eco East Coast Oracle Users Conference

Eastern States OATUG
ORACLE APPLICATIONS & TECHNOLOGY USERS GROUP

Leveraging XML & APIs to Import/Export Data for your Business Users

Tammy Vandermey
Sr. Technical Consultant
O2Works, LLC

Agenda

- Introduction
- XML and APIs
- Techniques
  - Process Wrapping
  - Notifications and Flat File Generation
  - Dynamic Boiler-plating
  - OAF Extension Reporting
  - Bursting to XML
- Q & A

Introduction

- Tammy Vandermey
  - Technical Oracle Consultant, O2Works, LLC
  - 25+ Yrs technical implementation experience in Oracle
  - Contact Information: tammy@o2works.com

About O2Works

O2Works is one of the leading Oracle application service providers offering the most experienced teams of functional and technical consultants in the industry. Our hands-on ***resources average 24+ years of experience*** focused exclusively on implementing, upgrading, integrating, and extending Oracle's EBS and Fusion cloud applications.

O2Works™
Putting Oracle To Work

ORACLE | Partner

Presentations, White Papers, and other information shared at: https://o2works.com/knowledge-works/

eco East Coast Oracle Users Conference

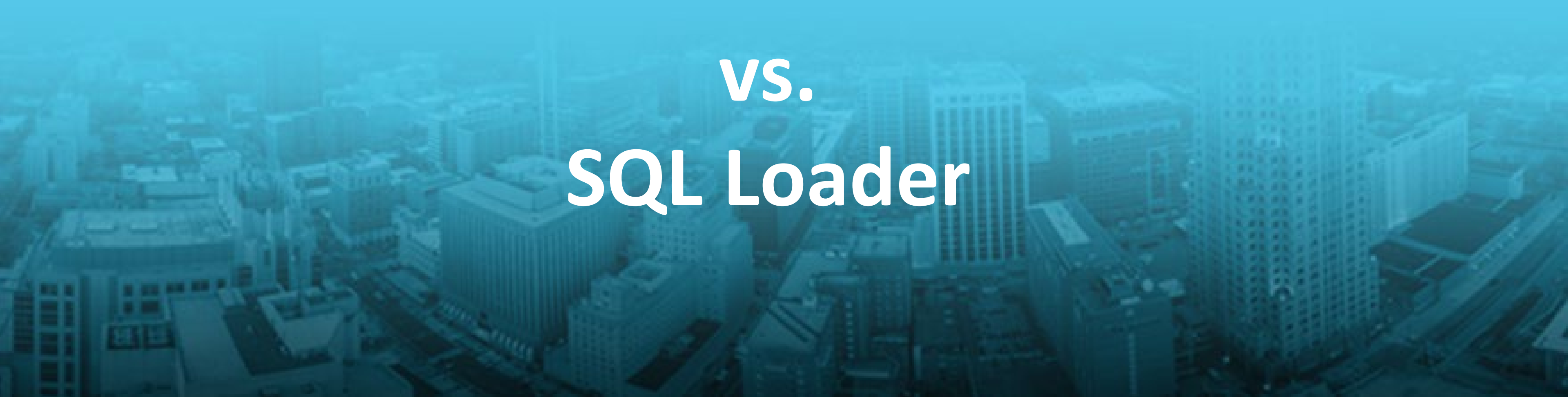
Eastern States OATUG ORACLE APPLICATIONS & TECHNOLOGY USERS GROUP

APIs and XML

eco East Coast Oracle Users Conference

Eastern States OATUG ORACLE APPLICATIONS & TECHNOLOGY USERS GROUP

External Tables vs. SQL Loader



External Tables vs. SQL Loader

SQL Loader

- Collect Data (csv/txt files)
- Create Staging Table
- Create Load Control File
- Run SQL Loader program to load file into table
- Review logs to verify success

External Table

- Create External Table
- Collect Data (csv/txt files
- Execute SQL to read data from table

External Tables vs SQL Loader

```
CREATE TABLE xxtst.xxoaug_test_ext_tbl(
id                          VARCHAR2(200),
name                        VARCHAR2(200),
account_number              VARCHAR2(200)
)
ORGANIZATION external
   (TYPE oracle_loader
    default directory XXTST_AR_CNV_DIR
    access parameters (records delimited by '\r\n'
                       characterset we8mswin1252
                       nologfile
                       skip 1
                       fields terminated by '|'
                       optionally enclosed by '"'
                       missing field values are null
                       )
                       location ('oaug_test_file.csv')
                       )
reject limit 0;
```

External Tables vs SQL Loader

External Table

- Use DBA_DIRECTORIES for determining file location
- File resides on Database Server
- Logfiles are produced when read errors occur

eco East Coast Oracle Users Conference

Eastern States OATUG
ORACLE APPLICATIONS & TECHNOLOGY USERS GROUP

Execution Modes Validate/Commit

Execution Modes

Autonomous Transactions

- Allows for the capture of what ‘would have occurred’ before committing to the Application Instance
- Prevents the need for repetitive and time-consuming database refreshes to re-load data when issues occur in Testing

Autonomous Transaction

```
PROCEDURE update_stg_ids(p_update_stg_rec      xxoaug_test_stg_tbl%ROWTYPE) IS
   PRAGMA AUTONOMOUS_TRANSACTION;
  BEGIN
    UPDATE xxoaug_test_stg_tbl(
         SET load_status                    = p_update_stg_rec.load_status,
               cust_account_id              = p_update_stg_rec.cust_account_id
     WHERE id = p_update_stg.id;

     COMMIT;
  END;
```

eco East Coast Oracle Users Conference

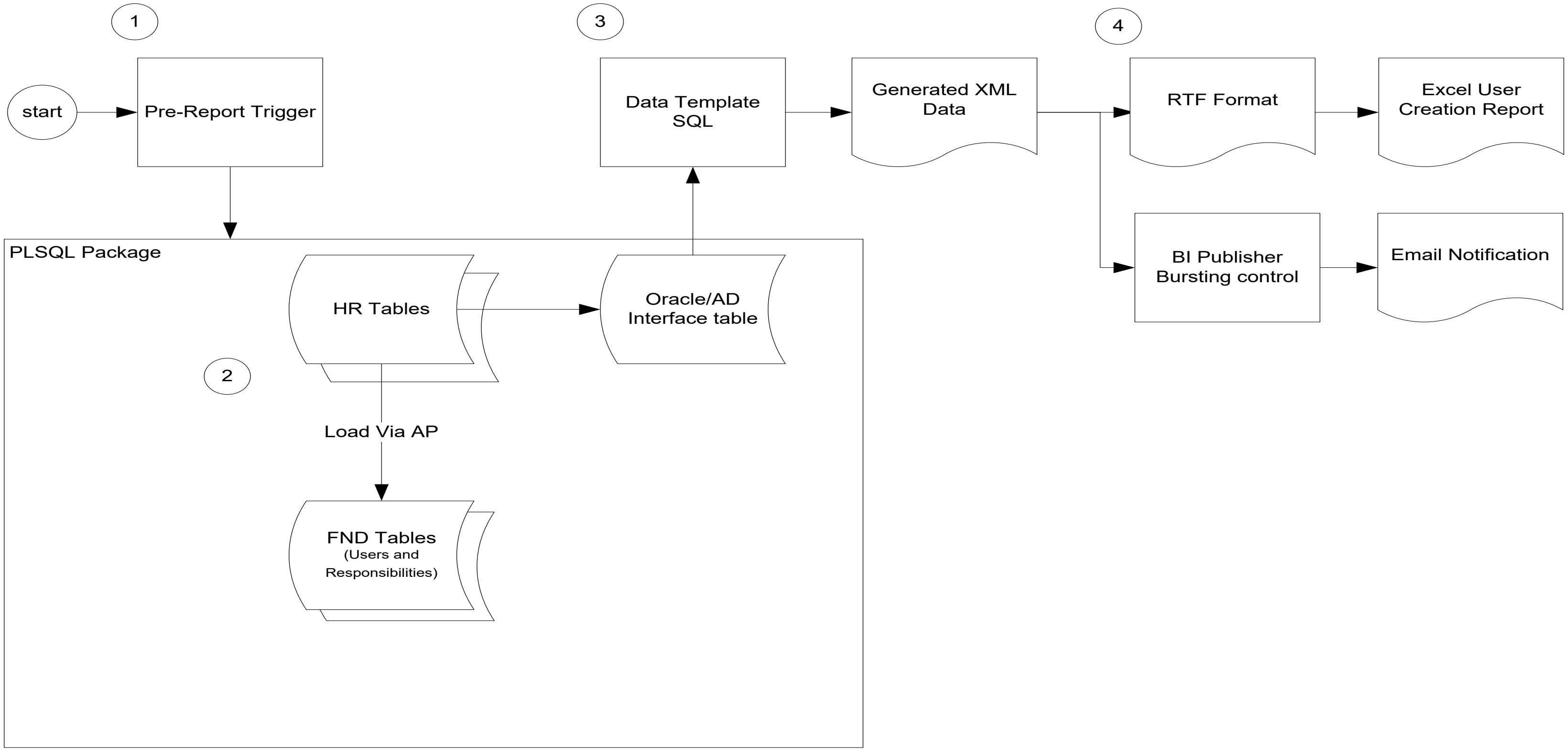
Eastern States OATUG ORACLE APPLICATIONS & TECHNOLOGY USERS GROUP

XML Process Wrapping

Process Wrapping

- BI Publisher is more than just a reprint tool.
- Can be used to ‘wrap’ entire processes, putting all functionality into one tidy ‘package’ of code
  - Before and after report triggers
  - Process Exception reporting
  - Notifications thru bursting

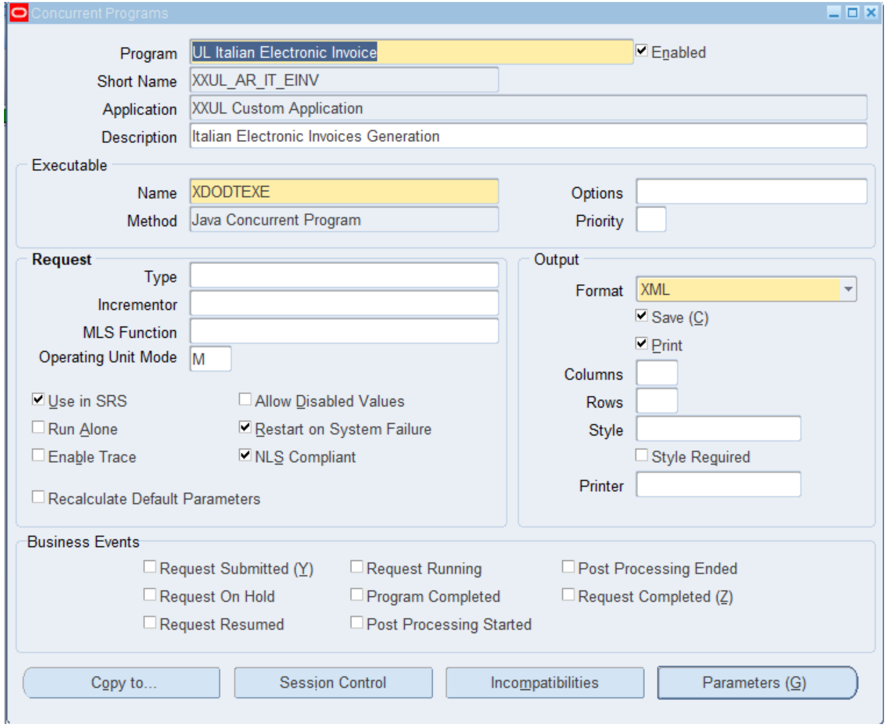
Overall Flow of Transaction Processing



Process Wrapping

Smaller Code Signature/Maintenance

- No longer need separate concurrent processes and executables
- Concurrent Programs Setup uses XDODTEXE executable to run



Concurrent Programs

Program: UL Italian Electronic Invoice ☑ Enabled
Short Name: XXUL_AR_IT_EINV
Application: XXUL Custom Application
Description: Italian Electronic Invoices Generation

Executable
Name: XDODTEXE
Options:
Method: Java Concurrent Program
Priority:

Request
Type:
Incrementor:
MLS Function:
Operating Unit Mode: M

☑ Use in SRS
☐ Allow Disabled Values
☐ Run Alone
☑ Restart on System Failure
☐ Enable Trace
☑ NLS Compliant
☐ Recalculate Default Parameters

Output
Format: XML
☑ Save (C)
☑ Print
Columns:
Rows:
Style:
☐ Style Required
Printer:

Business Events
☐ Request Submitted (Y)
☐ Request Running
☐ Post Processing Ended
☐ Request On Hold
☐ Program Completed
☐ Request Completed (Z)
☐ Request Resumed
☐ Post Processing Started

Process Wrapping

Benefits of XML Processing for Transaction Loads

- Before Report Triggers to call Oracle APIs to load data into staging.
- Autonomous writes to staging/logging allows for reporting of results back to the user.
- Multi-threaded processing supported with the use of control procedures in the function call.
- Post Process response files can be accommodated thru BI Publisher Bursting
- Exception notification can also be handled thru Bursting

Process Wrapping

Tips and Considerations

DON’T

- Create new concurrent program executables for every process
- Re-invent the wheel and write your own notification engine for exceptions

DO

- Use BI Publisher to ‘wrap’ your processes
- Use BI Publisher to generate well-formed error reports instead of relying on concurrent manager log files
- Use BI Publisher to expand exception notification processing in critical processes

eco East Coast Oracle Users Conference

Eastern States OATUG
ORACLE APPLICATIONS & TECHNOLOGY USERS GROUP

Notifications and Flat File Generation

Notifications and Flat File Generation

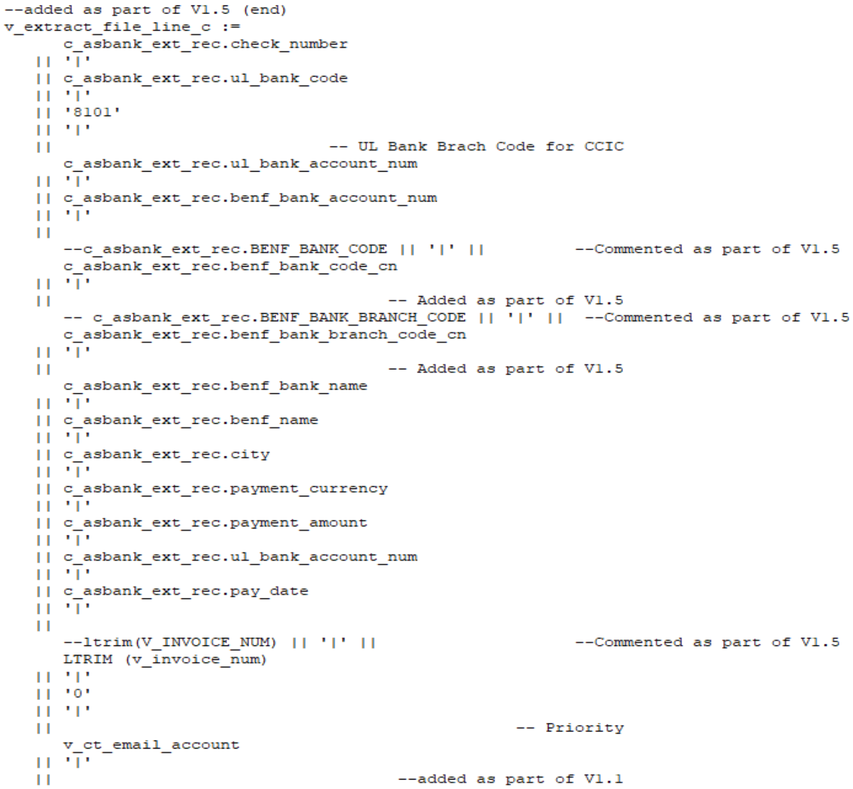
Option 1 – Code in PL/SQL

- Use UTL_FILE or FND_FILE to generate flat files specific to the business purpose
- To send file via email, use UTL_SMTP

- These functions can lead to excessive and unnecessary code maintenance issues

Notifications and Flat File Generation

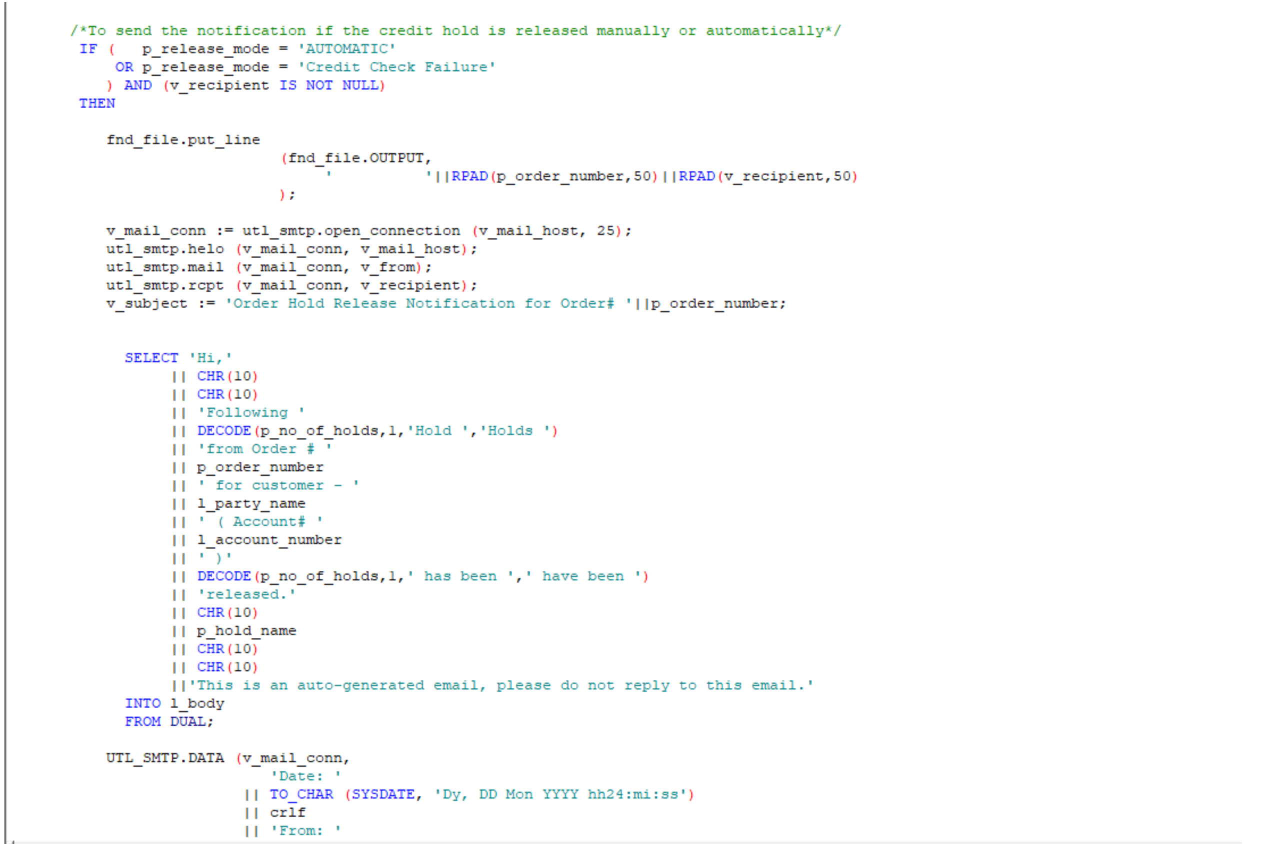
Flat File Example – Typical Code



```
--added as part of V1.5 (end)
v_extract_file_line_c :=
      c_asbank_ext_rec.check_number
   || '|'
   || c_asbank_ext_rec.ul_bank_code
   || '|'
   || '8101'
   || '|'
   ||                          -- UL Bank Brach Code for CCIC
      c_asbank_ext_rec.ul_bank_account_num
   || '|'
   || c_asbank_ext_rec.benf_bank_account_num
   || '|'
   ||
      --c_asbank_ext_rec.BENF_BANK_CODE || '|' ||         --Commented as part of V1.5
      c_asbank_ext_rec.benf_bank_code_cn
   || '|'
   ||                                -- Added as part of V1.5
      -- c_asbank_ext_rec.BENF_BANK_BRANCH_CODE || '|' ||  --Commented as part of V1.5
      c_asbank_ext_rec.benf_bank_branch_code_cn
   || '|'
   ||                                -- Added as part of V1.5
      c_asbank_ext_rec.benf_bank_name
   || '|'
   || c_asbank_ext_rec.benf_name
   || '|'
   || c_asbank_ext_rec.city
   || '|'
   || c_asbank_ext_rec.payment_currency
   || '|'
   || c_asbank_ext_rec.payment_amount
   || '|'
   || c_asbank_ext_rec.ul_bank_account_num
   || '|'
   || c_asbank_ext_rec.pay_date
   || '|'
   ||
      --ltrim(V_INVOICE_NUM) || '|' ||                    --Commented as part of V1.5
      LTRIM (v_invoice_num)
   || '|'
   || '0'
   || '|'
   ||                                            -- Priority
      v_ct_email_account
   || '|'
   ||                                 --added as part of V1.1
```

Notifications and Flat File Generation

Email Example - Typical Code



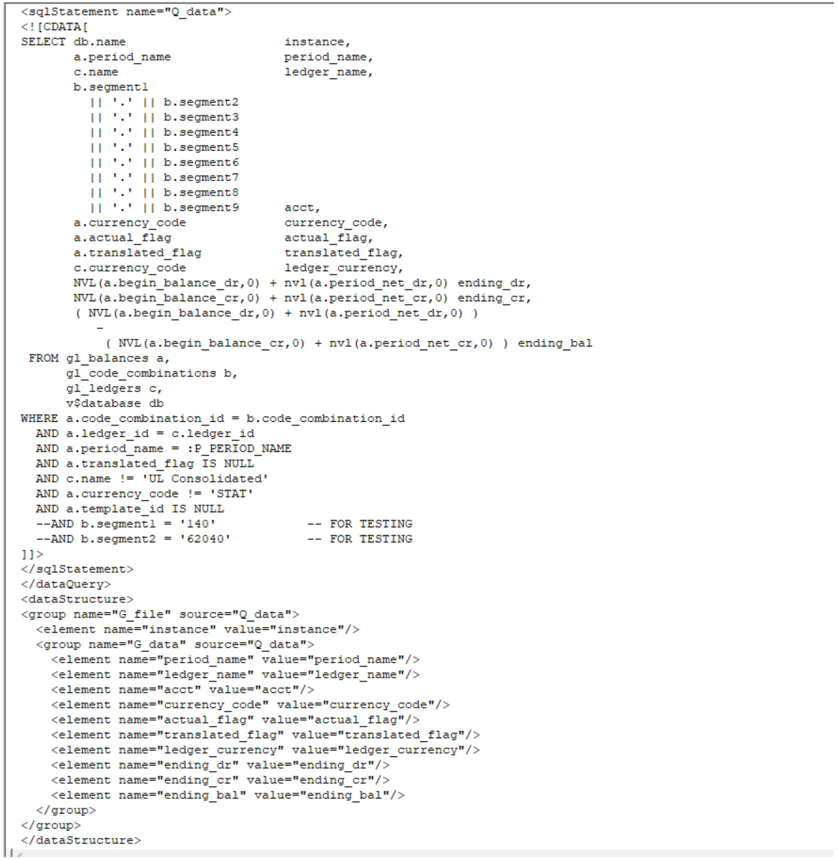
```
/*To send the notification if the credit hold is released manually or automatically*/
 IF (   p_release_mode = 'AUTOMATIC'
     OR p_release_mode = 'Credit Check Failure'
    ) AND (v_recipient IS NOT NULL)
 THEN

    fnd_file.put_line
                       (fnd_file.OUTPUT,
                          '          '||RPAD(p_order_number,50)||RPAD(v_recipient,50)
                       );

    v_mail_conn := utl_smtp.open_connection (v_mail_host, 25);
    utl_smtp.helo (v_mail_conn, v_mail_host);
    utl_smtp.mail (v_mail_conn, v_from);
    utl_smtp.rcpt (v_mail_conn, v_recipient);
    v_subject := 'Order Hold Release Notification for Order# '||p_order_number;


      SELECT 'Hi,'
           || CHR(10)
           || CHR(10)
           || 'Following '
           || DECODE(p_no_of_holds,1,'Hold ','Holds ')
           || 'from Order # '
           || p_order_number
           || ' for customer - '
           || l_party_name
           || ' ( Account# '
           || l_account_number
           || ' )'
           || DECODE(p_no_of_holds,1,' has been ',' have been ')
           || 'released.'
           || CHR(10)
           || p_hold_name
           || CHR(10)
           || CHR(10)
           ||'This is an auto-generated email, please do not reply to this email.'
      INTO l_body
      FROM DUAL;

    UTL_SMTP.DATA (v_mail_conn,
                      'Date: '
                   || TO_CHAR (SYSDATE, 'Dy, DD Mon YYYY hh24:mi:ss')
                   || crlf
                   || 'From: '
```

Notifications and Flat File Generation

Option 2 – Use BI Publisher

- SQL Code is placed in a simple Data Template
- Create an eText RTF template to format
- Burst the output to a designated location using a Bursting Control File

Notifications and Flat File Generation

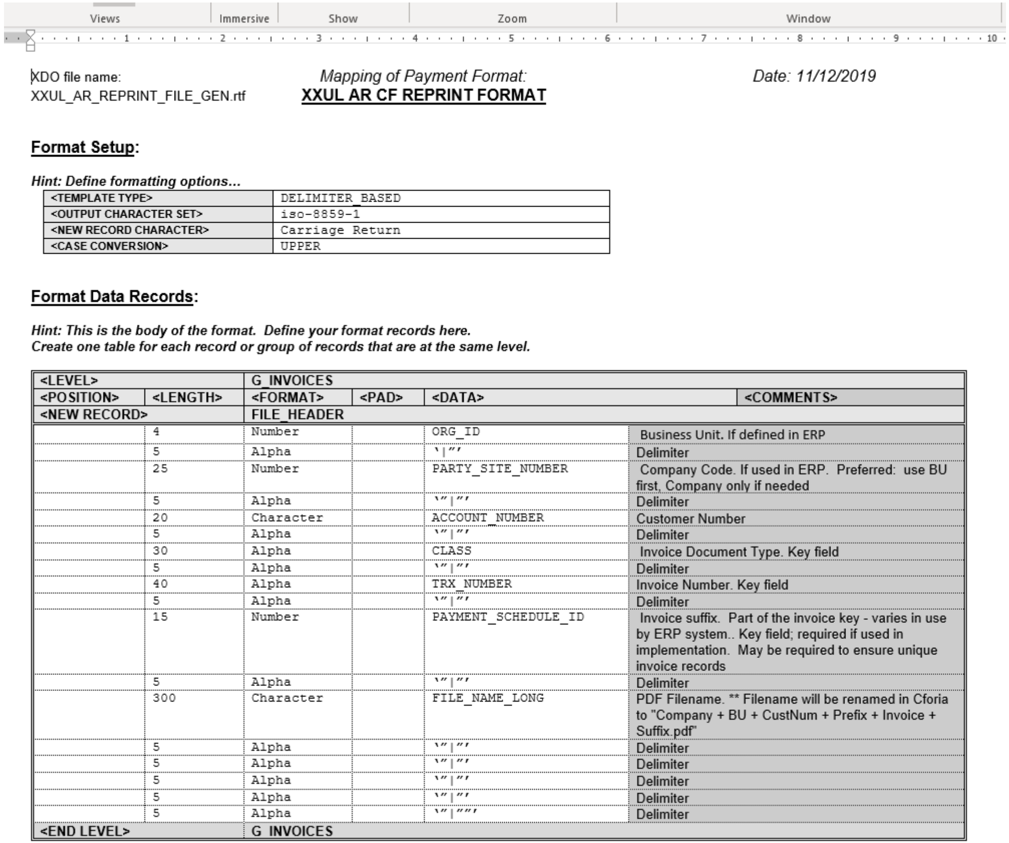


```
<sqlStatement name="Q_data">
<![CDATA[
SELECT db.name                     instance,
       a.period_name               period_name,
       c.name                      ledger_name,
       b.segment1
         || '.' || b.segment2
         || '.' || b.segment3
         || '.' || b.segment4
         || '.' || b.segment5
         || '.' || b.segment6
         || '.' || b.segment7
         || '.' || b.segment8
         || '.' || b.segment9      acct,
       a.currency_code             currency_code,
       a.actual_flag               actual_flag,
       a.translated_flag           translated_flag,
       c.currency_code             ledger_currency,
       NVL(a.begin_balance_dr,0) + nvl(a.period_net_dr,0) ending_dr,
       NVL(a.begin_balance_cr,0) + nvl(a.period_net_cr,0) ending_cr,
       ( NVL(a.begin_balance_dr,0) + nvl(a.period_net_dr,0) )
          -
           ( NVL(a.begin_balance_cr,0) + nvl(a.period_net_cr,0) ) ending_bal
 FROM gl_balances a,
      gl_code_combinations b,
      gl_ledgers c,
      v$database db
WHERE a.code_combination_id = b.code_combination_id
  AND a.ledger_id = c.ledger_id
  AND a.period_name = :P_PERIOD_NAME
  AND a.translated_flag IS NULL
  AND c.name != 'UL Consolidated'
  AND a.currency_code != 'STAT'
  AND a.template_id IS NULL
  --AND b.segment1 = '140'           -- FOR TESTING
  --AND b.segment2 = '62040'         -- FOR TESTING
]]>
</sqlStatement>
</dataQuery>
<dataStructure>
<group name="G_file" source="Q_data">
  <element name="instance" value="instance"/>
  <group name="G_data" source="Q_data">
    <element name="period_name" value="period_name"/>
    <element name="ledger_name" value="ledger_name"/>
    <element name="acct" value="acct"/>
    <element name="currency_code" value="currency_code"/>
    <element name="actual_flag" value="actual_flag"/>
    <element name="translated_flag" value="translated_flag"/>
    <element name="ledger_currency" value="ledger_currency"/>
    <element name="ending_dr" value="ending_dr"/>
    <element name="ending_cr" value="ending_cr"/>
    <element name="ending_bal" value="ending_bal"/>
  </group>
</group>
</dataStructure>
```

Flat File Generation

Data Template

Notifications and Flat File Generation



XDO file name:
XXUL_AR_REPRINT_FILE_GEN.rtf

Mapping of Payment Format:
XXUL AR CF REPRINT FORMAT

Date: 11/12/2019

Format Setup:

Hint: Define formatting options…

<TEMPLATE TYPE>	DELIMITER_BASED
<OUTPUT CHARACTER SET>	iso-8859-1
<NEW RECORD CHARACTER>	Carriage Return
<CASE CONVERSION>	UPPER

Format Data Records:

Hint: This is the body of the format. Define your format records here.
Create one table for each record or group of records that are at the same level.

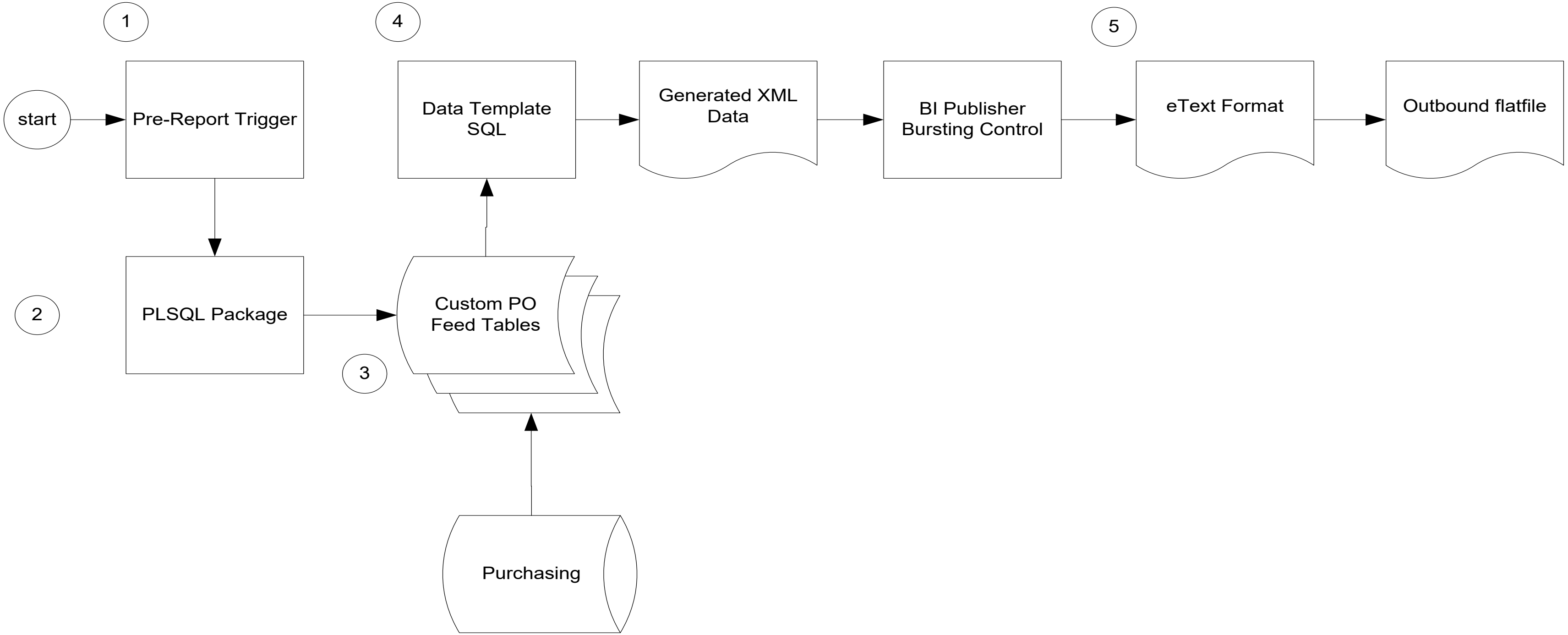
<LEVEL>		G_INVOICES			
<POSITION>	<LENGTH>	<FORMAT>	<PAD>	<DATA>	<COMMENTS>
<NEW RECORD>		FILE_HEADER			
	4	Number		ORG_ID	Business Unit. If defined in ERP
	5	Alpha		‘|”’	Delimiter
	25	Number		PARTY_SITE_NUMBER	Company Code. If used in ERP. Preferred: use BU first, Company only if needed
	5	Alpha		‘”|”’	Delimiter
	20	Character		ACCOUNT_NUMBER	Customer Number
	5	Alpha		‘”|”’	Delimiter
	30	Alpha		CLASS	Invoice Document Type. Key field
	5	Alpha		‘”|”’	Delimiter
	40	Alpha		TRX_NUMBER	Invoice Number. Key field
	5	Alpha		‘”|”’	Delimiter
	15	Number		PAYMENT_SCHEDULE_ID	Invoice suffix. Part of the invoice key - varies in use by ERP system.. Key field; required if used in implementation. May be required to ensure unique invoice records
	5	Alpha		‘”|”’	Delimiter
	300	Character		FILE_NAME_LONG	PDF Filename. ** Filename will be renamed in Cforia to "Company + BU + CustNum + Prefix + Invoice + Suffix.pdf"
	5	Alpha		‘”|”’	Delimiter
	5	Alpha		‘”|”’	Delimiter
	5	Alpha		‘”|”’	Delimiter
	5	Alpha		‘”|”’	Delimiter
	5	Alpha		‘”|””’	Delimiter
<END LEVEL>		G_INVOICES			

Flat File Generation

eText Format

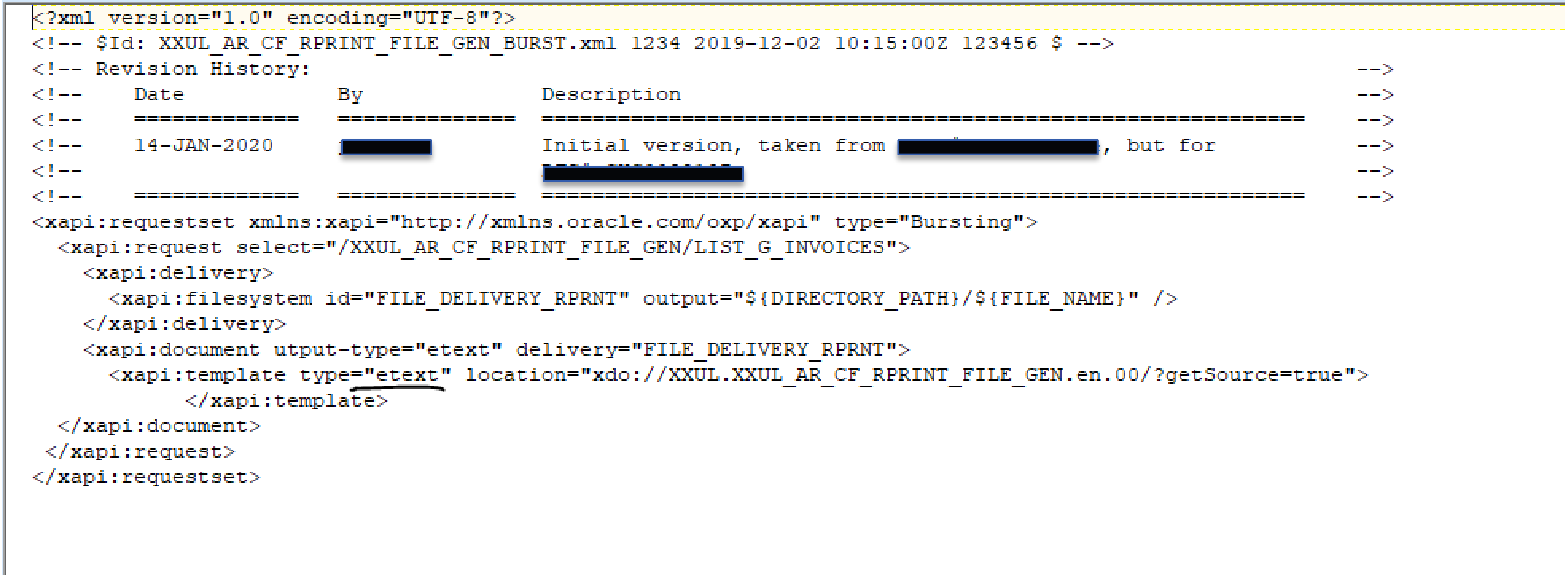
Notifications and Flat File Generation

Flat File Generation



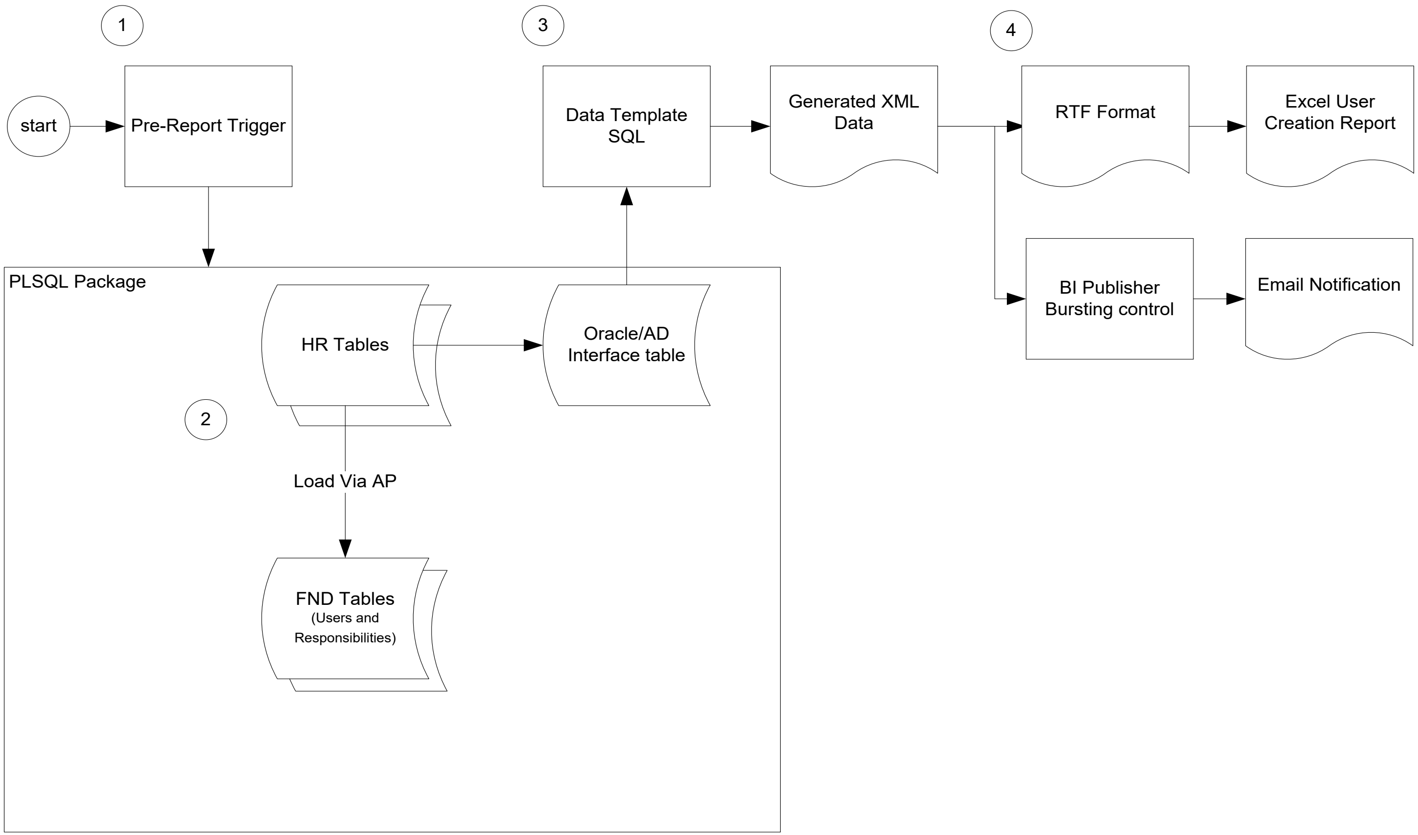
Notifications and Flat File Generation

Flat File Generation - Bursting Control File

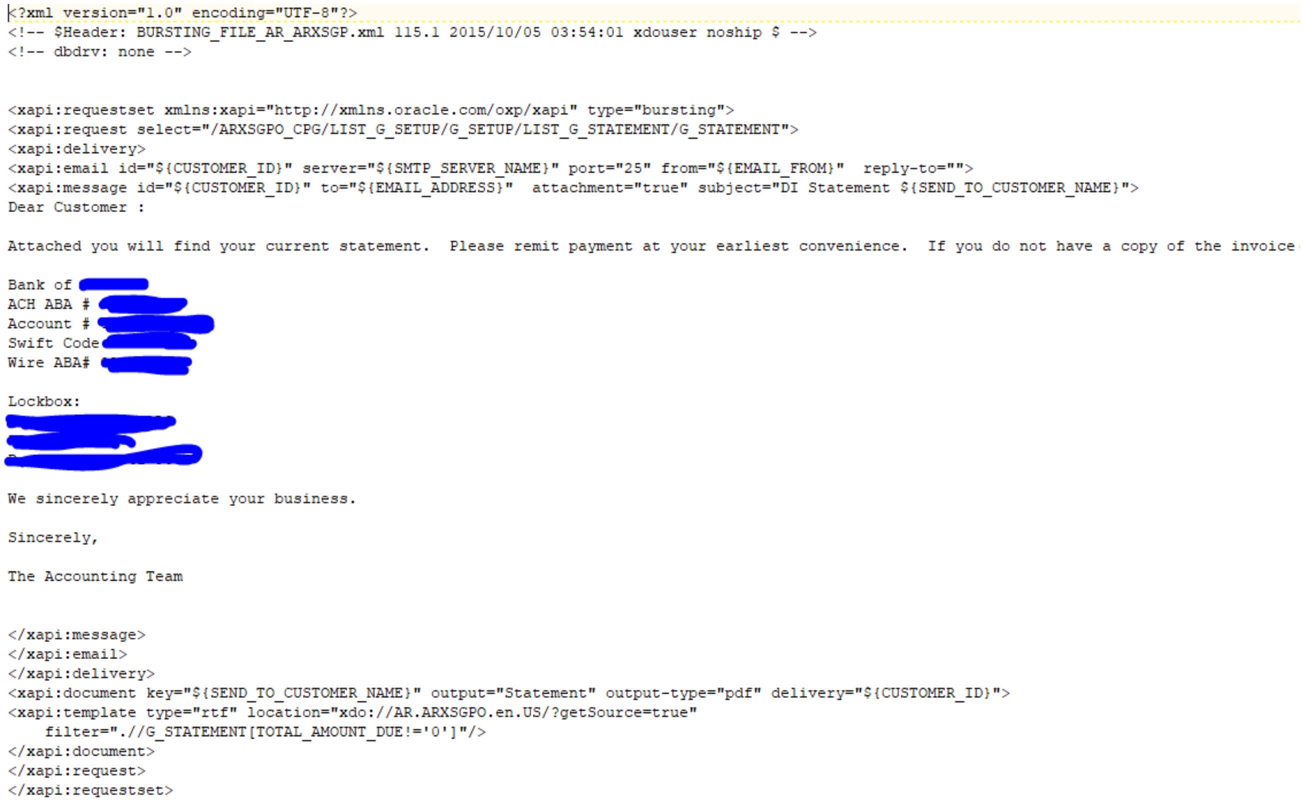


```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Id: XXUL_AR_CF_RPRINT_FILE_GEN_BURST.xml 1234 2019-12-02 10:15:00Z 123456 $ -->
<!-- Revision History:                                                                                  -->
<!--    Date            By              Description                                                     -->
<!--    =============   ==============  ============================================================    -->
<!--    14-JAN-2020     [redacted]      Initial version, taken from [redacted], but for                  -->
<!--                                    [redacted]                                                      -->
<!--    =============   ==============  ============================================================    -->
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi" type="Bursting">
  <xapi:request select="/XXUL_AR_CF_RPRINT_FILE_GEN/LIST_G_INVOICES">
    <xapi:delivery>
      <xapi:filesystem id="FILE_DELIVERY_RPRNT" output="${DIRECTORY_PATH}/${FILE_NAME}" />
    </xapi:delivery>
    <xapi:document utput-type="etext" delivery="FILE_DELIVERY_RPRNT">
      <xapi:template type="etext" location="xdo://XXUL.XXUL_AR_CF_RPRINT_FILE_GEN.en.00/?getSource=true">
            </xapi:template>
  </xapi:document>
 </xapi:request>
</xapi:requestset>
```

Notifications and Flat File Generation



Notifications and Flat File Generation



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- $Header: BURSTING_FILE_AR_ARXSGP.xml 115.1 2015/10/05 03:54:01 xdouser noship $ -->
<!-- dbdrv: none -->


<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi" type="bursting">
<xapi:request select="/ARXSGPO_CPG/LIST_G_SETUP/G_SETUP/LIST_G_STATEMENT/G_STATEMENT">
<xapi:delivery>
<xapi:email id="${CUSTOMER_ID}" server="${SMTP_SERVER_NAME}" port="25" from="${EMAIL_FROM}"  reply-to="">
<xapi:message id="${CUSTOMER_ID}" to="${EMAIL_ADDRESS}"  attachment="true" subject="DI Statement ${SEND_TO_CUSTOMER_NAME}">
Dear Customer :

Attached you will find your current statement.  Please remit payment at your earliest convenience.  If you do not have a copy of the invoice

Bank of [redacted]
ACH ABA # [redacted]
Account # [redacted]
Swift Code [redacted]
Wire ABA# [redacted]

Lockbox:
[redacted]

We sincerely appreciate your business.

Sincerely,

The Accounting Team


</xapi:message>
</xapi:email>
</xapi:delivery>
<xapi:document key="${SEND_TO_CUSTOMER_NAME}" output="Statement" output-type="pdf" delivery="${CUSTOMER_ID}">
<xapi:template type="rtf" location="xdo://AR.ARXSGPO.en.US/?getSource=true"
    filter=".//G_STATEMENT[TOTAL_AMOUNT_DUE!='0']"/>
</xapi:document>
</xapi:request>
</xapi:requestset>
```

Email Notification Generation

Bursting Control File

Notifications and Flat File Generation

Tips and Considerations

DON'T

- Propagate unmanageable code by always using UTL_FILE or FND_FILE to generate flat files
- Write unnecessary code using UTL_SMTP (or other means) to send emails

DO

- Simplify your code with a BI Publisher Data Template and eText formats to generate Flat Files
- Use BI Publisher Bursting to send email

eco East Coast Oracle Users Conference

Eastern States OATUG ORACLE APPLICATIONS & TECHNOLOGY USERS GROUP

Bursting to XML

Bursting to XML

BI Publisher Limitations

- Unable to use bursting to generate XML files
- Cannot burst thru XSL to generate XML files
- Can use an eText RTF to ‘construct’ an XML file

ALTERNATIVE APPROACH

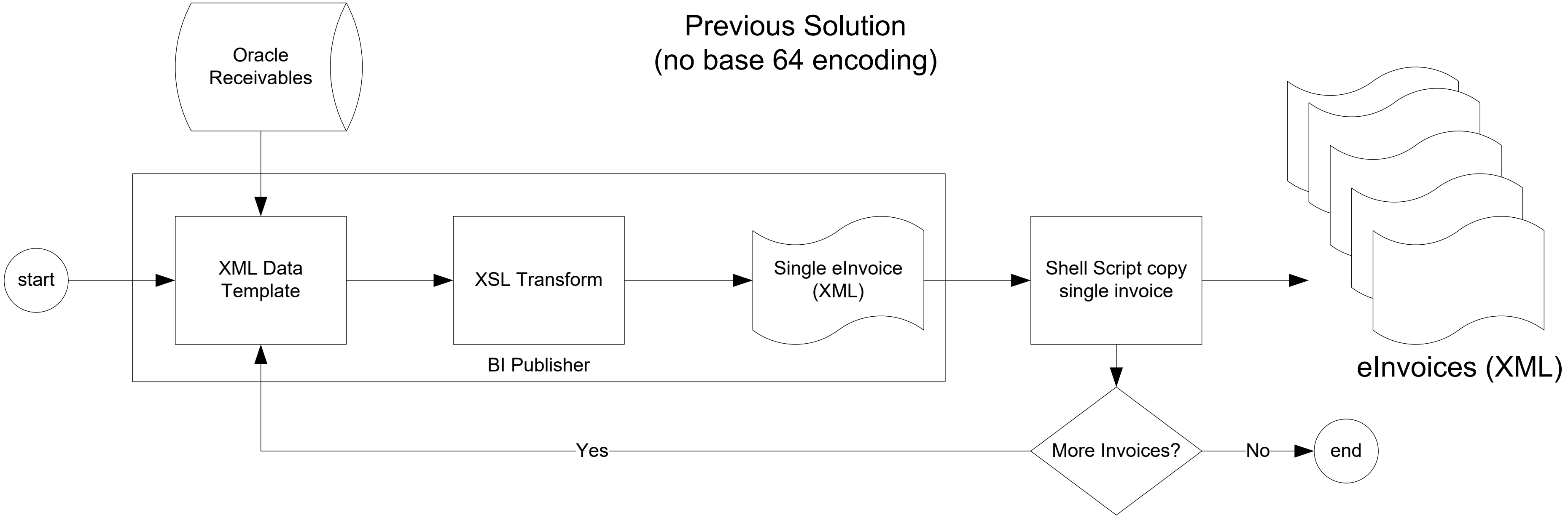
- Use native XML functions inside the Oracle database to build XML and burst the XML to an eText RTF.

Bursting to XML

- At our client, the Italian government required the generation of eInvoice files. These were XML data files.
  - Oracle provided a localization "patch" that generated these files
  - Functionality was very limited, and the files still needed to be processed through a 3rd party vendor. Customizations to the localization provided were still required.
  - The main limitation in the provided patch was that it could only generate one "eInvoice" file at a time.
- The initial solution built was severely limited by the one-at-a-time limitation. In order to generate each eInvoice file, a concurrent process was required for each file. Generating 5000 invoices required 5000 concurrent process executions.

Bursting to XML

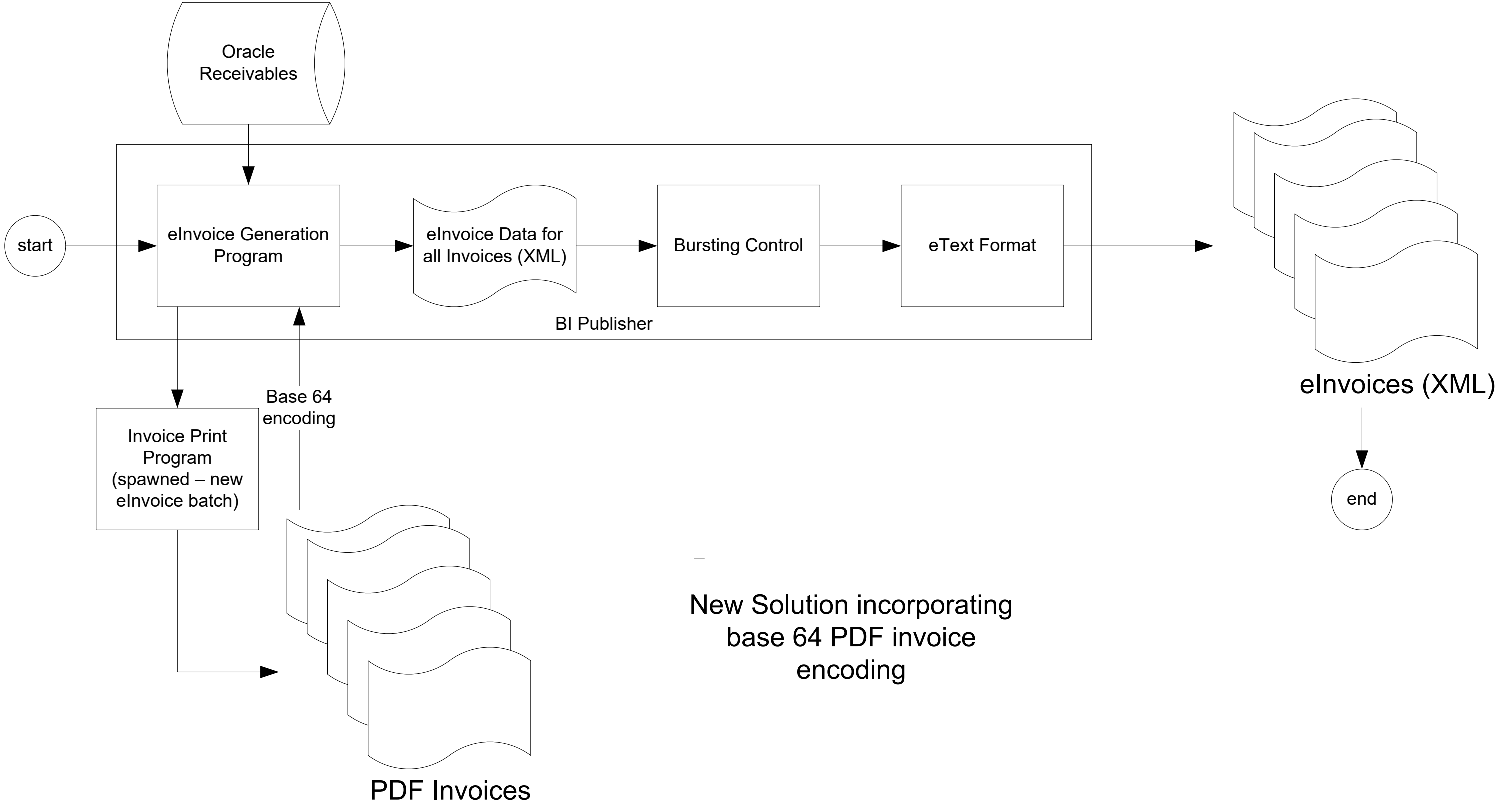
Previous Solution
(no base 64 encoding)



Bursting to XML

- After implementation, a new requirement was added. The PDF of each invoice was required to be added to the generated XML “eInvoice” so that the customer could print the invoice.
- We re-architected the solution to add the PDF invoice as a base 64 encoded string in the XML and, more importantly, moved the generation of the XML from BI Publisher to within the database
  - A Java utility was written to base 64 encode a PDF file
  - The SQL in the Oracle provided data template was moved into PLSQL code and “encased” in XML generating functions
  - The XSL transform used to generate the XML was not used in BI Publisher, but was instead fired from within the database code
  - A simple eText RTF formatting template was used in the bursting control file to generate the XML.

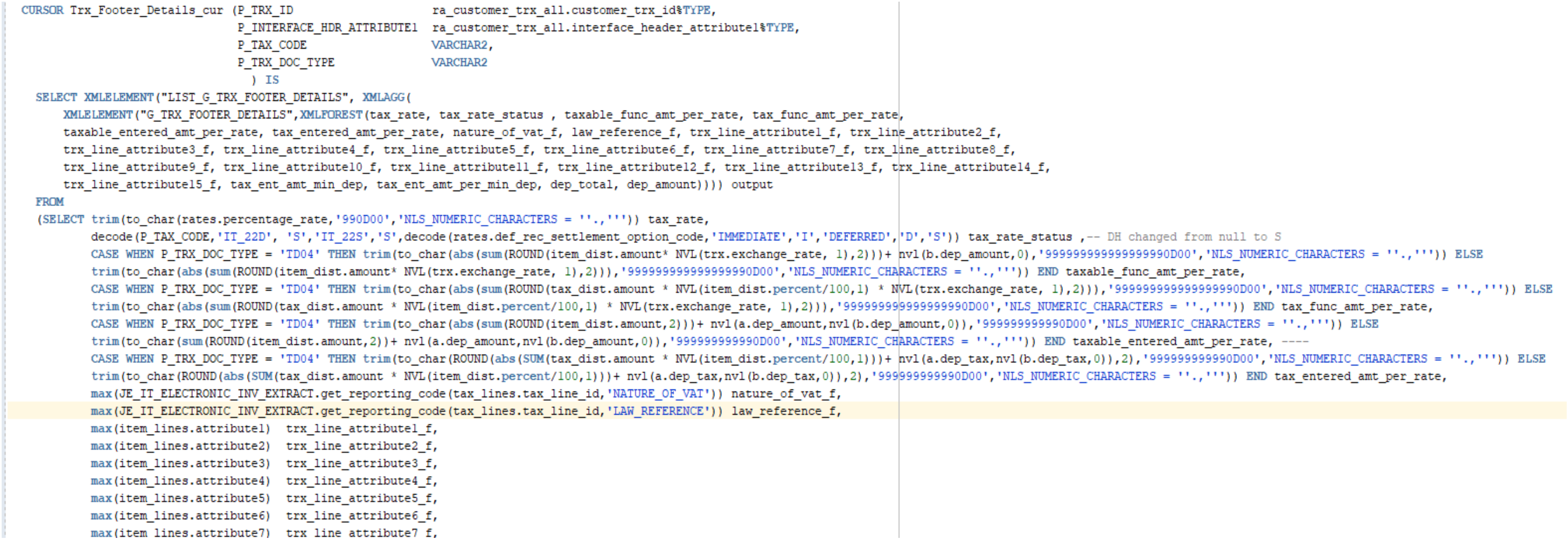
Bursting to XML


Oracle Receivables
start
eInvoice Generation Program
eInvoice Data for all Invoices (XML)
Bursting Control
eText Format
BI Publisher
eInvoices (XML)
end
Invoice Print Program (spawned – new eInvoice batch)
Base 64 encoding
PDF Invoices

New Solution incorporating base 64 PDF invoice encoding

Bursting to XML

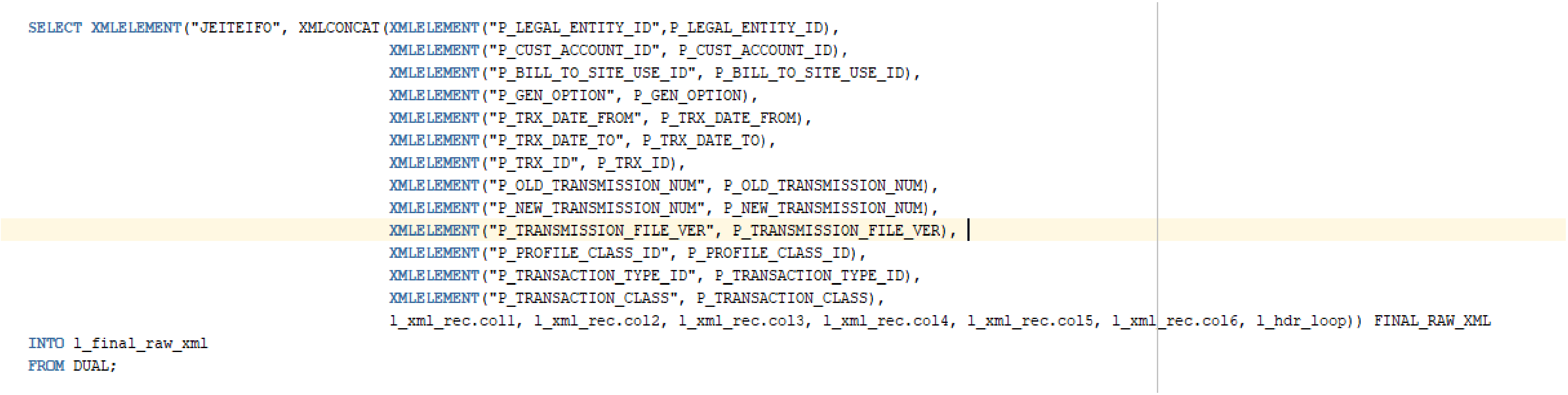
XML generation moved to the PL/SQL CODE



```sql
CURSOR Trx_Footer_Details_cur (P_TRX_ID                    ra_customer_trx_all.customer_trx_id%TYPE,
                               P_INTERFACE_HDR_ATTRIBUTE1  ra_customer_trx_all.interface_header_attribute1%TYPE,
                               P_TAX_CODE                  VARCHAR2,
                               P_TRX_DOC_TYPE              VARCHAR2
                                 ) IS
  SELECT XMLELEMENT("LIST_G_TRX_FOOTER_DETAILS", XMLAGG(
      XMLELEMENT("G_TRX_FOOTER_DETAILS",XMLFOREST(tax_rate, tax_rate_status , taxable_func_amt_per_rate, tax_func_amt_per_rate,
      taxable_entered_amt_per_rate, tax_entered_amt_per_rate, nature_of_vat_f, law_reference_f, trx_line_attribute1_f, trx_line_attribute2_f,
      trx_line_attribute3_f, trx_line_attribute4_f, trx_line_attribute5_f, trx_line_attribute6_f, trx_line_attribute7_f, trx_line_attribute8_f,
      trx_line_attribute9_f, trx_line_attribute10_f, trx_line_attribute11_f, trx_line_attribute12_f, trx_line_attribute13_f, trx_line_attribute14_f,
      trx_line_attribute15_f, tax_ent_amt_min_dep, tax_ent_amt_per_min_dep, dep_total, dep_amount)))) output
  FROM
  (SELECT trim(to_char(rates.percentage_rate,'990D00','NLS_NUMERIC_CHARACTERS = ''.,''')) tax_rate,
          decode(P_TAX_CODE,'IT_22D', 'S','IT_22S','S',decode(rates.def_rec_settlement_option_code,'IMMEDIATE','I','DEFERRED','D','S')) tax_rate_status ,-- DH changed from null to S
          CASE WHEN P_TRX_DOC_TYPE = 'TD04' THEN trim(to_char(abs(sum(ROUND(item_dist.amount* NVL(trx.exchange_rate, 1),2)))+ nvl(b.dep_amount,0),'999999999999999990D00','NLS_NUMERIC_CHARACTERS = ''.,''')) ELSE
          trim(to_char(abs(sum(ROUND(item_dist.amount* NVL(trx.exchange_rate, 1),2))),'999999999999999990D00','NLS_NUMERIC_CHARACTERS = ''.,''')) END taxable_func_amt_per_rate,
          CASE WHEN P_TRX_DOC_TYPE = 'TD04' THEN trim(to_char(abs(sum(ROUND(tax_dist.amount * NVL(item_dist.percent/100,1) * NVL(trx.exchange_rate, 1),2))),'999999999999999990D00','NLS_NUMERIC_CHARACTERS = ''.,''')) ELSE
          trim(to_char(abs(sum(ROUND(tax_dist.amount * NVL(item_dist.percent/100,1) * NVL(trx.exchange_rate, 1),2))),'999999999999999990D00','NLS_NUMERIC_CHARACTERS = ''.,''')) END tax_func_amt_per_rate,
          CASE WHEN P_TRX_DOC_TYPE = 'TD04' THEN trim(to_char(abs(sum(ROUND(item_dist.amount,2)))+ nvl(a.dep_amount,nvl(b.dep_amount,0)),'999999999990D00','NLS_NUMERIC_CHARACTERS = ''.,''')) ELSE
          trim(to_char(sum(ROUND(item_dist.amount,2))+ nvl(a.dep_amount,nvl(b.dep_amount,0)),'999999999990D00','NLS_NUMERIC_CHARACTERS = ''.,''')) END taxable_entered_amt_per_rate, ----
          CASE WHEN P_TRX_DOC_TYPE = 'TD04' THEN trim(to_char(ROUND(abs(SUM(tax_dist.amount * NVL(item_dist.percent/100,1)))+ nvl(a.dep_tax,nvl(b.dep_tax,0)),2),'999999999990D00','NLS_NUMERIC_CHARACTERS = ''.,''')) ELSE
          trim(to_char(ROUND(abs(SUM(tax_dist.amount * NVL(item_dist.percent/100,1)))+ nvl(a.dep_tax,nvl(b.dep_tax,0)),2),'999999999990D00','NLS_NUMERIC_CHARACTERS = ''.,''')) END tax_entered_amt_per_rate,
          max(JE_IT_ELECTRONIC_INV_EXTRACT.get_reporting_code(tax_lines.tax_line_id,'NATURE_OF_VAT')) nature_of_vat_f,
          max(JE_IT_ELECTRONIC_INV_EXTRACT.get_reporting_code(tax_lines.tax_line_id,'LAW_REFERENCE')) law_reference_f,
          max(item_lines.attribute1)  trx_line_attribute1_f,
          max(item_lines.attribute2)  trx_line_attribute2_f,
          max(item_lines.attribute3)  trx_line_attribute3_f,
          max(item_lines.attribute4)  trx_line_attribute4_f,
          max(item_lines.attribute5)  trx_line_attribute5_f,
          max(item_lines.attribute6)  trx_line_attribute6_f,
          max(item_lines.attribute7)  trx_line_attribute7_f,
```

Bursting to XML

All XML components assembled into one raw XML field



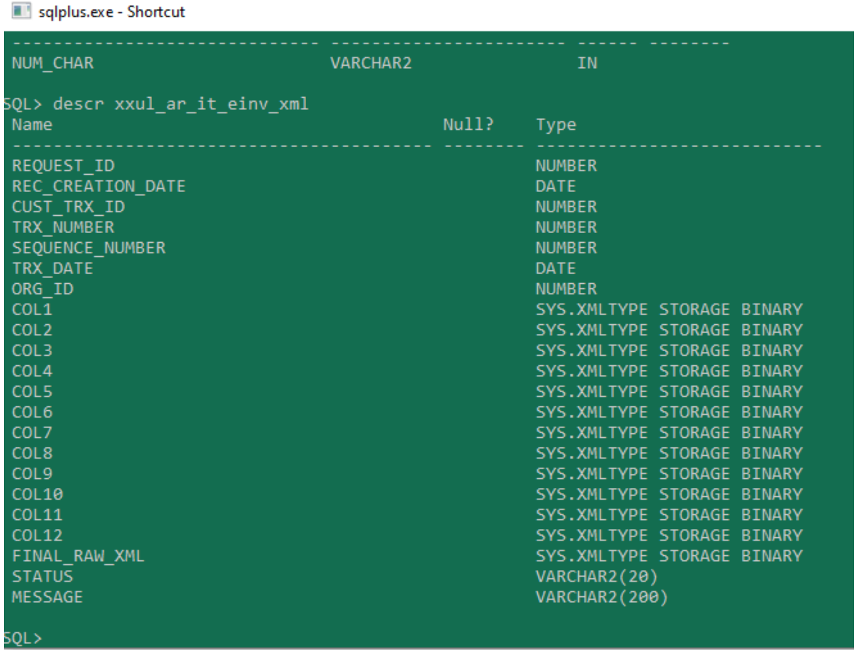
```
SELECT XMLELEMENT("JEITEIFO", XMLCONCAT(XMLELEMENT("P_LEGAL_ENTITY_ID",P_LEGAL_ENTITY_ID),
                                        XMLELEMENT("P_CUST_ACCOUNT_ID", P_CUST_ACCOUNT_ID),
                                        XMLELEMENT("P_BILL_TO_SITE_USE_ID", P_BILL_TO_SITE_USE_ID),
                                        XMLELEMENT("P_GEN_OPTION", P_GEN_OPTION),
                                        XMLELEMENT("P_TRX_DATE_FROM", P_TRX_DATE_FROM),
                                        XMLELEMENT("P_TRX_DATE_TO", P_TRX_DATE_TO),
                                        XMLELEMENT("P_TRX_ID", P_TRX_ID),
                                        XMLELEMENT("P_OLD_TRANSMISSION_NUM", P_OLD_TRANSMISSION_NUM),
                                        XMLELEMENT("P_NEW_TRANSMISSION_NUM", P_NEW_TRANSMISSION_NUM),
                                        XMLELEMENT("P_TRANSMISSION_FILE_VER", P_TRANSMISSION_FILE_VER),
                                        XMLELEMENT("P_PROFILE_CLASS_ID", P_PROFILE_CLASS_ID),
                                        XMLELEMENT("P_TRANSACTION_TYPE_ID", P_TRANSACTION_TYPE_ID),
                                        XMLELEMENT("P_TRANSACTION_CLASS", P_TRANSACTION_CLASS),
                                        l_xml_rec.col1, l_xml_rec.col2, l_xml_rec.col3, l_xml_rec.col4, l_xml_rec.col5, l_xml_rec.col6, l_hdr_loop)) FINAL_RAW_XML
INTO l_final_raw_xml
FROM DUAL;
```

Bursting to XML

East Coast Oracle Users Conference

Eastern States OATUG

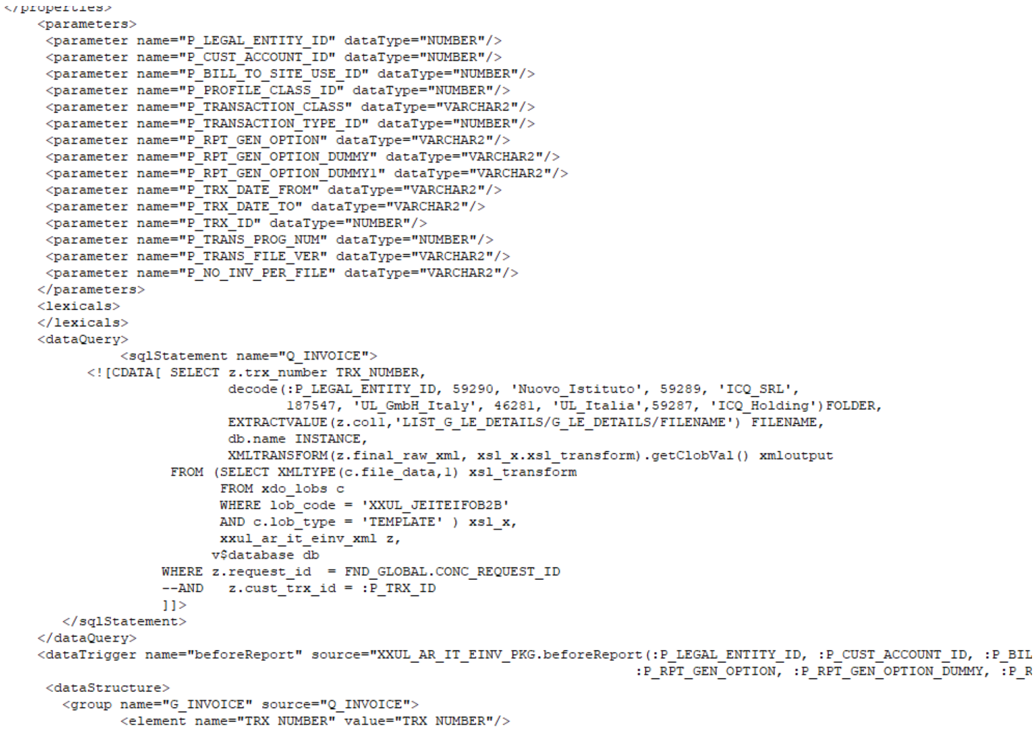
Assembled XML stored in a temporary table



```
 ------------------------------ ----------------------- ------ --------
 NUM_CHAR                       VARCHAR2                IN

SQL> descr xxul_ar_it_einv_xml
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REQUEST_ID                                         NUMBER
 REC_CREATION_DATE                                  DATE
 CUST_TRX_ID                                        NUMBER
 TRX_NUMBER                                         NUMBER
 SEQUENCE_NUMBER                                    NUMBER
 TRX_DATE                                           DATE
 ORG_ID                                             NUMBER
 COL1                                               SYS.XMLTYPE STORAGE BINARY
 COL2                                               SYS.XMLTYPE STORAGE BINARY
 COL3                                               SYS.XMLTYPE STORAGE BINARY
 COL4                                               SYS.XMLTYPE STORAGE BINARY
 COL5                                               SYS.XMLTYPE STORAGE BINARY
 COL6                                               SYS.XMLTYPE STORAGE BINARY
 COL7                                               SYS.XMLTYPE STORAGE BINARY
 COL8                                               SYS.XMLTYPE STORAGE BINARY
 COL9                                               SYS.XMLTYPE STORAGE BINARY
 COL10                                              SYS.XMLTYPE STORAGE BINARY
 COL11                                              SYS.XMLTYPE STORAGE BINARY
 COL12                                              SYS.XMLTYPE STORAGE BINARY
 FINAL_RAW_XML                                      SYS.XMLTYPE STORAGE BINARY
 STATUS                                             VARCHAR2(20)
 MESSAGE                                            VARCHAR2(200)

SQL>
```

Bursting to XML

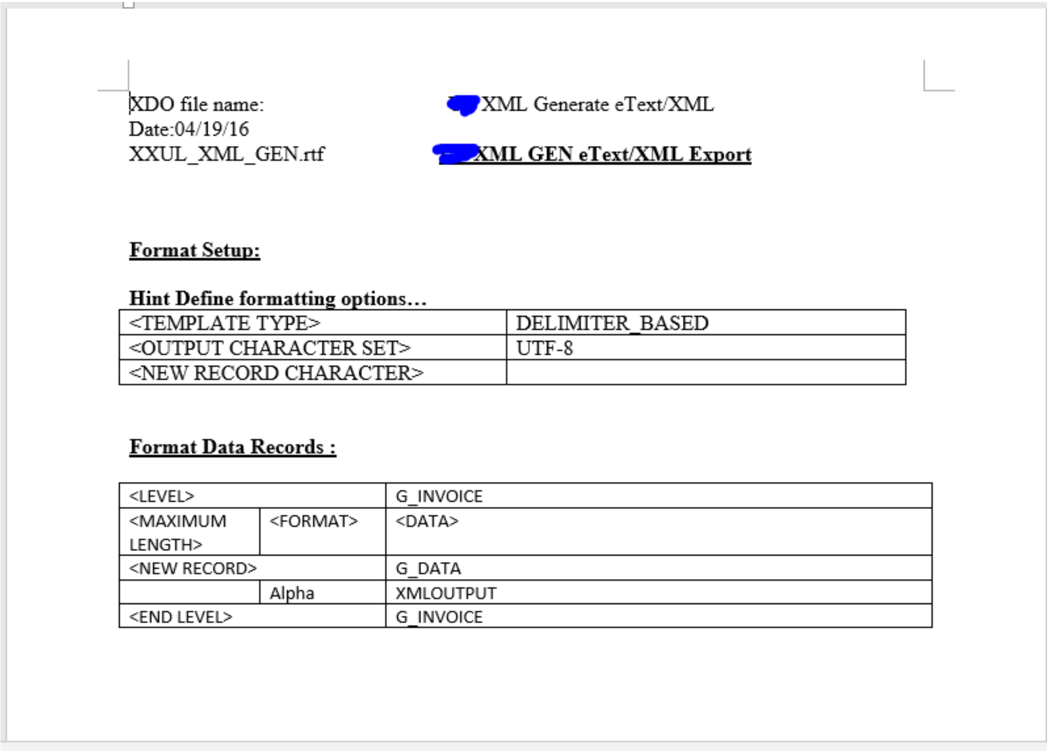
XML transformed via SQL in the Data Template



```
</properties>
    <parameters>
     <parameter name="P_LEGAL_ENTITY_ID" dataType="NUMBER"/>
     <parameter name="P_CUST_ACCOUNT_ID" dataType="NUMBER"/>
     <parameter name="P_BILL_TO_SITE_USE_ID" dataType="NUMBER"/>
     <parameter name="P_PROFILE_CLASS_ID" dataType="NUMBER"/>
     <parameter name="P_TRANSACTION_CLASS" dataType="VARCHAR2"/>
     <parameter name="P_TRANSACTION_TYPE_ID" dataType="NUMBER"/>
     <parameter name="P_RPT_GEN_OPTION" dataType="VARCHAR2"/>
     <parameter name="P_RPT_GEN_OPTION_DUMMY" dataType="VARCHAR2"/>
     <parameter name="P_RPT_GEN_OPTION_DUMMY1" dataType="VARCHAR2"/>
     <parameter name="P_TRX_DATE_FROM" dataType="VARCHAR2"/>
     <parameter name="P_TRX_DATE_TO" dataType="VARCHAR2"/>
     <parameter name="P_TRX_ID" dataType="NUMBER"/>
     <parameter name="P_TRANS_PROG_NUM" dataType="NUMBER"/>
     <parameter name="P_TRANS_FILE_VER" dataType="VARCHAR2"/>
     <parameter name="P_NO_INV_PER_FILE" dataType="VARCHAR2"/>
    </parameters>
    <lexicals>
    </lexicals>
    <dataQuery>
              <sqlStatement name="Q_INVOICE">
          <![CDATA[ SELECT z.trx_number TRX_NUMBER,
                           decode(:P_LEGAL_ENTITY_ID, 59290, 'Nuovo_Istituto', 59289, 'ICQ_SRL',
                                  187547, 'UL_GmbH_Italy', 46281, 'UL_Italia',59287, 'ICQ_Holding')FOLDER,
                           EXTRACTVALUE(z.coll,'LIST_G_LE_DETAILS/G_LE_DETAILS/FILENAME') FILENAME,
                           db.name INSTANCE,
                           XMLTRANSFORM(z.final_raw_xml, xsl_x.xsl_transform).getClobVal() xmloutput
                FROM (SELECT XMLTYPE(c.file_data,1) xsl_transform
                      FROM xdo_lobs c
                      WHERE lob_code = 'XXUL_JEITEIFOB2B'
                      AND c.lob_type = 'TEMPLATE' ) xsl_x,
                      xxul_ar_it_einv_xml z,
                     v$database db
               WHERE z.request_id  = FND_GLOBAL.CONC_REQUEST_ID
               --AND   z.cust_trx_id = :P_TRX_ID
               ]]>
       </sqlStatement>
    </dataQuery>
    <dataTrigger name="beforeReport" source="XXUL_AR_IT_EINV_PKG.beforeReport(:P_LEGAL_ENTITY_ID, :P_CUST_ACCOUNT_ID, :P_BIL
                                                                                  :P_RPT_GEN_OPTION, :P_RPT_GEN_OPTION_DUMMY, :P_R
     <dataStructure>
       <group name="G_INVOICE" source="Q_INVOICE">
              <element name="TRX_NUMBER" value="TRX_NUMBER"/>
```

Bursting to XML

eText Format to produce XML



XDO file name: [redacted] XML Generate eText/XML
Date:04/19/16
XXUL_XML_GEN.rtf [redacted] **XML GEN eText/XML Export**

Format Setup:

Hint Define formatting options…

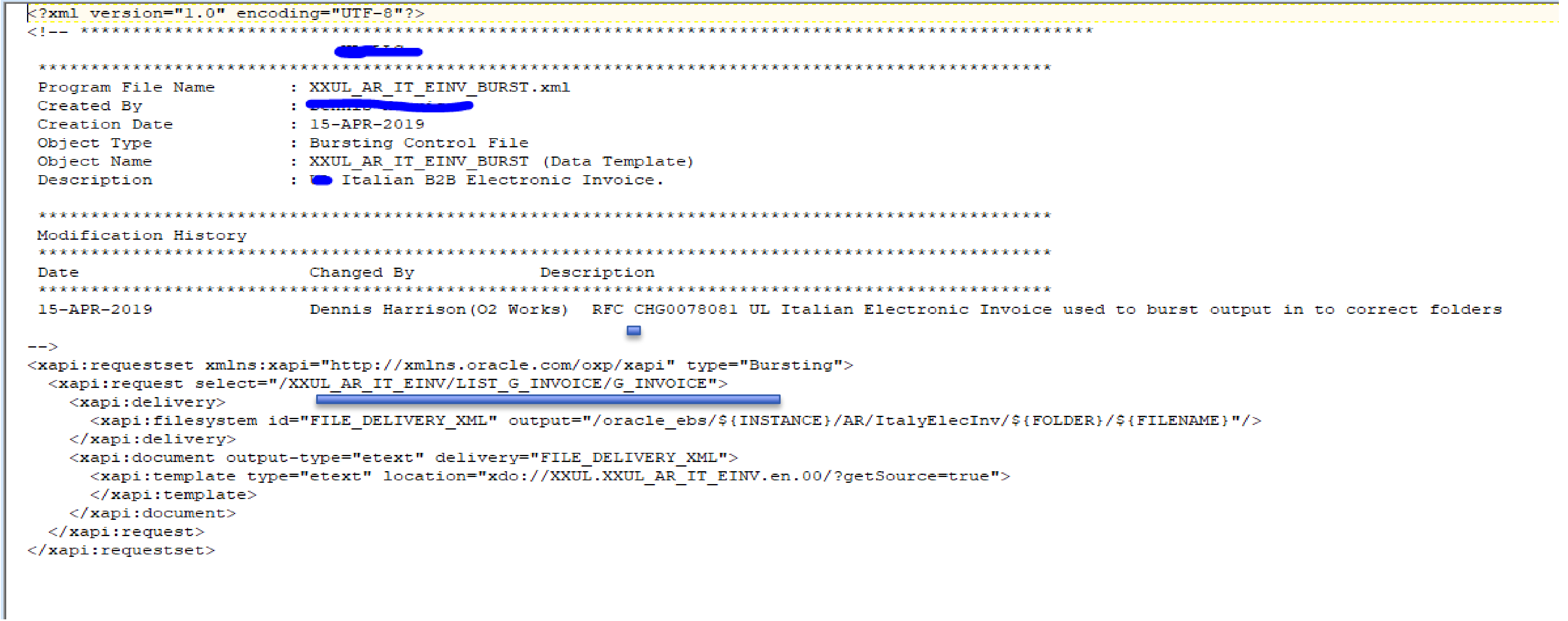
<TEMPLATE TYPE>	DELIMITER_BASED
<OUTPUT CHARACTER SET>	UTF-8
<NEW RECORD CHARACTER>	

Format Data Records :

<LEVEL>		G_INVOICE
<MAXIMUM LENGTH>	<FORMAT>	<DATA>
<NEW RECORD>		G_DATA
	Alpha	XMLOUTPUT
<END LEVEL>		G_INVOICE

Bursting to XML

Bursting Control to generate XML thru eText format



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- *************************************************************************************************
                              [redacted]
 *************************************************************************************************
 Program File Name        : XXUL_AR_IT_EINV_BURST.xml
 Created By               : [redacted]
 Creation Date            : 15-APR-2019
 Object Type              : Bursting Control File
 Object Name              : XXUL_AR_IT_EINV_BURST (Data Template)
 Description              : [redacted] Italian B2B Electronic Invoice.

 *************************************************************************************************
 Modification History
 *************************************************************************************************
 Date                       Changed By             Description
 *************************************************************************************************
 15-APR-2019                Dennis Harrison(O2 Works)  RFC CHG0078081 UL Italian Electronic Invoice used to burst output in to correct folders

-->
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi" type="Bursting">
  <xapi:request select="/XXUL_AR_IT_EINV/LIST_G_INVOICE/G_INVOICE">
    <xapi:delivery>
      <xapi:filesystem id="FILE_DELIVERY_XML" output="/oracle_ebs/${INSTANCE}/AR/ItalyElecInv/${FOLDER}/${FILENAME}"/>
    </xapi:delivery>
    <xapi:document output-type="etext" delivery="FILE_DELIVERY_XML">
      <xapi:template type="etext" location="xdo://XXUL.XXUL_AR_IT_EINV.en.00/?getSource=true">
      </xapi:template>
    </xapi:document>
  </xapi:request>
</xapi:requestset>
```

Bursting to XML

- The new solution built was greatly improved over previous
  - Only 1 process needed to generate all XML files, instead of one per invoice document
  - Run-time was reduced from three hours for a typical batch to 5 minutes
- The Design was scalable for future enhancements.
  - Recently, the business required that a second attachment be added to the generated XML for each invoice.

Bursting to XML

Tips and Considerations

DON’T

- Be limited by BI Publisher’s inability to burst to XML

DO

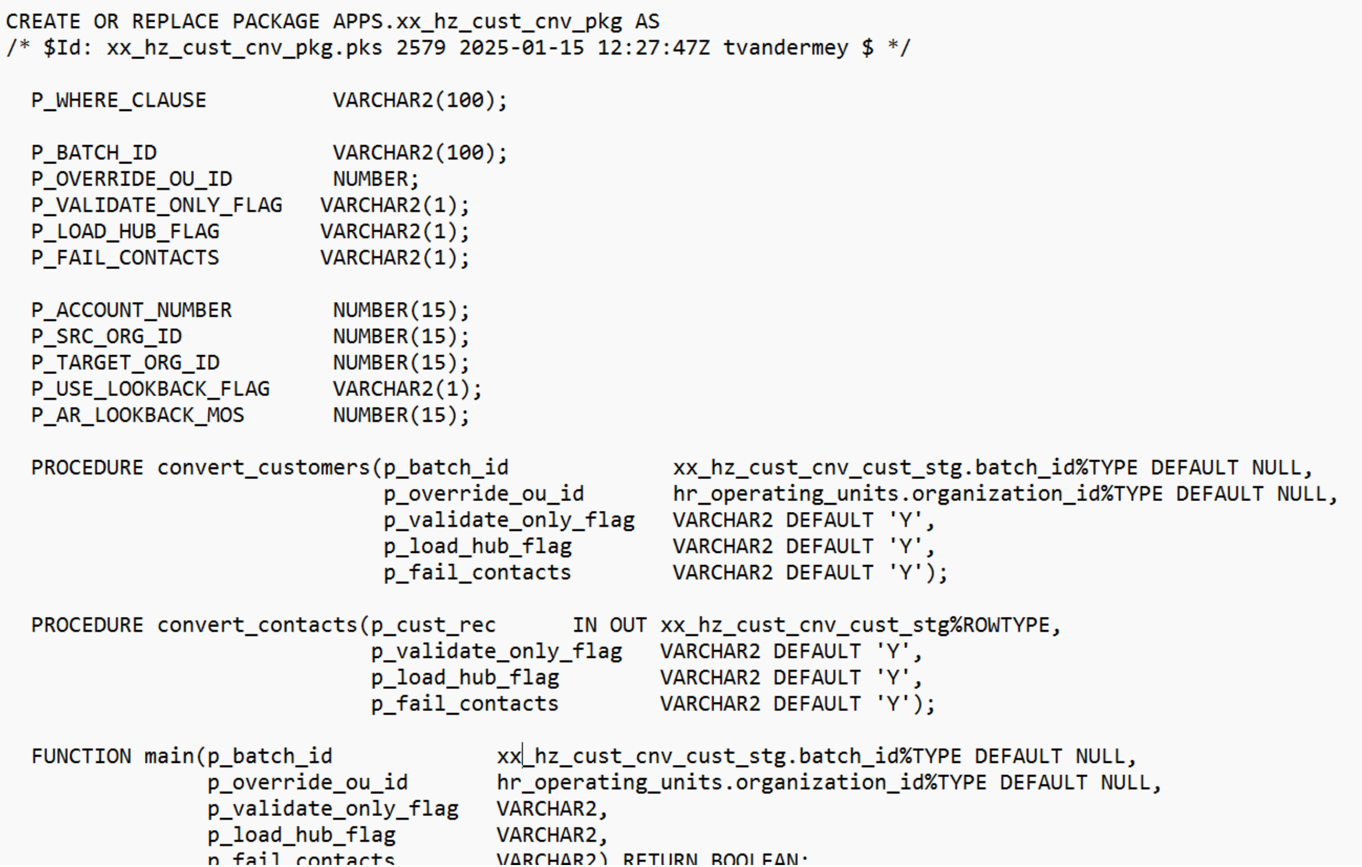
- Utilize native Oracle database XML functions to aggregate and build XML
- Use BI Publisher bursting to dynamically generate XML

PRACTICAL EXAMPLE Customer Import

Practical Example – Customer Import

- Build External Table
- Build Staging Table (external table fields plus reporting fields, ids, request ids, status)
- Create PL/SQL for API calls
  - Load staging from external table (adding request_id)
  - Pass record type thru API calls, updating values in record thru the process
  - Autonomous transaction to update the staging with API returned data
  - If validation mode, ‘rollback’, else, commit record.
- Create XML data template to call API process (before trigger) and create xml for output
- Create XML template for reporting results
- Create concurrent program to execute XML Process

PL/SQL Spec

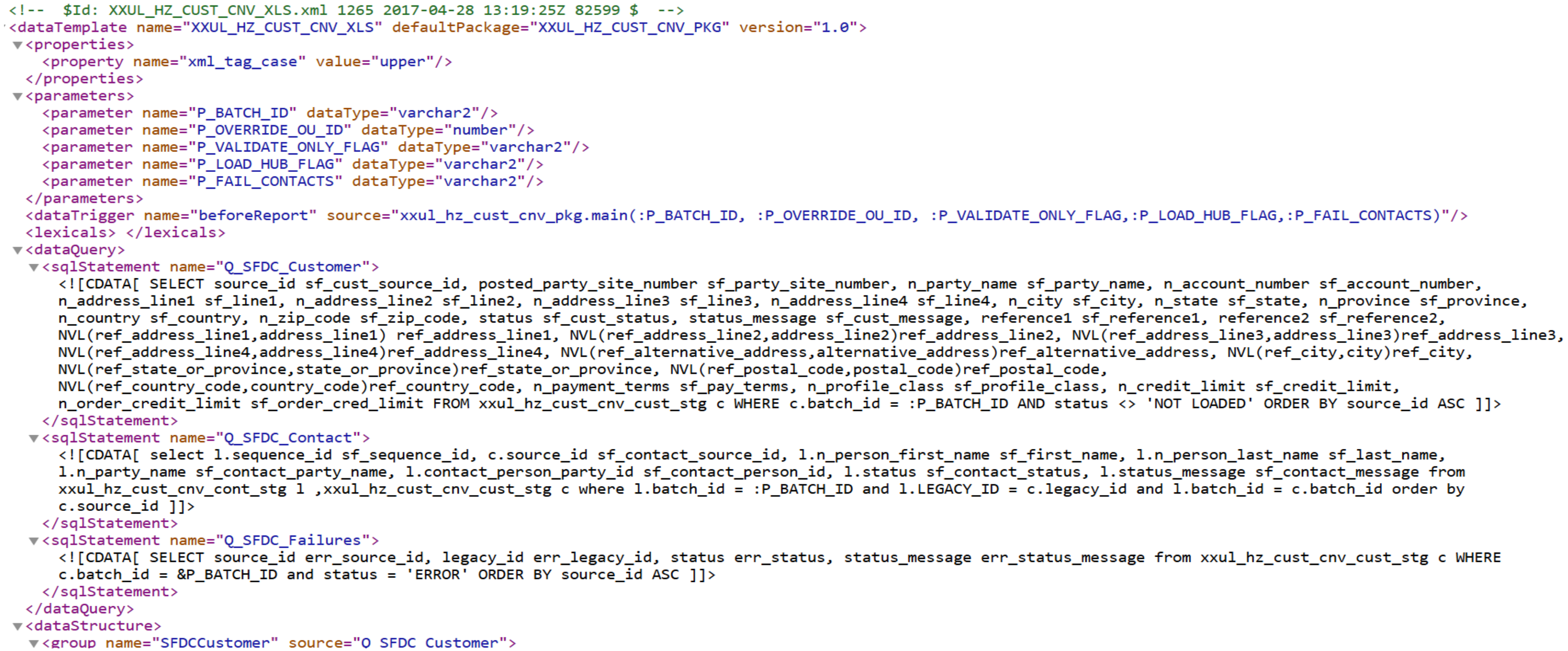


```
CREATE OR REPLACE PACKAGE APPS.xx_hz_cust_cnv_pkg AS
/* $Id: xx_hz_cust_cnv_pkg.pks 2579 2025-01-15 12:27:47Z tvandermey $ */

  P_WHERE_CLAUSE            VARCHAR2(100);

  P_BATCH_ID                VARCHAR2(100);
  P_OVERRIDE_OU_ID          NUMBER;
  P_VALIDATE_ONLY_FLAG     VARCHAR2(1);
  P_LOAD_HUB_FLAG          VARCHAR2(1);
  P_FAIL_CONTACTS          VARCHAR2(1);

  P_ACCOUNT_NUMBER          NUMBER(15);
  P_SRC_ORG_ID              NUMBER(15);
  P_TARGET_ORG_ID           NUMBER(15);
  P_USE_LOOKBACK_FLAG       VARCHAR2(1);
  P_AR_LOOKBACK_MOS         NUMBER(15);

  PROCEDURE convert_customers(p_batch_id              xx_hz_cust_cnv_cust_stg.batch_id%TYPE DEFAULT NULL,
                              p_override_ou_id        hr_operating_units.organization_id%TYPE DEFAULT NULL,
                              p_validate_only_flag    VARCHAR2 DEFAULT 'Y',
                              p_load_hub_flag         VARCHAR2 DEFAULT 'Y',
                              p_fail_contacts         VARCHAR2 DEFAULT 'Y');

  PROCEDURE convert_contacts(p_cust_rec      IN OUT xx_hz_cust_cnv_cust_stg%ROWTYPE,
                             p_validate_only_flag   VARCHAR2 DEFAULT 'Y',
                             p_load_hub_flag        VARCHAR2 DEFAULT 'Y',
                             p_fail_contacts        VARCHAR2 DEFAULT 'Y');

  FUNCTION main(p_batch_id              xx_hz_cust_cnv_cust_stg.batch_id%TYPE DEFAULT NULL,
                p_override_ou_id        hr_operating_units.organization_id%TYPE DEFAULT NULL,
                p_validate_only_flag    VARCHAR2,
                p_load_hub_flag         VARCHAR2,
                p_fail_contacts         VARCHAR2) RETURN BOOLEAN;
```

PL/SQL Body

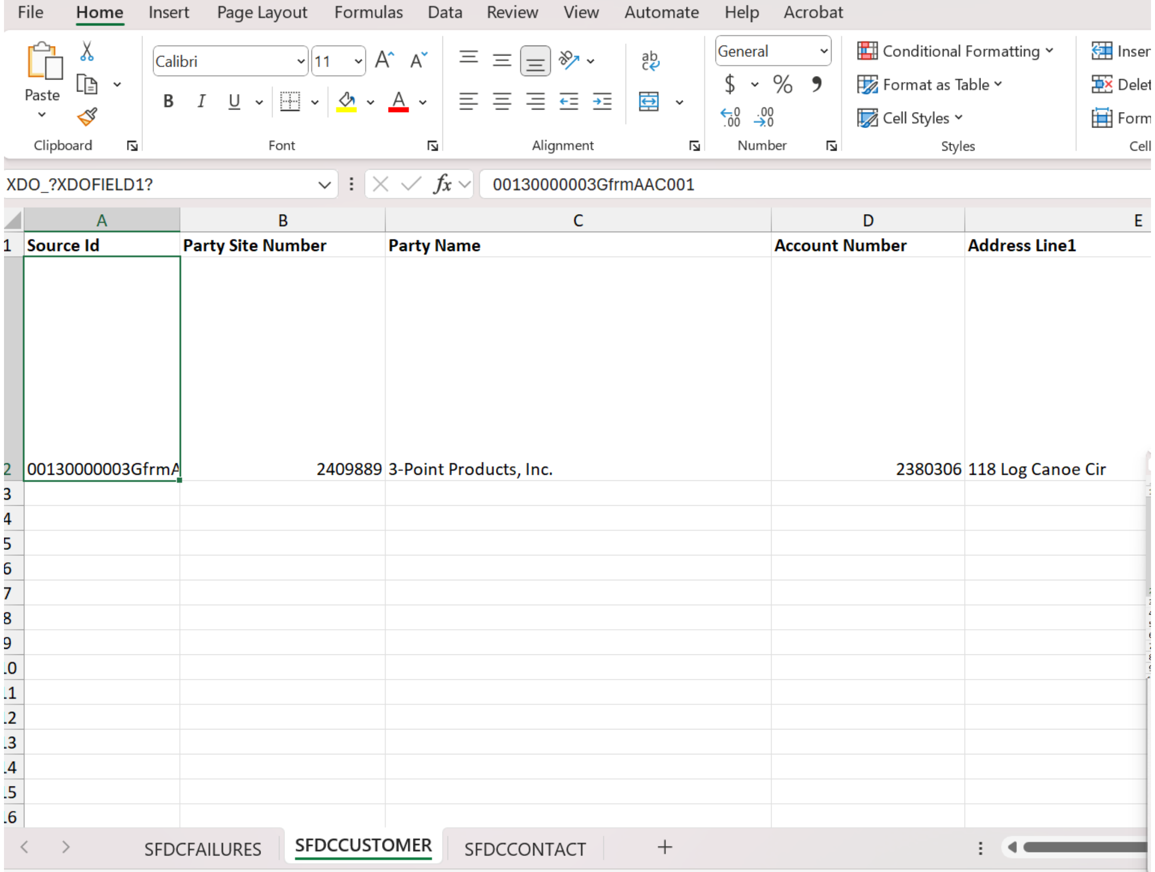
- Function Main (called in Before Trigger of XML)
  - Load staging tables from external
  - Loop thru staging and process all records
  - If validation mode = ‘COMMIT’ then COMMIT, else ‘ROLLBACK’

XML Data Template



```
<!--  $Id: XXUL_HZ_CUST_CNV_XLS.xml 1265 2017-04-28 13:19:25Z 82599 $  -->
<dataTemplate name="XXUL_HZ_CUST_CNV_XLS" defaultPackage="XXUL_HZ_CUST_CNV_PKG" version="1.0">
▼<properties>
   <property name="xml_tag_case" value="upper"/>
 </properties>
▼<parameters>
   <parameter name="P_BATCH_ID" dataType="varchar2"/>
   <parameter name="P_OVERRIDE_OU_ID" dataType="number"/>
   <parameter name="P_VALIDATE_ONLY_FLAG" dataType="varchar2"/>
   <parameter name="P_LOAD_HUB_FLAG" dataType="varchar2"/>
   <parameter name="P_FAIL_CONTACTS" dataType="varchar2"/>
 </parameters>
 <dataTrigger name="beforeReport" source="xxul_hz_cust_cnv_pkg.main(:P_BATCH_ID, :P_OVERRIDE_OU_ID, :P_VALIDATE_ONLY_FLAG,:P_LOAD_HUB_FLAG,:P_FAIL_CONTACTS)"/>
 <lexicals> </lexicals>
▼<dataQuery>
  ▼<sqlStatement name="Q_SFDC_Customer">
     <![CDATA[ SELECT source_id sf_cust_source_id, posted_party_site_number sf_party_site_number, n_party_name sf_party_name, n_account_number sf_account_number,
     n_address_line1 sf_line1, n_address_line2 sf_line2, n_address_line3 sf_line3, n_address_line4 sf_line4, n_city sf_city, n_state sf_state, n_province sf_province,
     n_country sf_country, n_zip_code sf_zip_code, status sf_cust_status, status_message sf_cust_message, reference1 sf_reference1, reference2 sf_reference2,
     NVL(ref_address_line1,address_line1) ref_address_line1, NVL(ref_address_line2,address_line2)ref_address_line2, NVL(ref_address_line3,address_line3)ref_address_line3,
     NVL(ref_address_line4,address_line4)ref_address_line4, NVL(ref_alternative_address,alternative_address)ref_alternative_address, NVL(ref_city,city)ref_city,
     NVL(ref_state_or_province,state_or_province)ref_state_or_province, NVL(ref_postal_code,postal_code)ref_postal_code,
     NVL(ref_country_code,country_code)ref_country_code, n_payment_terms sf_pay_terms, n_profile_class sf_profile_class, n_credit_limit sf_credit_limit,
     n_order_credit_limit sf_order_cred_limit FROM xxul_hz_cust_cnv_cust_stg c WHERE c.batch_id = :P_BATCH_ID AND status <> 'NOT LOADED' ORDER BY source_id ASC ]]>
   </sqlStatement>
  ▼<sqlStatement name="Q_SFDC_Contact">
     <![CDATA[ select l.sequence_id sf_sequence_id, c.source_id sf_contact_source_id, l.n_person_first_name sf_first_name, l.n_person_last_name sf_last_name,
     l.n_party_name sf_contact_party_name, l.contact_person_party_id sf_contact_person_id, l.status sf_contact_status, l.status_message sf_contact_message from
     xxul_hz_cust_cnv_cont_stg l ,xxul_hz_cust_cnv_cust_stg c where l.batch_id = :P_BATCH_ID and l.LEGACY_ID = c.legacy_id and l.batch_id = c.batch_id order by
     c.source_id ]]>
   </sqlStatement>
  ▼<sqlStatement name="Q_SFDC_Failures">
     <![CDATA[ SELECT source_id err_source_id, legacy_id err_legacy_id, status err_status, status_message err_status_message from xxul_hz_cust_cnv_cust_stg c WHERE
     c.batch_id = &P_BATCH_ID and status = 'ERROR' ORDER BY source_id ASC ]]>
   </sqlStatement>
 </dataQuery>
▼<dataStructure>
  ▼<group name="SFDCCustomer" source="Q_SFDC_Customer">
```

XML Template (XLS)



XDO_?XDOFIELD1? | 00130000003GfrmAAC001

	A	B	C	D	E
1	Source Id	Party Site Number	Party Name	Account Number	Address Line1
2	00130000003GfrmA	2409889	3-Point Products, Inc.	2380306	118 Log Canoe Cir

eco East Coast Oracle Users Conference

Eastern States OATUG ORACLE APPLICATIONS & TECHNOLOGY USERS GROUP

Thank You for Attending!

Tammy Vandermey
Sr. Technical Consultant
O2Works, LLC
tammy@o2works.com