|  |
| --- |
|  |

OUM

TA.070-8 DISEÑO TÉCNICO DE EXTENSIONES E

INTERFACES

Un dibujo de una cara feliz

Descripción generada automáticamente con confianza baja

ER03\_AR\_Timbrado de Complementos de Pagos

|  |  |
| --- | --- |
| Autor: | Forte Innovation |
| Fecha de creación: | Enero 15 2018 |
| Última actualización: | 28 de septiembre de 2021 |
| Código de referencia OUM: | DS.140 Design Specification |
| Versión:  **Aprobadores:** | 1.2 |
| Ivan Muñoz |  |

Ángel Flores



**Contenido**

1 Control de Documento 2

1.1 Bitácora de Cambios 2

1.2 Revisores 2

2 Resumen técnico 3

3 Diagrama de la integración 4

4 Lógica de implementación 5

4.1 Calendarización y Ejecución 5

4.2 PaymentComplementEnt 6

4.3 PaymentComplementBiz 12

4.4 DigitalStampGERTec 29

5 Diseño de Datos 35

5.1 Origen de Datos 35

5.2 Lógica de Validación 36

6 Diseño SQL 38

6.1 Sentencias SQL 38

7 Reglas de Negocio 39

7.1 Diseño del Servicio 40

8 Consideraciones de Rendimiento 41

8.1 Estrategia de Reinicio 41

8.2 Seguridad 41

8.3 Personalización 41

8.4 Catálogo de Errores 43

9 Consideraciones de Instalación 44

10 URL de Acceso y seguridad 45

11 Temas abiertos y cerrados 46

11.1 Temas Abiertos 46

11.2 Temas Cerrados 46

# Control de Documento

## Bitácora de Cambios

3

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Autor** | **Versión** | **Referencia del cambio** |
| 02-feb-2018 | Efrain Arellanes | 1.0 | No Previous Document |
| 23-feb-2021 | Pedro Sanchez | 1.1 | Flex para no Re timbrar |
| 10-09-2021 | Pedro Sanchez | 1.2 | Actualización PAC AMS. |
|  |  |  |  |

## Revisores

|  |  |
| --- | --- |
| **Nombre** | **Posición** |
|  |  |
|  |  |
|  |  |
|  |  |

# Resumen técnico

Esta especificación documenta el diseño para el timbrado de transacciones, cuyo caso de uso se describió en el documento **TA020-4\_ER03\_AR\_Timbrado de Complementos**.

El contenido del documento es el detalle para la representan la integración, de esta manera, su objetivo principal es identificar el flujo de la integración que está dividida en 3 capas.

* Enterprise (Ent)
* Business (Biz)
* Technical (Tec)

La capa **Ent** permite exponer aquellos servicios u operaciones de Oracle Service Oriented Architecture (SOA) mediante Oracle Service Bus (OSB), esto con la finalidad de tener una primera línea de seguridad sin exponer la arquitectura de los servidores internos de la empresa.

Por otro lado esta capa se encarga de administra las capas **Biz** utilizadas para esta integración, del mismo modo nos permite calendarizar en el servidor Enterprise Scheduler Service (EES) aquellas operaciones de SOA que se requieran ejecutar cada cierto tiempo.

La capa **Biz** tiene la finalidad de llevar acabo la lógica de negocio correspondiente a dicha integración, es decir, se encarga de orquestar aquellas capas técnicas y manejar aquellas reglas para el negocio.

La capa **Tec** nos permite ofrecer una línea de seguridad en cuestión a servicios de terceros mediante Oracle Service Bus (OSB); por otro lado esta capa nos permite conectar con aquellos orígenes de datos(Base de datos, Servicios Internos, EJB, etc.) permitiendo realizar diferentes operaciones para el guardado o manipulación de información.

Existen dos tipos de integraciones:

* **Inbound**: En este tipo de integraciones, las aplicaciones legadas llamarán a servicios web SOAP expuestos de lado de los servidores internos de la empresa, enviando la información hacia el ERP Cloud, es decir, son aquellas integraciones que nos permiten gestionar información referente a Clientes, Recibos, Transacciones, etc.
* **Outbound:** Estas integraciones nos permiten extraer información del ERP Cloud por medio de reportes a través de BI Publisher, posteriormente es enviada a diferentes sistemas legados según corresponda, algunas de las integraciones que aplican son: Timbrado de Facturas AR, validación de facturas AP, Validación de Complementos de Pago, etc.

Existen 4 fases importantes para este tipo de integraciones:

* 1. Consiste en la extracción de la información teniendo como fuente el ERP Cloud, donde se encuentran ciertas condiciones que permiten identificar qué información es apta para procesar.
  2. En esta fase consiste en el enriquecimiento del mensaje, es decir, implica realizar el complemento de información del lado de la capa de negocios en SOA para poder enviar al sistema externo (Capa Técnica), controlando cualquier tipo de error durante el flujo.
  3. En este punto se recupera la información devuelta por los servicios externos y se consolida la información de acuerdo a los resultados obtenidos.
  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.

# Diagrama de la integración



# 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.

## Calendarización y Ejecución

A continuación, se describe la programación de esta capa en el ESS, consta de 1 elemento, un job que se ejecuta cada 10 minutos:

|  |  |  |
| --- | --- | --- |
| Componente | Propiedad | Valor |
| Trabajo | Nombre | PaymentComplementEnt |
| Nombre mostrado | PaymentComplementEnt |
| Paquete | /oracle/apps/ess/custom |
| Descripción | Timbrado de complementos AR |
| Tipo de trabajo | SyncWebserviceJobType |
| Operación | StampPaymentComplement |
| Solicitud | <ns1:StampPaymentComplementRq xmlns:ns1="http://soa.estrellaroja.com.mx/PaymentComplementEnt"> <ns1:StartDate><childElement/></ns1:StartDate> <ns1:EndDate><childElement/></ns1:EndDate></ns1:StampPaymentComplementRq> |
| Planificación | Nombre | PaymentComplementEnt |
| Nombre mostrado | PaymentComplementEnt |
| Paquete | /oracle/apps/ess/custom |
| Descripción | Timbrado de complementos AR |
| Frecuencia | Cada 10 min |
|  | Zona horaria | UTC-06:00 Mexico City |

## PaymentComplementEnt

Diagrama

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

La implementación se basa en la aplicación compuesta mostrada en la imagen anterior, se hace uso de los componentes:

|  |  |
| --- | --- |
| Componente | Icono del componente |
| Servicio Web |  |
| Mediador |  |
| BPEL |  |

El módulo principal se denomina mediador, este componente es el encargado de enrutar las peticiones desde el servicio de entrada hacia los demás componentes que se encargan de diferente funcionalidad y viceversa, es decir, las respuestas de cada componente son enrutadas y así entregadas a la petición que se generó.

### Lista de Objetos

Los siguientes objetos fueron usados/creados para implementar la funcionalidad del servicio Web

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Tipo** | **Descripción** |
| PaymentComplementEnt | Web  Service | Servicio Expuesto |
| mediator PaymentComplementEnt | Mediator | Componente de SOA utilizado para integrar y enrutar los mensajes entre los servicios web |
| bpelStampPaymentComplement | Bpel | Componente de Orquestación |
| ws PaymentComplementBiz | Web  Service | Servicio externo que tiene la lógica de negocio del proceso |
| wsERPIntegrationBiz | Web    Service | Servicio externo que Reliza operaciones con el ERP importacion de templates y consulta de reportes |

### bpelStampPaymentComplement

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.

Diagrama, Aplicación

Descripción generada automáticamente

### Lista de objetos

La siguiente tabla contiene los objetos que fueron usados para la funcionalidad del bpel sendfileJournal.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Nombre** | **Tipo** | **Descripción** |
| 1 | assignWsPaymentComplementBizStampPaymentComplementRq | assign | Asignación de valores. |
| 2 | trBpelRqToWsERPIntegrationBizGenerateReportRq | transformation | Transformación de datos. |
| 3 | invokeWsERPIntegrationBizGenerateReport | invoke | Invocación de Servicio. |
| 4 | assignWsReportResult | assign | Asignación de valores. |
| 5 | trWsERPIntegrationBizGenerateReportRsToBpelRs | transformation | Transformación de datos. |
| 6 | trBpelRqToWsPaymentComplementBizStampPaymentComplementRq | transformation | Transformación de datos. |
| 7 | invokeWsPaymentComplementBizStampPaymentComplement | invoke | Invocación de Servicio. |
| 8 | trWsPaymentComplementBizStampPaymentComplementRsToBpelRs | transformation | Transformación de datos. |

### Pruebas

Las pruebas se ejecutarán en el ambiente de desarrollo tomando la dirección del siguiente WSDL:

[**http://ersoaprd-soa-0.snproduccin.vcner.oraclevcn.com:9073/soa-infra/services/PROD/PaymentComplementEnt/PaymentComplementEnt?WSDL**](http://ersoaprd-soa-0.snproduccin.vcner.oraclevcn.com:9073/soa-infra/services/PROD/PaymentComplementEnt/PaymentComplementEnt?WSDL)

A continuación, se muestran los parámetros de entrada y la respuesta que se presentan en esta operación:

**Request:** No se necesitan parámetros de entrada para la ejecución de esta operación ya que es calendarizada:

Texto

Descripción generada automáticamente

Response: El servicio no retorna nada.

Texto

Descripción generada automáticamente

## 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:

Diagrama

Descripción generada automáticamente

La implementación se basa en la aplicación compuesta mostrada en la imagen anterior, se hace uso de los componentes:

|  |  |
| --- | --- |
| Componente | Icono del componente |
| Servicio Web |  |
| Mediador |  |
| BPEL |  |
| SubProccess |  |

El módulo principal se denomina mediador, este componente es el encargado de enrutar las peticiones desde el servicio de entrada hacia los demás componentes que se encargan de diferente funcionalidad y viceversa, es decir, las respuestas de cada componente son enrutadas y así entregadas a la petición que se generó.

### Lista de Objetos

Los siguientes objetos fueron usados/creados para implementar la funcionalidad del servicio Web

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Tipo** | **Descripción** |
| PaymentComplementBiz | Web  Service | Servicio Expuesto |
| bpelStampPaymentComplement | Bpel | Componente de Orquestación |
| bpelHandleMergedStamping | Bpel | Componente de Orquestación |
| mediator PaymentComplementBiz | Mediator | Componente de SOA utilizado para integrar y enrutar los mensajes entre los servicios web |
| wsSOAUtilitiesTec | Web  Service | Servicio externo que Decodifica Archivo en este caso |
| wsCommonPaymentsTec | Web  Service | Servicio externo consulta a BD SOA |
| sbpelCheckAlreadyStamped | SubBpel | Componente de Orquestación sencillo |
| sbpelLogPayment | SubBpel | Componente de Orquestación sencillo |
| sbpelUpdateFlex | SubBpel | Componente de Orquestación sencillo |
| sbpelHandleMessage | SubBpel | Componente de Orquestación sencillo |
| sbpelAddAttachment | SubBpel | Componente de Orquestación sencillo |
| sbpelActivateCheckpoint | SubBpel | Componente de Orquestación sencillo |
| wsDigitalStampTec | Web  Service | Servicio externo |
| wsMessageTec | Web  Service | Servicio externo |
| wsIntegrationServiceTec | Web  Service | Servicio externo |
| wsAuditControlTec | Web  Service | Servicio externo |

### bpelStampPaymentCompement

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.

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

### Lista de objetos

La siguiente tabla contiene los objetos que fueron usados para la funcionalidad del bpel sendfileJournal.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Nombre** | **Tipo** | **Descripción** |
| 1 | assignWsJournalBizGetJournalRq | assign | Asignación de valores. |
| 2 | invokeWsJournalBizGetJournal | invoke | Invocación de Servicio. |
| 3 | IfValidate | if | Validación de datos. |
| 4 | assignVarInterfaceL1 | assign | Asignación de valores. |
| 5 | trWsJournalBizGetJournalRsToBpelRs | transformation | Transformación de datos. |
| 6 | invokeWsERPIntegrationBizImportToERPCloud | invoke | Invocación de Servicio. |
| 7 | assignVarInterfaceL2 | assign | Asignación de valores. |
| 8 | trWsJournalBizGetJournalRsToBpelRs | transformation | Transformación de datos. |
| 9 | invokeWsERPIntegrationBizImportToERPCloud | invoke | Invocación de Servicio. |

### bpelHandleMergeStamping

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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

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 o tiene error al realizar consulta en tabla.
2. Si hay error en la Consulta del query a la tabla se notifica y termina el proceso
3. Y esta timbrado el recibo, entonces no se procesa.
4. Si no ha sido timbrado entonces se evalúa si tiene aplicaciones si no tiene no se procesa
5. Se envía a timbrar el elemento que cumple con los requisitos primarios (no estar timbrado y tener aplicaciones)

Diagrama

Descripción generada automáticamente

1. 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.

Diagrama

Descripción generada automáticamente

1. 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.

Gráfico, Diagrama, Gráfico de cajas y bigotes

Descripción generada automáticamente

1. 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.

Diagrama, Gráfico de cajas y bigotes

Descripción generada automáticamente

1. 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.

Diagrama

Descripción generada automáticamente

1. 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.

Diagrama

Descripción generada automáticamente

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

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Son Seis 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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**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.

Diagrama

Descripción generada automáticamente

**SbpelCheckAlreadyStamped:**

1. Se asignan los valores para la variable que se enviará al servicio CommonPaymentTec
2. se ejecuta la operación GetPayment que consulta el Cash ReceipId.
3. Si se encuentra un registro
4. Si ya se timbro el Registro

5 se asigna la variable con el estatus de timbrado en true().

1. se ejecuta el subproceso que pobla el Flex de UUID
2. Si no a timbrado el Requistro.
3. se asigna la variable con el estatus de timbrado en false().

3. Si no se encuentra un registro

8 Evalua Si Error en la consulta

9 se asigna la variable con el v\_ErrorQuery en true().

1. Evalua Si Error en la consulta

10 se asigna la variable con el v\_ErrorQuery en true() y la variable con el estatus de timbrado en false().

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**SbpelHandleMessage:**

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

Interfaz de usuario gráfica, Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

**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.

Diagrama

Descripción generada automáticamente

**sbpelUpdateFlex:**

1. Este subproceso Realiza la Actualizacion de Flex en el complemento de Pago.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

### Lista de objetos

La siguiente tabla contiene los objetos que fueron usados para la funcionalidad del bpel sendfileJournal.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Nombre** | **Tipo** | **Descripción** |
| 1 | assignWsJournalBizGetJournalRq | assign | Asignación de valores. |
| 2 | invokeWsJournalBizGetJournal | invoke | Invocación de Servicio. |
| 3 | IfValidate | if | Validación de datos. |
| 4 | assignVarInterfaceL1 | assign | Asignación de valores. |
| 5 | trWsJournalBizGetJournalRsToBpelRs | transformation | Transformación de datos. |
| 6 | invokeWsERPIntegrationBizImportToERPCloud | invoke | Invocación de Servicio. |
| 7 | assignVarInterfaceL2 | assign | Asignación de valores. |
| 8 | trWsJournalBizGetJournalRsToBpelRs | transformation | Transformación de datos. |
| 9 | invokeWsERPIntegrationBizImportToERPCloud | invoke | Invocación de Servicio. |

### Pruebas

Las pruebas se ejecutarán en el ambiente de desarrollo tomando la dirección del siguiente WSDL:

<http://ersoaprd-soa-0.snproduccin.vcner.oraclevcn.com:9073/soa-infra/services/default/PaymentComplementEnt/PaymentComplementbIz?WSDL>

A continuación, se muestran los parámetros de entrada y la respuesta que se presentan en esta operación:

**Request:** contrato del arreglo de complementos a timbrar:

Texto

Descripción generada automáticamente con confianza baja

Response: no retorna nada en caso de proceso correcto .

Texto

Descripción generada automáticamente

## DigitalStampGERTec

Este servicio web utiliza una lógica de proceso de varios componentes de OSB que se encargan de enrutar las peticiones hacía los adaptadores de base. A continuación se muestra la lógica de implementación:

1. Petición del cliente dependiendo de la operación que desee ejecutar.
2. Invocación de pipeline de acuerdo con la petición.
3. Diagrama

   Descripción generada automáticamenteDespués de haber ejecutado una de las operaciones, el componente pipeline responde la petición.(se utilizan los adaptadores enmarcados)

La implementación OBS mostrada en la imagen anterior, se hace uso de los componentes:

|  |  |
| --- | --- |
| **Componente** | **Icono del componente** |
| Adaptador Rest | Interfaz de usuario gráfica  Descripción generada automáticamente |
| Tubería |  |

El módulo principal se denomina pipeline , este componente es el encargado de enrutar las peticiones desde el servicio de entrada hacia los demás componentes que se encargan de diferente funcionalidad y viceversa, es decir, las respuestas de cada componente son enrutadas y así entregadas a la petición que se generó. A continuación, se muestra la interface del asistente que se utiliza en tiempo de implementación:

### Lista de objetos

Los siguientes objetos fueron usados/creados para implementar la funcionalidad del servicio Web.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Nombre** | **Tipo** | **Descripción** | **GIT** |
| 1 | DigitalStampGERTecPS | Web  Service | Servicio Expuesto |  |
| 2 | DigitalStampGERApi | Adaptador | Adaptador de REST |  |
| 3 | DigitalStampGERTecPL | Tubería | Flujo de tubería |  |

### createTrxCFDI

La operación **createTrxCFDI** realiza una serie de pasos basados dentro de un pipeline, a continuación, se enlistan los pasos a seguir para consumir dicha operación:

1. Se asigna la entrada del servicio a una variable “createTrxCFDIRq”.
2. Se toma el tipo de documento que se va a procesar de document\_type\_description y se coloca en v\_operation.
3. Se procede a mapeo de datos con la transformación DigitalStampRq\_to\_DigitalStampAPIRq.xsl para generar la petición del servicio Rest a consumir y se guarda en la variable varXMLJsonSchema.
4. Ejecutamos un traslate que convierte nuestro Shema XML a Json .
5. Se asigna nuestro json generado al cuerpo de la petición a enviar.
6. Se rutea al servicio saliente correspondiente a esta operación.
7. Dependiendo de la v\_operation ejecutamos la API correspondiente y enviamos la petición generada al Adaptador DigitalStampGERApi
8. Si la respuesta es correcta convertimos la respuesta de Json a XML con un traslate.
9. Después ejecutamos el mapeo de datos de DigitalStampGERApi a DigitalStampGERTecPS con la transformación DigitalStampAPIRs\_to\_DigitalStampRs.xsl.
10. Una ves Realizado esto remplazamos el resultado en el cuerpo de respuesta.

Nota: En caso de existir algún error al momento de invocar al servicio externo, este será devuelto como la salida del servicio.

### Lista de objetos

La siguiente tabla contiene los objetos que fueron usados para la funcionalidad del DigitalStampGERTecPL

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Nombre** | **Tipo** | **Descripción** |
| 1 | BranchNode | BranchNode | Menú de operaciones del servicio |
| 2 | Pipeline Pair Node | Pipeline Pair Node | Flujo de petición y response del servicio |
| 3 | Assing | assign | Assingna un valor a una variable |
| 4 | nXSD Translate | nXSD Translate | Transforma datos de un esquema nativo a otros o viceversa |
| 5 | Error Handler | Error Handler | Cacha los errores ocurridos. |
| 6 | If Then | If Then | operación que condición si . |
| 7 | Java Callout | Java Callout | Ejecuta clase java. |
| 8 | Replace | Replace | Remplaza una variable por otra |
| 9 | Condicional Branch | Condicional Branch | Condiciona que servicios ejecutar. |
| 10 | insert | insertt | Inserta valores a la petición o salida. |

### Pruebas

Las pruebas se ejecutarán en el ambiente de desarrollo tomando la dirección de Direct Biding:

sb://ersoaprd-soa-0.snproduccin.vcner.oraclevcn.com:9073/DigitalStampGERTecPS

A continuación, se muestran los parámetros de entrada y la respuesta que se presentan en esta operación:

**Request:** Valores de entrada no requeridos

Texto

Descripción generada automáticamente

Response: El servicio retorna los datos que están contenidos en la base de datos de acuerdo a los parámetros seleccionados.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# Diseño 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.

## Origen de Datos

Hace referencia a las tablas custom empleadas en esta integración, dichas tablas son solo de control.

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 |

## 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

* No debe tener UUID.
* No debe tener Folio.
* Debe tener un estatus de Reprocesar, Pago con Anticipo o bien no debe tener un valor.
* El tipo de documento debe ser Factura o Nota de Crédito.

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

# Diseño 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.

## Sentencias SQL

### Query para obtener los recibos a timbrar.



### Query para obtener las aplicaciones del recibo.



# Reglas de Negocio

Las reglas de negocio se describen a continuación, para saber más detalle puedes observar el documento TA0204\_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.

## 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.

# 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.

## 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.

## Seguridad

* Se recomienda el monitoreo oportuno de la base de datos para asegurar el correcto espacio para su crecimiento.
* El proceso principal para las pólizas es JournalEnt 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.

## 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:

Consideraciones de Rendimiento

|  |  |  |
| --- | --- | --- |
| 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?rfcuuid-&file\_type=pdf |
| URL\_XML | http://wsestrellaroja.testsolucionesdfacture.com/api/downloadfile?rfcuuid-&file\_type=xml |
| ENTITY\_NAME | AR\_CASH\_RECEIPTS\_ALL |
| CATEGORY\_NAME | CUSTOMER\_RECEIPT |
| ALLOW\_DUPLICATE | yes |

## 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.

# 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.



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

* sca\_PaymentComplementBiz.jar
* sca\_PaymentComplementEnt.jar

 

# 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 |
| DigitalStampGERTec | sb://<hostname>:<puerto>/DigitalStampGERTecPs |

# Temas abiertos y cerrados

## Temas Abiertos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Tema** | **Solución** | **Responsabilidad** | **Fecha Objetivo** | **Fecha impacto** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Temas Cerrados

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Tema** | **Solución** | **Responsabilidad** | **Fecha Objetivo** | **Fecha impacto** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |