Sage X3

E-invoicing : How To Guide XML mapping

Contents

Context

This "How To Guide XML mapping" is designed to help developers make advanced setup modifications to handle multiple E-invoicing use cases.

A single example of modification is provided to illustrate the necessary updates in Sage X3.
This guide will not cover all possible E-invoicing use cases.

Example of adaptation of the delivered UBL21 mapping

Goal: Adapt the provided UBL21 mapping by adding the BG-11 and BG-12 tag groups according to the UBL format of the AFNOR standard.

Extract of the scheme including the BG-11 and BG-12 group of tags

PR XP Z12-012 - V1.2 : Formats et Profils des messages Factures et Statuts de cycle de vie, constitutifs du socle mi

ID	Flux 1, 10 ?	Cardinalité SEMANTIQUE EN16931	Cardinalité SEMANTIQUE EN16931	Cardinalité SEMANTIQUE EN16931	Structure du format					Path de la norme UBL	
					N1	N2	N3	N4	N5	Facture ou avoir	Chemin
BG-11	X	0..1	0..1	0..1	REPRÉSENTANT FISCAL DU VENDEUR					/Invoice /CreditNote	/cac:TaxRepresentativeParty
BT-62		1..1	1..1	1..1		Nom du représentant fiscal du vendeur				/Invoice /CreditNote	/cac:TaxRepresentativeParty/cac:PartyName/cbc:Name
BT-63	X	1..1	1..1	1..1		Identifiant à la TVA du représentant fiscal du vend				/Invoice /CreditNote	/cac:TaxRepresentativeParty/cac:PartyTaxScheme/cbc:CompanyID
BT-63-0	X	1..1	1..1	1..1			Identifiant du schéma de l'identifiant TVA du			/Invoice /CreditNote	/cac:TaxRepresentativeParty/cac:PartyTaxScheme/cac:TaxScheme/cbc:ID
BG-12		1..1	1..1	1..1		ADRESSE POSTALE DU REPRÉSENTANT FISCAL DU V				/Invoice /CreditNote	/cac:TaxRepresentativeParty/cac:PostalAddress

XML file definition is provided in Excel file “XP_Z12-012_Annexe_A_2026_V1.3.xlsx”, tab “FE EN16931 + EXTENDED”
Download available on the AFNOR website: [Norme XP Z12-012]

Data snapshot

The **Data snapshot** is used to freeze the data from the posted invoices in the following list of Data snapshot tables:
ESINVH: Invoices
ESINVD: Invoice lines
ESINVTRDPTY: Invoice third parties
ESINVHVAT: Invoice taxes
ESINVHSFI: Invoice footer elements
ESINVNOT: Invoice notes

For each table, a corresponding persistent class exists with the same name as the table name.

Tax management:
The taxes are generated with several lines for each VATCAT and VATRAT and then aggregate with a view
VESINVHVAT: Invoice taxes

The VESINVHVAT view is used to generate the XML, please see the annex “View VESINVHVAT”

Writing schema in Data snapshot tables with entry points:

The following scheme should be respected to add new properties from the Sales invoice using the existing entry point.

```
START SALES INVOICE
 |
 ├─ Load Sage standard [F] class ESINVH
 |
 ├─ Load ESINVD snapshot table
 |
 ├─ Loop on SINVOICED
 |    ├─ Load Sage standard [F] class ESINVD
 |    ├─ Entry point before write LOAD_CLASS_SID_ESID
 |    ├─ Write [F:ESID]
 |    └─ Next
 |
 ├  Load ESINVHVAT snapshot table
 |
 ├─ Loop on SVCRVAT
 |    ├─ Load Sage standard [F] class ESINVHVAT
 |    ├─ Entry point before write LOAD_CLASS_SVV_ESIHV
 |    ├─ Write [F:ESIHV]
 |    └─ Next
 |
 ├─ Load ESINVHSFI snapshot table
```

```
|
├─ Loop on SFOOTINV / SVCRFOOT (INCDCR = 2)
|   ├─ Load Sage standard [F] class ESINVHVAT
|   ├─ Entry point before write LOAD_CLASS_SIH_SVFSFI
|   ├─ Write [F:ESIHF]
|   └─ Next
|
├─ Loop on SFOOTINV / SVCRFOOT (INCDCR = 1)
|   ├─ Load Sage standard [F] class ESINVHVAT
|   ├─ Entry point before write LOAD_CLASS_SIH_SVFSFI
|   ├─ Write [F:ESIHF]
|   └─ Next
|
├─ Load ESINVTRDPTY snapshot table
|
├─ Seller
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_SELLER
|   └─ Write [F:ESI3P]
|
├─ Buyer
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_BUYER
|   └─ Write [F:ESI3P]
|
├─ Invoicee
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_INVOICEE
|   └─ Write [F:ESI3P]
|
├─ Payer
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_PAYER
|   └─ Write [F:ESI3P]
|
├─ Header delivery
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_DELIVERY_H
|   └─ Write [F:ESI3P]
|
├─ Line delivery
|   ├─ Loop on SINVOICED
|   |   ├─ Load Sage standard [F] class ESINVTRDPTY
|   |   ├─ Entry point before write LOAD_CLASS_ESI3P_DELIVERY_D
|   |   ├─ Write [F:ESI3P]
|   |   └─ Next
|
```

```
├─ Entry point CREATE_SIH_ESI3P
│   → optional additional third parties
│
├─ Load ESINVNOT snapshot table
│
├─ Entry point CREATE_SIH_ESIN
│   → optional additional notes
│
├─ Entry point before write LOAD_CLASS_ESIH
│
└─ Write [F:ESIH]
END SALES INVOICE
```

The following scheme should be respected to add new properties from the BP Customer invoice using the existing entry point.

```
START CUSTOMER INVOICE
 │
 ├─ Load Sage standard [F] class ESINVH
 │
 ├─ Load ESINVD snapshot table
 │
 ├─ Loop on BPCINVLIG
 │    ├─ Load Sage standard [F] class ESINVD
 │    ├─ Entry point before write LOAD_CLASS_SIL_ESID
 │    ├─ Write [F:ESID]
 │    └─ Next
 │
 ├  Load ESINVHVAT snapshot table
 │
 ├─ Loop on SVCRVAT (SINVOICE fields TAX())
 │    ├─ Load Sage standard [F] class ESINVHVAT
 │    ├─ Entry point before write LOAD_CLASS_SVV_ESIHV
 │    ├─ Write [F:ESIHV]
 │    └─ Next
 │
 ├─ Load ESINVHSFI snapshot table
 │
 ├─ Loop on SFOOTINV / SVCRFOOT (INCDCR = 2)
 │    ├─ Load Sage standard [F] class ESINVHVAT
 │    ├─ Entry point before write LOAD_CLASS_SIH_SVFSFI
 │    ├─ Write [F:ESIHF]
 │    └─ Next
 │
 ├─ Loop on SFOOTINV / SVCRFOOT (INCDCR = 1)
```

```
|   ├─ Load Sage standard [F] class ESINVHVAT
|   ├─ Entry point before write LOAD_CLASS_SIH_SVFSFI
|   ├─ Write [F:ESIHF]
|   └─ Next
|
├─ Load ESINVTRDPTY snapshot table
|
├─ Seller
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_SELLER
|   └─ Write [F:ESI3P]
|
├─ Buyer
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_BUYER
|   └─ Write [F:ESI3P]
|
├─ Invoicee
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_INVOICEE
|   └─ Write [F:ESI3P]
|
├─ Payer
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_PAYER
|   └─ Write [F:ESI3P]
|
├─ Header delivery
|   ├─ Load Sage standard [F] class ESINVTRDPTY
|   ├─ Entry point before write LOAD_CLASS_ESI3P_DELIVERY_H
|   └─ Write [F:ESI3P]
|
├─ Line delivery
|   ├─ Loop on BPCINVLIG
|   |   ├─ Load Sage standard [F] class ESINVTRDPTY
|   |   ├─ Entry point before write LOAD_CLASS_ESI3P_DELIVERY_D
|   |   ├─ Write [F:ESI3P]
|   |   └─ Next
|
├─ Entry point CREATE_SIH_ESI3P
|   → optional additional third parties
|
├─ Load ESINVNOT snapshot table
|
├─ Entry point CREATE_SIH_ESIN
|   → optional additional notes
|
```

```
├─ Entry point before write LOAD_CLASS_ESIH
│
└─ Write [F:ESIH]
END CUSTOMER INVOICE
```

Third Party Type Management

• Third Party Type Management adding a new type in the Miscellaneous table 210.

- Add Third Party Types: New third party types must be added to the 210 table to manage cases such as “tax representative”. ⚠️ Miscellaneous table 210 is delivered as non-editable. You can change temporarily the definition of the table to add a new value, but you will have to update it each time you install a new release. The entry point CREATE_SIH_ESI3P has been provided to create a new third party.

- Impact on Mapping and Display: The addition of new third-party types must be carefully considered to avoid impacts on display on EINVUBL Crystal Reports (this PDF document should display the same data as the UBL) and XML generation.

XML structure

The following Basic classes are filled from the persistent classes defined previously

The structure of classes corresponds to the XML structure. Each group of tags of the XML has a corresponding class. For example, the group of tags BG-1 has the corresponding UBLBG0001 class. The sub-classes, like for example UBLBGS001, have been defined sequentially each time a new subclass was needed.

- UBLBG0000: Main invoice class
- UBLBG0000CN: Main Credit Note class
- UBLBG0000CN1
- UBLBG0000IV
- UBLBG0001
- UBLBG0003
- UBLBG0004
- UBLBG0005
- UBLBG0006
- UBLBG0013
- UBLBG0014
- UBLBG0016
- UBLBG0017

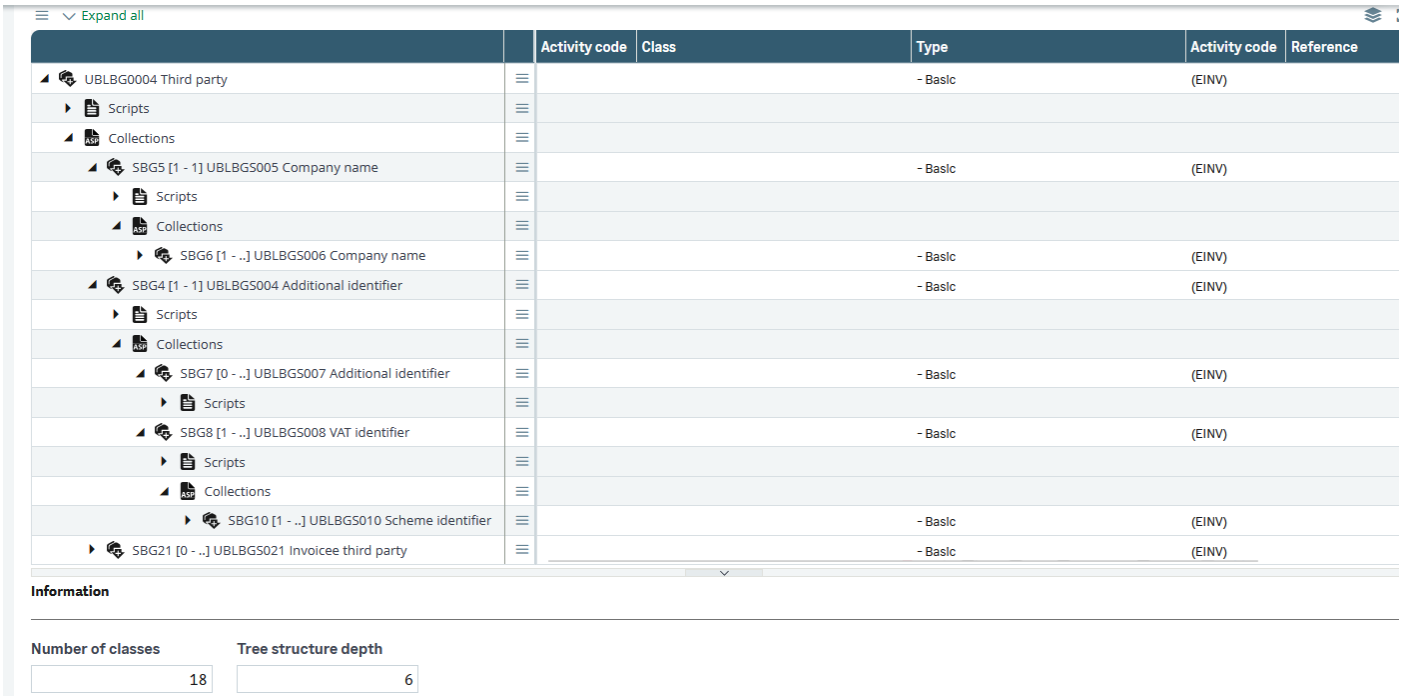
- UBLBG0019
- UBLBG0020
- UBLBG0022
- UBLBG0023
- UBLBG0025
- UBLBG0029
- UBLBG0030
- UBLBG0031
- UBLBGS001
- UBLBGS002
- UBLBGS003
- UBLBGS004
- UBLBGS005
- UBLBGS006
- UBLBGS007
- UBLBGS008
- UBLBGS009
- UBLBGS010
- UBLBGS011
- UBLBGS012
- UBLBGS013
- UBLBGS014
- UBLBGS015
- UBLBGS016
- UBLBGS017
- UBLBGS018
- UBLBGS019
- UBLBGS020
- UBLBGS021
- UBLBGS022
- UBLBGS023
- UBLBGS024
- UBLEXTFRFE06
- UBLEXTFRFE11
- UBLEXTFRFE12
- UBLEXTFRFE13
- UBLEXTFRFS01

In GESACLA, you can see the tree structure of the classes and subclasses.

For example, if you select the UBLBG0004 class (that correspond to the group of tag BG-04: Seller party) and you click on “Tree structure” you will obtain the following structure that correspond to the XML structure.

⚠ It is currently not possible to obtain the tree structure from the main class or from the representation UBLBG0000 due to a technical limitation that will be fixed.

Linked classes



Representation used to expose the properties:

- UBLBG0000

Each added property must be loaded in the class SPE script (method LOAD) and exposed in the UBLBG0000 representation

Adding and configuring groups BG-11 and BG-12

• Steps to add Group BG-11 (tax representative) and Group BG-12 (representative's address) to the corresponding UBL class, cardinality management, and associated scripts.

- Creating Group BG-11: (Tax Representative) Adding Group BG-11 to the UBL class, using a specific coding (XUBLBG00011) and based on the standard structure, taking in account cardinality constraints.

- Cardinality Management and Scripts: Cardinality management: use 0..1 for BG11 (with the automatically generated “addline” method) and script structure, recommending the use of methods rather than functions to benefit from the execution context (`this.ACTX`), and always testing error statuses with appropriate constants.

- Adding the BG-12 subgroup: (representative address), specifying the tag structure (BT64, BT66, etc.) and ensuring adherence to the standard, without using basic types for properties to avoid X3 constraints.

- Comments and Parameters: Add and/or correct comments in the code, particularly to properly document method parameters, and the importance of always passing the header document and the current line as parameters to allow for correct navigation within collections.

- Entry Point Management and Mapping: Place the new fields in the appropriate snapshot table, and not in the ESINVH header table, to ensure data consistency when generating XML, taking in account the normalized structure and the handling of multi-tier cases.

Mapping update

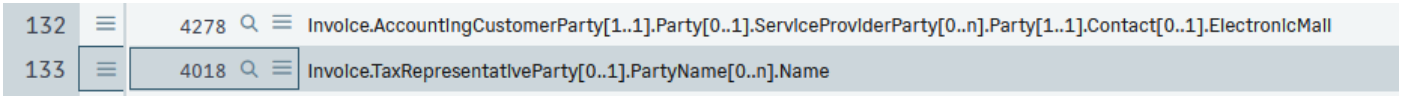
1. Duplicate the existing mapping setup that is not updatable (SAGE factory owner)
   Careful: the specific mapping must be updated manually each time Sage will deliver an updated mapping
2. Execute the Generation action
3. **Validate the mapping to generate the mapping.json and prototype.json used by the Syracuse script generateXmlInvoice**

Specificity on conditions and attributes

4. Use condition to display the tag
5. Use attribute to classify the data (Currency, Unit Of Measure, Type of identifier)

The Identifier of each tag of the mapping rules are stored into the table EINVMAPNODES.
These identifiers are generated by the “Generation” Button in the GESEINVMAP function based on the list of XSD uploaded files. This generation action calls a Syracuse script parseXsd.
In case of missing group of tags used in the Standard UBL, you should update the existing XSD or add a new XSD and run the Generation action in GESEINVMAP function.
For example, the group of elements PayeeParty is not generated because the following line is in comments in the XSD <!-- xsd:element name="PayeeParty" type="PartyType"/--> . You should uncomment it on the UBL-CommonAggregateComponents-2.1.xsd, line 406 and run the “Generation” action in GESEINVMAP function.
⚠ After Generation, many new identifiers will be added into EINVMAPNODES table and potentially existing identifier could have a new number. You may need to revalidate all the mapping you are using.

The tag mapping for group BG-11 begins with tag BT-62. This tag must be mapped with the Identifier 4018 after the tag group AccountingCustomerParty.



The mapping of the BG-12 group begins with the tag BT-65. This tag must be mapped with the identifier 4009 after the BG-11 group of tags.

Documentation and UBL mapping process

• The mapping logic between snapshot tables, UBL classes, and the generated XML using the AFNOR rules Excel is documented, and the naming of the properties should respect the AFNOR structure to facilitate customization by partners for example if you want to add the BT-62, you should name the property XBT62 to avoid conflict with Sage in the future.

- Link between Snapshot, Classes, and XML: Documentation describing the logic that connects snapshot tables, UBL classes, and the XML mapping, making the process transparent and reproducible for partners, since the class structure must follow the tree hierarchy defined by the AFNOR.

- Use of AFNOR Documentation: The Excel documentation provided by AFNOR, which details the tag hierarchy and its level, allows you to determine where and how to add or modify properties in the mapping.

- Adaptation Cases and Best Practices: Two levels of adaptation: using existing tags in the standard mapping on the one hand, adding new groups or fields on the other hand.

Miscellaneous technical points and code optimization

• Code optimization is important, including error status management, method usage, and activity code protection, to ensure the robustness and maintainability of development.

- Error Status Management: Always test error statuses with appropriate constants (less than ERROR) to guarantee proper message handling and avoid unnecessary blocking during code evolution.
- Method and Context Usage: It is important to use methods rather than functions to benefit from the execution context (this.ACTX in classes and GACTX in classic) and avoid using global variables in classes.

If you need to add a note or a tag in the UBL containing the description from a message, you should add it in the language of the customer.
You can use the following routine from the EINVOUTGEN_L_ESIN script

```
##########################################################################################
Funprog GET_MESS_BPR_LAN(ACTX, SLAN, NCHAP, NNUMMSG)
Variable Instance ACTX Using =[V]CST_C_NAME_CLASS_CONTEXT
```

```
Value Char SLAN
Value Integer NCHAP
Value Integer NNUMMSG

Local Clbfile SARENDRE(1)
Local Char SLANREAD(GLONLAN)

 If SLAN = ACTX.LAN Then
  # If asked language is the connection language we can use "mess"
  SARENDRE = mess(NNUMMSG,NCHAP,1)
 Else
  Local File APLSTD [AST]
  # -------------------------
  For SLANREAD = SLAN, ACTX.AFOLD.ALANGDEF
   Read [AST]CLE=NCHAP;NNUMMSG;SLANREAD
   If fstat = [V]CST_AOK Then
    SARENDRE = [F:AST]LANMES
    Break
   Endif
  Next
 Endif
End SARENDRE
```

Annexes

View VESINVHVAT:

```
WITH SUMS AS (
  SELECT
    SIVNUM_0,
    VATCAT_0,
    VATRAT_0,
    SUM(BASVAT_0) AS SUM_BASVAT_0,
    SUM(AMTVAT_0) AS SUM_AMTVAT_0
  FROM ESINVHVAT
  GROUP BY SIVNUM_0, VATCAT_0, VATRAT_0
)
SELECT
  S.SIVNUM_0,
  S.VATCAT_0,
  S.VATRAT_0,
  F.EXEVATREACOD_0,
  F.EXEVATREADES_0,
  S.SUM_BASVAT_0,
  S.SUM_AMTVAT_0
FROM SUMS AS S
CROSS APPLY (
  SELECT TOP (1)
    E.EXEVATREACOD_0,
    E.EXEVATREADES_0
  FROM ESINVHVAT AS E
  WHERE E.SIVNUM_0 = S.SIVNUM_0
   AND E.VATCAT_0 = S.VATCAT_0
   AND E.VATRAT_0 = S.VATRAT_0
  ORDER BY E.UPDDATTIM_0 ASC
) AS F
```

sage.com