



PL/PDF Factur-X

PL/PDF Factur-X / ZUGFeRD User Guide

v0.11.0

PL/SQL programming reference



PL/PDF Factur-X

Contents

1. Introduction.....	3
2. Public constants, types and programming conventions.....	5
3. Data-first invoice input.....	9
4. Validation and XML output.....	22
5. XML-first input.....	22
6. Factur-X PDF/A-3 generation.....	23
7. Renderer integration and read-only canonical model.....	25
8. End-to-end examples and diagnostics.....	30



PL/PDF Factur-X

1. Introduction

PL/PDF Factur-X / ZUGFeRD is a PL/SQL add-on for creating, importing, validating, and packaging hybrid electronic invoices directly inside Oracle Database. It maintains a canonical invoice model, serializes or imports Cross Industry Invoice (CII) XML, and prepares the PL/PDF document as a Factur-X PDF/A-3b carrier with the required embedded XML and XMP extension metadata.

The public programming interface is the PLPDF_FACTUR package. Applications can build invoice data with the data-first setter API, load an existing Factur-X/ZUGFeRD CII document with LOADXML, retrieve the canonical XML with GETXML, and create a hybrid PDF through PREPAREPDFCARRIER. The optional PLPDF_FACTUR_INVOICE_EXAMPLE package is a reference visual renderer built entirely on the read-only canonical model API.

1.1 Standards and prerequisites

Component	Supported baseline
PLPDF_FACTUR package	0.11.0
Factur-X	1.09.2
ZUGFeRD	2.5.2
Hybrid PDF conformance	PDF/A-3b
Required PL/PDF version	5.32 or later
EN 16931 code-list baseline	v17b, published 2026-04-16, effective 2026-05-15

1.2 Supported profiles

Profile constant	Profile	CII line items	PDF attachment relationship
c_profile_minimum	MINIMUM	Not serialized	Data
c_profile_basic_wl	BASIC WL	Not serialized	Data
c_profile_basic	BASIC	Serialized	Alternative
c_profile_en16931	EN 16931	Serialized	Alternative
c_profile_extended	EXTENDED	Serialized	Alternative

MINIMUM and BASIC WL deliberately permit richer canonical data for the human-readable PDF than is serialized into their reduced XML profiles. Renderer code can therefore display lines or other visual information that is intentionally absent from the embedded lower-profile CII document.

1.3 Main workflows

Workflow	Description
Data-first XML	INIT, populate the canonical model, VALIDATE, then GETXML.



PL/PDF Factur-X

Workflow	Description
Data-first hybrid PDF	INIT and populate the model, then prepare the Factur-X carrier before the visible PL/PDF content is finalized.
XML-first	LOADXML validates and imports an existing supported CII document and preserves its exact source bytes for the hybrid PDF carrier.
Reference renderer	PLPDF_FACTUR_INVOICE_EXAMPLE.GENERATE creates a simple visible invoice from the read-only canonical model and packages the Factur-X carrier.
Custom renderer	Application code reads the canonical snapshot getters and uses normal PL/PDF drawing/text APIs between PREPAREPDFCARRIER and PLPDF.SENDDOC.

1.4 Canonical model and package state

PLPDF_FACTUR is stateful within the Oracle session. INIT starts a new data-first invoice model for the selected profile. LOADXML atomically replaces the current model only after the supplied XML has been parsed and validated successfully. A failed INIT or LOADXML does not destroy the previously valid model.

All data-first setters and Add* procedures operate on the current session model. Calling a data-first mutation after LOADXML invalidates the preserved exact XML source; subsequent GETXML and PDF carrier generation serialize the modified canonical model instead of reusing the original XML bytes.

1.5 Input, output, and transaction ownership

Input BLOBs passed to LOADXML are borrowed/read-only and remain owned by the caller. The package copies the XML it needs internally and does not free the caller locator.

GETXML returns a caller-owned temporary CLOB. Release it with DBMS_LOB.FREETEMPORARY when it is no longer required. PREPAREPDFCARRIER owns an internal temporary XML BLOB until RELEASEPDFCARRIER, the next successful INIT, or an error cleanup releases it.

PLPDF_FACTUR does not COMMIT or ROLLBACK the caller transaction. Transaction boundaries remain under application control.

1.6 Invoice and credit-note sign conventions

The public invoice-type constants cover the most common document types, but SETINVOICE is not restricted to those seven constants: other invoice type codes from the supported code list can be used when the applicable profile and business rules permit them.

Credit-note document types 381 and 261 use positive monetary totals; the document type carries the credit-note meaning. Negative invoice representation is supported for



PL/PDF Factur-X

invoice-like types 380, 384, 386, 389, and 751. VALIDATE enforces these sign semantics.

2. Public constants, types and programming conventions

2.1 Profile constants

Constant	Value	Purpose
c_profile_minimum	MINIMUM	Factur-X MINIMUM profile.
c_profile_basic_wl	BASIC WL	Factur-X BASIC WL profile.
c_profile_basic	BASIC	Factur-X BASIC profile.
c_profile_en16931	EN 16931	EN 16931 profile; default for INIT.
c_profile_extended	EXTENDED	Factur-X EXTENDED profile.

2.2 Common invoice type constants

Constant	Code	Meaning
c_invoice_type_commercial	380	Commercial invoice.
c_invoice_type_credit_note	381	Credit note.
c_invoice_type_corrective	384	Corrective invoice.
c_invoice_type_prepayment	386	Prepayment invoice.
c_invoice_type_self_billed	389	Self-billed invoice.
c_invoice_type_self_billed_credit_note	261	Self-billed credit note.
c_invoice_type_accounting_information	751	Invoice information for accounting purposes.

2.3 Adjustment constants

Constant	Value	Meaning
c_adjustment_level_document	DOCUMENT	Document-level allowance/charge returned by GETADJUSTMENT.
c_adjustment_level_line	LINE	Line-level allowance/charge returned by GETADJUSTMENT.
c_adjustment_kind_allowance	ALLOWANCE	Allowance record.
c_adjustment_kind_charge	CHARGE	Charge record.

2.4 Canonical snapshot record types

The read-only getter API returns package record types. These records expose the same canonical state that is used by validation and XML serialization, so a custom visual renderer can remain synchronized with the embedded invoice XML.



PL/PDF Factur-X

2.4.1 T_INVOICE

Field	Meaning
business_process_type	BT-23 Business process type.
invoice_id	BT-1 Invoice number.
issue_date	BT-2 Invoice issue date.
invoice_type_code	BT-3 Invoice type code.
currency_code	BT-5 Invoice currency code.
vat_accounting_currency_code	BT-6 VAT accounting currency code.
buyer_reference	BT-10 Buyer reference.

2.4.2 T_PARTY

Field	Meaning
name	Party legal/display name.
trading_name	Trading name.
identifier / identifier_scheme	Party identifier and scheme.
legal_registration_id / scheme	Legal registration identifier and scheme.
vat_id	VAT identifier.
tax_registration_id / scheme	Tax registration identifier and scheme.
electronic_address / scheme	Electronic address and EAS scheme.
address_line1..3	Postal address lines.
postcode / city / country_subdivision / country_code	Postal location.
contact_point / contact_department	Contact person or department; use only one of the two name forms.
contact_phone / contact_email	Contact details.
additional_legal_information	Seller additional legal information.

2.4.3 T_REFERENCES

Field	Meaning
purchase_order_reference	Purchase order reference.
sales_order_reference	Sales order reference.
contract_reference	Contract reference.
project_reference	Project reference.
buyer_accounting_reference	Buyer accounting reference / cost centre.
despatch_advice_reference	Despatch advice reference.
receiving_advice_reference	Receiving advice reference.
tender_or_lot_reference	Tender or lot reference.
invoiced_object_identifier / scheme	Invoiced object identifier and qualifier.



PL/PDF Factur-X

2.4.4 T_PRECEDING_INVOICE

Field	Meaning
invoice_id	BT-25 preceding invoice number.
issue_date	BT-26 preceding invoice issue date; optional.

2.4.5 T_PAYMENT

Field	Meaning
means_code / means_text	Payment means code and optional text.
payment_account_id	Creditor payment account identifier, e.g. IBAN.
payment_account_name	Payment account name.
payment_service_provider_id	Payment service provider identifier, e.g. BIC.
debited_account_id	Debited account identifier.
creditor_id	Creditor identifier for direct debit.
mandate_reference	Mandate reference.
remittance_information	Remittance information.
due_date	Payment due date.
terms	Payment terms text.

2.4.6 T_NOTE

Field	Meaning
subject_code	BT-21 note subject code.
note_text	BT-22 invoice note text.

2.4.7 T_LINE

Field	Meaning
line_id	BT-126 invoice line identifier.
line_note	BT-127 line note.
item_name / item_description	Item name and description.
seller_item_id / buyer_item_id	Seller and buyer item identifiers.
standard_item_id / scheme	Standard item identifier and scheme.
country_of_origin	Country of origin code.
purchase_order_line_reference	Purchase order line reference.
buyer_accounting_reference	Line buyer accounting reference.
quantity / unit_code	Billed quantity and unit code.
gross_unit_price / unit_discount / net_unit_price	Price model.
price_base_quantity / price_base_unit_code	Price base quantity and unit.



PL/PDF Factur-X

Field	Meaning
vat_category_code / vat_rate	Line VAT category and rate.
period_start_date / period_end_date	Line invoicing period.
line_net_amount	BT-131 line net amount.

2.4.8 T_ADJUSTMENT

Field	Meaning
adjustment_level / adjustment_kind	DOCUMENT or LINE; ALLOWANCE or CHARGE.
line_index	1-based source line index for line adjustments; NULL for document adjustments.
amount / base_amount / percentage	Adjustment monetary/percentage values.
reason_code / reason_text	Allowance/charge reason.
vat_category_code / vat_rate	VAT classification for document-level adjustments.

2.4.9 T_VAT_BREAKDOWN

Field	Meaning
category_code	VAT category code.
rate	VAT rate.
taxable_amount	VAT category taxable amount.
tax_amount	VAT category tax amount.
exemption_reason_code / exemption_reason_text	VAT exemption reason when applicable.

2.4.10 T_TOTALS

Field	Meaning
line_net_total	BT-106 sum of line net amounts.
allowance_total	BT-107 document allowance total.
charge_total	BT-108 document charge total.
tax_basis_total	BT-109 invoice total amount without VAT.
tax_total	BT-110 invoice total VAT amount.
tax_total_accounting_currency	BT-111 VAT total in accounting currency.
grand_total	BT-112 invoice total amount with VAT.
paid_amount	BT-113 paid amount.
rounding_amount	BT-114 rounding amount.
due_payable_amount	BT-115 amount due for payment.



2.5 Code-list validation

VALIDATE applies the code lists embedded for the supported Factur-X/ZUGFeRD release baseline. The public model currently validates, where applicable, invoice type codes, currencies, countries, party identifier schemes, legal registration schemes, electronic address schemes, note subject codes, payment means, VAT category codes, allowance/charge reason codes, reference qualifiers, and UNECE unit codes.

EXTENDED intentionally uses the wider official party GlobalID scheme set and the wider VAT category set defined by the EXTENDED profile. Applications should use the exact code required by the selected profile rather than treating a code accepted in EXTENDED as automatically valid in EN 16931 or BASIC.

3. Data-first invoice input

Data-first processing starts with INIT. The application then populates the canonical model with Set* and Add* procedures. VALIDATE should be called before consuming the XML or PDF output; PREPAREPDFCARRIER and the reference renderer also call VALIDATE internally.

3.1 INIT

Starts a new empty canonical invoice model for the selected profile.

Syntax

```
procedure Init(  
    p_profile varchar2 default c_profile_en16931  
);
```

Parameters

Parameter	Type / default	Description
p_profile	IN varchar2 DEFAULT c_profile_en16931	One of the public profile constants. Common spelling variants are normalized; an unsupported profile raises an error.

Example

```
plpdf_factur.Init(plpdf_factur.c_profile_en16931);
```

3.2 SETINVOICE

Sets the document identity and header-level invoice information.

Syntax

```
procedure SetInvoice(  
    p_invoice_id          varchar2,  
    p_issue_date          date,  
    p_invoice_type_code   varchar2,  
    p_currency_code       varchar2,  
    p_business_process_type varchar2 default null,  
    p_buyer_reference      varchar2 default null,  
    p_vat_accounting_currency_code varchar2 default null  
);
```



PL/PDF Factur-X

Parameters

Parameter	Type / default	Description
p_invoice_id	IN varchar2	Invoice number (BT-1).
p_issue_date	IN date	Invoice issue date (BT-2).
p_invoice_type_code	IN varchar2	Invoice type code (BT-3). Use a public convenience constant or another supported code.
p_currency_code	IN varchar2	Invoice currency (BT-5).
p_business_process_type	IN varchar2 DEFAULT null	Business process type (BT-23).
p_buyer_reference	IN varchar2 DEFAULT null	Buyer reference (BT-10).
p_vat_accounting_currency_code	IN varchar2 DEFAULT null	VAT accounting currency (BT-6), when applicable.

Example

```
plpdf_factur.SetInvoice(  
  p_invoice_id      => 'INV-2026-001',  
  p_issue_date      => date '2026-08-29',  
  p_invoice_type_code => plpdf_factur.c_invoice_type_commercial,  
  p_currency_code   => 'EUR',  
  p_business_process_type => 'urn:example:billing',  
  p_buyer_reference  => 'P0-4711'  
);
```

3.3 SETSELLER

Sets seller identity, address, tax, electronic-address, and contact information.

Syntax

```
procedure SetSeller(  
  p_name                varchar2,  
  p_country_code        varchar2,  
  p_trading_name        varchar2 default null,  
  p_identifier          varchar2 default null,  
  p_identifier_scheme   varchar2 default null,  
  p_legal_registration_id varchar2 default null,  
  p_legal_registration_scheme varchar2 default null,  
  p_vat_id              varchar2 default null,  
  p_tax_registration_id varchar2 default null,  
  p_tax_registration_scheme varchar2 default null,  
  p_electronic_address  varchar2 default null,  
  p_electronic_address_scheme varchar2 default null,  
  p_address_line1       varchar2 default null,  
  p_address_line2       varchar2 default null,  
  p_address_line3       varchar2 default null,  
  p_postcode            varchar2 default null,  
  p_city                varchar2 default null,  
  p_country_subdivision varchar2 default null,  
  p_contact_point       varchar2 default null,  
  p_contact_department  varchar2 default null,  
  p_contact_phone       varchar2 default null,  
  p_contact_email       varchar2 default null,  
  p_additional_legal_information varchar2 default null  
);
```



PL/PDF Factur-X

Parameters

Parameter	Type / default	Description
p_name	IN varchar2	Seller name.
p_country_code	IN varchar2	Seller country code.
p_trading_name	IN varchar2 DEFAULT null	Seller trading name.
p_identifier	IN varchar2 DEFAULT null	Seller identifier.
p_identifier_scheme	IN varchar2 DEFAULT null	Scheme of the seller identifier.
p_legal_registration_id	IN varchar2 DEFAULT null	Seller legal registration identifier.
p_legal_registration_scheme	IN varchar2 DEFAULT null	Scheme of the legal registration identifier.
p_vat_id	IN varchar2 DEFAULT null	Seller VAT identifier.
p_tax_registration_id	IN varchar2 DEFAULT null	Seller tax registration identifier.
p_tax_registration_scheme	IN varchar2 DEFAULT null	Tax registration scheme; current fiscal scheme support includes FC.
p_electronic_address	IN varchar2 DEFAULT null	Seller electronic address.
p_electronic_address_scheme	IN varchar2 DEFAULT null	Electronic Address Scheme (EAS) identifier.
p_address_line1..3	IN varchar2 DEFAULT null	Seller postal address lines.
p_postcode	IN varchar2 DEFAULT null	Postcode.
p_city	IN varchar2 DEFAULT null	City.
p_country_subdivision	IN varchar2 DEFAULT null	Country subdivision.
p_contact_point	IN varchar2 DEFAULT null	Contact person/name.
p_contact_department	IN varchar2 DEFAULT null	Contact department. Do not supply together with p_contact_point when the target profile requires a single contact-name representation.
p_contact_phone	IN varchar2 DEFAULT null	Contact telephone.
p_contact_email	IN varchar2 DEFAULT null	Contact email.
p_additional_legal_information	IN varchar2 DEFAULT null	Additional seller legal information.

Example

```
plpdf_factur.SetSeller(  
  p_name          => 'Example Seller GmbH',
```



PL/PDF Factur-X

```
p_country_code      => 'DE',
p_vat_id            => 'DE123456789',
p_electronic_address => 'seller@example.test',
p_electronic_address_scheme => 'EM',
p_address_line1     => 'Example Strasse 1',
p_postcode          => '10115',
p_city              => 'Berlin'
);
```

3.4 SETBUYER

Sets buyer identity, address, electronic-address, and contact information.

Syntax

```
procedure SetBuyer(
  p_name          varchar2,
  p_country_code  varchar2,
  p_trading_name  varchar2 default null,
  p_identifier     varchar2 default null,
  p_identifier_scheme varchar2 default null,
  p_legal_registration_id varchar2 default null,
  p_legal_registration_scheme varchar2 default null,
  p_vat_id        varchar2 default null,
  p_electronic_address varchar2 default null,
  p_electronic_address_scheme varchar2 default null,
  p_address_line1  varchar2 default null,
  p_address_line2  varchar2 default null,
  p_address_line3  varchar2 default null,
  p_postcode       varchar2 default null,
  p_city           varchar2 default null,
  p_country_subdivision varchar2 default null,
  p_contact_point  varchar2 default null,
  p_contact_department varchar2 default null,
  p_contact_phone  varchar2 default null,
  p_contact_email  varchar2 default null
);
```

Parameters

Parameter	Type / default	Description
p_name	IN varchar2	Buyer name.
p_country_code	IN varchar2	Buyer country code.
p_trading_name	IN varchar2 DEFAULT null	Buyer trading name.
p_identifier	IN varchar2 DEFAULT null	Buyer identifier.
p_identifier_scheme	IN varchar2 DEFAULT null	Scheme of the buyer identifier.
p_legal_registration_id	IN varchar2 DEFAULT null	Buyer legal registration identifier.
p_legal_registration_scheme	IN varchar2 DEFAULT null	Scheme of the legal registration identifier.
p_vat_id	IN varchar2 DEFAULT null	Buyer VAT identifier.
p_electronic_address	IN varchar2 DEFAULT null	Buyer electronic address.
p_electronic_address_scheme	IN varchar2 DEFAULT null	Electronic Address Scheme (EAS) identifier.



PL/PDF Factur-X

Parameter	Type / default	Description
p_address_line1..3	IN varchar2 DEFAULT null	Buyer postal address lines.
p_postcode	IN varchar2 DEFAULT null	Postcode.
p_city	IN varchar2 DEFAULT null	City.
p_country_subdivision	IN varchar2 DEFAULT null	Country subdivision.
p_contact_point	IN varchar2 DEFAULT null	Contact person/name.
p_contact_department	IN varchar2 DEFAULT null	Contact department.
p_contact_phone	IN varchar2 DEFAULT null	Contact telephone.
p_contact_email	IN varchar2 DEFAULT null	Contact email.

Example

```
plpdf_factur.SetBuyer(  
  p_name           => 'Example Buyer Kft.',  
  p_country_code   => 'HU',  
  p_electronic_address => 'buyer@example.test',  
  p_electronic_address_scheme => 'EM',  
  p_address_line1  => 'Minta utca 1.',  
  p_postcode       => '1111',  
  p_city           => 'Budapest'  
);
```

3.5 SETREFERENCES

Sets document-level order, contract, project, logistics, accounting, tender, and invoiced-object references.

Syntax

```
procedure SetReferences(  
  p_purchase_order_reference  varchar2 default null,  
  p_sales_order_reference     varchar2 default null,  
  p_contract_reference        varchar2 default null,  
  p_project_reference         varchar2 default null,  
  p_buyer_accounting_reference varchar2 default null,  
  p_despatch_advice_reference varchar2 default null,  
  p_receiving_advice_reference varchar2 default null,  
  p_tender_or_lot_reference   varchar2 default null,  
  p_invoiced_object_identifier varchar2 default null,  
  p_invoiced_object_scheme    varchar2 default null  
);
```

Parameters

Parameter	Type / default	Description
p_purchase_order_reference	IN varchar2 DEFAULT null	Purchase order reference.
p_sales_order_reference	IN varchar2 DEFAULT null	Sales order reference.



PL/PDF Factur-X

Parameter	Type / default	Description
p_contract_reference	IN varchar2 DEFAULT null	Contract reference.
p_project_reference	IN varchar2 DEFAULT null	Project reference.
p_buyer_accounting_reference	IN varchar2 DEFAULT null	Buyer accounting reference / cost centre.
p_despatch_advice_reference	IN varchar2 DEFAULT null	Despatch advice reference.
p_receiving_advice_reference	IN varchar2 DEFAULT null	Receiving advice reference.
p_tender_or_lot_reference	IN varchar2 DEFAULT null	Tender or lot reference.
p_invoiced_object_identifier	IN varchar2 DEFAULT null	Invoiced object identifier.
p_invoiced_object_scheme	IN varchar2 DEFAULT null	Qualifier/scheme for the invoiced object identifier.

Example

```
plpdf_factur.SetReferences(  
  p_purchase_order_reference => 'P0-4711',  
  p_contract_reference       => 'CONTRACT-2026-01',  
  p_despatch_advice_reference => 'DESPATCH-17'  
);
```

3.6 ADDPRECEDINGINVOICE

Adds a BG-3 preceding invoice reference. The procedure is repeatable.

Syntax

```
procedure AddPrecedingInvoice(  
  p_invoice_id varchar2,  
  p_issue_date date default null  
);
```

Parameters

Parameter	Type / default	Description
p_invoice_id	IN varchar2	Preceding invoice identifier (BT-25).
p_issue_date	IN date DEFAULT null	Preceding invoice issue date (BT-26).

Example

```
plpdf_factur.AddPrecedingInvoice(  
  p_invoice_id => 'INV-2026-0007',  
  p_issue_date => date '2026-08-01'  
);
```

3.7 SETPAYMENT

Sets payment means, account, remittance, direct-debit, due-date, and payment-terms information.



PL/PDF Factur-X

Syntax

```
procedure SetPayment(  
    p_means_code          varchar2 default null,  
    p_means_text          varchar2 default null,  
    p_payment_account_id  varchar2 default null,  
    p_payment_account_name varchar2 default null,  
    p_payment_service_provider_id varchar2 default null,  
    p_debited_account_id  varchar2 default null,  
    p_creditor_id         varchar2 default null,  
    p_mandate_reference   varchar2 default null,  
    p_remittance_information varchar2 default null,  
    p_due_date            date default null,  
    p_terms               varchar2 default null  
);
```

Parameters

Parameter	Type / default	Description
p_means_code	IN varchar2 DEFAULT null	Payment means code (BT-81).
p_means_text	IN varchar2 DEFAULT null	Payment means text.
p_payment_account_id	IN varchar2 DEFAULT null	Creditor account identifier, e.g. IBAN.
p_payment_account_name	IN varchar2 DEFAULT null	Account name.
p_payment_service_provider_id	IN varchar2 DEFAULT null	Payment service provider identifier, e.g. BIC.
p_debited_account_id	IN varchar2 DEFAULT null	Debited account identifier.
p_creditor_id	IN varchar2 DEFAULT null	Creditor identifier.
p_mandate_reference	IN varchar2 DEFAULT null	Direct-debit mandate reference.
p_remittance_information	IN varchar2 DEFAULT null	Remittance information.
p_due_date	IN date DEFAULT null	Payment due date.
p_terms	IN varchar2 DEFAULT null	Payment terms text.

Example

```
plpdf_factur.SetPayment(  
    p_means_code          => '58',  
    p_payment_account_id  => 'DE02120300000000202051',  
    p_due_date            => date '2026-09-28',  
    p_terms               => 'Payable within 30 days.'  
);
```

3.8 ADDNOTE

Adds an invoice-level note. Multiple notes can be added.

Syntax

```
procedure AddNote(  
    p_note_text    varchar2,
```



PL/PDF Factur-X

```
p_subject_code varchar2 default null  
);
```

Parameters

Parameter	Type / default	Description
p_note_text	IN varchar2	Invoice note text (BT-22).
p_subject_code	IN varchar2 DEFAULT null	Optional note subject code (BT-21).

Example

```
plpdf_factur.AddNote('Invoice issued electronically.', 'AAI');
```

3.9 ADDLINE

Adds one canonical invoice line. Line indices used by ADDLINEALLOWANCE and ADDLINECHARGE are 1-based in insertion order.

Syntax

```
procedure AddLine(  
    p_line_id                varchar2,  
    p_item_name              varchar2,  
    p_quantity               number,  
    p_unit_code              varchar2,  
    p_net_unit_price         number,  
    p_vat_category_code      varchar2,  
    p_vat_rate               number default null,  
    p_line_net_amount        number default null,  
    p_line_note              varchar2 default null,  
    p_item_description        varchar2 default null,  
    p_seller_item_id         varchar2 default null,  
    p_buyer_item_id          varchar2 default null,  
    p_standard_item_id       varchar2 default null,  
    p_standard_item_scheme   varchar2 default null,  
    p_country_of_origin      varchar2 default null,  
    p_purchase_order_line_reference varchar2 default null,  
    p_buyer_accounting_reference varchar2 default null,  
    p_gross_unit_price       number default null,  
    p_unit_discount          number default null,  
    p_price_base_quantity    number default null,  
    p_price_base_unit_code   varchar2 default null,  
    p_period_start_date      date default null,  
    p_period_end_date        date default null  
);
```

Parameters

Parameter	Type / default	Description
p_line_id	IN varchar2	Invoice line identifier.
p_item_name	IN varchar2	Item name.
p_quantity	IN number	Billed quantity.
p_unit_code	IN varchar2	UNECE unit code.
p_net_unit_price	IN number	Item net unit price.
p_vat_category_code	IN varchar2	VAT category code.
p_vat_rate	IN number DEFAULT null	VAT rate.



PL/PDF Factur-X

Parameter	Type / default	Description
p_line_net_amount	IN number DEFAULT null	Line net amount. Supply the calculated business value expected by the invoice.
p_line_note	IN varchar2 DEFAULT null	Line note.
p_item_description	IN varchar2 DEFAULT null	Item description.
p_seller_item_id	IN varchar2 DEFAULT null	Seller item identifier.
p_buyer_item_id	IN varchar2 DEFAULT null	Buyer item identifier.
p_standard_item_id	IN varchar2 DEFAULT null	Standard item identifier.
p_standard_item_scheme	IN varchar2 DEFAULT null	Standard item identifier scheme.
p_country_of_origin	IN varchar2 DEFAULT null	Country of origin.
p_purchase_order_line_reference	IN varchar2 DEFAULT null	Purchase order line reference.
p_buyer_accounting_reference	IN varchar2 DEFAULT null	Line buyer accounting reference.
p_gross_unit_price	IN number DEFAULT null	Gross unit price.
p_unit_discount	IN number DEFAULT null	Unit discount.
p_price_base_quantity	IN number DEFAULT null	Price base quantity.
p_price_base_unit_code	IN varchar2 DEFAULT null	Price base unit code.
p_period_start_date	IN date DEFAULT null	Line period start.
p_period_end_date	IN date DEFAULT null	Line period end.

Example

```
plpdf_factur.AddLine(  
  p_line_id      => '1',  
  p_item_name    => 'Software licence',  
  p_quantity     => 1,  
  p_unit_code    => 'C62',  
  p_net_unit_price => 100,  
  p_vat_category_code => 'S',  
  p_vat_rate     => 19,  
  p_line_net_amount => 100  
);
```

3.10 ADDDOCUMENTALLOWANCE

Adds a document-level allowance.



PL/PDF Factur-X

Syntax

```
procedure AddDocumentAllowance(  
    p_amount          number,  
    p_vat_category_code varchar2,  
    p_vat_rate         number default null,  
    p_base_amount      number default null,  
    p_percentage        number default null,  
    p_reason_code       varchar2 default null,  
    p_reason_text       varchar2 default null  
);
```

Parameters

Parameter	Type / default	Description
p_amount	IN number	Allowance/charge amount.
p_vat_category_code	IN varchar2	VAT category code.
p_vat_rate	IN number DEFAULT null	VAT rate.
p_base_amount	IN number DEFAULT null	Calculation base amount.
p_percentage	IN number DEFAULT null	Percentage.
p_reason_code	IN varchar2 DEFAULT null	Allowance/charge reason code.
p_reason_text	IN varchar2 DEFAULT null	Allowance/charge reason text.

Example

```
plpdf_factur.AddDocumentAllowance(10, 'S', 19, 250, 4, '95', 'Discount');
```

3.11 ADDDOCUMENTCHARGE

Adds a document-level charge.

Syntax

```
procedure AddDocumentCharge(  
    p_amount          number,  
    p_vat_category_code varchar2,  
    p_vat_rate         number default null,  
    p_base_amount      number default null,  
    p_percentage        number default null,  
    p_reason_code       varchar2 default null,  
    p_reason_text       varchar2 default null  
);
```

Parameters

Parameter	Type / default	Description
p_amount	IN number	Allowance/charge amount.
p_vat_category_code	IN varchar2	VAT category code.
p_vat_rate	IN number DEFAULT null	VAT rate.
p_base_amount	IN number DEFAULT null	Calculation base amount.



PL/PDF Factur-X

Parameter	Type / default	Description
p_percentage	IN number DEFAULT null	Percentage.
p_reason_code	IN varchar2 DEFAULT null	Allowance/charge reason code.
p_reason_text	IN varchar2 DEFAULT null	Allowance/charge reason text.

Example

```
plpdf_factur.AddDocumentCharge(3, 'S', 19, null, null, 'ZZZ', 'Service charge');
```

3.12 ADDLINEALLOWANCE

Adds an allowance to an existing line.

Syntax

```
procedure AddLineAllowance(  
  p_line_index pls_integer,  
  p_amount     number,  
  p_base_amount number default null,  
  p_percentage number default null,  
  p_reason_code varchar2 default null,  
  p_reason_text varchar2 default null  
);
```

Parameters

Parameter	Type / default	Description
p_line_index	IN pls_integer	1-based index of the line previously added with ADDLINE.
p_amount	IN number	Allowance/charge amount.
p_base_amount	IN number DEFAULT null	Calculation base amount.
p_percentage	IN number DEFAULT null	Percentage.
p_reason_code	IN varchar2 DEFAULT null	Allowance/charge reason code.
p_reason_text	IN varchar2 DEFAULT null	Allowance/charge reason text.

Example

```
plpdf_factur.AddLineAllowance(1, 5, 100, 5, '95', 'Line discount');
```

3.13 ADDLINECHARGE

Adds a charge to an existing line.

Syntax

```
procedure AddLineCharge(  
  p_line_index pls_integer,  
  p_amount     number,  
  p_base_amount number default null,  
  p_percentage number default null,  
  p_reason_code varchar2 default null,  
  p_reason_text varchar2 default null  
);
```



PL/PDF Factur-X

```
p_reason_text varchar2 default null  
);
```

Parameters

Parameter	Type / default	Description
p_line_index	IN pls_integer	1-based index of the line previously added with ADDLINE.
p_amount	IN number	Allowance/charge amount.
p_base_amount	IN number DEFAULT null	Calculation base amount.
p_percentage	IN number DEFAULT null	Percentage.
p_reason_code	IN varchar2 DEFAULT null	Allowance/charge reason code.
p_reason_text	IN varchar2 DEFAULT null	Allowance/charge reason text.

Example

```
plpdf_factur.AddLineCharge(1, 2, null, null, 'ZZZ', 'Line charge');
```

3.14 ADDVATBREAKDOWN

Adds one VAT breakdown row. Add one row for each VAT category/rate combination represented by the invoice totals.

Syntax

```
procedure AddVatBreakdown(  
    p_category_code    varchar2,  
    p_taxable_amount   number,  
    p_tax_amount       number,  
    p_rate             number default null,  
    p_exemption_reason_code varchar2 default null,  
    p_exemption_reason_text varchar2 default null  
);
```

Parameters

Parameter	Type / default	Description
p_category_code	IN varchar2	VAT category code.
p_taxable_amount	IN number	Taxable amount for the category.
p_tax_amount	IN number	VAT amount for the category.
p_rate	IN number DEFAULT null	VAT rate.
p_exemption_reason_code	IN varchar2 DEFAULT null	Exemption reason code when required.
p_exemption_reason_text	IN varchar2 DEFAULT null	Exemption reason text when required.

Example

```
plpdf_factur.AddVatBreakdown('S', 100, 19, 19);
```



3.15 SETTOTALS

Sets document-level invoice monetary totals.

Syntax

```
procedure SetTotals(  
    p_line_net_total      number,  
    p_tax_basis_total     number,  
    p_grand_total         number,  
    p_due_payable_amount  number,  
    p_allowance_total     number default null,  
    p_charge_total        number default null,  
    p_tax_total           number default null,  
    p_tax_total_accounting_currency number default null,  
    p_paid_amount         number default null,  
    p_rounding_amount     number default null  
);
```

Parameters

Parameter	Type / default	Description
p_line_net_total	IN number	BT-106 sum of invoice line net amounts.
p_tax_basis_total	IN number	BT-109 invoice total amount without VAT.
p_grand_total	IN number	BT-112 invoice total amount with VAT.
p_due_payable_amount	IN number	BT-115 amount due.
p_allowance_total	IN number DEFAULT null	BT-107 document allowance total.
p_charge_total	IN number DEFAULT null	BT-108 document charge total.
p_tax_total	IN number DEFAULT null	BT-110 invoice VAT total.
p_tax_total_accounting_currency	IN number DEFAULT null	BT-111 VAT total in accounting currency.
p_paid_amount	IN number DEFAULT null	BT-113 paid amount.
p_rounding_amount	IN number DEFAULT null	BT-114 rounding amount.

Example

```
plpdf_factur.SetTotals(  
    p_line_net_total      => 100,  
    p_tax_basis_total     => 100,  
    p_tax_total           => 19,  
    p_grand_total         => 119,  
    p_due_payable_amount => 119  
);
```



PL/PDF Factur-X

4. Validation and XML output

4.1 VALIDATE

Validates the current canonical model against the selected profile, supported business rules, invoice-type sign semantics, arithmetic consistency, and the pinned code lists.

Syntax

```
procedure Validate;
```

Parameters

Validation failures are raised as Oracle application errors. Messages include a rule identifier such as BR-, CII-SR-*, FX-*, or FX-CODE-* where applicable.*

Example

```
plpdf_factur.Validate;
```

4.2 GETXML

Returns the current invoice as Factur-X/ZUGFeRD CII XML.

Syntax

```
function GetXML return clob;
```

Parameters

Return value

Caller-owned temporary CLOB. In XML-first mode, if the model has not been mutated since LOADXML, the logical XML content corresponds to the imported source; the exact original source bytes are retained separately for PDF carrier embedding.

Example

```
l_xml := plpdf_factur.GetXML;
-- Consume l_xml here.

if l_xml is not null and dbms_lob.isemporary(l_xml) = 1 then
  dbms_lob.freemporary(l_xml);
end if;
```

4.3 Validation and serialization behavior by profile

MINIMUM and BASIC WL validate and serialize only the structured content represented by those lower profiles; canonical fields used solely for the visual PDF can remain present without being emitted into the CII XML. BASIC reuses the EN 16931 business-rule core and then applies BASIC-specific restrictions. EXTENDED accepts the modeled EN 16931 terms plus the supported EXTENDED profile extensions represented by the public canonical model.

5. XML-first input

5.1 LOADXML

Loads an existing supported Factur-X/ZUGFeRD Cross Industry Invoice XML document into the canonical model. The operation is atomic: parsing or validation failure leaves the previous valid package state unchanged.



PL/PDF Factur-X

Syntax

```
procedure LoadXML(  
  p_xml blob  
);
```

Parameters

Parameter	Type / default	Description
p_xml	IN blob	UTF-8 Factur-X/ZUGFeRD CII XML. Borrowed/read-only; ownership remains with the caller.

LOADXML cannot run while a PDF carrier is currently prepared. Call RELEASEPDFCARRIER first if a previous custom-rendering workflow has not yet been released.

Example

```
plpdf_factur.LoadXML(l_facturx_xml_blob);
```

5.2 Exact source preservation

After a successful LOADXML, PLPDF_FACTUR stores an internal exact copy of the imported XML bytes. PREPAREPDFCARRIER uses those exact source bytes, so an XML-first hybrid PDF can embed the original CII document without reserialization.

Any subsequent data-first mutation - for example SETSELLER, ADDLINE, or SETTOTALS - invalidates that exact source copy. The canonical model remains available, and GETXML/PREPAREPDFCARRIER then serialize the modified model.

5.3 XML-first workflow example

```
declare  
  l_pdf blob;  
begin  
  plpdf_factur.LoadXML(l_facturx_xml_blob);  
  
  -- Optional: inspect the canonical model with GetInvoice/GetSeller/etc.  
  plpdf_factur_invoice_example.Generate(  
    p_pdf      => l_pdf,  
    p_moddate => sysdate  
  );  
  
  -- l_pdf is now a hybrid Factur-X PDF/A-3b document.  
end;
```

6. Factur-X PDF/A-3 generation

Hybrid invoice generation combines the canonical CII XML with a human-readable PDF. PREPAREPDFCARRIER configures the active PL/PDF document for PDF/A-3b, registers the Factur-X XMP extension schema, sets the Factur-X XMP values, and attaches factur-x.xml with the profile-appropriate AFRelationship.

6.1 PREPAREPDFCARRIER

Prepares the active PL/PDF document as a Factur-X/ZUGFeRD PDF/A-3b carrier. Call PLPDF.INIT before this procedure when writing a custom visual renderer.



PL/PDF Factur-X

Syntax

```
procedure PreparePDFCarrier(  
    p_moddate date default null  
);
```

Parameters

Parameter	Type / default	Description
p_moddate	IN date DEFAULT null	Modification date for the embedded factur-x.xml attachment. NULL uses SYSDATE.

The procedure calls VALIDATE. In XML-first mode it embeds the exact imported XML source bytes; in data-first mode it serializes the current canonical model.

Example

```
plpdf_factur.PreparePDFCarrier(p_moddate => sysdate);
```

6.2 RELEASEPDFCARRIER

Releases the internal temporary BLOB that backs the embedded Factur-X XML attachment.

Syntax

```
procedure ReleasePDFCarrier;
```

Parameters

For custom PL/PDF rendering, call RELEASEPDFCARRIER after PLPDF.SENDDOC. Error handlers should also attempt best-effort release before re-raising the original exception.

Example

```
plpdf_factur.ReleasePDFCarrier;
```

6.3 Carrier metadata

Property	Value / behavior
Embedded file name	factur-x.xml
MIME type	text/xml
PDF conformance	PDF/A-3b
XMP namespace	urn:factur-x:pdfa:CrossIndustryDocument:invoice:1p0#
XMP DocumentType	INVOICE
XMP Version	1.0
XMP ConformanceLevel	MINIMUM, BASIC WL, BASIC, EN 16931, or EXTENDED according to the active profile.
AFRelationship	Data for MINIMUM/BASIC WL; Alternative for BASIC/EN 16931/EXTENDED.

6.4 Custom PL/PDF rendering sequence

```
declare  
    l_pdf blob;  
    l_invoice plpdf_factur.t_invoice;
```



PL/PDF Factur-X

```
begin
  plpdf_factur.Validate;
  l_invoice := plpdf_factur.GetInvoice;

  plpdf.Init;
  plpdf_factur.PreparePDFCarrier(p_moddate => sysdate);

  -- Configure an embedded font, then generate the visible invoice.
  plpdf.NewPage;
  plpdf.SetPrintFont('DejaVuSans', null, 10);
  plpdf.PrintCell(
    p_w  => 180,
    p_h  => 6,
    p_txt => 'INVOICE ' || l_invoice.invoice_id,
    p_ln  => 1
  );
  -- ... custom visual content ...

  plpdf.SendDoc(
    p_blob  => l_pdf,
    p_fretemp => false
  );

  plpdf_factur.ReleasePDFCarrier;
exception
  when others then
    begin
      plpdf_factur.ReleasePDFCarrier;
    exception
      when others then
        null; -- best-effort cleanup; preserve the original error
      end;
    raise;
end;
```

6.5 Reference invoice renderer

The optional `PLPDF_FACTUR_INVOICE_EXAMPLE` package is a minimal reference renderer. It validates the current `PLPDF_FACTUR` state, reads only the public canonical getter API, initializes PL/PDF, prepares the Factur-X carrier, writes a simple visible invoice, finalizes the PDF, and releases the carrier.

Syntax

```
procedure plpdf_factur_invoice_example.Generate(
  p_pdf      out nocopy blob,
  p_moddate  date default null
);
```

Example

```
plpdf_factur_invoice_example.Generate(
  p_pdf      => l_pdf,
  p_moddate  => sysdate
);
```

7. Renderer integration and read-only canonical model

The getter functions expose read-only snapshots of the current canonical model. They are intended for visual invoice renderers and application inspection. Getter indices are 1-based and invalid indices raise an application error.



7.1 GETINVOICE

Returns the current T_INVOICE snapshot.

Syntax

```
function GetInvoice return t_invoice;
```

Parameters

Return value

T_INVOICE record.

Example

```
l_invoice := plpdf_factur.GetInvoice;
```

7.2 GETSELLER

Returns the current seller T_PARTY snapshot.

Syntax

```
function GetSeller return t_party;
```

Parameters

Return value

T_PARTY record.

Example

```
l_seller := plpdf_factur.GetSeller;
```

7.3 GETBUYER

Returns the current buyer T_PARTY snapshot.

Syntax

```
function GetBuyer return t_party;
```

Parameters

Return value

T_PARTY record.

Example

```
l_buyer := plpdf_factur.GetBuyer;
```

7.4 GETREFERENCES

Returns document-level references.

Syntax

```
function GetReferences return t_references;
```

Parameters

Return value

T_REFERENCES record.

Example

```
l_refs := plpdf_factur.GetReferences;
```



7.5 GETPAYMENT

Returns payment information.

Syntax

```
function GetPayment return t_payment;
```

Parameters

Return value

T_PAYMENT record.

Example

```
l_payment := plpdf_factur.GetPayment;
```

7.6 GETTOTALS

Returns invoice monetary totals.

Syntax

```
function GetTotals return t_totals;
```

Parameters

Return value

T_TOTALS record.

Example

```
l_totals := plpdf_factur.GetTotals;
```

7.7 GETPRECEDINGINVOICECOUNT

Returns the number of preceding invoice references.

Syntax

```
function GetPrecedingInvoiceCount return pls_integer;
```

Parameters

Return value

Number of T_PRECEDING_INVOICE records.

Example

```
for i in 1 .. plpdf_factur.GetPrecedingInvoiceCount loop  
  null;  
end loop;
```

7.8 GETPRECEDINGINVOICE

Returns one preceding invoice reference.

Syntax

```
function GetPrecedingInvoice(p_index pls_integer) return t_preceding_invoice;
```

Parameters

Parameter	Type / default	Description
p_index	IN pls_integer	1-based index in the corresponding canonical collection.

**Return value**

T_PRECEDING_INVOICE record.

Example

```
l_prev := plpdf_factur.GetPrecedingInvoice(1);
```

7.9 GETNOTECOUNT

Returns the number of invoice notes.

Syntax

```
function GetNoteCount return pls_integer;
```

Parameters**Return value**

Number of T_NOTE records.

Example

```
l_count := plpdf_factur.GetNoteCount;
```

7.10 GETNOTE

Returns one invoice note.

Syntax

```
function GetNote(p_index pls_integer) return t_note;
```

Parameters

Parameter	Type / default	Description
p_index	IN pls_integer	1-based index in the corresponding canonical collection.

Return value

T_NOTE record.

Example

```
l_note := plpdf_factur.GetNote(1);
```

7.11 GETLINECOUNT

Returns the number of canonical invoice lines.

Syntax

```
function GetLineCount return pls_integer;
```

Parameters**Return value**

Number of T_LINE records.

Example

```
l_count := plpdf_factur.GetLineCount;
```

7.12 GETLINE

Returns one canonical invoice line.



PL/PDF Factur-X

Syntax

```
function GetLine(p_index pls_integer) return t_line;
```

Parameters

Parameter	Type / default	Description
p_index	IN pls_integer	1-based index in the corresponding canonical collection.

Return value

T_LINE record.

Example

```
l_line := plpdf_factur.GetLine(1);
```

7.13 GETADJUSTMENTCOUNT

Returns the total number of canonical allowances and charges, including document- and line-level adjustments.

Syntax

```
function GetAdjustmentCount return pls_integer;
```

Parameters

Return value

Number of T_ADJUSTMENT records.

Example

```
l_count := plpdf_factur.GetAdjustmentCount;
```

7.14 GETADJUSTMENT

Returns one canonical allowance/charge record.

Syntax

```
function GetAdjustment(p_index pls_integer) return t_adjustment;
```

Parameters

Parameter	Type / default	Description
p_index	IN pls_integer	1-based index in the corresponding canonical collection.

Return value

T_ADJUSTMENT record.

Example

```
l_adj := plpdf_factur.GetAdjustment(1);
```

7.15 GETVATBREAKDOWNCOUNT

Returns the number of VAT breakdown rows.

Syntax

```
function GetVatBreakdownCount return pls_integer;
```



PL/PDF Factur-X

Parameters

Return value

Number of T_VAT_BREAKDOWN records.

Example

```
l_count := plpdf_factur.GetVatBreakdownCount;
```

7.16 GETVATBREAKDOWN

Returns one VAT breakdown row.

Syntax

```
function GetVatBreakdown(p_index pls_integer) return t_vat_breakdown;
```

Parameters

Parameter	Type / default	Description
p_index	IN pls_integer	1-based index in the corresponding canonical collection.

Return value

T_VAT_BREAKDOWN record.

Example

```
l_vat := plpdf_factur.GetVatBreakdown(1);
```

7.17 Renderer iteration example

```
declare
  l_invoice plpdf_factur.t_invoice;
  l_line    plpdf_factur.t_line;
begin
  l_invoice := plpdf_factur.GetInvoice;

  for i in 1 .. plpdf_factur.GetLineCount loop
    l_line := plpdf_factur.GetLine(i);
    -- Render l_line.item_name, quantity, prices, VAT, etc.
  end loop;
end;
```

8. End-to-end examples and diagnostics

8.1 EN 16931 data-first hybrid invoice

```
declare
  l_pdf blob;
begin
  plpdf_factur.Init(plpdf_factur.c_profile_en16931);

  plpdf_factur.SetInvoice(
    p_invoice_id      => 'INV-2026-001',
    p_issue_date      => date '2026-08-29',
    p_invoice_type_code => plpdf_factur.c_invoice_type_commercial,
    p_currency_code   => 'EUR',
    p_buyer_reference  => 'PO-4711'
  );

  plpdf_factur.SetSeller(
    p_name      => 'Example Seller GmbH',
    p_country_code => 'DE',
  );
```



PL/PDF Factur-X

```
p_vat_id      => 'DE123456789',
p_address_line1 => 'Example Strasse 1',
p_postcode    => '10115',
p_city        => 'Berlin'
);

plpdf_factur.SetBuyer(
  p_name      => 'Example Buyer Kft.',
  p_country_code => 'HU',
  p_address_line1 => 'Minta utca 1.',
  p_postcode    => '1111',
  p_city        => 'Budapest'
);

plpdf_factur.AddLine(
  p_line_id      => '1',
  p_item_name     => 'Software licence',
  p_quantity      => 1,
  p_unit_code     => 'C62',
  p_net_unit_price => 100,
  p_vat_category_code => 'S',
  p_vat_rate      => 19,
  p_line_net_amount => 100
);

plpdf_factur.AddVatBreakdown('S', 100, 19, 19);

plpdf_factur.SetPayment(
  p_means_code      => '58',
  p_payment_account_id => 'DE0212030000000002051',
  p_due_date        => date '2026-09-28'
);

plpdf_factur.SetTotals(
  p_line_net_total    => 100,
  p_tax_basis_total   => 100,
  p_tax_total         => 19,
  p_grand_total       => 119,
  p_due_payable_amount => 119
);

plpdf_factur.Validate;

plpdf_factur_invoice_example.Generate(
  p_pdf      => l_pdf,
  p_moddate  => sysdate
);
end;
```

8.2 Credit note example

For document types 381 and 261, monetary totals remain positive. Use ADDPRECEDINGINVOICE when the business case references the invoice being credited.

```
plpdf_factur.Init(plpdf_factur.c_profile_en16931);
plpdf_factur.SetInvoice(
  p_invoice_id      => 'CN-2026-001',
  p_issue_date      => date '2026-08-29',
  p_invoice_type_code => plpdf_factur.c_invoice_type_credit_note,
  p_currency_code   => 'EUR'
);
plpdf_factur.AddPrecedingInvoice('INV-2026-001', date '2026-08-01');
-- Add seller, buyer, positive credit-note line/tax/totals, then Validate/Generate.
```



PL/PDF Factur-X

8.3 Negative invoice example

A negative invoice uses an invoice-like BT-3 code and negative quantity/tax/totals while unit price remains a normal positive unit price. The package accepts negative totals only for the supported negative-invoice types.

```
plpdf_factur.SetInvoice(  
  p_invoice_id      => 'INV-NEG-001',  
  p_issue_date      => date '2026-08-29',  
  p_invoice_type_code => plpdf_factur.c_invoice_type_commercial,  
  p_currency_code   => 'EUR'  
);  
plpdf_factur.AddLine(  
  p_line_id      => '1',  
  p_item_name     => 'Correction',  
  p_quantity      => -1,  
  p_unit_code     => 'C62',  
  p_net_unit_price => 100,  
  p_vat_category_code => 'S',  
  p_vat_rate      => 19,  
  p_line_net_amount => -100  
);  
plpdf_factur.AddVatBreakdown('S', -100, -19, 19);  
plpdf_factur.SetTotals(-100, -100, -119, -119, p_tax_total => -19);
```

8.4 Common application errors

Oracle code	Category	Meaning
-20501	Dependency	Installed PL/PDF version is older than the required baseline.
-20502	Profile	Invalid profile passed to INIT.
-20503	State	Operation requires an initialized/loaded invoice model.
-20504	Index	Invalid 1-based getter or line-adjustment index.
-20505	Profile	Requested validation/carrier behavior is not implemented for the active profile.
-20510	Validation	Business-rule, arithmetic, invoice-type, or code-list validation failure.
-20511	Serialization	CII XML serialization failure.
-20512	Carrier	Factur-X PDF carrier preparation failure.
-20513	XML input	LOADXML input/parsing/state error.

8.5 Recommended error-handling pattern

```
begin  
  -- build or load the PLPDF_FACTUR model  
  plpdf_factur.Validate;  
  plpdf_factur_invoice_example.Generate(l_pdf, sysdate);  
exception  
  when others then  
    -- Log SQLCODE/SQLERRM. Do not replace the original PLPDF_FACTUR error.  
    raise;  
end;
```



PL/PDF Factur-X

8.6 Validation scope and external acceptance

The package validates the supported public canonical model before XML or hybrid-PDF generation. Release acceptance for this baseline was exercised across MINIMUM, BASIC WL, BASIC, EN 16931, EXTENDED, credit-note, and negative-invoice artifacts. Applications should still validate their own production documents in the target interoperability environment and keep profile/code-list selection aligned with their business and regulatory requirements.