OUM

TA.070-8 DISEÑO TÉCNICO DE EXTENSIONES E INTERFACES

ER03_AR_Timbrado de Complementos de Pagos

Autor:	Oracle Consulting
Fecha de creación:	Enero 16, 2018
Última actualización:	febrero 14, 2018
Código de referencia OUM:	DS.140 Design Specification
Version:	1.0

Aprobadores:

Ivan Muñoz _____

Ángel Flores _____

ORACLE

1 Control de Documento

1.1 Bitácora de Cambios

Fecha	Autor	Versión	Referencia del cambio
16-ene-2018	Gloria Jiménez	1.0	No hay documento previo

1.2 Revisores

Nombre	Posición

Contenido

2 Resumen técnico

Esta especificación documenta el diseño para el timbrado de complementos de pagos, cuyo caso de uso se describió en el documento TA020-4_ER03_AR_Timbrado de Complementos de Pago. La integración se compone de 3 capas las cuales son:

- Ent
- Biz
- Tec

La capa **Ent** administra las capas **Biz** utilizadas para esta integración y es la que se tiene programada en el servidor SOA para ejecutarse en los horarios establecidos, que son a las 9:00, 14:00 y 21:00 horas cada día.

A continuación, se describe la programación de esta capa en el ESS, consta de tres elementos, un job y dos programaciones para abarcar los horarios solicitados:

Componente	Propiedad	Valor
Trabajo	Nombre	PaymentComplementJobDev
	Nombre Mostrado	PaymentComplementJobDev
	Paquete	/oracle/apps/ess/custom/soa
	Descripción	Job de complementos de Pagos
	Tipo de trabajo	SyncWebserviceJobType
	WSDL	http://129.150.110.0:80/soa-infra/services/test/PaymentComplementEnt/PaymentComplementEnt?WSDL
	Tipo de puerto	PaymentComplementEntPort
	Operación	StampPaymentComplement
Planificación	Nombre	PaymentComplementSch
	Nombre Mostrado	PaymentComplementSch
	Paquete	/oracle/apps/ess/custom/soa
	Descripción	Programación diaria a las 2 pm
	Frecuencia	Diariamente
	Zona horaria	(UTC-06:00) Ciudad de México
Planificación	Nombre	PaymentComplementSch2
	Nombre Mostrado	PaymentComplementSch2
	Paquete	/oracle/apps/ess/custom/soa
	Descripción	Programación cada 12 hrs, iniciando a las 9am
	Frecuencia	Cada Hora/Minuto (Cada 12 horas iniciando a las 9 am)
	Zona horaria	(UTC-06:00) Ciudad de México

La capa **Biz** es la que se encarga de toda la lógica que requiere el envío de la información a timbrar como gestión de errores, archivos adjuntos, log en base de datos, etc.

La capa **Tec** es donde se programa las funcionalidades para la extracción de datos de las diferentes fuentes.

Existen dos tipos de integraciones:

- **Inbound:** En este tipo de integraciones, las aplicaciones legadas llamarán a servicios web SOAP expuestos en la nube, enviando los archivos para ser cargados en ERP Cloud por los mecanismos de integración de datos y estándares que requiera la aplicación.
- **Outbound:** En el ERP Cloud se dispondrán de reportes a través de BI Publisher, que mediante servicios web SOAP serán leídos y la información contenida se procederá a procesar según corresponda, en este caso, se envía a timbrar los complementos de pagos y se tratan los resultados.

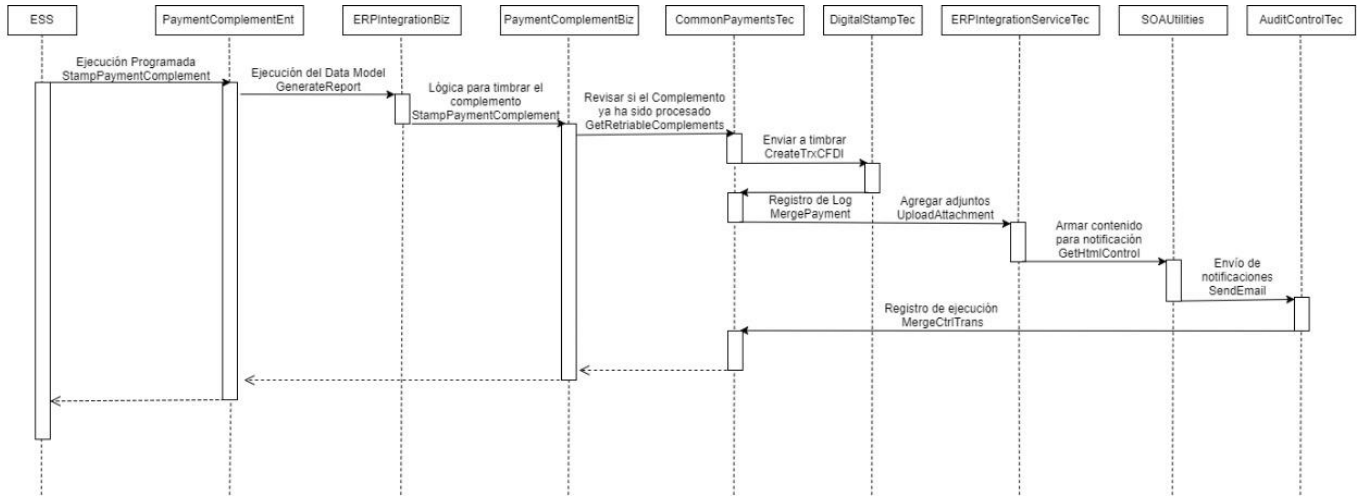
La integración de complementos de pagos es de tipo **Outbound** ya que se toma como fuente el ERP Cloud para enviar los recibos a timbrar al PAC en donde tenemos cuatro fases importantes que se realizan:

1. La primera consiste en la extracción de la información teniendo como fuente el ERP CLOUD, donde se encuentran ciertas condiciones que permiten identificar que complemento de pago es apto para enviar a timbrar.
2. La segunda fase implica el envío a timbrar y controlar todos posibles errores tanto de consistencia de datos del recibo como las validaciones que el propio servicio de timbrado del PAC realiza.
3. Como tercera etapa tenemos el almacenamiento de los errores en caso de que estos existan o la carga de archivos adjuntos del complemento de pago (URL de xml, URL de pdf y UUID).
4. Por último, se envía una notificación con la información de todos los registros procesados, número de éxitos y errores con su detalle.

Estas fases están distribuidas en las 3 capas mencionadas anteriormente (Ent, Biz, Tec) las cuales se explicarán a detalle en los siguientes puntos.

2.1 Diagrama de la integración

El siguiente diagrama muestra la secuencia que se tiene para timbrado de complementos de pago, iniciando en la programación de la integración donde se llama la capa **Ent** y esta a su vez hace los llamados a servicios **Biz** requeridos.

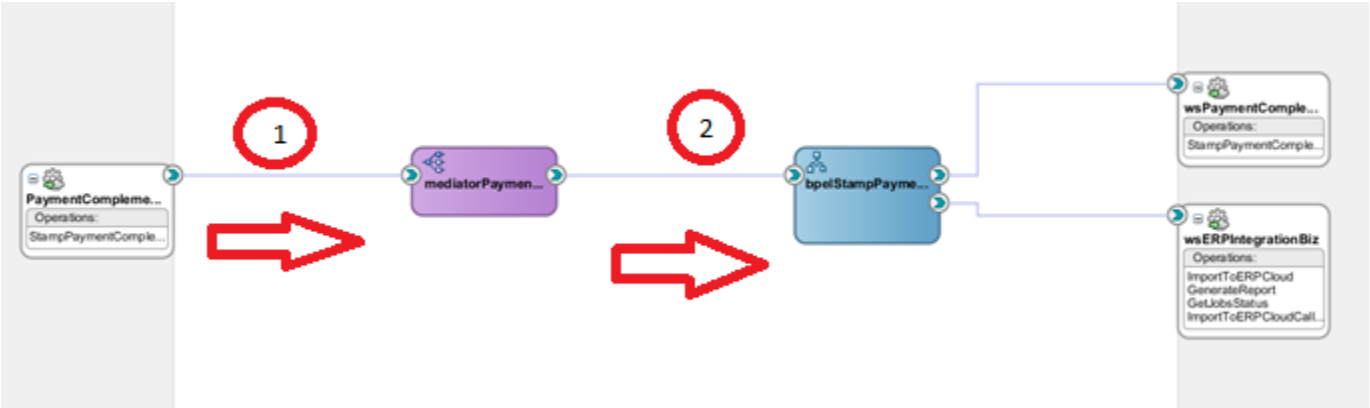


3 Lógica de implementación

Como mencionamos anteriormente, se cuenta con tres capas en la integración de complementos de pagos, en esta sección describiremos la lógica que se utilizó en cada una de ellas.

3.1 PaymentComplementEnt

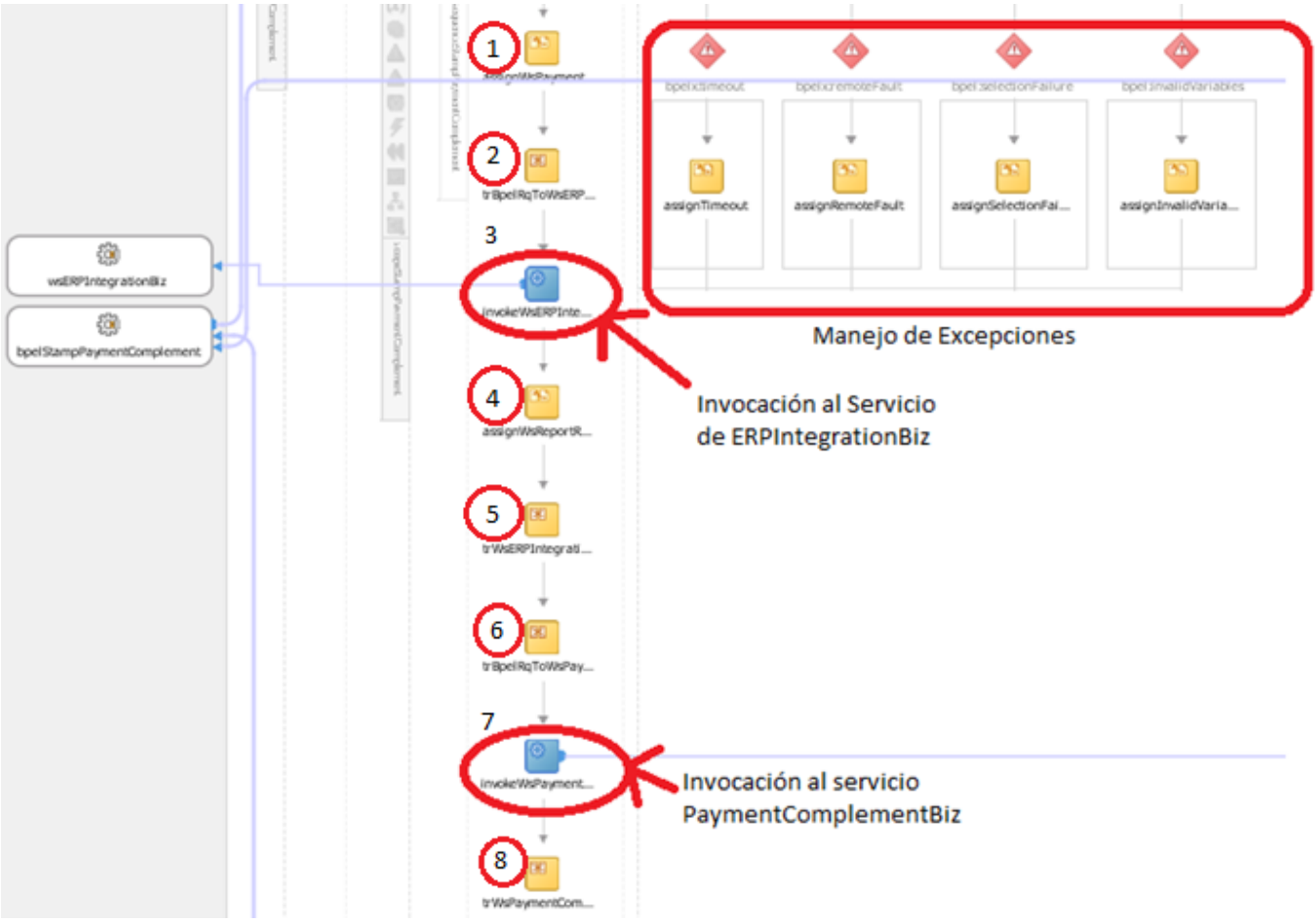
En la capa PaymentComplementEnt se realiza una serie de pasos concentrados en un bpel como se muestra en la siguiente imagen:



Como servicio expuesto de lado izquierdo tenemos la operación **StampPaymentComplement** que es la operación programada en el ESS, posterior utilizamos un mediator para dirigir a **bpelStampPaymentComplement** donde se concentra la lógica para hacer el llamado a las capas Biz correspondientes (PaymentComplementBiz y ErpIntegrationBiz).

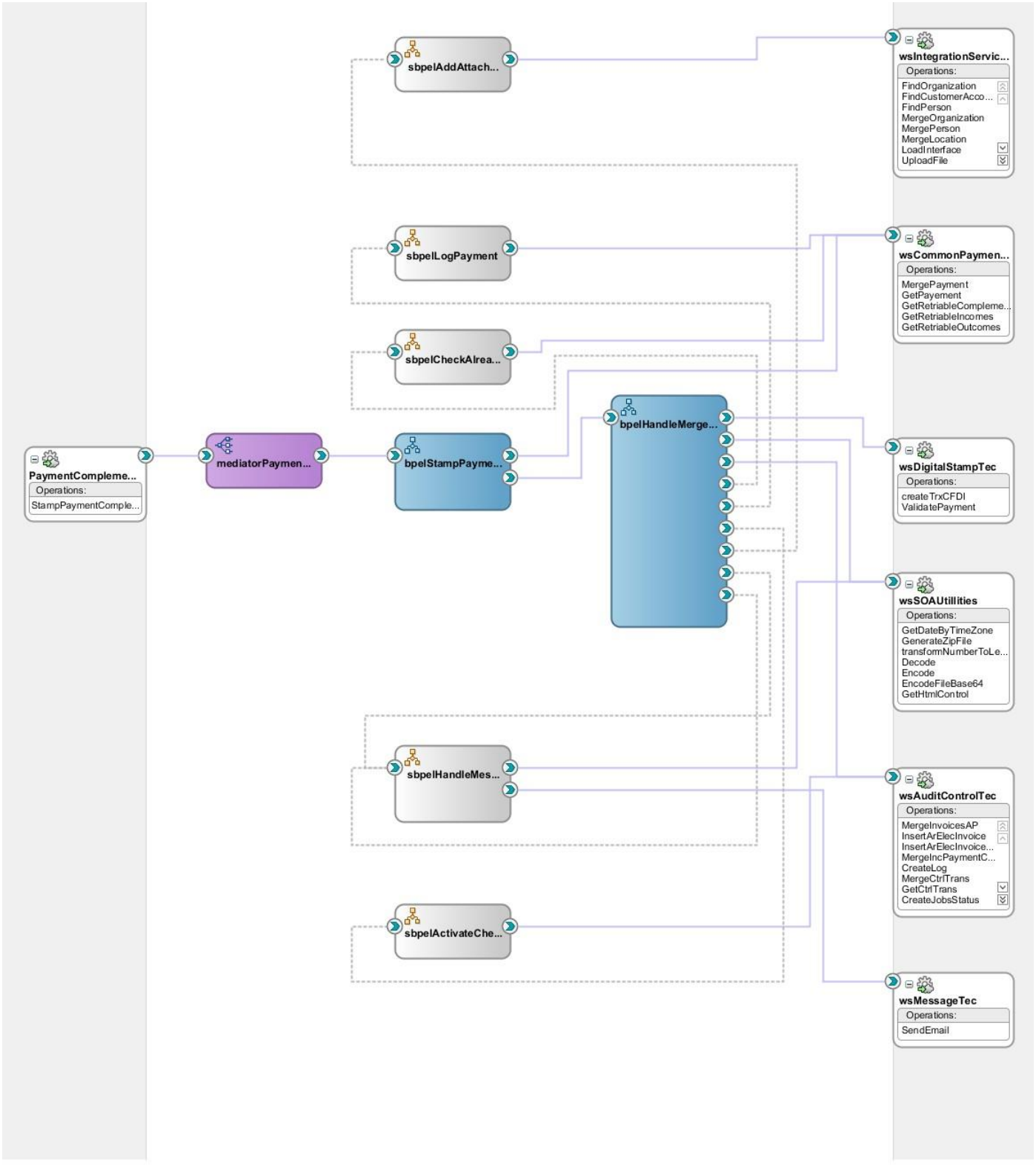
El bpel realiza las siguientes acciones:

1. Se crea la variable de Request a enviar.
2. Se llena con los datos necesarios para la solicitud.
3. Se invoca el servicio de ERPIntegractionBiz.
4. Se asigna el resultado a una colección
5. Como las variables de salida del servicio son diferentes a las de entrada del siguiente servicio a llamar, se mapean los datos.
6. Se pasan los datos a la variable que se enviará al siguiente servicio.
7. Se invoca el servicio de PaymentComplementBiz.
8. Se guarda la respuesta en la variable de salida.



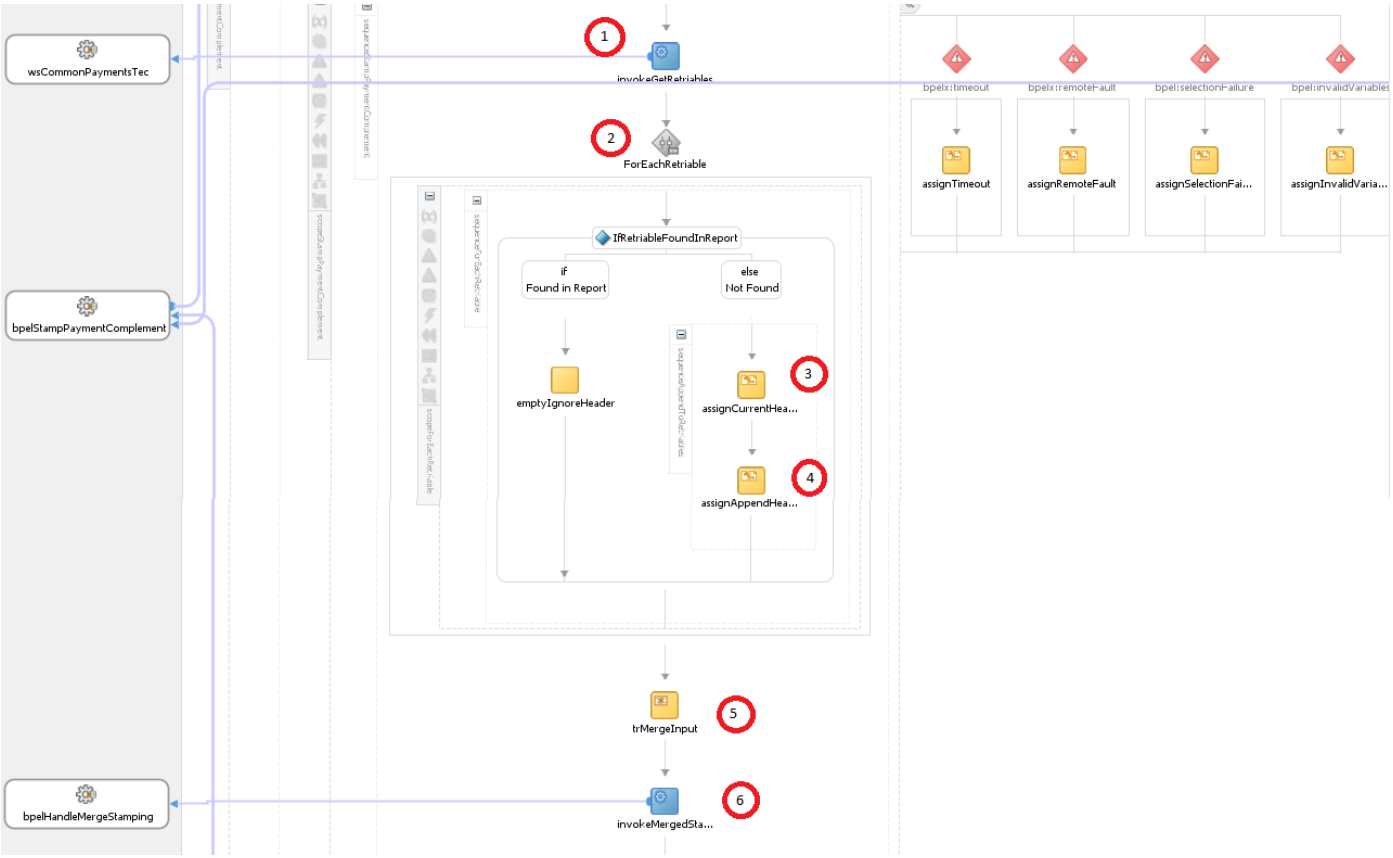
3.2 PaymentComplementBiz

En la capa **PaymentComplementBiz** se tiene una mayor cantidad de pasos a realizar debido a que se gestiona toda la lógica del timbrado. Se parte de la operación expuesta StampPaymentComplement, posterior se tiene un mediador que direcciona al bpelPaymentComplement y este gestiona una serie de pasos que invoca al bpleHandleMerge, a continuación, se explicará a detalle cada uno de ellos. El proceso general se muestra en la siguiente imagen:


sbpelAddAttach...
wsIntegrationServic...
Operations:
FindOrganization
FindCustomerAcco...
FindPerson
MergeOrganization
MergePerson
MergeLocation
LoadInterface
UploadFile
sbpelLogPayment
wsCommonPaymen...
Operations:
MergePayment
GetPayement
GetRetriableCompleme...
GetRetriableIncomes
GetRetriableOutcomes
sbpelCheckAlrea...
bpelHandleMerge...
PaymentCompleme...
Operations:
StampPaymentComple...
mediatorPaymen...
bpelStampPayme...
wsDigitalStampTec
Operations:
createTrxCFDI
ValidatePayment
wsSOAUtillities
Operations:
GetDateByTimeZone
GenerateZipFile
transformNumberToLe...
Decode
Encode
EncodeFileBase64
GetHtmlControl
sbpelHandleMes...
wsAuditControlTec
Operations:
MergeInvoicesAP
InsertArElecInvoice
InsertArElecInvoice...
MergeIncPaymentC...
CreateLog
MergeCtrlTrans
GetCtrlTrans
CreateJobsStatus
sbpelActivateChe...
wsMessageTec
Operations:
SendEmail

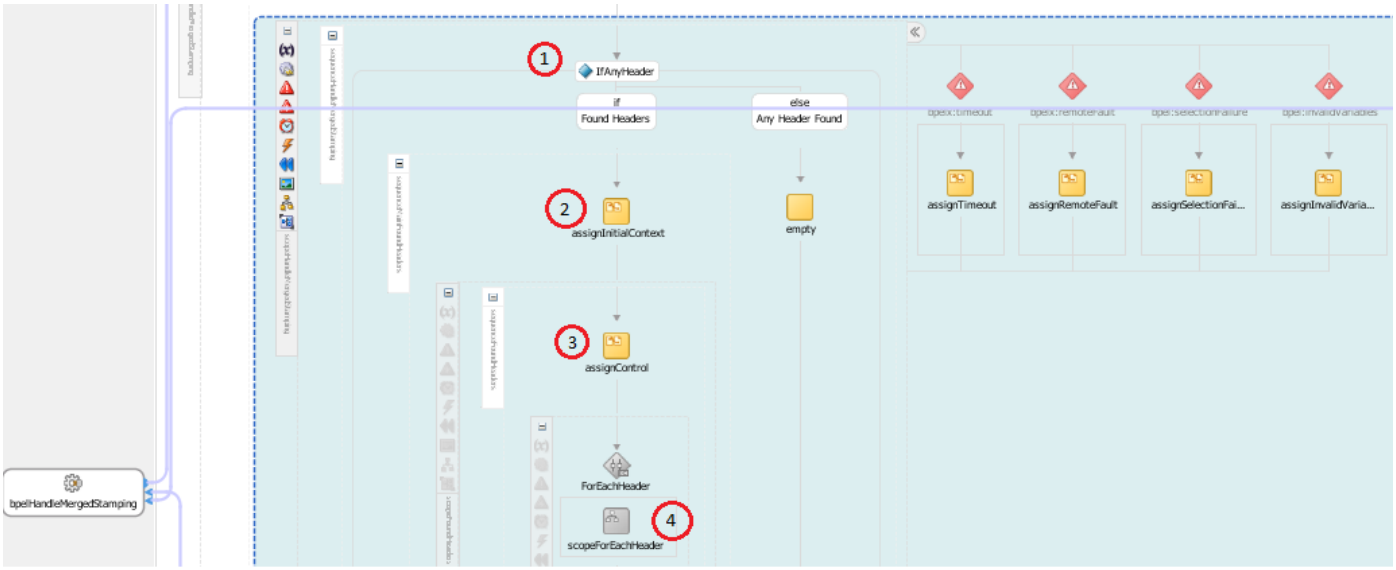
El **bpelStampPaymentCompement** contiene los siguientes pasos:

1. Llama al servicio de CommonPaymentTec para obtener los registros a reprocesar en caso de que haya habido un error de conexión al PAC previamente.
2. Por cada registro que se obtiene se le da un tratamiento, si no se obtiene nada, se pasa en automático al paso 5 de lo contrario se realiza el paso 3.
3. Se hace un mapeo para tener los datos obtenidos en la variable de request que necesitamos.
4. Se agrega el registro a una colección donde se tendrán todos los registros encontrados en el tipo de variable que corresponde.
5. Se realiza un Merge entre los datos a reprocesar y los datos nuevos.
6. Se invoca el bpelHandleMergeStamping para seguir con el proceso de timbrado.



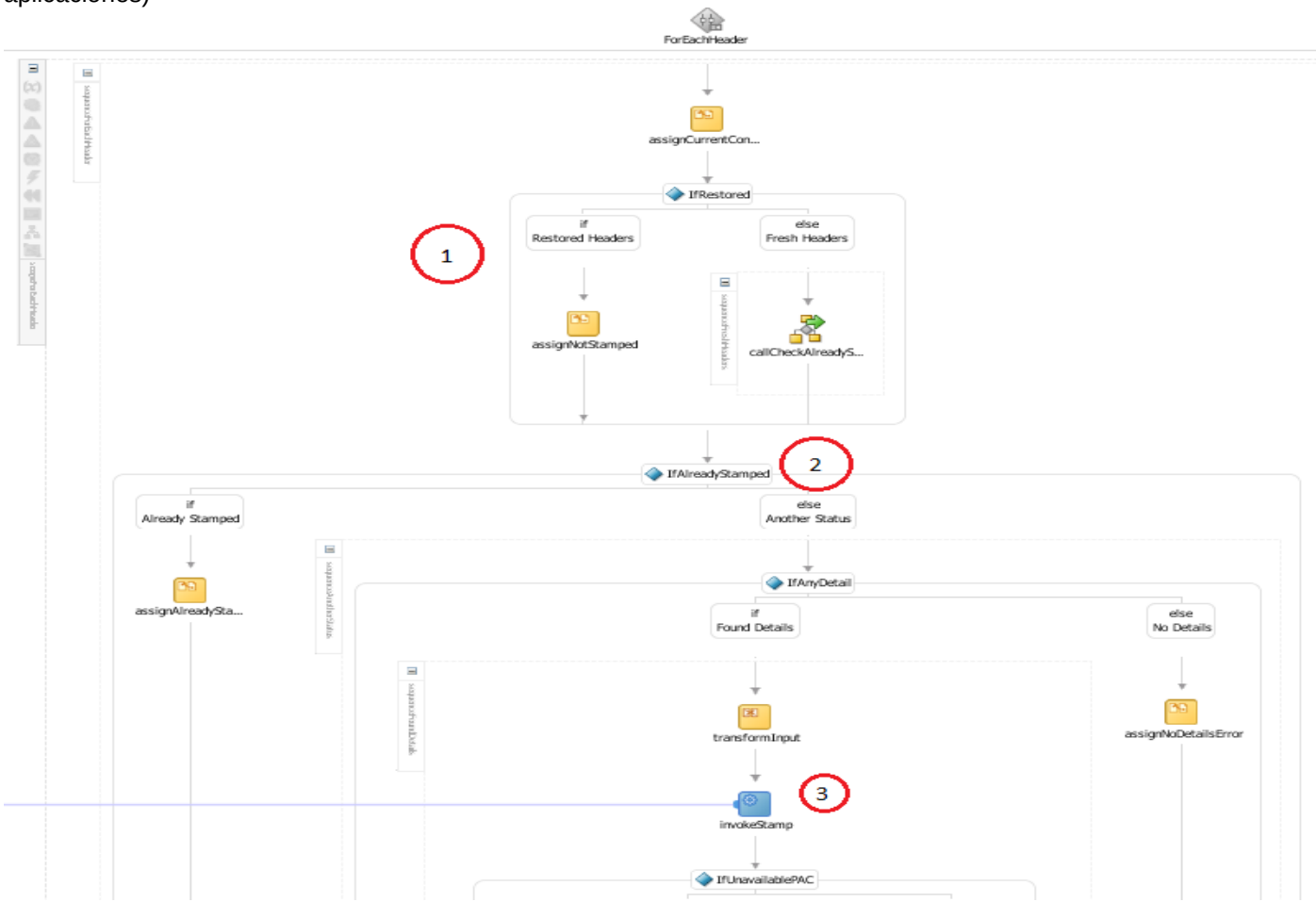
El **bpelHandleMergeStamping** contiene la mayor lógica en el proceso realizando los siguientes pasos:

1. Se evalúa si hay recibos por procesar, si no hay nada que procesar no se realiza ninguna acción, si existe registro se realiza el paso 2.
2. Se inicializan las variables de contexto para almacenar los errores y sus códigos.
3. Se inicializan las variables de control para el conteo de registros procesados, éxitos y errores.
4. Por cada registro a procesar se realiza una serie de pasos explicados más adelante.

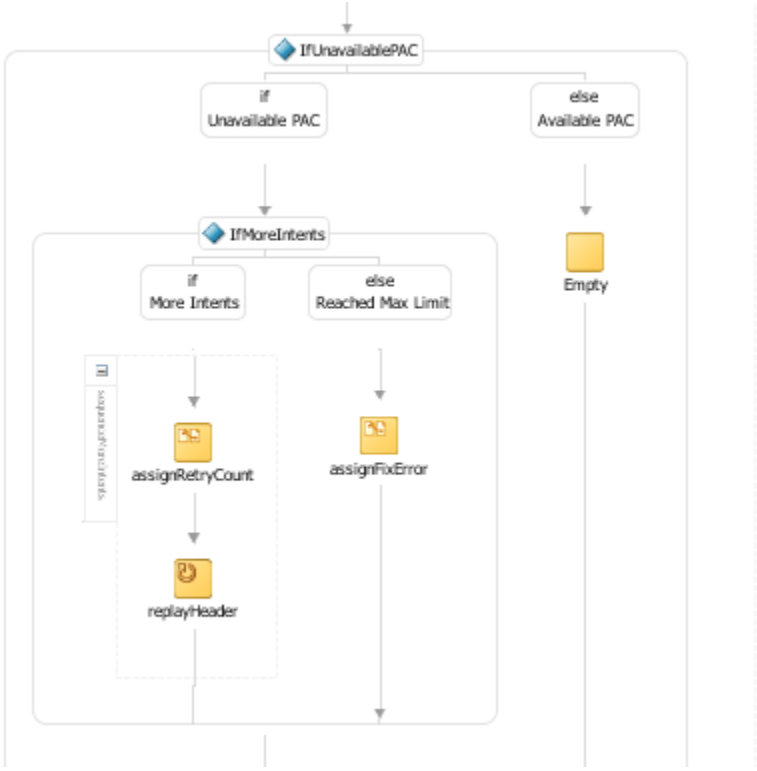


En la siguiente imagen se describen los pasos contenidos en el scope de ForEachHeader:

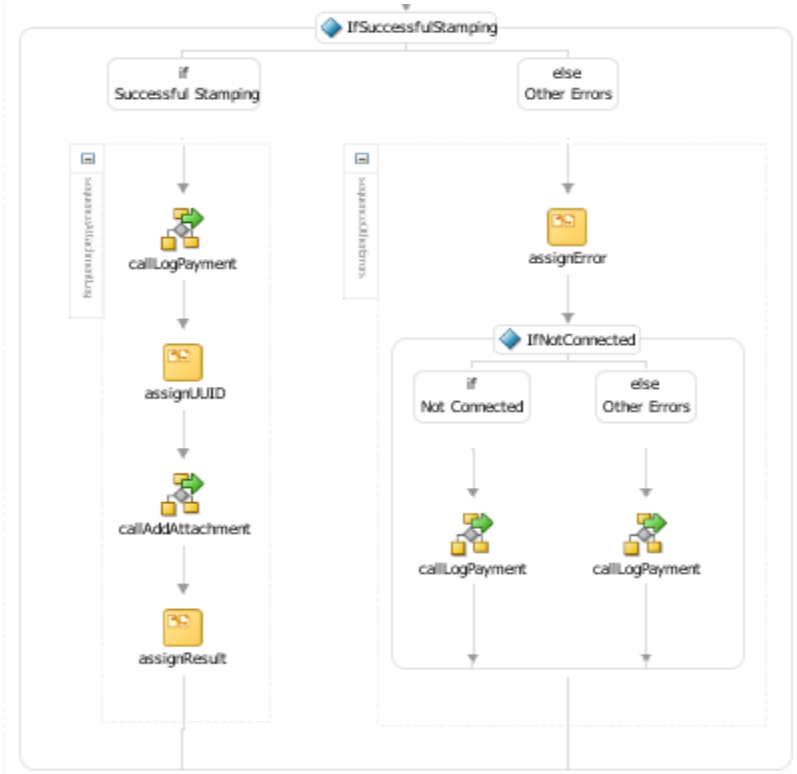
1. Se evalúa si es un registro a reprocesar y de ser así, si este ya está timbrado.
2. Y esta timbrado el recibo, entonces no se procesa. Si no ha sido timbrado entonces se evalúa si tiene aplicaciones si no tiene no se procesa
3. Se envía a timbrar el elemento que cumple con los requisitos primarios (no estar timbrado y tener aplicaciones)



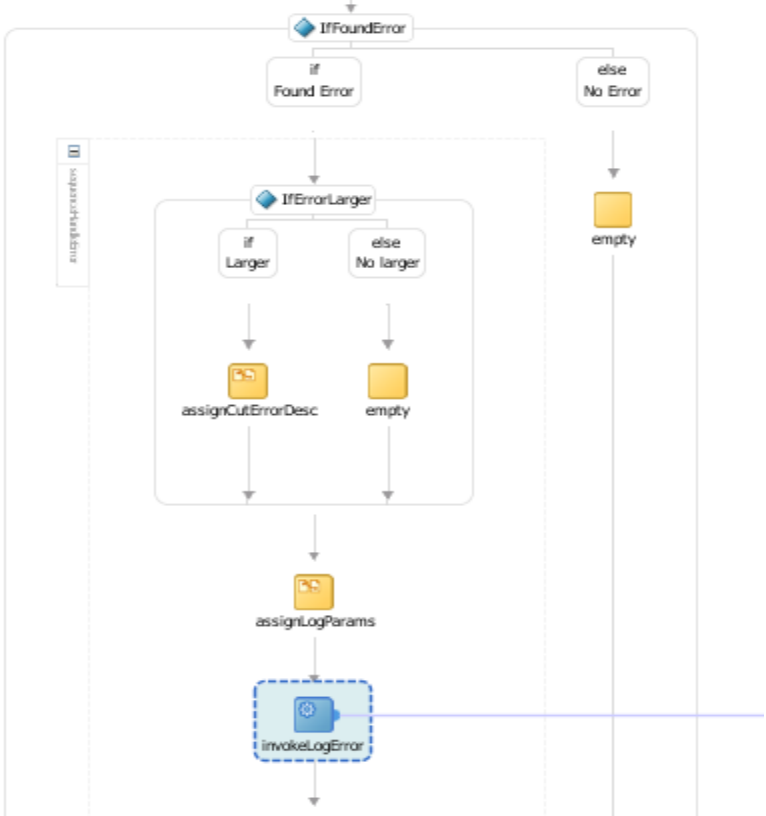
4. Si no está disponible la conexión con el PAC entonces se reintenta 5 veces y después el registro se manda a error si no se realiza la conexión exitosamente.



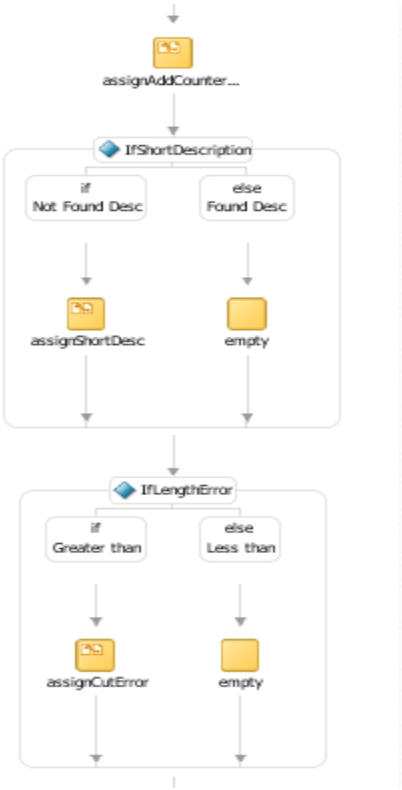
5. Si el registro se timbró correctamente entonces se almacena en las tablas de control del esquema custom de base de datos y se agrega los adjuntos al recibo, de lo contrario solo se almacena el estatus con el que termino el recibo.



6. Si se encuentra un error entonces se verifica que su longitud no sea mayor a 1000 de ser así se corta para poder ser guardado en la tabla de errores del esquema custom.

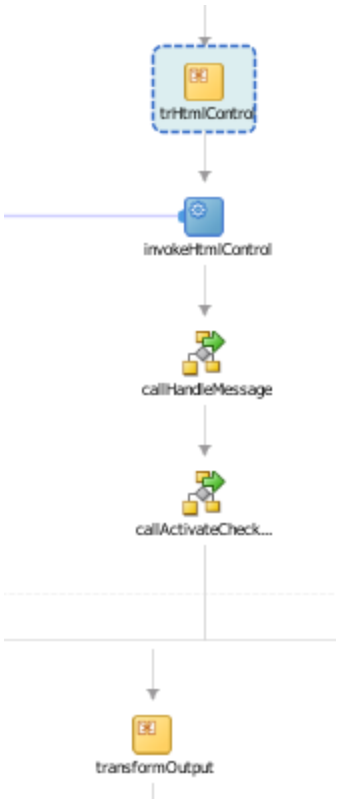


7. Se agregan los errores al contador y se evalúa si se cuenta con una Descripción corta, de no contar con una se asigna la descripción larga, si la descripción es mayor a 140 se corta para poder ser enviada por notificación



8. Posterior, se evalúa si es el primer error. Si es el primer error se asigna sin problema a la colección, pero si no, se crea primero un nodo en la colección y posterior se asignan los datos.

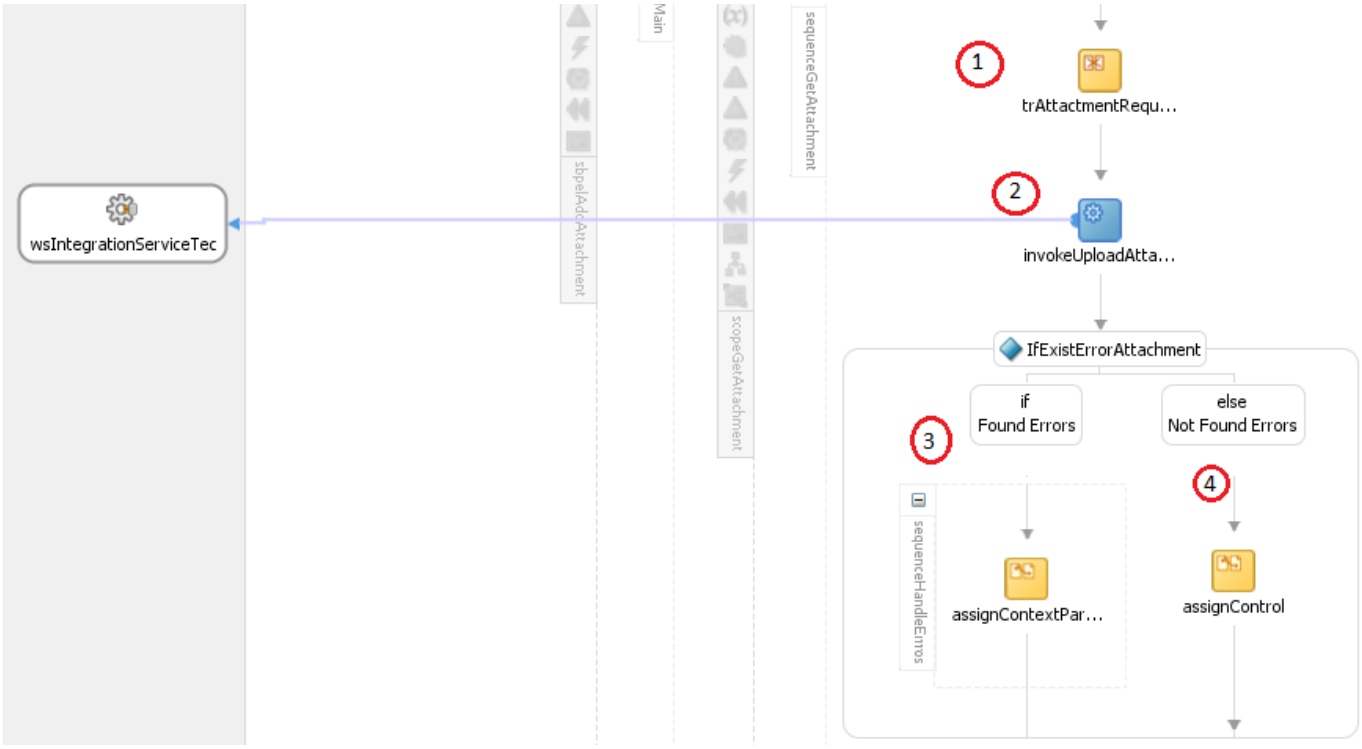
9. Finalmente se configura el formato de la notificación se envía, se almacena la hora de la ejecución y se termina el proceso.



Son cinco subprocesos que se manda a ejecutar en este proceso, los cuales se describen detalladamente a continuación.

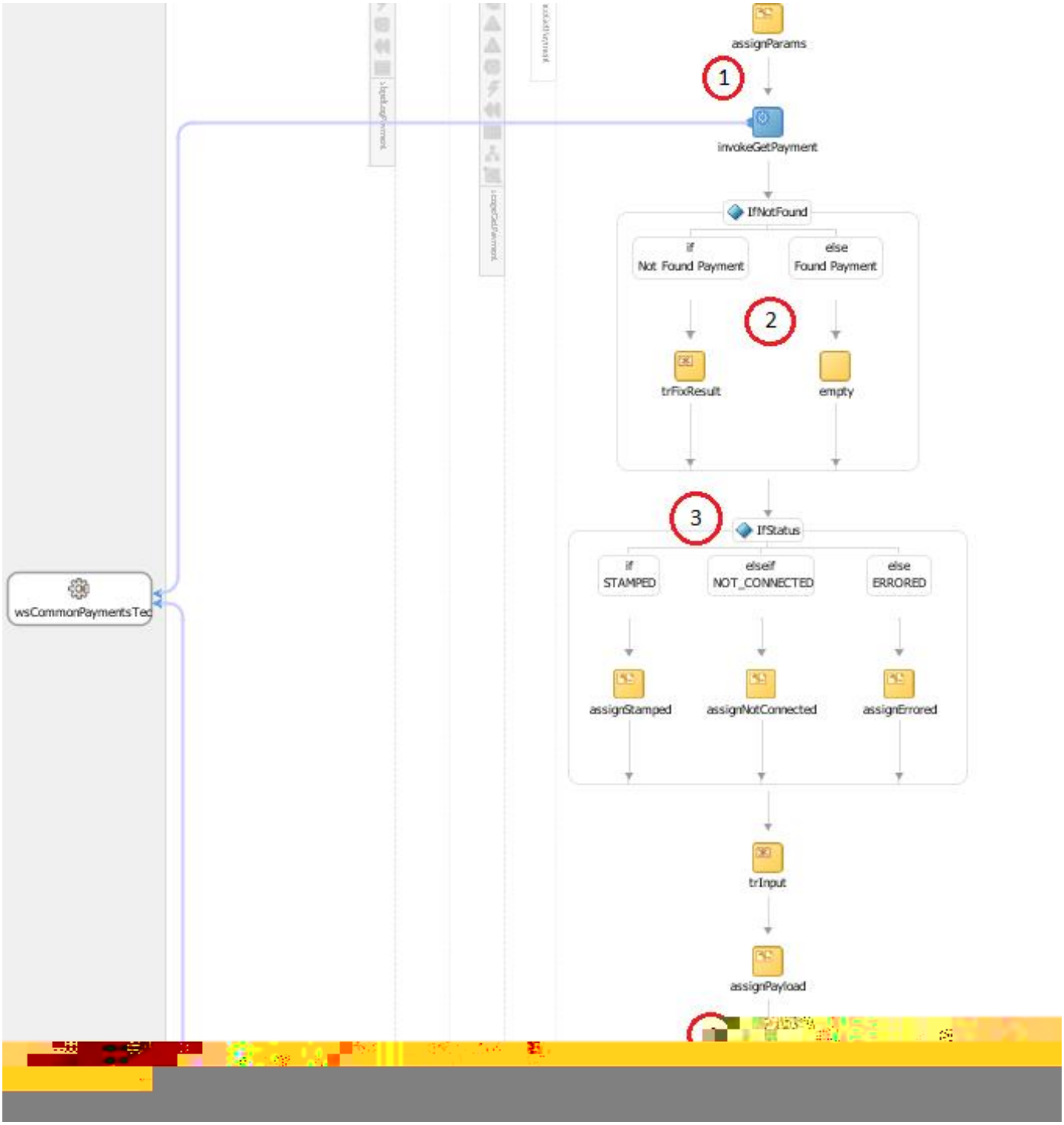
SpbelAddAttachment:

1. Se asignan los valores a la variable request que se enviara a IntegracionServiceTec.
2. Se invoca la operación UploadAttachment en IntegatrionServiceTec
3. Se evalúa si existen errores, archivos que no se adjuntaron, en este caso se asignan las variables de error.
4. Se asigna las variables de control.



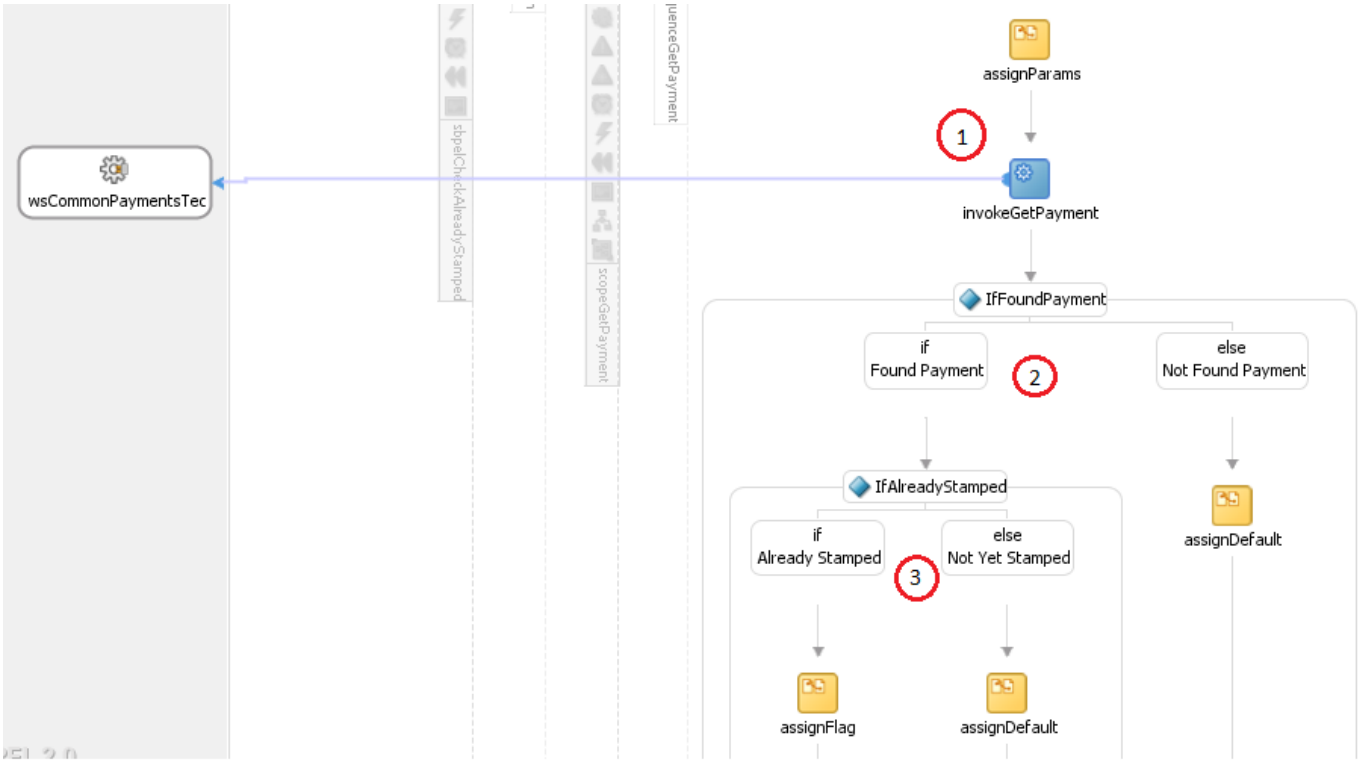
SbelLogPayment:

1. Asignamos los valores a la variable de request para el servicio CommonPaymentTec y mandamos a ejecutar la operación GetPayment para saber si el registro a insertar en el Log ya existe.
2. Si no existe, el registro se crea
3. Se actualiza el estatus del registro dependiendo cual haya sido el resultado, exitoso, no conectado o error.



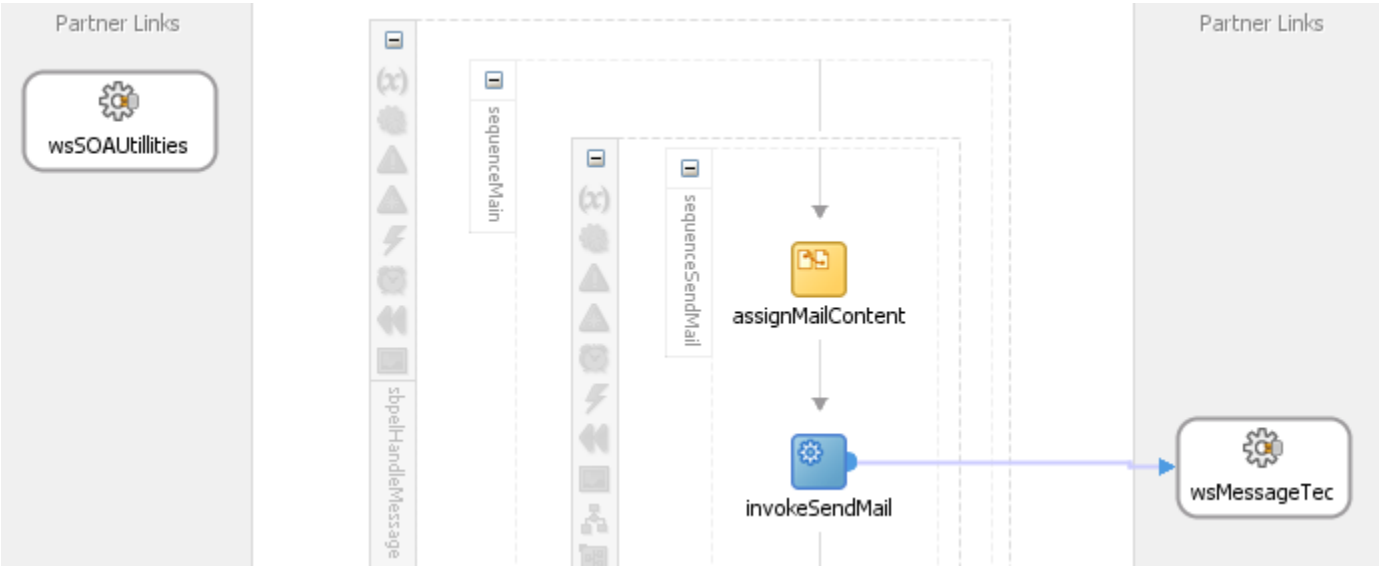
SbpelCheckAlreadyStamped:

1. Se asignan los valores para la variable que se enviará al servicio CommonPaymentTec y se ejecuta la operación GetPayment.
2. Si se encuentra un registro, se evalúa si ha sido timbrado de lo contrario se asigna la variable con el estatus de timbrado en false().
3. Cuando se evalúa si ya está timbrado se asigna la variable correspondiente a timbrado true(), de lo contrario se pone false().



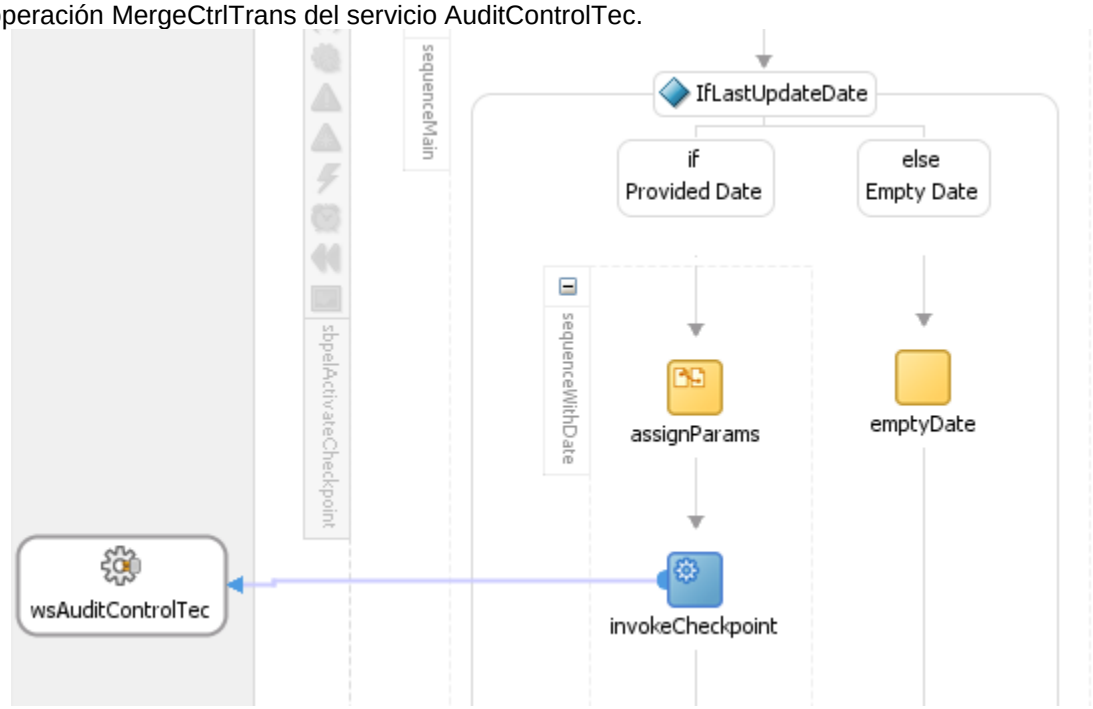
SbpelHandleMessage:

1. Se asignan los valores a enviar al servicio de MessageTec y se ejecuta la operación SendEmail.



SbpelActivateCheckpoint:

1. Se verifica si ya se cuenta con una fecha de ejecución, de no contar con una se asigna la fecha y se ejecuta la operación MergeCtrlTrans del servicio AuditControlTec.



3.3 Pruebas

Las pruebas de complementos de pagos se realizaron mediante la ejecución del compuesto desde el llamado de su endpoint con los parametros requeridos. Donde se puede modificar la fecha de inicio y fin que deseamos y ejecutarlo de manera manual.

URL del servicio en el ambiente de desarrollo:

http://129.150.110.0:80/soa-infra/services/default/PaymentComplementEnt/PaymentComplementEnt?WSDL

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:pay="http://soa.estrellaroja.com.mx/PaymentComplementEnt">
   <soap:Header/>
   <soap:Body>
      <pay:StampPaymentComplementRq>
         <!--Optional:-->
         <pay:StartDate>2018-01-12</pay:StartDate>
         <!--Optional:-->
         <pay:EndDate></pay:EndDate>
      </pay:StampPaymentComplementRq>
   </soap:Body>
</soap:Envelope>
```

4 Diseño de Datos

En este punto abordaremos todo lo referente a la parte de diseño de los datos utilizados durante la integración.

4.1 Tabla de diseño de datos

Los bpels utilizados en esta integración cuentan con los siguientes componentes.

bpelStampPaymentComplement.bpel:

#	Nombre	Tipo	Descripción
1	invokeGetRetriables	invoke	Invocación de Servicio.
2	ForEachRetriable	forEach	Componente de repetición.
3	trMergeInput	transformation	Transformación de datos.
4	invokeMergedStamping	invoke	Invocación de Servicio.
5	IfRetriableFoundInReport	if	Validación de datos.
6	assignCurrentHeader	assign	Asignación de valores.
7	assignAppendHeader	assign	Asignación de valores.

bpelHandleMergeStamping:

#	Nombre	Tipo	Descripción
1	IfAnyHeader	if	Validación de datos.
2	transformOutput	transformation	Transformación de datos.
3	assignInitialContext	assign	Asignación de valores.
4	trHtmlControl	transformation	Transformación de datos.
5	invokeHtmlControl	invoke	Invocación de Servicio.
6	callCheckAlreadyStamped	extensionActivity	Se llama a un subproceso
7	callLogPayment	extensionActivity	Se llama a un subproceso
8	assignControl	assign	Asignación de valores.
9	sequenceForEachHeader	scope	Secuencia del iterador de Header
10	assignCurrentContext	assign	Asignación de valores.
11	IfRestored	if	Validación de datos.
12	IfAlreadyStamped	if	Validación de datos.
13	IfFoundError	if	Validación de datos.
14	callAddAttachment	extensionActivity	Se llama a un subproceso
15	IfAnyDetail	if	Validación de datos.
16	transformInput	transformation	Transformación de datos.
17	invokeStamp	invoke	Invocación de Servicio.
18	IfUnavailablePAC	if	Validación de datos.

19	IfSuccessfulStamping	if	Validación de datos.
20	assignRetryCount	assign	Asignación de valores.
21	callHandleMessage	extensionActivity	Llamado a un subproceso
22	assignUUID	assign	Asignación de valores.
23	callActivateCheckpoint	extensionActivity	Llamado a un subproceso
24	assignResult	assign	Asignación de valores.
25	assignError	assign	Asignación de valores.
26	IfNotConnected	if	Validación de datos.
27	IfErrorLarger	if	Validación de datos.
28	assignLogParams	assign	Asignación de valores.
29	invokeLogError	invoke	Invocación de Servicio.
30	assignAddCounterError	assign	Asignación de valores.
31	IfShortDescription	if	Validación de datos.
32	IfLengthError	if	Validación de datos.
33	IfFirstError	if	Validación de datos.
34	assignPosition	assign	Asignación de valores.
35	assigAppedError	assign	Asignación

4.2 Origen de Datos

Hace referencia a las tablas custom empleadas en esta integración, como es que se consultan y se almacena la información correspondiente a Complementos de pago.

Tabla: **XXER_PAYMENTS**

Columna	Tipo de dato	Descripción
PAYMENT_ID	VARCHAR2(150 CHAR)	Identificador unico del Pago
CASH_RECEIPT_ID	VARCHAR2(3 CHAR)	Identificador del Recibo AR
RECEIPT_NUMBER	VARCHAR2(150 CHAR)	Número de recibo
CUSTOMER_NUMBER	NUMBER	Número de cliente
ACCOUNT_NUMBER	NUMBER	Número de cuenta
SITE_NUMBER	NUMBER	Número de Site
UUID	CLOB	UUID generado por el SAT
UUID_DATE	VARCHAR2(150 CHAR)	Fecha de creación del UUID
STATUS	VARCHAR2(150 CHAR)	Estatus { 'STAMPED', 'ERRORED', 'NOT_CONNECTED'}
PAYMENT_TYPE	VARCHAR2(150 CHAR)	Type { 'INCOME', 'OUTCOME', 'COMPLEMENT'}
RECEIP_METHOD	VARCHAR2(150 CHAR)	Metodo del recibo
CONTENT	TIMESTAMP(6)	Payload del registro
CREATED_BY	VARCHAR2(150 CHAR)	Indica el usuario que creo el registro
CREATION_DATE	TIMESTAMP(6)	Indica la fecha de creación
LAST_UPDATED_BY	VARCHAR2(150 CHAR)	Indica el último usuario que modificó el registro
LAST_UPDATE_DATE	TIMESTAMP(6)	Indica la última fecha de modificación
SOURCE_LAST_UPDATED_BY	VARCHAR2(150 CHAR)	Indica el usuario que actualizó por última vez el recibo

		en el momento del procesamiento
OBJECT_VERSION_NUMBER	NUMBER(9,0)	Número de modificaciones del registro

Tabla: **XXER_PAYMENT_DETAILS**

Columna	Tipo de dato	Descripción
PAYMENT_DETAIL_ID	NUMBER(18,0)	Identificador unico del detalle de pago
PAYMENT_ID	NUMBER(18,0)	ID del Payment
INVOICE_NUMBER	VARCHAR2(150 CHAR)	Número de Factura
INSTALLMENT_NUMBER	NUMBER(3,0)	Parcialidad
RECEIVABLE_APPLICATION_ID	NUMBER(18,0)	AR Application ID
TRX_NUMBER	VARCHAR2(20 CHAR)	AR Referencia de Aplicación
BALANCE_BEFORE	NUMBER	Saldo Previo
BALANCE_AFTER	NUMBER	Saldo Pendiente
UUID	VARCHAR2(150 CHAR)	UUIDde la factura
STATUS	VARCHAR2(150 CHAR)	Estatus { ''STAMPED'', ''ERRORED'', ''NOT_CONNECTED''}
CURRENCY_CODE	VARCHAR2(3 CHAR)	Código de moneda
CREATED_BY	VARCHAR2(150 CHAR)	Indica el usuario que creo el registro
CREATION_DATE	TIMESTAMP(6)	Indica la fecha de creación
LAST_UPDATED_BY	VARCHAR2(150 CHAR)	Indica el último usuario que modificó el registro
LAST_UPDATE_DATE	TIMESTAMP(6)	Indica la última fecha de modificación
OBJECT_VERSION_NUMBER	NUMBER(9,0)	Número de modificaciones del registro

4.3 Lógica de Validación

Esta lógica hace referencia a las validaciones que se emplean en los origines para poder tener la información congruente y poder procesarla para finalmente timbrar el complemento de pago.

Sin embargo, podemos mencionar que las principales validaciones de negocio son las siguientes:

- El recibo debe estar listo para timbrar
- No debe ser un anticipo
- Debe ser una creación manual y no provenir de la superinterface.
- Sus aplicaciones debieron ser previamente timbradas
- Sus aplicaciones deben ser de tipo de pago 99-Por definir
- Solo nos interesan las aplicaciones de tipo factura

Las cuales se pueden observar en los queries del tema 5.1 Sentencias SQL, también se cuenta con las validaciones explicadas en el punto 3. Lógica de implementación

5 Diseño SQL

En el diseño SQL se abarca principalmente las consultas utilizadas para la extracción de la información del ERP Cloud.

5.1 Sentencias SQL

En esta integración existen principalmente dos consultas, una obtiene todos los recibos pertinentes que se enviarán a timbrar y la otra obtiene todas las aplicaciones del recibo.

Query para obtener los recibos a timbrar.

```
SELECT
cra.CASH_RECEIPT_ID,
cra.RECEIPT_NUMBER,
xr.REGISTRATION_NUMBER issuer_rfc,
xr.REGISTERED_NAME issuer_name,
hl.POSTAL_CODE issuer_zip_code,
--TO_CHAR(cra.RECEIPT_DATE, 'yyyy-mm-dd"T"hh24:mi:ss') invoice_date,
cra.RECEIPT_DATE invoice_date,
'' document_type, -- DVM 'P'
'' additional_information,
'' total_words,
'' description_itm,--DVM 'Pago'
'' prod_serv_key_itm,--DVM 84111506
1  quantity_itm,
0  vat_percentage_itm,
'' unit_key_itm, --DVM ACT
0  amount_itm,
0  subtotal_itm,
0  vat_itm  ,
0  total_itm,
0  total_vat,
0  subtotal,
0  total,
'' serie, -- DVM PQ
'' branch_name, -- MATRIZ DVM
hl.POSTAL_CODE branch_zip_code, -- Mismo dato que issuer_zip_code
hp.PARTY_NAME receiver_name,
DECODE(hp.party_type,'ORGANIZATION', op.JGZZ_FISCAL_CODE, pp.JGZZ_FISCAL_CODE)
receiver_rfc,
'' receiver_phone, --NI EX
hzl.ADDRESS1 receiver_address,
'' receiver_ext_number,
'' receiver_int_number,
hzl.ADDR_ELEMENT_ATTRIBUTE3 receiver_colony,
hzl.ADDR_ELEMENT_ATTRIBUTE2 receiver_county,
cra.ATTRIBUTE3 receiver_cfdi_use, --DEBE SER 'P01'
'' currency, --DVM VALOR POR DEFECTO
'' issuer_tax_regime, --DVM 624
'' payment_status,--No va
cra.COMMENTS comments,
'' service, --DVM PAQUER
SYSDATE payment_date_pc, --pc (payment complements)
cra.ATTRIBUTE4 payment_way_pc,
cra.CURRENCY_CODE currency_pc,
0  currency_exchange_pc,
```

```
 (SELECT SUM(ara.amount_applied)
     FROM ar_receivable_applications_all ara
     WHERE 1=1
           AND ara.status = 'APP'
         AND ara.cash_receipt_id = cra.cash_receipt_id
     GROUP BY ara.status
 ) amount_pc,
1  operation_number_pc,
'' bank_name_pc,
'' payer_account_pc,
'' ben_acc_issuer_rfc_pc,
'' beneficiary_account_pc,
'' payment_string_type_pc,
'' payment_cert_pc,
'' payment_string_pc,
'' payment_seal_pc,
1 version_pc,
hp.PARTY_NUMBER AS CUSTOMER_NUMBER,
hca.ACCOUNT_NUMBER,
csu.LOCATION AS SITE_NUMBER,
cra.LAST_UPDATE_DATE AS LAST_UPDATE_DATE,
cra.LAST_UPDATED_BY AS LAST_UPDATED_BY,
ho.name BUSINESS_UNIT,
rm.name RECEIPT_METHOD
FROM HR_OPERATING_UNITS ho,
     XLE_ENTITY_PROFILES xe,
     XLE_REGISTRATIONS xr,
       HZ_LOCATIONS hl,
       AR_CASH_RECEIPTS_ALL cra,
       HZ_CUST_ACCOUNTS hca,
       HZ_PARTIES hp,
       HZ_organization_profiles op,
       HZ_person_profiles pp,
       HZ_CUST_SITE_USES_ALL csu,
       HZ_CUST_ACCT_SITES_ALL cas,
       HZ_CUST_ACCOUNTS hca1,
       HZ_PARTIES hp1,
       HZ_PARTY_SITES ps,
       HZ_LOCATIONS hzl,
       AR_RECEIPT_METHODS rm,
        AR_RECEIPT_CLASSES rc
WHERE 1=1
AND cra.ORG_ID = ho.ORGANIZATION_ID
AND xe.LEGAL_ENTITY_ID           =  ho.DEFAULT_LEGAL_CONTEXT_ID
AND xe.LEGAL_ENTITY_ID           =  xr.SOURCE_ID
AND hl.LOCATION_ID = xr.LOCATION_ID
AND hp.party_id = hca.party_id
AND op.party_id(+)=hp.party_id
AND pp.party_id(+)=hp.party_id
AND cra.PAY_FROM_CUSTOMER = hca.CUST_ACCOUNT_ID
AND cra.ATTRIBUTE1 = 'Y'    --Listo para timbrar
AND csu.SITE_USE_ID = cra.CUSTOMER_SITE_USE_ID
AND csu.CUST_ACCT_SITE_ID =cas.CUST_ACCT_SITE_ID
AND hca1.CUST_ACCOUNT_ID = cas.CUST_ACCOUNT_ID
AND hp1.party_id = hca1.party_id
AND cas.PARTY_SITE_ID= ps.PARTY_SITE_ID
AND ps.LOCATION_ID=hzl.LOCATION_ID
AND rm.RECEIPT_METHOD_ID =cra.RECEIPT_METHOD_ID
```

```
AND rm.RECEIPT_CLASS_ID=rc.RECEIPT_CLASS_ID
AND NVL(rc.ATTRIBUTE1,'N')<> 'Y' --Es anticipo
AND NVL(cra.ATTRIBUTE6,'X') <>'Y'-- Es recibo Manual, no proviene de la super
interface
--boundary dates
 AND (cra.LAST_UPDATE_DATE > TO_TIMESTAMP(:p_startDate, 'yyyy-MM-dd hh24:mi:SS.FF9')
 AND cra.LAST_UPDATE_DATE <= NVL(TO_TIMESTAMP(:p_endDate, 'yyyy-MM-dd
hh24:mi:SS.FF9'), CURRENT_TIMESTAMP)
  );
```

Query para obtener las aplicaciones del recibo:

```
SELECT
cra.CASH_RECEIPT_ID,
rct.TRX_NUMBER INVOICE_NUMBER,
rct.ATTRIBUTE3 DOCUMENT_ID,
'PQ' SERIE, --DVM PQ
rct.ATTRIBUTE5 FOLIO,
rct.INVOICE_CURRENCY_CODE CURRENCY,
0 CURRENCY_EXCHANGE,
rct.ATTRIBUTE12 PAYMENT_METHOD,
(SELECT TO_CHAR(COUNT(ara.APPLIED_CUSTOMER_TRX_ID))
          FROM AR_RECEIVABLE_APPLICATIONS_ALL arai
          WHERE 1 = 1
            AND arai.APPLIED_CUSTOMER_TRX_ID = rct.CUSTOMER_TRX_ID
            AND arai.RECEIVABLE_APPLICATION_ID <= ara.RECEIVABLE_APPLICATION_ID
        )AS PARTIALITY_NUMBER,
 (SELECT SUM(rctl.EXTENDED_AMOUNT)
         FROM RA_CUSTOMER_TRX_LINES_ALL rctl
         WHERE 1 = 1
           AND rctl.CUSTOMER_TRX_ID = rct.CUSTOMER_TRX_ID) - (SELECT
NVL(SUM(arai.AMOUNT_APPLIED),0)
                                                              FROM
AR_RECEIVABLE_APPLICATIONS_ALL arai
                                                              WHERE 1 = 1
                                                                AND
arai.APPLIED_CUSTOMER_TRX_ID = rct.CUSTOMER_TRX_ID
                                                                AND
arai.RECEIVABLE_APPLICATION_ID < ara.RECEIVABLE_APPLICATION_ID
        ) AS PREVIOUS_BALANCE,
TO_CHAR(ara.AMOUNT_APPLIED) PAID_AMOUNT,
(SELECT SUM(rctl.EXTENDED_AMOUNT)
         FROM RA_CUSTOMER_TRX_LINES_ALL rctl
         WHERE 1 = 1
           AND rctl.CUSTOMER_TRX_ID = rct.CUSTOMER_TRX_ID) - (SELECT
NVL(SUM(arai.AMOUNT_APPLIED),0)
                                                              FROM
AR_RECEIVABLE_APPLICATIONS_ALL arai
                                                              WHERE 1 = 1
                                                                AND
arai.APPLIED_CUSTOMER_TRX_ID = rct.CUSTOMER_TRX_ID
                                                                AND
arai.RECEIVABLE_APPLICATION_ID <= ara.RECEIVABLE_APPLICATION_ID
        ) AS OUTSTANDING_BALANCE,
rct.CUSTOMER_TRX_ID CUSTOMER_TRX_LINE_ID,
cra.RECEIPT_NUMBER,
```

```
 (SELECT COUNT(ara.APPLIED_CUSTOMER_TRX_ID)
         FROM AR_RECEIVABLE_APPLICATIONS_ALL arai
         WHERE 1 = 1
           AND arai.APPLIED_CUSTOMER_TRX_ID = rct.CUSTOMER_TRX_ID
           AND arai.RECEIVABLE_APPLICATION_ID <= ara.RECEIVABLE_APPLICATION_ID
        ) AS INSTALLMENT_NUMBER,
        ara.RECEIVABLE_APPLICATION_ID AS RECEIVABLE_APPLICATION_ID,
        rct.TRX_NUMBER AS TRX_NUMBER
FROM AR_RECEIVABLE_APPLICATIONS_ALL ara,
       RA_CUSTOMER_TRX_ALL rct,
          AR_CASH_RECEIPTS_ALL cra,
         AR_RECEIPT_METHODS rm,
         AR_RECEIPT_CLASSES rc
  WHERE 1 = 1
    AND ara.APPLIED_CUSTOMER_TRX_ID = rct.CUSTOMER_TRX_ID
      AND rm.RECEIPT_METHOD_ID =cra.RECEIPT_METHOD_ID
    AND rm.RECEIPT_CLASS_ID=rc.RECEIPT_CLASS_ID
      AND rct.ATTRIBUTE3 is not null --UUID no es nulo
      AND rct.ATTRIBUTE10 = '99'  --Tipo de pago
    AND rct.TRX_CLASS = 'INV'  --Solo facturas
      AND ara.CASH_RECEIPT_ID = cra.CASH_RECEIPT_ID
    AND ara.ORG_ID = cra.ORG_ID;
```

6 Reglas de Negocio

Las reglas de negocio se describen a continuación, para saber más detalle puedes observar el documento TA020-4_ER03_AR_Timbrado de Complementos de Pago donde se especifica todo el requerimiento.

- Solo se timbrarán los recibos que son creados de forma manual dentro del ERP Cloud.
- Se timbrará un único recibo con todas sus aplicaciones como un complemento de pago.
- Solo se tomarán en cuenta las aplicaciones de recibo a facturas que tienen como forma de pago “99 por definir”.
- Un complemento de pago puede tener una o más facturas relacionadas.
- Solo se timbrarán complementos de pago que están debidamente identificados, es decir, tienen un cliente asociado.
- Para el caso de los recibos no debe tener la bandera de anticipo activada, la cual es un flex field a nivel método de recibo. Esta bandera se activa o desactiva de forma manual.
- Se timbrarán los recibos que tengan activa la bandera de “listo para timbre”.
- El número de parcialidad se definirá con la siguiente regla:

  Se buscarán todas las aplicaciones de recibo a la factura en las tablas de recibos y el número de aplicaciones (pagos) a la misma factura encontradas, será el número de parcialidad a enviar. Ej. Se encontraron 3 aplicaciones (pagos) de una factura, por lo tanto, se envía como número de parcialidad 3.

- Los recibos que provienen de la súper interface y son complementos de pago deben subir al ERP Cloud con su respectivo UUID.
- Cuando el PAC devuelva un mensaje de error en el timbrado, el cual fue notificado por correo electrónico, Estrella Roja en caso de requerir reprocesar el recibo (complemento de pago) deberá actualizar el campo “Reprocesar” para que este recibo vuelva a ser tomado por la integración. Es importante mencionar que solo reprocesará los recibos que no han sido timbrados. Ejemplo: Datos faltantes de clientes.

6.1 Diseño del Servicio

Para llevar a cabo el inicio de la integración se tiene que ejecutar el compuesto de PaymentComplementEnt, el cual recibe 2 parámetros opcionales, los cuales se describen a continuación.

Argument	Prompt	Value Set	Default Value	Example
p_startDate	StartDate	dateTime	Fecha Actual a las 00:00:00 horas	2017-01-19 00:00:00
p_endDate	EndDate	dateTime	Fecha de Ejecución	2017-01-19 14:00:00

Los tiempos de ejecución se almacenan en la tabla XXER_INTEGRATIONS_TRANSACTIONS de la base de datos custom lo que hace que, al ejecutar la integración, esta tome la última fecha de ejecución como fecha de inicio para obtener los datos del ERP Cloud

7 Consideraciones de Rendimiento

Este requerimiento ha sido probado con un conjunto de recibos pequeños, tener contemplado que el compuesto puede presentar demora con una cantidad de registros muy grande a procesar.

7.1 Estrategia de Reinicio

- Para llevar a cabo un reinicio de la aplicación no es necesario realizar movimiento en base de datos, la aplicación al ser reiniciada continuará con la ejecución que corresponde.
- Supervisar que al momento del reinicio no existan instancia del proceso en ejecución, de ser así, esperar a que estas terminen para asegurar la congruencia de datos.

7.2 Seguridad

- Se recomienda el monitoreo oportuno de la base de datos para asegurar el correcto espacio para su crecimiento.
- El proceso principal para iniciar el timbrado de complementos de pagos es PaymentComplementEnt el cual solo se encuentra expuesto en el ambiente interno de Estrella Roja.
- En caso de requerir exponerlo a un mayor nivel, se recomienda pasar por un servicio OSB para no poner en riesgo la infraestructura del dominio SOA.

7.3 Personalización

- En caso de requerir modificar el proyecto, tomar la versión más reciente del controlador de versiones con el que se cuente.
- La versión del IDE de desarrollo de JDeveloper con la que se implementó dicha solución es JDEVADF_12.2.1.2.0_GENERIC_161008.1648.S
- La integración de Complemento de Pagos cuenta con parámetros que pueden ser personalizables, los cuales se describen a continuación.

En la parte de PaymentComplementEnt, dentro del proyecto se tiene la carpeta \SOA\Dvms donde encontraremos un catálogo actualizable de valores predeterminados, los que podemos modificar son los siguientes:

Código	Valor
DOCUMENT_TYPE	P
DESCRIPTION	Pago
UNIT_KEY	ACT
SERIE	PQ
BRANCH_NAME	MATRIZ
ISSUER_TAX_REGIME	624

Tenemos otros catálogos del lado de PaymentComplementBiz, en la carpeta \SOA\Dvms donde podemos encontrar StampDefaultValues que tiene los siguientes valores a personalizables:

Código	Valor
DOCUMENT_TYPE	P
DESCRIPTION_ITM	Pago
PROD_SERV_KEY_ITM	84111506
UNIT_KEY_ITM	ACT
SERIE	PQ
BRANCH_NAME	MATRIZ
CURRENCY	XXX
ISSUER_TAX_REGIME	624
SERVICE	PAQUER

Y el catálogo Configuration que cuenta con los siguientes valores:

Código	Valor
ALREADY_STAMPED_RECEIPT	Ya tiene asignado un UUID; por lo cual, ya está timbrado.
NO_APPLICATIONS	No contiene ninguna aplicación.
MAX_RETRY_COUNT	5
URL_PDF	http://wsestrellaroja.testsolucionesdfacture.com/api/downloadfile?rfc-uuid-&file_type=pdf
URL_XML	http://wsestrellaroja.testsolucionesdfacture.com/api/downloadfile?rfc-uuid-&file_type=xml
ENTITY_NAME	AR_CASH_RECEIPTS_ALL
CATEGORY_NAME	CUSTOMER_RECEIPT
ALLOW_DUPLICATE	yes

7.4 Catálogo de Errores

Se cuenta con una lista de errores en el archivo CatalogosErroresEstrellaRoja.xlsx, de manera muy particular particular de esta aplicación, se cuenta con los siguiente:

Código	Descripción
SOA-00021	Fallo al agregar archivos adjuntos
SOA-00022	El recibo ya está timbrado
SOA-00023	El recibo no tiene aplicaciones

Sin embargo, también se utilizan códigos generales compartidos con otras integraciones, ver el archivo mencionado para mayor detalle.

8 Consideraciones de Instalación

Como Prerrequisitos se requiere un esquema de base de datos (DBCS) previamente configurado, con los data sources correspondientes, consultar el manual de configuración del ambiente.

En este esquema deben existir las siguientes tablas:

- XXER_PAYMENTS.
- XXER_PAYMENT_DETAILS
- XXER_INTEGRATION_TRANSACTIONS
- XXER_COMPOSITE_ERRORS

De lo contrario se requiere crearlas, el script de estas tablas se encuentra en el siguiente archivo, para ejecutarlo se debe contar con los permisos necesarios ya sea desde sqlplus o un entorno gráfico.

SriptTablasPayment Complement.sql

Instalación de la integración de complemento de pago tanto capa Ent como Biz, instalar como aplicación SOA:

- sca_PaymentComplementBiz.jar
- sca_PaymentComplementEnt.jar

9 URL de Acceso y seguridad

A continuación, se definen los servicios utilizados en la integración cuyas URL están definidas con comodines que deben adaptarse dependiendo del ambiente del que se trate. Las dos primeras son propias de las integraciones de complemento de Pagos, las demás se incorporan para brindar funcionalidades especificas descritas anteriormente en el punto 3. Lógica de implementación

Servicio	URL
PaymentComplementEnt	**http://<hostname>:<puerto>/soa-infra/services/default/PaymentComplementEnt/PaymentComplementEnt?WSDL**
PaymentComplementBiz	**http://<hostname>:<puerto>/soa-infra/services/default/PaymentComplementBiz/PaymentComplementBiz?WSDL**
ERPIntegrationBiz	http://<hostname>:<puerto>/soa-infra/services/default/ERPIntegrationBiz/ERPIntegrationBiz?WSDL
AuditControlTec	sb://<hostname>:<puerto>/AuditControlTecPs
SOAUtilitiesTec	http://<hostname>:<puerto>/soa-infra/services/default/SOAUtilitiesTec/SOAUtilitiesTec?WSDL
MessageTec	http://<hostname>:<puerto>/soa-infra/services/default/MessageTec/MessageTec?WSDL
DigitalStampTec	sb://<hostname>:<puerto>/DigitalStampTecPs

10 Temas abiertos y cerrados

10.1 Temas Abiertos

ID	Tema	Solución	Responsabilidad	Fecha Objetivo	Fecha impacto

10.2 Temas Cerrados

ID	Tema	Solución	Responsabilidad	Fecha Objetivo	Fecha impacto