aaacorp.co.ph		
	Delivery Address			
	Airway bill number	Airway bill number date	Bill of lading number date	

Item Name	Description of Item	Quantity	Unit	Unit Cost	Sales Amount	Regular Discount Amount	Special Discount Amount	Net of Item Sales
396357	Micardis 40mg PS 1's (own test do not us	1	Set	1,000.00	1,000.00	0.00	0.00	1,000.00

Discount Amount	Senior Citizen	0.00	Deduct : Creditable Withholding Tax	Income Tax	0.00
	PWD	0.00		Business VAT	0.00
	Regular	0		Business Percentage	0%
	Special	0.00		Service Charge	0.00
	Remark	-	Local Tax	0.00	

Total Net Sales Amount	1,000.00
Currency	0.00
Currency Amount	0.00
Total Net Sales After Discount	1,000.00
VAT Amount	0.00
Net Amount Payable	1,000.00

[Correction Issue](#)
[Copy Issue](#)
[Back](#)
[Print](#)

e-invoice on EIS (VAT Exempt Sales) #3

■ Details of e-Tax Invoice

Unique Invoice ID		20201110-20000001-00000003				Invoice No. or B/S No. or SOA No.		55510000028		
SELLER	TIN	000-000-111	Branch Code	00000	VAT or Non-VAT	VAT	TIN	444-444-444	Branch Code	00000
	Name	Pharmaceutical Company				Name	AAA Corporation			
	Business Name	Pharmaceutical Company				Business Name				
	Address	Manalao Avenue 1632 Taguig City				Address	23rd Pir. Cisbank Tower 5741 Paseo de Roxax. Saicedo Village 1227			
	E-mail Address	Pharmaceutical@gmail.com				E-mail Address	purchase@aaacorp.co.ph			
						Delivery Address				
						Airway bill number		Airway bill number date		
Reason for correction	05				Bill of lading number		Bill of lading number date			
Issuance Date	2020.11.10		Note							
Item Name	Description of Item	Quantity	Unit	Unit Cost	Sales Amount	Regular Discount Amount	Special Discount Amount	Net of Item Sales		
116676	Glyxambi Tablet 10mg/5mg 30's	1	Set	50.00	50.00	0.00	0.00	50.00		
Discount Amount	Senior Citizen	0.00	Deduct : Creditable Withholding Tax	Income Tax	0.00	Total Net Sales Amount		50.00		
	PWD	0.00		Business VAT	0.00	Currency		0.00		
	Regular	0		Business Percentage	0%	Currency Amount		0.00		
	Special	0.00		Service Charge	0.00	Total Net Sales After Discount		50.00		
	Remark	-		Local Tax	0.00	VAT Amount		0.00		
					Net Amount Payable		50.00			

Correction Issue Copy Issue Back Print

5.3 Sample JSON

Based on the above example, a sample of three JSON files that are transmitted separately are as follows.

JSON-01: VATable Sales

```
{
  "CompInvoiceId": "55510000028",
  "IssueDtm": "20201110",
  "EisUniqueId": "202011102000000100000001",
  "CorrYN": "N",
  "TransClass": "01",
  "DocType": "01",
  "Rmk1": "",
  "SellerInfo": {
    "Tin": "123456789",
    "BranchCd": "00000",
    "Type": "0",
    "RegNm": "Pharmaceutical Company",
    "BusinessNm": "Pharmaceutical Company",
    "Email": "Pharmaceutical@gmail.com",
    "RegAddr": "Manalao Avenue 1632 Taguig City"
  },
  "BuyerInfo": {
    "Tin": "444444444",
    "BranchCd": "00000",
    "RegNm": "AAA Corporation",
    "BusinessNm": "AAA Corporation",
    "Email": "purchase@aaacorp.co.ph",
    "RegAddr": "23rd Pir. Cisbank Tower 5741 Paseo de Roxax. Saicedo Village 1227"
  }
}
```

```

"ItemList": [
  {
    "Nm": "301357",
    "Desc": "MICARDIS.TABS/30/40MG BLIBR",
    "Qty": 5,
    "Unit": "Set",
    "UnitCost": 89.29,
    "SalesAmt": 446.43,
    "RegDscntAmt": 133.95,
    "SpeDscntAmt": 0.00,
    "NetSales": 312.50
  },
  {
    "Nm": "301357",
    "Desc": "MICARDIS.TABS/30/40MG BLIBR",
    "Qty": 5,
    "Unit": "Set",
    "UnitCost": 89.29,
    "SalesAmt": 446.43,
    "RegDscntAmt": 0.00,
    "SpeDscntAmt": 0.00,
    "NetSales": 446.43
  }
],
"TotNetItemSales": 758.93,
"Discount": {
  "ScAmt": 0.00,
  "PwdAmt": 0.00,
  "RegAmt": 0.00,
  "SpeAmt": 0.00
},
"OtherTaxRev": 0.00,
"TotNetSalesAftDisct": 758.93,
"VATAmt": 91.07,
"WithholdIncome": 0.00,
"WithholdBusVAT": 0.00,
"WithholdBusPT": 0.00,
"OtherNonTaxCharge": 0.00,
"NetAmtPay": 850.00,
"PtuNum": "FP012345-011-0123456-00001",
}

```

JSON-02: Zero-Rated Sales

```

{
  "CompInvoiceId": "55510000028",
  "IssueDtm": "20201110",
  "EisUniqueId": "202011102000000100000002",
  "CorrYN": "N",
  "TransClass": "02",
  "DocType": "01",
  "Rmk1": "",
  "SellerInfo": {
    "Tin": "123456789",
    "BranchCd": "00000",
    "Type": "0",
    "RegNm": "Pharmaceutical Company",
    "BusinessNm": "Pharmaceutical Company",
    "Email": "Pharmaceutical@gmail.com",
    "RegAddr": "Manalao Avenue 1632 Tagulg City"
  },
  "BuyerInfo": {
    "Tin": "444444444",
    "BranchCd": "00000",
    "RegNm": "AAA Corporation",
    "BusinessNm": "AAA Corporation",
    "Email": "purchase@aaacorp.co.ph",
    "RegAddr": "23rd Pir. Cisbank Tower 5741 Paseo de Roxax. Saicedo Village 1227"
  },
  "ItemList": [
    {
      "Nm": "396357",

```

```

        "Desc": "Micardis 40mg PS 1's (own test do not us",
        "Qty": 1,
        "Unit": "Set",
        "UnitCost": 1000.00,
        "SalesAmt": 1000.00,
        "RegDscntAmt": 0.00,
        "SpeDscntAmt": 0.00,
        "NetSales": 1000.00
    }
],
"TotNetItemSales": 1000.00,
"Discount": {
    "ScAmt": 0.00,
    "PwdAmt": 0.00,
    "RegAmt": 0.00,
    "SpeAmt": 0.00
},
"OtherTaxRev": 0.00,
"TotNetSalesAftDisct": 1000.00,
"VATAmt": 0.00,
"WithholdIncome": 0.00,
"WithholdBusVAT": 0.00,
"WithholdBusPT": 0.00,
"OtherNonTaxCharge": 0.00,
"NetAmtPay": 1000.00,
"PtuNum": "FP012345-011-0123456-00001"
}

```

JSON-03: Exempt Sales

```

{
    "CompInvoiceId": "55510000028",
    "IssueDtm": "20201110104521",
    "TransDtm": "20201110104000",
    "EisUniqueId": "202011102000000100000003",
    "CorrYN": "N",
    "TransClass": "03",
    "DocType": "01",
    "Rmk": "",
    "SellerInfo": {
        "Tin": "123456789",
        "BranchCd": "00000",
        "Type": "0",
        "RegNm": "Pharmaceutical Company",
        "BusinessNm": "Pharmaceutical Company",
        "Email": "Pharmaceutical@gmail.com",
        "RegAddr": "Manalao Avenue 1632 Tagulig City"
    },
    "BuyerInfo": {
        "Tin": "444444444",
        "BranchCd": "00000",
        "RegNm": "AAA Corporation",
        "BusinessNm": "AAA Corporation",
        "Email": "purchase@aaacorp.co.ph",
        "RegAddr": "23rd Pir. Cisbank Tower 5741 Paseo de Roxax. Saicedo Village 1227"
    },
    "ItemList": [
        {
            "Nm": "116676",
            "Desc": "Giyxambi Tablet 10mg/5mg 30's",
            "Qty": 1,
            "Unit": "Set",
            "UnitCost": 50.00,
            "SalesAmt": 50.00,
            "RegDscntAmt": 0.00,
            "SpeDscntAmt": 0.00,
            "NetSales": 50.00
        }
    ],
    "TotNetItemSales": 50.00,
    "Discount": {
        "ScAmt": 0.00,

```

```
    "PwdAmt": 0.00,  
    "RegAmt": 0.00,  
    "SpeAmt": 0.00  
  },  
  "OtherTaxRev": 0.00,  
  "TotNetSalesAftDisct": 50.00,  
  "VATAmt": 0.00,  
  "WithholdIncome": 0.00,  
  "WithholdBusVAT": 0.00,  
  "WithholdBusPT": 0.00,  
  "OtherNonTaxCharge": 0.00,  
  "NetAmtPay": 50.00,  
  "PtuNum": "FP012345-011-0123456-00001",  
}
```

6. Security

6.1 Summary

EIS e-invoice API is a RESTful Service API that may be weak in security compared to the SOAP protocol using WS-Security. To cover these security vulnerabilities, various security techniques have been applied.

6.2 7steps of API security safeguard

EIS e-invoice API applied 7 steps of security safeguard to prevent e-invoice data leakage or tampering/falsification.

1. Verify user using the User ID and PW of EIS Accreditation Website
2. Application verification using Accreditation ID, registered Application ID, Application Secret Key issued from EIS Accreditation Website.
3. Encrypt inter-transmission communication using TLS(SSL)
4. Encrypt transmitting data using temporary private key exchange valid only during the Session.
5. Secure integrity of data using HMAC (Hash-based Message Authentication Code)
6. Digital signature of e-invoice document using EIS own Key-pair
7. Strengthen the security applying IP White List Tunneling

6.3 invoice security

The following 'document security' process is required to secure the invoice to send from taxpayers to the BIR.

First, the digital signature of the invoice issuer is needed.

Second, the encryption process is needed for the security of online data.

6.3.1 Main standards and Algorithms

The following is the main standards used for the invoice security

Use	Standard and Algorithm	Purpose of Use
Digital signature	IETF RFC 7515 JSON Web Signature (JWS)	JSON digital signature standard
	IETF RFC 3447 RSASSA-PKCS1-v1.5 SHA-256	Invoice JWS digital signature Hash Algorithm
Encryption	ISO / IEC 18033-3 AES (Advanced Encryption Standard)	Symmetric key data encryption/decryption
	IETF RFC 3447 PKCS #1 RSA Cryptography Standard	Asymmetric keys data encryption/decryption
Transmission Security	IETF RFC 2014 HMAC	Transmission message integrity verification

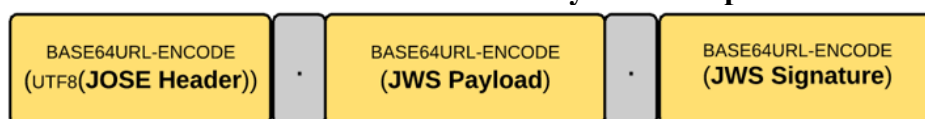
6.4 JWS digital signature

The digital signature is needed on the e-invoice JSON in order to ensure that the data has not been tampered with/changed since it was prepared by the taxpayer.

The algorithm used in the digital signature is RSA PKCS#1 signature with SHA-256, and the method of digital signature is JSON Web Signature (JWS) considering the consideration below;

- JWS IETF standard (RFC7516).
- The development is not complicated because the JWS does not need to canonicalize the JSON.
 - The original JSON Format is maintained in serialized format. (Format encoded in base64url to be exact)
- There is a various library which supports JWS, therefor it is easy to implement the digital signature.
 - Nimbus (Java): <http://connect2id.com/products/nimbus-jose-jwt>
 - Nuget (.NET): <http://www.nuget.org/packages/jose-jwt/>
 - pyjwt (Python): <https://github.com/jpadilla/pyjwt/>
 - jsrsasign (javascript): <https://github.com/kjur/jsrsasign>
 - php-jwt (PHP): <https://github.com/firebase/php-jwt>

The structure of a JWS token is formed by JWS compact serialization



6.5 Transmission Data Encryption

6.5.1 Overview

Invoices sent to BIR must be kept confidential and must comply with the following encryption process.

6.5.2 Encryption algorithm

The algorithm used in the encryption can be divided into two groups.

There is a symmetric key algorithm used to directly encrypt the target data and a public key (asymmetric key) algorithm that transmits the symmetric key so that only the recipient can decrypt it.

The Public Key used in the public key algorithm is registered on the EIS Accreditation Website and can be used by anyone.

The asymmetric key algorithm is used ONLY in the Authentication API Request to deliver the Authentication Temporary Secret Key (32- digits string in plain text) randomly generated by the taxpayer's system to the EIS.

For other API calls, the AES-256 algorithm is used considering both security and performance.

6.5.3 Target data to be encrypted

Encryption between API transmissions is performed in the following cases below, and the target and algorithm for each encryption are as follows.

Case	Taxpayer's Action	Encryption Target	Algorithm	Encryption Secret Key
Request Authentication API	Encryption	<ul style="list-style-type: none"> ● User ID ● Password ● Authentication ● Temporary Secret Key 	RSA	EIS Public Key
Response Authentication API	Decryption	<ul style="list-style-type: none"> ● Accreditation ID ● User ID ● Authentication Token ● Session Secret Key ● Token Expiry Date 	AES-256	Authentication Temporary Secret Key
Request Invoice Issuance API	Encryption	<ul style="list-style-type: none"> ● Invoice JSON Data 	AES-256	Session Secret Key
Response Invoice Issuance API	Decryption	<ul style="list-style-type: none"> ● Accreditation ID ● User ID ● Reference Submit ID ● Acknowledgment ID ● Response DateTime 	AES-256	Session Secret Key

6.5.4 Asymmetric encryption sample code

[Code Sample - Java]

```
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.interfaces.RSAPublicKey;
import java.util.Base64;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

public class RSA {
    private static final String RSA = "RSA";

    /**
     * RSA Encryption
     *
     * @param content
     * @param publicKey
     *
     * @return Encrypted content
     *
     * @throws NoSuchPaddingException - This exception is thrown when a particular
     *     * cryptographic algorithm is requested but is not available in the environment.
     * @throws NoSuchAlgorithmException - This exception is thrown when a particular padding
     *     * mechanism is requested but is not available in the environment.
     * @throws InvalidKeyException - This is the exception for invalid Keys (invalid encoding,
     *     * wrong length, uninitialized, etc).
     * @throws BadPaddingException - This exception is thrown when a particular padding
     *     * mechanism is expected for the input data but the data is not padded properly.
     * @throws IllegalBlockSizeException - This exception is thrown when the length of data
     *     * provided to a block cipher is incorrect, i.e., does not match the block size
     *     * of the cipher.
     * @throws UnsupportedEncodingException - The Character Encoding is not supported.
     */
    public static String encrypt(String content, RSAPublicKey publicKey) throws
        NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
        IllegalBlockSizeException, BadPaddingException, UnsupportedEncodingException {

        // Specify RSA Algorithm
        Cipher cipher = Cipher.getInstance(RSA);
        // Set to Encryption Mode using RSAPublicKey
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        // Cipher converts content to byte array
        byte[] bytePlain = cipher.doFinal(content.getBytes());
        // Encrypt content
        String encrypted = Base64.getEncoder().encodeToString(bytePlain);

        return encrypted;
    }
}
```

6.5.5 Symmetric encryption/decryption sample code

[Code Sample - Java]

```
package ph.eis.acc.common.util;

import java.io.UnsupportedEncodingException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class AES {
    private static final String AES = "AES";
    private static final String UTF_8 = "UTF-8";
    private static final String TRANSFORMATION = "AES/CBC/PKCS5Padding";

    /**
     * AES Encryption
     *
     * @param content
     * @param secretKey
     *
     * @return encrypt content
     *
     * @throws NoSuchPaddingException - This exception is thrown when a particular
     * cryptographic algorithm is requested but is not available in the environment.
     * @throws NoSuchAlgorithmException - This exception is thrown when a particular padding
     * mechanism is requested but is not available in the environment.
     * @throws InvalidKeyException - This is the exception for invalid Keys (invalid encoding,
     * wrong length, uninitialized, etc).
     * @throws BadPaddingException - This exception is thrown when a particular padding
     * mechanism is expected for the input data but the data is not padded properly.
     * @throws IllegalBlockSizeException - This exception is thrown when the length of data
     * provided to a block cipher is incorrect, i.e., does not match the block size of the
     * cipher.
     * @throws UnsupportedEncodingException - The Character Encoding is not supported.
     * @throws InvalidAlgorithmParameterException - This is the exception for invalid or
     * inappropriate algorithm parameters.
     */
    public static String encrypt(String content, String secretKey) throws
        NoSuchAlgorithmException, NoSuchPaddingException,
        InvalidKeyException, InvalidAlgorithmParameterException, IllegalBlockSizeException,
        BadPaddingException, UnsupportedEncodingException {

        // SecretKey converts to byte array
        byte[] secretKeyByte = secretKey.getBytes();
        // Set to SecretKeySpec to AES
        SecretKey key = new SecretKeySpec(secretKeyByte, AES);
        // Specify AES Algorithm
        Cipher cipher = Cipher.getInstance(TRANSFORMATION);
        // Set to Encryption Mode using AES
        cipher.init(Cipher.ENCRYPT_MODE, key, new IvParameterSpec(
            secretKey.substring(0, 16).getBytes()));
        // Cipher converts content to byte array
        byte[] encrypted = cipher.doFinal(content.getBytes(UTF_8));
        // Encrypt content
        String str = Base64.getEncoder().encodeToString(encrypted);

        return str;
    }
}
```

```

* AES Encryption
*
* @param content
* @param secretKey
*
* @return encrypt content
*
* @throws NoSuchPaddingException - This exception is thrown when a particular
* cryptographic algorithm is requested but is not available in the environment.
* @throws NoSuchAlgorithmException - This exception is thrown when a particular padding
* mechanism is requested but is not available in the environment.
* @throws InvalidKeyException - This is the exception for invalid Keys (invalid encoding,
* wrong length, uninitialized, etc).
* @throws BadPaddingException - This exception is thrown when a particular padding
* mechanism is expected for the input data but the data is not padded properly.
* @throws IllegalBlockSizeException - This exception is thrown when the length of data
* provided to a block cipher is incorrect, i.e., does not match the block size of the
* cipher.
* @throws UnsupportedEncodingException - The Character Encoding is not supported.
* @throws InvalidAlgorithmParameterException - This is the exception for invalid or
* inappropriate algorithm parameters.
*/
public static String decrypt(String encryptContent, String secretKey) throws
    NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
    InvalidAlgorithmParameterException, UnsupportedEncodingException,
    IllegalBlockSizeException, BadPaddingException {

    // SecretKey converts to byte array
    byte[] secretKeyByte = secretKey.getBytes();
    // Set to SecretKeySpec to AES
    SecretKey key = new SecretKeySpec(secretKeyByte, AES);
    // Specify AES Algorithm
    Cipher cipher = Cipher.getInstance(TRANSFORMATION);
    // Set to Decryption Mode using AES
    cipher.init(Cipher.DECRYPT_MODE, key, new IvParameterSpec(
        secretKey.substring(0, 16).getBytes(UTF_8)));

    // Decode encrypted content
    byte[] byteStr = Base64.getDecoder().decode(encryptContent.getBytes());

    return new String(cipher.doFinal(byteStr), UTF_8);
}

```

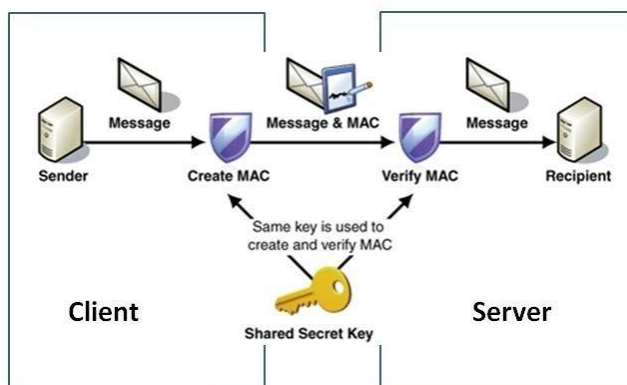
6.6 Transmission security

6.6.1 HMAC Signature

HMAC is an acronym for Hash Message Authentication Code, RFC2014 standard encryption protocol.

The EIS e-invoice API is build based on the HMAC, and it is required to send the generated HMAC signature in all request header 'Authorization'.

HMAC Signature certification process



1. The client processes the Key + message with the HMAC algorithm to create a hash value (signature).
2. The HASH value and the values required to generate the hash value are sent to the API server by putting them in the header as an HTTP request.
3. The server combines the request received from the client and generates an HMAC hash value.
4. It compares whether the HASH transmitted from the client and the HASH generated by the server are the same. If they are the same, authentication is successful

6.6.2 HMAC Signature Rule

		Rule
HMAC Message Value		DATETIME(yyyyMMddHHmmss) + HTTP_METHOD + API URL (Without domain)
HMAC Secret Key	Authentication API	Use Application Secret Key
	Other API	Use Session Secret Key

HMAC Signature generating process

1. Create HMAC message value

$$\text{hmac_value} = \text{DATETIME} + \text{HTTP_METHOD} + \text{API_URL}$$
2. Create hash value with HMAC SHA 256 algorithm

$$\text{hash_value} = \text{hmac_sha_256_hash_function}(\text{HMAC Secret Key}, \text{hmac_value})$$
3. Create HMAC signature value

$$\text{signature} = \text{Authentication Type} + " " + \text{hash_value}$$

Ex)

hmac_value = "20220217" + "POST" + "/api/v1/authentication"

hash_value = hmac_sha_256_hash_function("X8TD8UUEDTK7", hmacValue)

signature = "Bearer" + " " + hash_value

6.6.3 Sample Signature, Request Header transmitted by the taxpayer

Header Attribute		Header Value
All API	accreditationId	20111111
	applicationId	A1B2C3D4
	authorization	Bearer 4C9VQAMV2Y18EN8C20PHT5O128SFONM3
	datetime	20220217093015
Invoice Issuance API / Inquiry Result API	authToken	NGHQH08X9RPP6I0S0Y00F4DY3TDX15QG

6.6.4 HMAC Signature generation + API request sample

[Code Sample - Java]

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.util.Base64;

public class Hmac {

    private final String charsetName = "UTF-8";
    // Algorithms constants for using SHA-256
    private final String HmacSHA256 = "HmacSHA256";

    /**
     * Generate Hmac Signature
     *
     * @param secretKey - Use T_APPLICATION's APPLICATION_SECRET column for secret key if
     * before login, the secret key of session for secret key if after login.
     * @param httpMethod - Only use GET, POST, PUT, PATCH, DELETE
     * @param datetime - yyyyMMddHHmmss(ex. 20210514213014)
     * @param url (without domain) - ex (/api/v1/invoices)
     *
     * @return signature
     */
    public String generate(String secretKey, String dateTime, String httpMethod, String url)
        throws Exception {

        SecretKeySpec key = new SecretKeySpec(secretKey.getBytes(), HmacSHA256);
        Mac mac = Mac.getInstance(HmacSHA256);
        mac.init(key);

        String value = dateTime + httpMethod + url;
        byte[] rawHmac = mac.doFinal(value.getBytes(Charset.forName(charsetName)));
        String hmac = "Bearer " + Base64.getEncoder().encodeToString(rawHmac);

        return hmac;
    }
}
```

7. API Specification

7.1 Authentication

API that receives Authentication Token for an external linkage API call

In order to call Invoice issuance, inquiry result API, the Authentication token should be issued by calling the API. An issued token is valid for 6 hours, there is no need to call this API again unless it expires.

7.1.1 Path

- Method: POST
- URL: /api/{version}/authentication

Example Endpoint

<https://eis-api.bir.gov.ph/v2/authentication>

7.1.2 Request Parameters

7.1.2.1 Header

No.	Attribute	Type	Required	Description
1	accreditationId	String	Y	Accreditation ID issued by the EIS Accreditation website. It consists of 8digit number characters. The first 2digit of accreditation id indicates issuance system type. <ul style="list-style-type: none"> - EIS Web portal: 10 - CAS: 20 - POS: 31 - CRM: 32 - ASP-CAS: 51 - ASP-POS: 52
2	applicationId	String	Y	Application ID created by the EIS Accreditation website.
3	authorization	String	Y	HMAC signature generated using the application secret key created by the EIS Accreditation website
4	datetime	String	Y	It is used for HMAC authentication and must be within 10 minutes of the time received from the server. (Format: yyyyMMddHHmmss)

7.1.2.2 Body

No.	Attribute	Level	Type	Required	Description
1	data	1	String	Y	JSON Data encrypted using RSA algorithm - Encrypted with EIS Public Key - EIS Public Key is shared by the EIS Accreditation website
2	forceRefreshToken	1	Boolean	N	Whether to force authentication token refresh

7.1.2.3 Request Example

```
{
  "data": "eneAZDnJP/5B6r/X6RyAlP3Jv6ZnnC1CzxENxpgpGKMjPK+HbfHK/xp4dArKeqeH6+rI/oCdI74afQrdpg801BkXc13usgGxbl1JeRYUj2VcnxbWV4NEp9yeWchpz7mmAt67fnTxfloen7MjuzE256DtnCaUqhZuSR27vGQ28X41/Q001ZsFUd+M1aGwbLHvWcTMEtB3QxLQiSsZfIbzQFUYxPeXvhGgLqbXYeF85zVx1TcatRpyzof212JuCkb9y4uZZtnvL01XzPCeenU08fbogSk6EODPOJ/PW00YIGerPyMm1G5X0BHpZ/rnblYxEYqBTbHXgDHBw7RDQz6+",
  "forceRefreshToken": true
}
```

7.1.2.4 Request “data” Attribute Specification

JSON Format				
No.	Attribute	Type	Required	Description
1	userId	String	Y	User ID of the EIS Accreditation website
2	password	String	Y	User password of the EIS Accreditation website in plain text format
3	authKey	String	Y	Authentication Secret Key: Encryption key randomly generated by the user in plaintext format (AES-256) - 32 character string

7.1.2.5 Decrypted JSON in “data” attribute

```
{
  "userId": "douzone",
  "password": "abc!123",
  "authKey": "eQNWGyPMNyEzTlYBs9Vy#Z7rJub13r^6"
}
```

7.1.3 Response Message

7.1.3.1 Response Status

Status Code	Response Message	Description
200	OK	When authentication token issuance is successful
400	Bad Request	In case the request header or body contains an incorrect value in the form or type
401	Unauthorized	In case the password or application_secret_key or HMAC signature does not match
404	Not Found	In case the user ID or application ID does not exist
500	Internal Server Error	Other errors occur

7.1.3.2 Body

No.	Attribute	Level	Type	Required	Description
1	status	1	String	Y	Authentication request status (values 1-Success and 0- Failure)
2	data	1		N	If Status is '1' - Encrypted using the AuthKey provided by the taxpayer's application (AES-256)
3	errorDetails	1		N	If Status is '0'
4	errorCode	2	String	Y	Unique error code
5	errorMessage	2	String	Y	Error description

7.1.3.3 Error Spec

E0X - Request Header / Auth Error

Error Code	Error Message	Description
E01	Invalid accreditationId	Accreditation ID that does not exist or is not registered for production environment
E02	Invalid applicationId	Invalid application ID does not exist or does not match its accreditation ID
E03	Invalid authToken	Expired or invalid authentication token
E04	Invalid HMAC Signature	The HMAC Signature in the Authorization field of the Request Header is incorrect.
E05	Invalid Datetime	The Datetime field in the Request Header is invalid.
E06	Expired EIS Key-pair	EIS Key-pair of application has been expired.
E07	Non-whitelisted IP	Request from unregistered IP in the white-list on the Accreditation system.
E08	Locked Sandbox accreditationId	If the accreditationId is under the approval process on the EIS-CERT system, or when the EIS-CERT approval or EIS-PTU is completed, the Sandbox API test can't carry out anymore.

E1X - Request Body Error

Error Code	Error Message	Description
E11	Invalid Request Body JSON	JSON Data in Request Body is invalid
E12	Invalid eisKeyId	Non-existent EIS Key-pair ID
E13	Expired eisKeyId	Expired EIS Key-pair ID
E14	Duplicated submitId	Submit ID is already exists
E15	Invalid submitId	Submit ID that does not exist or has expired after submission
E16	Not existent result document	Submit ID of non-existent processing result document

E2X - Request Data Error

Error Code	Error Message	Description
E21	It could not decrypt the "data" field	The field "data" in the request body cannot be decrypted.
E22	Invalid JSON data in the "data" field	
E23	Invalid User ID or Password	User ID does not exist or Password does not match
E24	Invalid authKey	AuthKey has an invalid format.

E99 - Internal Server Error

Error Code	Error Message	Description
E99	Internal Server Error	In case of an error due to internal server problems

7.1.3.4 Response Example

CASE 01 - Success

```
{
  "status": "1",
  "data":
  "uIBdWCRrNnpUKfd+RcYYwgHtFi6M3ggK9N+tgP36WdQBFmzU2q06Qe7vEFITwtDnmeKQ6mom5ZX3qCz/SPPshtsS3SrL9YTDVw42Rsg
ZYYK5pbl4kpR6Xqr0HjjBrUiv6PvgCtZneCqGtPAX/4EScePKIBVdKpF0m9TtNY3NjmZXvrMRYMCOjNeZDMb49vxuk0FD56XIAdVncSm
bvEP5n6OC+T7YSMPsbdJYZmNhLBobfUfpv9Uve1a5Ggv3fOnFHQZA4RCtYP4hrzHT4dJTbN11KnVwHBw7Z8F8mJZId2R0xk1K7ae6G05
8TMQ2fX1S"
}
```

CASE 02 - Failure

```
{
  "status": "0",
  "errorDetails": {
    "errorCode": "E23",
    "errorMessage": "Invalid User ID or Password"
  }
}
```

7.1.3.5 Response "data" Attribute Specification

JSON Format				
No.	Attribute	Type	Required	Description
1	accreditationId	String	Y	Accreditation ID
2	userId	String	Y	User ID
3	authToken	String	Y	Authentication token (Number+Uppercase alphabet 32character - 256bit)

4	sessionKey	String	Y	Session Secret Key - Symmetric key to use for data encryption during the session
5	tokenExpiry	String	Y	Token expiry date, Date format is 'yyyy-MM-ddThh:mm:ss'

7.1.3.6 Decrypted JSON in "data" attribute

```
{
  "accreditationId": "test accreditation id",
  "userId": "testuser",
  "authToken": "53A8CJFLEK3CE9MQ7L2X9V76TUIPZ4YU",
  "sessionKey": "WmZq4t7w!z$C&F)J@NcRfUjXn2r5u8x/",
  "tokenExpiry": "2021-06-16T12:20:00"
}
```

7.2 Invoice Issuance

API to transmit/send invoices issued by taxpayers' system (CAS/CRM/POS) to the EIS system for single up to 100 cases.

Return acknowledgment message (success/failure) immediately in JSON data for the call.

7.2.1 Path

- Method: POST
- URL: /api/{version}/invoices

Example Endpoint

<https://eis-api.bir.gov.ph/v2/invoices>

7.2.2 Request Parameters

7.2.2.1 Header

No.	Attribute	Type	Required	Description
1	accreditationId	String	Y	Accreditation ID issued by the EIS Accreditation website
2	applicationId	String	Y	Application ID created by the EIS Accreditation website
3	authToken	String	Y	Authentication token received as a result of Authentication API
4	authorization	String	Y	HMAC signature generated using the session secret key received as a result of Authentication API
5	datetime	String	Y	It is used for HMAC authentication and must be within 10 minutes of the time received from the server. (Format: yyyyMMddHHmmss)

7.2.2.2 Body

No.	Attribute	Level	Type	Required	Description
1	submitId	1	String	Y	ID to differentiate submission units when submitting one invoice or multiple invoices at once - Give unique value for each submission unit - Used as a key-value when checking the processing result later - EIS_Accreditation_ID + '-' + {YYYYMMDD} + '-' + (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f) 12 repetitions of the randomly selected one - Example>12345678-20090325-be3cfa2c0b1e
2	data	1	String	Y	AES256 Encrypted Invoice JSON in JWS format (1-100) Encrypted with the provided Session Secret Key when issuing the authentication token.

7.2.2.3 Request Example

```
{
  "submitId": "12345678-20210325-be3cfa2c0b1e",
  "data":
    "cwMtQm1x/upJuEqKfTMTQ6RH1s6SOQwm9SknTMBvHm9CFEjCCs4otZZbdf76sP/xpR4K3H5qv1fFmSrIFw1kmcPG1fSrE+AhXZjicr7dFsFmngugxwTu4BScAdA75FGto1VbvDnDgvzJygmDwXmIM3Mkbt7by5XC9CqVqqhnPSri9R5UE3PJgVKcMsM3aBewBdNGiQ/2Z9FI TZwCp1l/z6h6pMDd93GT30I8nxMsn3frJfUJZF+VW0LDdaHRYZi06mE3oBELTKgLvH6QCLyaa1F+UtwjVCWNci1nI10Lcr8fyZg0b8hqe 4BoVRxKscK128+69NuI3QI1vPbYwrqz+q1+vhK7b6gMT7tOT+8us/LvTSEFdD501jFD8PcB0uqHPeeHRNyYc1n9M0yEewWwue+JoXH PlWT4RVQsJdcaEVC2EdXoYdHQY8t+PmXCAOpdHnqLVuHMLdHyDvdsknzY35Sdvpba4MgYu40VMyoPYIq1XJB39/P0CiQR6IA1S/YY7 6ndFB+QqJ1In3ayvuw/wD4K6lY9L86T0Sa4ssF0e1F75axL8E9yU02aAJhphafF+Syhj5sLR/mRxxRd6s5y01mL5Vthw6ua+KRRY4IwG P9dJgmJ/Tbnf3FT413oFyG/At5oR5LHf2ckndDZ2Jbriv4zLBWewhYKTeYvPOU60K+terW/Y22k7FwFrwpZpdeUdQ0j9WAMEL7qhW5WAMeL7qhW5 RVY1K1Be3/fp+qG0HFETaf/Z50d4aqztp8NH7jE5jQqFyyRvN0iSv9ZFatJERqTp7+/sL4q2XEkcICqpbQ7ShpqeNMvaqP3D1z4zhBZE rVZONvSVj0bt1eHexIPibMYNeuIFevULQBchd10xwqrZiZyyi+5224N53EVjys8nPiw2ZFBVcEviNsOwvb9nEw9PqD2iMFvTTFH6hWM0 feEeL7R0n5S3q3bPX1D+RugS0xgdghwhhFLiP68XhphWL0jI/74NmF8vNUIAwvJnZGB1cmOgW9oHiNNszuhHe/DydLg1lQaCtSLCcA+Ee e0GhQJ0CULVNDrwbVwP2N5nVajdzPOU60K+terW/Y22k7FwFrwpZpdeUdQ0j9WAMEL7qhW5SRVY1K1Be3/fp+qG0HFETafZy1k8+sXFPi S6AopkEaXLXzhKJI0alapVD4nw6axcAoQQZeBugsfb/EyycgL7Zc8xUSGJj1XZQQcSReiKaNiG2wbmrGn1jDJgh0GUjMRbI7p+AB1csJ c77BuXC8yYLXkxuGj2Q0Q/mpRde7bKxerXV6DGBRhwQ3ISs9DJgx9009OmUi6RRaonFed131B09ajZt8LJ7DFc8xn+Ty131xAGE0v2Q qxaaIi24dzxhZSLwAtpcED46mWw8og2Y1LNky0L1pxtyIj7rIDc8QQQwh/oVbL2gR9K0W1E6yw07FLD0Mdel8xQyMgunD7+YHdc3k3yY vAcF4R5XG6nR6R51kwbTcZG6yheU0dirK0YfOm8z+YbHybPjP/CqiazXhQ7doyQXtZF8mZvYsLsiCS0kmtcIYZn1lvZyo5H0d842V2j 8FEPL5/GQpS5CD4+xrQ1i8c1fBZ9KQX3FthAAJ0vQ47CmMejpaCxxalJymLDrftFV50D/T0gZ11EGBrgx00QqGLe4eTkd5Gaip6Z1zj r9eHgSd8Wrt3VSs4jWwEBFIi/kLvo1yn6ZZ1RsvofmDqQSQKwCHWDmDVLwv0j4I9CPQRfX8X2FgIdqQWYUjuCff7+P/Qz/DiPND91Qor NEcSbvpaFdf6EbkBF LG3fijjNBRAbEzu+dtKk6UuedshXtaoBiITScd+wM09y+bmQLLCzYmY2FsJb6Eprrr+CxM2FJGca+03Iu3ABJQVE wylYI0Dia7jMZj07KR/SBD0DXCImYurvszJKiNkHQ5AsAtkyldq1MQ4L1PzvHJHwy4hs7AhrPmztKNFNE2Sg+DqMfsEGYq99j6Znsew 1CI/dbY3bqCVB3jntRcJr5QVFOVW15NE73tC55MR2Rj6H8wy70QX9hdeKie4EVK01XivMu0Sypc60AadwqZp3PfgiuI3UYFberHhWtE 8meMnJov15VjbTRBq9SHyJLGDHCoDM9Gq0x00LIK4C/SjS8hvKX18Gw87aXITo1a1Uhd0LUq6ULvE2CM0eL6tzlR07DgLFyv+2Th9Ajj wC1k3+LPjiEDRS6Xsiod2XzL21AAQGuDJaPjNQFDjs4INDjV7mFah1NQva7Srbbknz0A77wdc/HBRrk5TggcYGgm6Uaid1AQEjJntT88 tP11dZaGvqIdzKLxDQDDALx7JHGmlalGOpE4su9gFHFq+PV1k+VBUp5m9t0YpCHgl557+nk+Bw3Q2Dh83ULxFuYYPN6M4di+kGiZa/k Ncp13A0ebUPMktNmiiTPFRJoTvJPybFCrtD3cy6C1fiBdNY0v9pJ1zhUgYk/NUyMmk9I1oIQZzdFAueiGFwP20fTrCcRnh4H6ABSQ4+3eZ 5g7Z1RVUit5Tkvw8+7bPnRKQJ5CYtA5pmppYtK4ris4I3Wj/yvMbazHaLZfus00F8XMAg7C/E8Jen16HEb4CceTfPeAw4ank5gVM0vz1C G/DvTMxMQCqG5Zt1I yV9dWmPlE80btJkz4k/CzRC/7yN7g=="
```

7.2.2.4 Request “data” Attribute Specification

Refer to e-invoice JSON format

7.2.2.5 Decrypted JSON in “data” attribute

String value that each e-invoice JSON (JWS format) is separated by a delimiter (,), e-invoice JSON is allowed to add a minimum of 1 and a maximum of 100

e-invoice JSON JWS1,e-invoice JSON JWS2,...,e-invoice JSON JWS100

7.2.3 Response Message

7.2.3.1 Response Status

Status Code	Response Message	Description
200	OK	When invoice receiving is successful
400	Bad Request	In case the request header or body contains an incorrect value in the form or type
401	Unauthorized	In case the HMAC signature does not match
404	Not Found	In case the accreditation ID or application ID does not exist
500	Internal Server Error	Other errors occur

7.2.3.2 Body

No.	Attribute	Level	Type	Required	Description
1	status	1	String	Y	Authentication request status (values 1-Success and 0- Failure)
2	data	1		N	If Status is '1' - Encrypted with the provided Session Secret Key when issuing the authentication token. (AES-256)
3	errorDetails	1		N	If Status is '0'
4	errorCode	2	String	Y	Unique error code
5	errorMessage	2	String	Y	Error description

7.2.3.3 Error Spec

Refer to 7.1.3.3. Error Spec

7.2.3.4 Response Example

CASE 01 - Success

```
{
  "status": "1",
  "data":
  "uIBdWCRrNnpUKfd+RcYYwgHtFi6M3ggK9N+tgP36WdQBFmZU2q06Qe7vEFITwtDnmeKQ6mom5ZX3qCz/SPPshtsS3SrL9YTDVw42Rsg
ZYYK5pbL4kpR6Xqr0HjjBrUiv6PvgCtZneCqGtPAx/4EScePKIBVdKpF0m9TtNY3NjmZXvrMRymCOjNeZDMb49vxuk0FD56XIAdVncSm
bvEP5n60C+T7YSMPsbDJYZmNhLBobfUfpv9Uve1a5Ggv3fOnFHQZA4RCtYP4hrzHT4dJTbN11KnVwHBw7Z8F8mJZId2R0xk1K7ae6G05
8TMQ2fX1S"
}
```

CASE 02 - Failure

```
{
  "status": "0",
  "errorDetails": {
    "errorCode": "E03",
    "errorMessage": "Invalid authToken"
  }
}
```

7.2.3.5 Response “data” Attribute Specification

JSON Format				
No.	Attribute	Type	Required	Description
1	accreditationId	String	Y	Accreditation ID
2	userId	String	Y	User ID
3	refSubmitId	String	Y	Transmitted submit ID
4	ackId	String	Y	Receiving acknowledgment ID - 'BIR' + '-' + yyyyMMddhhmmss' + '-' + 00000(5 digit random value in the combination of numbers and uppercase letters)
5	responseDtm	String	Y	Response datetime, Date format is 'yyyy-MM-ddThh:mm:ss'
6	description	String	Y	Note

7.2.3.6 Decrypted JSON in "data" attribute

```
{
  "accreditationId": "test accreditation id",
  "userId": "testuser",
  "refSubmitId": "12345678-20210325-be3cfa2c0b1e",
  "ackId": "BIR-20210325154527-E8QN3",
  "responseDtm": "2021-03-25T15:45:28",
  "description": "Transmission has been successfully processed. EIS will process the final registration within the next business day."
}
```

7.3 Inquiry Result

The API that returns the processing result document for the invoice issuance with Submit ID

The response of the API is

When the requested Submit ID is still on the processing of registration and verification: return the response 'processing'.

When the requested Submit ID is completed with the registration and verification: return the JSON data.

7.3.1 Path

- Method: GET
- URL: /api/{version}/invoice_result/{submitId}

Example Endpoint

https://eis-api.bir.gov.ph/v2/invoice_result/12345678-20090325-be3cfa2c0b1e

7.3.2 Request Parameters

7.3.2.1 Header

No.	Attribute	Type	Required	Description
1	accreditationId	String	Y	Accreditation ID issued by the EIS Accreditation website
2	applicationId	String	Y	Application ID created by the EIS Accreditation website
3	authToken	String	Y	Authentication token received as a result of Authentication API
4	authorization	String	Y	HMAC signature generated using the session secret key received as a result of Authentication API
5	datetime	String	Y	It is used for HMAC authentication and must be within 10 minutes of the time received from the server. (Format: yyyyMMddHHmmss)

7.3.2.2 URL Parameter

No.	Attribute	Level	Type	Required	Description
1	submitId	1	String	Y	Submit ID used when transmitting e-invoice

7.3.2.3 Request Example

```
https://eis-api.bir.gov.ph/v2/invoice_result/12345678-20090325-be3cfa2c0b1e
```


7.3.3 Response Message

7.3.3.1 Response Status

Status Code	Response Message	Description
200	OK	When returning processing result is successful
400	Bad Request	In case the request header or body contains an incorrect value in the form or type
401	Unauthorized	In case the HMAC signature does not match
404	Not Found	In case the submit ID or application ID does not exist
500	Internal Server Error	Other errors occur

7.3.3.2 Body

No.	Attribute	Level	Type	Required	Description
1	status	1	String	Y	Authentication request status (values 1-Success and 0- Failure)
2	data	1		N	If Status is '1'
3	errorDetails	1		N	If Status is '0'
4	errorCode	2	String	Y	Unique error code
5	errorMessage	2	String	Y	Error description

7.3.3.3 Error Spec

Refer to 7.1.3.3. Error Spec

7.3.3.4 Response Example

CASE 01 - Success

```
{
  "status": "1",
  "data": {
    "accreditationId": "test accreditation id",
    "userId": "testuser",
    "refSubmitId": "12345678-20210325-be3cfa2c0b1e",
    "ackId": "BIR-20210325154527-12345",
    "responseDtm": "2021-03-25T15:45:28",
    "processStatusCode": "01",
    "totalCountQuantity": 3,
    "successCountQuantity": 2,
    "failCountQuantity": 1,
    "processedDocuments": [
      {"invoiceUid": "202103251234567800000001", "resultStatusCode": "SUC001"},
      {"invoiceUid": "202103251234567800000002", "resultStatusCode": "SUC001"}
    ]
  }
}
```

```
{
  "invoiceUid": "202103251234567800000003", "resultStatusCode": "SYN004", "description":
  "Detailed error message"
}
```

CASE 02 - Failure

```
{
  "status": "0",
  "errorDetails": {
    "errorCode": "E03",
    "errorMessage": "Invalid authToken"
  }
}
```

7.3.3.5 Response “data” Attribute Specification

JSON Format					
No.	Attribute	Level	Type	Required	Description
1	accreditationId	1	String	Y	Accreditation ID
2	userId	1	String	Y	User ID
3	refSubmitId	1	String	Y	Transmitted submit ID
4	ackId	1	String	Y	Receiving acknowledgment ID - 'BIR' + '-' + yyyyMMddhhmmss' + '-' + 00000(Sequence)
5	responseDtm	1	String	Y	Date format is 'yyyy-MM-ddThh:mm:ss'
6	processStatusCode	1	String	Y	Processing status code 01: Processing completed 02: Processing in progress 03: Processing not available
7	failReasonStatusCode	1	String	N	Fail reason status code - Required when processStatusCode is '03'
8	totalCountQuantity	1	Number	N	Total submitted invoice count - Required only when processStatusCode is '01'
9	successCountQuantity	1	Number	N	Processing success invoice count - Required only when processStatusCode is '01'
10	failCountQuantity	1	Number	N	Processing failure invoice count - Required only when processStatusCode is '01'
11	processdDocuments	1		N	Processed document list (JSON Array) - Required only when processStatusCode is '01'

12	invoiceUid	2	String	Y	EIS Unique ID
13	resultStatusCode	2	String	Y	Verification result status code
14	description	2	String	N	Detailed error message - Required only if an error occurred

7.3.3.6 Inner JSON of “data” from Response Sample

CASE 01 – In case processStatusCode is ‘01’ (Processing completed)

```
{
  "accreditationId": "test accreditation id",
  "userId": "testuser",
  "refSubmitId": "12345678-20210325-be3cfa2c0b1e",
  "ackId": "BIR-20210325154527-12345",
  "responseDtm": "2021-03-25T15:45:28",
  "processStatusCode": "01",
  "totalCountQuantity": 3,
  "successCountQuantity": 2,
  "failCountQuantity": 1,
  "processedDocuments": [
    {"invoiceUid": "202103251234567800000001", "resultStatusCode": "SUC001"},
    {"invoiceUid": "202103251234567800000002", "resultStatusCode": "SUC001"},
    {"invoiceUid": "202103251234567800000003", "resultStatusCode": "SYN004", "description":
      "Detailed error message"}
  ]
}
```

CASE 02 – In case processStatusCode is '02' (Processing in progress)

```
{
  "accreditationId": "test accreditation id",
  "userId": "testuser",
  "refSubmitId": "12345678-20210325-be3cfa2c0b1e",
  "ackId": "BIR-20210325154527-12345",
  "responseDtm": "2021-03-25T15:45:28",
  "processStatusCode": "02"
}
```

CASE 03 – In case processStatusCode is '03' (Processing not available)

```
{
  "accreditationId": "test accreditation id",
  "userId": "testuser",
  "refSubmitId": "12345678-20210325-be3cfa2c0b1e",
  "ackId": "BIR-20210325154527-12345",
  "responseDtm": "2021-03-25T15:45:28",
  "processStatusCode": "03",
  "failReasonStatusCode": "FRS002"
}
```

7.3.3.7 Processing Result Code

7.3.3.7.1 Process Status Code

Code Name	Code Value	Definition	Description
ProcessStatusCode	01	Processing completed	When all verification of e-invoices submitted by the Taxpayer has completed, and e-invoices have been saved in BIR
	02	In processing	When verification of e-invoices submitted by Taxpayer is in progressing
	03	Unable to process	When verification task of e-invoices of submitted by Taxpayer is impossible because of decryption error, etc.

7.3.3.7.2 Fail Reason Status Code

Code Name	Code Value	Definition	Description
FailReasonCode	FRS001	Not exists submitId	If Taxpayer requested inquiry result API by non-existence submit-ID.
	FRS002	Decryption failure	If an AES-256 encrypted invoice JSON data submitted by Taxpayer is unable to decrypt.

7.3.3.7.3 Process Result Status Code

Code Name	Code Value	Definition	Description
ResultStatusCode	SUC001	Success	If the submitted data has passed all verification processes.
	SYN002	Invalid signature	If the digital signature of e-invoice is not valid.
	SYN003	Duplicated EisUniqueId	If the EisUniqueId of submitted e-invoice is already registered as a normal e-invoice in the BIR.
	SYN004	e-invoice schema error	In case of invalid e-invoice structure (field redundancy, non-existence of required fields, code errors, data type errors, various number format errors, etc.)
	ERR001	Seller TIN error	The seller's TIN is not registered in BIR
	ERR002	Issuance datetime error	<ul style="list-style-type: none"> ● If the Issuance datetime is invalid ● If the Issuance datetime is after the Transmission datetime ● If the Issuance date in EisUniqueId is different from Issuance date.
	ERR003	Transmission datetime error	<ul style="list-style-type: none"> ● If the Transmission date is invalid
	ERR004	Total Sales Amount, VAT Amount error	<ul style="list-style-type: none"> ● If the plus/minus sign of Total sales amount value and the VAT amount is different ● If VAT amount value is not '0' when the invoice tax type is 'Exempt' or 'Zero-rated'.
	ERR005	Code type error	If there are errors in various classification codes.

	ERR006	Accreditation ID error	If the Accreditation ID in the EisUniqueID is different from the Accreditation ID of the Taxpayer's transmission system.
	ERR007	E-invoice Unique ID to be corrected error	<ul style="list-style-type: none"> ● If the PrevUniqueID entered in the corrected e-invoice does not exist or is not the seller's EisUniqueID. ● If the PrevUniqueID is entered when e-invoice is a general invoice.
	ERR999	Other undefined error	Other errors that do not correspond to the errors listed above.

7.4 Invoice Result Callback

The API to deliver results to the taxpayer for the Invoice issuance process per-Submit ID.

It is a callback REST API that is configured in the Taxpayer server if the taxpayer wants to receive the JSON data of the processing result from the EIS as soon as the invoice registration and verification processing is completed in the Taxpayer system (Optional)

7.4.1 Path

- Method: PUT
- URL: {Taxpayer URL}/api/{version}/receive_invoice_result

Example Endpoint

https://eis-linkage.large_company.com/api/v1/receive_invoice_result

7.4.2 Request Parameters

7.4.2.1 Header

No.	Attribute	Type	Required	Description
1	accreditationId	String	Y	Accreditation ID issued by the EIS Accreditation website
2	applicationId	String	Y	Application ID created by the EIS Accreditation website
3	authorization	String	Y	HMAC signature generated using the application secret key created by the EIS Accreditation website
4	datetime	String	Y	It is used for HMAC authentication and must be within 10 minutes of the time received from the server. (Format: yyyyMMddHHmmss)

7.4.2.2 Body

No.	Attribute	Level	Type	Required	Description
1	accreditationId	1	String	Y	Accreditation ID
2	userId	1	String	Y	User ID

3	refSubmitId	1	String	Y	Transmitted submit ID
4	ackId	1	String	Y	Receiving acknowledgment ID - 'BIR' + '-' + yyyyMMddhhmmss' + '-' + 00000(Sequence)
5	responseDtm	1	String	Y	Date format is 'yyyy-MM-ddThh:mm:ss'
6	processStatusCode	1	String	Y	Processing status code 01: Processing completed 02: Processing in progress 03: Processing not available
7	failReasonStatusCode	1	String	N	Fail reason status code - Required when processStatusCode is '03'
8	totalCountQuantity	1	Number	N	Total submitted invoice count - Required only when processStatusCode is '01'
9	successCountQuantity	1	Number	N	Processing success invoice count - Required only when processStatusCode is '01'
10	failCountQuantity	1	Number	N	Processing failure invoice count - Required only when processStatusCode is '01'
11	processdDocuments	1		N	Processed document list (JSON Array) - Required only when processStatusCode is '01'
12	invoiceUid	2	String	Y	EIS Unique ID
13	resultStatusCode	2	String	Y	Verification result status code
14	description	2	String	N	Detailed error message - Required only if an error occurred

7.4.2.3 Request Example

CASE 01 – In case processStatusCode is '01' (Processing completed)

```
{
  "accreditationId": "test accreditation id",
  "userId": "testuser",
  "refSubmitId": "AB123456-20200512-be3cfa2c0b1e",
  "ackId": "BIR-20210325154527-12345",
  "responseDtm": "2021-03-25T15:45:28",
  "processStatusCode": "01",
  "totalCountQuantity": 3,
  "successCountQuantity": 2,
  "failCountQuantity": 1,
  "processedDocuments": [
    {"invoiceUid": "202103251234567800000001", "resultStatusCode": "S001"},
    {"invoiceUid": "202103251234567800000002", "resultStatusCode": "S001"},
    {"invoiceUid": "202103251234567800000003", "resultStatusCode": "E001", "description":
      "Unregistered Seller TIN"}
  ]
}
```

CASE 02 – In case processStatusCode is '03' (Processing not available)

```
{
  "accreditationId": "test accreditation id",
  "userId": "testuser",
  "refSubmitId": "12345678-20210325-be3cfa2c0b1e",
  "ackId": "BIR-20210325154527-12345",
  "responseDtm": "2021-03-25T15:45:28",
  "processStatusCode": "03",
  "failReasonStatusCode": "PR001"
}
```

7.4.3 Response Message

7.4.3.1 Response Status

Status Code	Response Message	Description
200	OK	When authentication token issuance is successful
400	Bad Request	In case the request header or body contains an incorrect value in the form or type
401	Unauthorized	In case HMAC signature does not match
404	Not Found	In case the accreditation ID or application ID or refSubmitId does not exist
500	Internal Server Error	Other errors occur

7.4.3.2 Body

No.	Attribute	Level	Type	Required	Description
1	status	1	String	Y	Authentication request status (values 1-Success and 0- Failure)
2	data	1		N	If Status is '1' - Encrypted using the AuthKey provided by the taxpayer's application (AES-256)
3	errorDetails	1		N	If Status is '0'
4	errorCode	2	String	Y	Unique error code
5	errorMessage	2	String	Y	Error description

7.4.3.3 Error Spec

Refer to 7.1.3.3. Error Spec

7.4.3.4 Response Example

CASE 01 - Success

```
{
  "status": "1",
  "data": {
    "accreditationId": "test accreditation id",
    "userId": "testuser",
```

```

    "refSubmitId": "12345678-20210325-be3cfa2c0b1e",
    "taxpayerAckId": "20210325-20210601135218-12345",
    "description": "Test description message"
  }
}

```

CASE 02 - Failure

```

{
  "status": "0",
  "errorDetails": {
    "errorCode": "E15",
    "errorMessage": "Invalid submit ID"
  }
}

```

7.4.3.5 Response “data” Attribute Specification

JSON Format				
NO	Attribute	Type	Required	Description
1	accreditationId	String	Y	Accreditation ID
2	userId	String	Y	User ID
3	refSubmitId	String	Y	Transmitted Submit ID
4	taxpayerAckId	String	Y	Taxpayer's receiving acknowledgment ID - {AccreditationID} + '-' + yyyyMMddhhmmss' + '-' + 00000(Sequence)
5	description	String	Y	Note

8. API interface example

The process of creating an invoice through the API interface with an external system, sending it to the EIS, and checking the registration result is described in the following order through examples.

8.1 Authentication API Call

First, in order to use the API of EIS, you must first obtain an authentication token. To issue an authentication token, it can be issued by calling the Authentication API.

The following is a Java example of the code for issuing an authentication token.

8.1.1 Temporary Secret Key Generation

[Code Sample - Java]

```
import java.util.Random;
import java.util.stream.IntStream;

public class AuthKey {
    private final int length = 32;
    private final char[] base71chars =
        "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz@#$$%^&*()"
        .toCharArray();

    public String generate() {
        Random random = new Random();
        StringBuffer sb = new StringBuffer(length);

        IntStream.rangeClosed(1, length).forEach(len ->
            sb.append(base71chars[random.nextInt(base71chars.length)])
        );

        return sb.toString();
    }
}
```

8.1.2 Application authentication information asymmetric key encryption

[Code Sample - Java]

```
import java.security.KeyFactory;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

import javax.crypto.Cipher;

import com.fasterxml.jackson.databind.ObjectMapper;

public class Data {

    // EIS Public Key String
    private String publicKey
    ="MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEALUCQZso6P43gKqw0CFTlwYb3N+m4v6IME4nPA3Wxe52wFpDM/JCFW
    SdXa7BewlwzDYjb1glwL4u59CPxNTPTh7LTD4xX0aGDJHjX5+YgqK4fb9rsImjMpIACrND/LAdrq5mctWwz3Utw3F+o+sNwIZM8
    n65ysS+Vhq9IypFlfuQbWrKjAcwZ3u1iLtplzyf/pjh0EyyZiBUUnh6D219+pMiE9nhCpc4xkH1gnlGszIDBqZMMULtGJvFXydA1
    vv5HxxCYJ2ydEzmAKYxVgA9BGXPEGE89dQbeJsieTj+FSsp9oTm+4vi345opRvH8DWhmZc4OPSwBEL8pwgS7cUnKPtwIDAQAB";

    private String userId = "user_id"; // EIS Accreditation system user id
    private String password = "user_password"; // EIS Accreditation system user password
    private String authKey = AuthKey.generate();
}
```

```

public String generate() throws Exception {
    // Data converts to JSON
    Map<String, String> map= new HashMap<>();
    map.put("userId", userId);
    map.put("password", password);
    map.put("authKey", authKey);
    String json = new ObjectMapper().writeValueAsString(map);

    // Decode PublicKey
    byte[] decoedPublicKey = Base64.getDecoder().decode(publicKey);

    // PublicKey converts to RSAPublicKey
    RSAPublicKey rsaPublicKey = (RSAPublicKey) KeyFactory.getInstance("RSA").
        generatePublic(new X509EncodedKeySpec(decoedPublicKey));

    // Specify RSA Algorithm
    Cipher cipher = Cipher.getInstance("RSA");
    // Set to Encryption Mode using RSAPublicKey
    cipher.init(Cipher.ENCRYPT_MODE, rsaPublicKey);
    // Cipher converts content to byte array
    byte[] bytePlain = cipher.doFinal(json.getBytes());
    // Encrypt content
    String encryptedData = Base64.getEncoder().encodeToString(bytePlain);

    return encryptedData;
}
}

```

8.1.3 Request Authentication API

[Code Sample - Java]

```

import com.fasterxml.jackson.databind.ObjectMapper;
import com.sample.eis.entity.Hmac;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.HttpClients;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.Map;

public class Authentication {
    private final ObjectMapper objectMapper = new ObjectMapper();

    private final String charsetName = "UTF-8";
    // Authentication API endpoint URL
    private final String url = "https://SAMPLE_API_URL/api/authentication";
    private final String uri = "/api/authentication";
    private final String contentType = "application/json; charset=utf-8";

    private final String accreditationId = "20146IT4"; // Sample Accreditation ID
    private final String applicationId = "60xLDIL1"; // Sample Application ID
    // Whether to force authentication token refresh (Optional)
    private final String forceRefreshToken = "false";
    private final String secretKey = "X8TD8UUEDTK7"; // Sample Secret Key

    /**
     * Request Authentication API
     */
    public HttpResponse send() throws Exception {
        HttpClient httpClient = HttpClients.createDefault();

```

```

// Set Request URL
HttpPost httpPost = new HttpPost(url);

// Set Request Header
String dateTime = LocalDateTime.now().format(
    DateTimeFormatter.ofPattern("yyyyMMddHHmmss")); // Asia/Manila
httpPost.addHeader("datetime", dateTime);

String hmac = new Hmac().generate(secretKey, dateTime, httpPost.getMethod(), uri);
httpPost.addHeader("authorization", hmac);

httpPost.addHeader("accreditationId", accreditationId);
httpPost.addHeader("applicationId", applicationId);
httpPost.addHeader("Content-Type", contentType);

// Set Request body
Map<String, String> map= new HashMap<>();
map.put("data", new com.sample.eis.entity.authentication.Data().generate());
map.put("forceRefreshToken", forceRefreshToken);
String json = objectMapper.writeValueAsString(map);

// Set Request Body
httpPost.setEntity(new StringEntity(json, charsetName));

return httpClient.execute(httpPost);
}
}

```

8.2 Sending invoices by the invoice issuance API

8.2.1 Generate issued invoice with e-invoice JWS

Converts SI/OR issued from CAS or CRM/POS, etc. to e-invoice JSON. The encoding of the JSON file must be set to UTF-8, and since JSON is a plain text file, it can be created without a separate tool.

8.2.1.1 EIS Unique Invoice ID numbering

EIS Unique Invoice ID is a string consisting of 24 digits and letters, and it is composed of combinations of the following elements.

1. Transaction Date (YYYYMMDD)
- + 2. Accreditation ID (2 digit + 6 digit)
- + 3. Sequence Or Unique Random Key (8 alphanumeric [26 upper case letters and 10 digits])

8.2.1.2 Generate Invoice JSON

The following is a code sample that generates invoice JSON using Java.

[Code Sample - Java]

```

import com.fasterxml.jackson.annotation.JsonAutoDetect;
import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.*;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

```

```

import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.stream.IntStream;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@JsonAutoDetect(fieldVisibility = JsonAutoDetect.Visibility.ANY)
public class CAS {

    // Management Information
    @JsonProperty("CompInvoiceId")
    private String compInvoiceId;
    @JsonProperty("IssueDtm")
    private String issueDtm;

    // Basic Information
    @JsonProperty("EisUniqueId")
    private String eisUniqueId;
    @JsonProperty("DocType")
    private String docType;
    @JsonProperty("TransClass")
    private String transClass;
    @JsonProperty("CorrYN")
    private String corrYN;
    @JsonProperty("CorrectionCd")
    private String correctionCd;
    @JsonProperty("PrevUniqueId")
    private String prevUniqueId;
    @JsonProperty("Rmk1")
    private String rmk1;

    @JsonProperty("SellerInfo")
    private Seller sellerInfo;
    @JsonProperty("BuyerInfo")
    private Buyer buyerInfo;
    @JsonProperty("ItemList")
    private List<Item> itemList;
    @JsonProperty("TotNetItemSales")
    private BigDecimal totNetItemSales;
    @JsonProperty("Discount")
    private Discount discount;
    @JsonProperty("OtherTaxRev")
    private BigDecimal otherTaxRev;

    @JsonProperty("TotNetSalesAftDisct")
    private BigDecimal totNetSalesAftDisct;
    @JsonProperty("VATAmt")
    private BigDecimal vatAmt;
    @JsonProperty("WithholdIncome")
    private BigDecimal withholdIncome;
    @JsonProperty("WithholdBusVAT")
    private BigDecimal withholdBusVat;
    @JsonProperty("WithholdBusPT")
    private Double withholdBusPt;
    @JsonProperty("OtherNonTaxCharge")
    private BigDecimal otherNonTaxCharge;
    @JsonProperty("NetAmtPay")
    private BigDecimal netAmtPay;

    @JsonProperty("ForCur")
    private ForCur forCur;
    @JsonProperty("PtuNum")
    private String ptuNum;

    @Getter
    @Setter

```

```

@Builder
public static class Seller {
    @JsonProperty("Tin")
    private String tin;
    @JsonProperty("BranchCd")
    private String branchCd;
    @JsonProperty("Type")
    private String type;
    @JsonProperty("RegNm")
    private String regNm;
    @JsonProperty("BusinessNm")
    private String businessNm;
    @JsonProperty("Email")
    private String email;
    @JsonProperty("RegAddr")
    private String regAddr;
}

@Getter
@Setter
@Builder
public static class Buyer {
    @JsonProperty("Tin")
    private String tin;
    @JsonProperty("BranchCd")
    private String branchCd;
    @JsonProperty("RegNm")
    private String regNm;
    @JsonProperty("BusinessNm")
    private String businessNm;
    @JsonProperty("Email")
    private String email;
    @JsonProperty("RegAddr")
    private String regAddr;
    @JsonProperty("DevAddr")
    private String devAddr;
    @JsonProperty("AirNum")
    private String airNum;
    @JsonProperty("AirNumDt")
    private String airNumDt;
    @JsonProperty("LadNum")
    private String ladNum;
    @JsonProperty("LadNumDt")
    private String ladNumDt;
}

@Getter
@Setter
@Builder
public static class Item {
    @JsonProperty("Nm")
    private String nm;
    @JsonProperty("Desc")
    private String desc;
    @JsonProperty("Qty")
    private Long qty;
    @JsonProperty("Unit")
    private String unit;
    @JsonProperty("UnitCost")
    private BigDecimal unitCost;
    @JsonProperty("SalesAmt")
    private BigDecimal salesAmt;
    @JsonProperty("RegDscntAmt")
    private BigDecimal regDscntAmt;
    @JsonProperty("SpeDscntAmt")
    private BigDecimal speDscntAmt;
    @JsonProperty("NetSales")
    private BigDecimal netSales;
}

```

```

@Getter
@Setter
@AllArgsConstructor
public static class Discount {
    @JsonProperty("ScAmt")
    BigDecimal scAmt;
    @JsonProperty("PwdAmt")
    BigDecimal pwdAmt;
    @JsonProperty("RegAmt")
    BigDecimal regAmt;
    @JsonProperty("SpeAmt")
    BigDecimal speAmt;
    @JsonProperty("Rmk2")
    String rmk2;

    BigDecimal getTotalDiscountAmount() {
        return this.scAmt.add(pwdAmt).add(regAmt).add(speAmt);
    }
}

@Getter
@Setter
@AllArgsConstructor
public static class ForCur {
    @JsonProperty("Currency")
    private String currency;
    @JsonProperty("ConvRate")
    private BigDecimal convRate;
    @JsonProperty("ForexAmt")
    private BigDecimal forexAmt;
}

private transient final char[] base62chars =
    "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
        .toCharArray();
private transient final int length = 8;

private String getSequence() {
    Random random = new Random();
    StringBuffer sb = new StringBuffer(length);

    IntStream.rangeClosed(1, length).forEach(len ->
        sb.append(base62chars[random.nextInt(base62chars.length)])
    );

    return sb.toString();
}

public CAS CAS(String accreditationId) {
    String currentLocalDate =
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMdd"));
    String currentLocalDateTime =
LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMddHHmmss"));

    String eisUniqueId = currentLocalDate + accreditationId + getSequence();

    Seller seller = getSampleSeller();
    Buyer buyer = getSampleBuyer();

    List<Item> itemList = getSampleItemList();
    BigDecimal totNetItemSales = itemList.stream()
        .map(item -> item.getNetSales())
        .reduce(BigDecimal.ZERO, BigDecimal::add);

    Discount discount = getSampleDiscount();
    ForCur forCur = getSampleForCur();

    BigDecimal otherTaxRev = new BigDecimal(200);
    BigDecimal totNetSalesAftDisct = totNetItemSales
        .add(otherTaxRev)

```

```

        .subtract(discount.getTotalDiscountAmount());
BigDecimal vatAmount = totNetSalesAftDisct.multiply(new BigDecimal(0.12));
BigDecimal withholdIncome = new BigDecimal(3);
BigDecimal withholdBusVat = new BigDecimal(7);
Double withholdBusPt = 0.1;
BigDecimal otherNonTaxCharge = new BigDecimal(100);
BigDecimal netAmtPay = totNetSalesAftDisct
    .add(vatAmount)
    .add(otherNonTaxCharge)
    .subtract(withholdIncome)
    .subtract(withholdBusVat);

CAS cas = CAS.builder()
    .compInvoiceId("000000001")
    .issueDtm(currentLocalDateTime)
    .eisUniqueId(eisUniqueId)
    .docType("01")
    .transClass("01")
    .corrYN("N")
    .correctionCd("") // if corrYN is "Y" then this field must not blank
    .prevUniqueId("") // if corrYN is "Y" then this field must not blank
    .rmk1("Invoice Remark 1")
    .sellerInfo(seller)
    .buyerInfo(buyer)
    .itemList(itemList)
    .totNetItemSales(totNetItemSales)
    .discount(discount)
    .otherTaxRev(otherTaxRev)
    .totNetSalesAftDisct(totNetSalesAftDisct)
    .vatAmt(vatAmount)
    .withholdIncome(withholdIncome)
    .withholdBusVat(withholdBusVat)
    .withholdBusPt(withholdBusPt)
    .otherNonTaxCharge(otherNonTaxCharge)
    .netAmtPay(netAmtPay)
    .forCur(forCur)
    .ptuNum("20146IT4-0000001")
    .build();

return cas;
}

private Seller getSampleSeller() {
    return new Seller(
        "11111111",
        "0000",
        "0",
        "Seller Registered Name",
        "Seller Business Name",
        "seller@bir.gov.ph",
        "Seller Registered Address"
    );
}

private Buyer getSampleBuyer() {
    return new Buyer(
        "22222222",
        "0000",
        "Buyer Register Name",
        "Buyer Business Name",
        "buyer@bir.gov.ph",
        "Buyer Register Address",
        "Buyer Development Address",
        "0123456789",
        "20220217",
        "9876543210",
        "20220217"
    );
}

```

```

private List<Item> getSampleItemList() {
    List<Item> itemList = new ArrayList<>();

    Item item = new Item(
        "Item 1",
        "Item Description",
        Long.valueOf(2),
        "EA",
        new BigDecimal(85490),
        new BigDecimal(170980),
        new BigDecimal(500),
        new BigDecimal(620),
        new BigDecimal(170400)
    );

    itemList.add(item);
    return itemList;
}

private Discount getSampleDiscount() {
    return new Discount(
        new BigDecimal(510),
        new BigDecimal(470),
        new BigDecimal(290),
        new BigDecimal(20),
        "Discount Remark"
    );
}

private ForCur getSampleForCur() {
    return new ForCur(
        "USD",
        new BigDecimal(950),
        new BigDecimal(90)
    );
}
}

```

8.2.2 e-invoice JSON digital signature

The generated e-invoice JSON must be digitally signed to ensure that it was created in the LT.

The digital signature is digitally signed using the key-pair issued by EIS, and the JSON standard digital signature format uses the JWS (JSON Web Signature) format.

Detailed description of JWS is provided in [6.4 JWS digital signature](#) of this document.

The following is a code sample that creates JWS by digitally signing invoice JSON generated using Java.

[Code Sample - Java]

```

import com.nimbusds.jose.*;
import com.nimbusds.jose.crypto.RSASSASigner;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.security.KeyFactory;
import java.security.interfaces.RSAPrivateKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;

```



```

public class JWS {

    public String generate(String id, String privateKey, String data) throws Exception {
        JWSHeader jwsHeader = new JWSHeader.Builder(JWSAlgorithm.RS256).keyID(id).build();
        Payload payload = new Payload(data);

        JWSObject jwsObject = new JWSObject(jwsHeader, payload);
        byte[] decodePrivateKey = Base64.getDecoder().decode(privateKey);
        RSAPrivateKey rsaPrivateKey = (RSAPrivateKey)KeyFactory.getInstance("RSA").generatePrivate(
            new PKCS8EncodedKeySpec(decodePrivateKey));
        JWSSigner signer = new RSASSASigner(rsaPrivateKey);
        jwsObject.sign(signer);

        return jwsObject.serialize();
    }

    public String encrypt(String sessionSecretKey, String jws) throws Exception {
        byte[] rawSecretKey = sessionSecretKey.getBytes();
        SecretKey secretKey = new SecretKeySpec(rawSecretKey, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

        cipher.init(Cipher.ENCRYPT_MODE, secretKey,
            new IvParameterSpec(sessionSecretKey.substring(0, 16).getBytes()));
        byte[] encrypted = cipher.doFinal(jws.getBytes("UTF-8"));

        return Base64.getEncoder().encodeToString(encrypted);
    }
}

```

8.2.3 Encrypt generated JWS using Session Secret Key

[Code Sample - Java]

```

public String encrypt(String sessionSecretKey, String jws) throws Exception {
    byte[] rawSecretKey = sessionSecretKey.getBytes();
    SecretKey secretKey = new SecretKeySpec(rawSecretKey, "AES");
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

    cipher.init(Cipher.ENCRYPT_MODE, secretKey,
        new IvParameterSpec(sessionSecretKey.substring(0, 16).getBytes()));
    byte[] encrypted = cipher.doFinal(jws.getBytes("UTF-8"));

    return Base64.getEncoder().encodeToString(encrypted);
}

```

8.2.4 Request invoice issuance API

8.2.4.1 Generate Submit ID

Submit ID is an Identifier to differentiate submission units when submitting one invoice or multiple invoices at once.

- Give unique value for each submission unit
- Used as a key-value when checking the processing result later
- EIS_Accreditation_ID + '-' + {YYYYMMDD} + '-' + (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f) 12 repetitions of the randomly selected one
- Example> 12345678-20090325-be3cfa2c0b1e

[Code Sample - Java]

```

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Random;
import java.util.stream.IntStream;

public class SubmitId {

    private final int length = 12;
    private final char[] uniqueChars = "0123456789abcdef".toCharArray();

    public String generate(String accreditationId) {
        Random random = new Random();
        StringBuffer sb = new StringBuffer(length);

        IntStream.rangeClosed(1, length).forEach(len ->
            sb.append(uniqueChars[random.nextInt(uniqueChars.length)])
        );

        return accreditationId
            + "-"
            + LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMdd"))
            + "-"
            + sb.toString();
    }
}

```

8.2.4.2 Request & Response Issuance API**[Code Sample - Java]**

```

import java.nio.charset.Charset;
import java.text.SimpleDateFormat;
import java.util.Base64;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

import javax.crypto.Cipher;
import javax.crypto.Mac;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.HttpClients;

import com.fasterxml.jackson.databind.ObjectMapper;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

public class Invoice {
    String HMAC_SHA_256 = "HmacSHA256";
    String secretKey = "#Go(YD3aiML3tTCGArQtFd(e$691c%jk"; // Sample Secret Key

    public void execute() throws Exception{
        // API Call
        HttpResponse response =this.send();

        // Get Response
    }
}

```

```

Entity entity = this.response(response);

if (entity.getData() != null){
    String data = this.decodeData(entity);
}

}

public HttpResponse send() throws Exception {
    String url = "http://SAMPLE_URL/api/invoices";
    String accreditationId = "20111111"; // Sample Accreditation ID
    String applicationId = "AAAAA"; // Sample Application ID
    // Sample Authentication Token
    String authToken = "ANGKWMVLS4PLGE6GVV0KLR14XH2F6BM";
    // Sample EIS Key-pair private key
    String jwsKey
="MIIIEvGIBADANBgkqhkiG9w0BAQEFAASCBKggggSkAgEAAoIBAQC41fMeX0jW6WZoMqLxscN5vfcmDKsj071cYb4eqEuVxNVsr
TKklEH+GNTFCMqE/LLDpWghZniPdLV4ncjgleIYZ2anOrJfm7k0K2FAHKNNAVsoZS/N8uGmBtrcdQmoeqqJ0MkK5KhXnM1cwwdb
wpfZC9gztIn9JbxbvV4k4qV9hhURGU82pV76k1ekkEKUjQ1XyentP1LKgAr877Uww8Ub83KI1ZqWkcTzV+ifhtnrX/9ydQfQhwi
EpW5txcRqv6A8atZSM6HvYKAlw8+EzCMmh9IbneDRJJiPnf0hJeVzeP0VI6M+ftEc6v04GLkyI/iXpPxL3b1GBkAeSuJCyU4Fag
MBAAECggEAevr30tGjL/MwGV4nyGssxLfh3TsZbEbXb0410NYzzSg4Gc4u+JdKkixQMwBm4xbE0u8JgT9EE4R9EfgPaY8a3S0
k+uoQigj1Pzp+mmGwH++hihPwFse8Ax3Jrw7UV4tKuzKElNbMXKqf61pbskt4I2/ugPq9xwzV6vD5E6AjK6NuTorBmzLX1Pj7Q
IUfpi/t9Y31qtIafUe1D+NxZh1cFc5qYsIj8iLUH+9PjWDR9L/i10TQAtDYMbmtPH5M1KwzBiSLyhjbfbjkYYP/hEGTm0x/031aC
BZ0L8+Jdel17jv31rvAIBHwvR9ysFejmNXyaEgaSfm3xUIKuzOooXQKBgQD5YjKXbQxsIPfkC4cqvdHGbkatu98X5vBhPSuNX8
bopMTBD1qi5+4603tjN1Hm4gg7LR1o803DNVNTvc7W9w08NwfGD0e4te47J53S4weJqXaAGjDNzL5E+kuNR3SN/YcyvRbrpIuqv
t+CwnhjJAh+fiRra+ky40ZvaHwFAoV0swKBgQC9vViPKW79FZEcj5qo9tVpn5b+z3PjDShHOYDa5ZIG17AZz5S01yTc1j0X5iHB
KP08H1gqvXW6a8jDsOQULVioGIPHUFY6aXT1l//8ywcF9wgQWKTax+dzWh4hvx5+13qES9fWPrA71Z52r7YAAgrxn3TWsM71N9k
gTJDHudw+ZwKBgQDP+46X7XFbyI8VGGwtPvEAPb4ibLndEjmxzzNAX61jHu+c1pz0mkF71BZ2ornhpYcZ20FY4P5aPGH28WxkqX
PrB7Yq/V5QFSLPIPYmCAOkUrmqkq07JdNVca2K8RoKxXN45Gx2MC/vQerkgc2nWUjgPdGf6Up+jz+R8Ht6Z6TnswKBgQC8RNZL7
5v+qli7005uW8Y1MTK9khF2eQa/bbKLAYJ27i4B3nSXJ70fITg+gA58QPHT13R6rfKccPjCCHiKAL7YxMJ1kiSNfziL3MAYEUJq
q00mFXU1NWhwkFQXvjE/AYWDI1J/duKdrP942y8nYN+FE2j24qfY0Aq1PuPlszdr/wKBGf2NFxmc4b3e/7RngWQNP0sg5bJJIR
wPpqQmR/u53Jv1r+PbstYf9W78uSCeauNHv2NF43pqNXq0QmJGq+IFIw2e990KEH494yZAWkB9nddHBLGbBCFFADgZbyojX13r
2U5vBFhxlTmdD1VrZJq6LUm2928MsxQ2pCYb3YpwuT";
    SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMddHHmmss");
    Date current = new Date();
    String dateTime = formatter.format(current);

    JWS jws = new JWS();
    CAS cas = new CAS();
    String encryptJws = jws.encrypt(
        secretKey,
        jws.generate("Akeypair00", jwsKey, cas.createSampleJson(accreditationId)));

    HttpClient httpClient = HttpClient.createDefault();
    HttpPost httpPost = new HttpPost(url);

    // HMAC
    String value = dateTime + httpPost.getMethod() + "/invoices";
    SecretKeySpec key = new SecretKeySpec(secretKey.getBytes(), HMAC_SHA_256);
    Mac mac = Mac.getInstance(HMAC_SHA_256);
    mac.init(key);
    byte[] rawHmac = mac.doFinal(value.getBytes(Charset.forName("UTF-8")));
    String signature= "Bearer " + Base64.getEncoder().encodeToString(rawHmac);

    httpPost.setHeader("authorization",signature);
    httpPost.setHeader("authToken",authToken);
    httpPost.setHeader("applicationId",applicationId);
    httpPost.setHeader("accreditationId",accreditationId);
    httpPost.setHeader("datetime",dateTime);
    httpPost.setHeader("Content-Type","application/json; charset=utf-8");

    Map<String, String> map= new HashMap<>();
    String submitId = new SubmitId().generate(accreditationId);
    map.put("submitId", submitId);
    map.put("data", encryptJws);

    ObjectMapper mapper = new ObjectMapper();
    String json = mapper.writeValueAsString(map);

    httpPost.setEntity(new StringEntity(json, ContentType.APPLICATION_JSON));

```

```

        return httpClient.execute(httpPost);
    }

    /**
     * Receive receipt information
     *
     * @param response
     */
    public Entity response(HttpResponse response) throws Exception {
        ObjectMapper mapper = new ObjectMapper();
        Entity entity = mapper.readValue(response.getEntity().getContent(), Entity.class);

        return entity;
    }

    /**
     * Deocode Data
     *
     * @param entity
     */
    public String decodeData(Entity entity) throws Exception {
        // SecretKey converts to bytearray
        byte[] secretKeyByte = secretKey.getBytes();
        // Set to SecretKeySpec to AES
        SecretKey key = new SecretKeySpec(secretKeyByte, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        // Set to DecryptionMode using AES
        cipher.init(
            Cipher.DECRYPT_MODE, key,
            new IvParameterSpec(secretKey.substring(0, 16).getBytes("UTF-8")));
        // Decode encrypted content
        byte[] byteStr = Base64.getDecoder().decode(entity.getData().getBytes());

        return new String(cipher.doFinal(byteStr), "UTF-8");
    }

    @ToString
    @Getter //lombok
    static class Entity {
        String status;
        String data;
        ErrorDetails errorDetails;
    }

    @Setter
    @Getter
    @ToString
    static class ErrorDetails {
        String errorCode;
        String errorMessage;
    }
}

```

8.3 Checking the registration result through "Invoice Issuance Result Check API"

8.3.1 Request Inquiry Result API

[Code Sample - Java]

```

import com.fasterxml.jackson.databind.ObjectMapper;
import com.sample.eis.entity.Hmac;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClients;

```

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class InquiryResult {

    private final String submitId = "20146IT4-20220216-3200a2752e72"; // Sample Submit Id

    private final String url = "https://SAMPLE_URL/api/invoice_result/" + submitId;
    private final String uri = "/api/invoice_result/" + submitId;
    private final String contentType = "application/json; charset=utf-8";

    private final String secretKey = "zsmV5%Yc9z0AIINQG)40R3YN@FFM(kXz"; // Sample Secret Key
    private final String authToken = "G6SE9E5PS2JAOJHIK506XQUR4XKEC3JQ"; // Sample Session Key
    private final String accreditationId = "20146IT4"; // Sample Accreditation ID
    private final String applicationId = "60xLDIIL1"; // Sample Application ID

    public HttpResponse send() throws Exception {
        HttpClient httpClient = HttpClient.createDefault();

        // Set Request URL
        HttpGet httpGet = new HttpGet(url);

        // Set Request Header
        String dateTime = LocalDateTime.now().format(
            DateTimeFormatter.ofPattern("yyyyMMddHHmmss")); // Asia/Manila
        httpGet.addHeader("datetime", dateTime);

        String hmac = new Hmac().generate(secretKey, dateTime, httpGet.getMethod(), uri);
        httpGet.addHeader("authorization", hmac);

        httpGet.addHeader("authToken", authToken);
        httpGet.addHeader("accreditationId", accreditationId);
        httpGet.addHeader("applicationId", applicationId);
        httpGet.addHeader("Content-Type", contentType);

        return httpClient.execute(httpGet);
    }
}
```