|  |
| --- |
|  |

OUM

TA.070-8 DISEÑO TÉCNICO DE EXTENSIONES E

INTERFACES

Un dibujo de una cara feliz

Descripción generada automáticamente con confianza baja

ER27\_AR\_Carta\_Porte

|  |  |
| --- | --- |
| Autor: | Forte Innovation |
| Fecha de creación: | 15 de marzo de 2022 |
| Última actualización: | 15 de marzo de 2022 |
| Código de referencia OUM: | DS.140 Design Specification |
| Versión:  **Aprobadores:** | 1.1 |
| Juan Gonzalez |  |

Gregorio 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 Ejecución 5

4.2 LetterPorteBiz 7

4.3 LetterPorteTec 21

5 Diseño de Datos 27

5.1 Origen de Datos 27

5.2 Lógica de Validación 28

6 Diseño SQL 29

6.1 Sentencias SQL 29

7 Reglas de Negocio 30

7.1 Diseño del Servicio 30

8 Consideraciones de Rendimiento 31

8.1 Estrategia de Reinicio 31

8.2 Seguridad 31

8.3 Personalización 31

8.4 Catálogo de Errores 32

9 Consideraciones de Instalación 33

10 URL de Acceso y seguridad 34

11 Temas abiertos y cerrados 35

11.1 Temas Abiertos 35

11.2 Temas Cerrados 35

# Control de Documento

## Bitácora de Cambios

3

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Autor** | **Versión** | **Referencia del cambio** |
| 15/03/2022 | Pedro Sanchez | 1.1 | Agregar proceso a facturas cre y estatus |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Revisores

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

# Resumen técnico

Esta especificación documenta el diseño para la generación de relaciones de los CFDIs y carga de URL de relaciones.

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.

## Ejecución

Servicio se ejecuta najo de manda por las integraciones de facturación o si es necesario esta equipado para ejecución manual:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente Una captura de pantalla de una computadora

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## LetterPorteBiz

En la capa LetterPorteBiz se realiza una serie de pasos concentrados en un bpel como 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 |  |

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** |
| LetterPorteBiz | Web  Service | Servicio Expuesto |
| mediatorLetterPorteBiz | Mediator | Componente de SOA utilizado para integrar y enrutar los mensajes entre los servicios web |
| BPELUploadURLTransactions | Bpel | Componente de Orquestación |
| BPELRelationCFDIs | Bpel | Componente de Orquestación |
| wsSOAUtilitiesTec | Web  Service | Servicio externo que tiene la lógica de negocio del proceso |
| wsMessageTec | Web    Service | Servicio externo que Reliza operaciones con el ERP importacion de templates y consulta de reportes |
| wsERPIntegrationServiceTec | Web  Service | Servicio externo que tiene la lógica de negocio del proceso |
| wsLetterPorteTec | Web  Service | Servicio externo que tiene la lógica de negocio del proceso |

### BPELUploadURLTransactions

El **BPELUploadURLTransactions** contiene los siguientes pasos:

1. Se inicializan valores de respuesta de servicio.
2. Si hay valores de entrada en las transacciones ejecutamos consulta.
3. Validamos si es un arreglo de datos a buscar o esta vacío.
4. En caso de ser arreglo mapeamos en dato.
5. En caso de no ser un arreglo hacemos una búsqueda general.
6. Ejecutamos el servicio que consulta el reporte wsERPIntegrationServiceTec- RunDataModel.
7. Decodificamos la respuesta de B64 a string.
8. Parseamos el resultado a un objeto XML para su manipulación .
9. Si esta habilitado el envio de errores.
10. Asignamos msg de error a variable WsSOAUtilitiesGetHtmlControlRq.
11. Invocamos servicio de generación de msg de notificación SOAUtilities-GetHtmlControl.
12. Asignamos msg de error a variable Rq de envio de notifiaciones WsMessage\_SendEmailRq.
13. Invocamos servicio de envio de correos MessageTec- SendEmail

Diagrama

Descripción generada automáticamente

1. Mapeamos información consultada en el reporte a variable de entrada .
2. Ciclo for que recorre todas las variable de entrada todas las transacciones.
3. Asignamos asignación contador de numero procesamientos
4. If evalua si la transacción existe el customer id.
5. Asignamos asignación parámetros para WsErpIntegrationTecRunDataModelRq request de reporte de customers.
6. Invocamos servicio ErpIntegrationTec-RunDataModel.
7. Decodificamos el B64 recibido en el reponse del servicio.
8. Asignación de variable String en formato XML a objeto XML para su manipulación.

Diagrama

Descripción generada automáticamente con confianza baja

1. Si no existe el customer mandamos msj de error de lo contrario procedemos a realizar las asignaciones de valores.
2. Asignación de error de que no existe el customer TRX.
3. Proceso que se ejecuta en paralelo.
4. Si no se tiene el customer TRXID se asigna con lo consultado.
5. Asignación de error variable customer TRX.
6. Si no se tiene el BusinessUnit se asigna con lo consultado.
7. Asignación de error variable BusinessUnit.
8. Si no se tiene el TransactionSource se asigna con lo consultado.
9. Asignación de error variable TransactionSource.
10. Asignación de petición de servicio WsERPIntegrationServiceTec-UploadAttachmentRq encargado de realizar la carga de la url en APOLO.

Diagrama

Descripción generada automáticamente

1. Invocamos servicio WsERPIntegrationServiceTec-UploadAttachment encargado de realizar la carga de url en los attachements de APOLO.
2. Si la ejecución es correcta asignamos msj de success de lo contrario mapeamos el error.
3. Asignamos msj de success.
4. Asignamos msj de error.
5. Si la notificación de success esta habilitada.
6. Invocamos servicio WsSOAUtilitiesGetHtmlControlRq que genera body del correo de notificación.
7. Asignamos petición de servicio que envia email de notificación WsMessage-SendEmailRq.
8. Invocamos servicio de notifiaciones de email WsMessageTec-SendEmail.

Diagrama

Descripción generada automáticamente

### Lista de objetos

La siguiente tabla contiene los objetos que fueron usados para la funcionalidad del **BPELUploadURLTransactions**.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Nombre** | **Tipo** | **Descripción** |
| 1 | assignProceso | assign | Asignación de valores. |
| 2 | Trs\_mapeo | transformation | Transformación de datos. |
| 3 | WsService | invoke | Invocación de Servicio. |
| 5 | If\_\* | if | Condicional de información. |
| 6 | Flow1 | flow | Flujo paralelo de información. |
| 7 | XQTrs\_to\_input | traslate | Invocación de Servicio. |
| 8 | JavaDecoded | Java Embedding | Java embebido |

### 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/LetterPorteBiz/LetterPorteBiz?WSDL**](http://ersoaprd-soa-0.snproduccin.vcner.oraclevcn.com:9073/soa-infra/services/PROD/LetterPorteBiz/LetterPorteBiz?WSDL%20)

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 se puede habilitar un msj email de notificación del proceso.

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

### BPELRelationCFDIs

El **BPELRelationCFDIs** contiene los siguientes pasos:

1. Asignación de variables iniciales.
2. Invocamos servicio que consulta la información del CSV que vamos a generar.
3. Su existen relaciones generamos el mapeo de lo contrario se genera un archivo vacio.
4. Transformación de mapeo a schema de salida CSV.
5. NXSD que genera string separado por comas.
6. Generamos petición para convertir String a B64 que vamos a retornar en la url.
7. Asignamos string vacio sin registros.
8. Cacht all de errores.
9. Si esta habilitado el envio de errores.
10. Asignamos msg de error a variable WsSOAUtilitiesGetHtmlControlRq.
11. Invocamos servicio de generación de msg de notificación SOAUtilities-GetHtmlControl.
12. Asignamos msg de error a variable Rq de envio de notifiaciones WsMessage\_SendEmailRq.
13. Invocamos servicio de envio de correos MessageTec- SendEmail.

Diagrama

Descripción generada automáticamente

1. Invocamos servicio WsSOAUtilities-Encode encargado de decodificar el string a B64.
2. Asignamos el b64 y el nombre del archivo a la salida del servicio.
3. Si la notificación de success esta habilitada.
4. Invocamos servicio WsSOAUtilitiesGetHtmlControlRq que genera body del correo de notificación.
5. Asignamos petición de servicio que envia email de notificación WsMessage-SendEmailRq.
6. Invocamos servicio de notifiaciones de email WsMessageTec-SendEmail.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

### Lista de objetos

La siguiente tabla contiene los objetos que fueron usados para la funcionalidad del **BPELRelationCFDIs**.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Nombre** | **Tipo** | **Descripción** |
| 1 | assignProceso | assign | Asignación de valores. |
| 2 | Trs\_mapeo | transformation | Transformación de datos. |
| 3 | WsService | invoke | Invocación de Servicio. |
| 5 | If\_\* | if | Condicional de información. |
| 6 | Flow1 | flow | Flujo paralelo de información. |
| 7 | XQTrs\_to\_input | traslate | Invocación de Servicio. |
| 8 | JavaDecoded | Java Embedding | Java embebido |
| 1 | assignProceso | assign | Asignación de valores. |
| 2 | Trs\_mapeo | 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/LetterPorteBiz/LetterPorteBiz?WSDL>

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

**Request:** recibe el numero de factura a buscar:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Response: retorna el b64 de la búsqueda y el nombre del archivo .

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## LetterPorteTec

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

Diagrama

Descripción generada automáticamente

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

|  |  |
| --- | --- |
| **Componente** | **Icono del componente** |
| Adaptador BD |  |
| Tubería |  |
| Servicio SOAP |  |
| Servicio DB |  |

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** |
| 1 | RelationshipCFDIsPS | Web  Service | Servicio Expuesto |
| 2 | LetterPorteBiz | Adaptador | Adaptador SOAP |
| 3 | RelationshipCFDIsPL | Tubería | Flujo de tubería |
| 4 | LetterPorteBTecPS | Web  Service | Servicio Expuesto |
| 5 | LetterPorteBTecPL | Tubería | Flujo de tubería |
| 6 | dbGetRelationsCFDIs | Adaptador | Base de datos |

### GetRelationsCFDIs

La operación **GetRelationsCFDIs** 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 “**GetRelationsCFDIs**”.
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 GetRelationsCFDIsRq\_to\_ GetRelationsCFDIsRq.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 GetRelationsCFDIs
8. Si la respuesta es correcta convertimos la respuesta de Json a XML con un traslate.
9. Después ejecutamos el mapeo de datos de GetRelationsCFDIs a GetRelationsCFDIsPS con la transformación GetRelationsCFDIsRs\_to\_GetRelationsCFDIsRs.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 LetterPorteTecPL

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **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 | insert | 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/LetterPorteTecPS

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

**Request:** Valores de entrada no requeridos

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

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

### RelationshipCFDIs

La operación **RelationshipCFDIs** 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 “RelationshipCFDIsRq”.
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 RelationshipCFDIsRq\_to\_RelationshipCFDIsRq.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 RelationshipCFDIs
8. Si la respuesta es correcta convertimos la respuesta de Json a XML con un traslate.
9. Después ejecutamos el mapeo de datos de RelationshipCFDIs a RelationshipCFDIsPS con la transformación RelationshipCFDIsRs\_to\_RelationshipCFDIsRs.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 RelationshipCFDIsPL

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **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 | insert | Inserta valores a la petición o salida. |

### Pruebas

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

<http://150.136.16.44/LetterPorteTec/RelationshipCFDIsPS>

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

**Request:** Valores de entrada no requeridos numero de documento por url

http://150.136.16.44/LetterPorteTec/RelationshipCFDIsPS?transaction=PQD-270453

**Response**: El servicio retorna la descarga de un archivo CSV con la información que retorna la consulta en tiempo real a la Base.

Interfaz de usuario gráfica, Aplicación

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 de Archivo CSV generado por la integración.

**Template CSV**

|  |  |  |
| --- | --- | --- |
| **Columna Tipo de dato** | | **Descripción** |
| Transaccion | VARCHAR2(150 CHAR) | Numero de documento |
| Numero Guia | VARCHAR2(3 CHAR) | Numero de guía en TPAK |
| Monto Total | NUMBER | Monto total del timbrado |
| UUID Ingreso | VARCHAR2(150 CHAR) | Uuid de factura de ingreso |
| Estatus Timbrado Ingreso | VARCHAR2(150 CHAR) | Estatus actual del timbrado |
| URL XML Ingreso | VARCHAR2(150 CHAR) | URL de descarga de XML de factura de ingreso |
| URL PDF Ingreso | VARCHAR2(150 CHAR) | URL de descarga de PDF de factura de ingreso |
| Fecha Ingreso | VARCHAR2(150 CHAR) | Fecha de de timbrado de factura de ingreso |
| UUID CP | VARCHAR2(150 CHAR) | Uuid de factura de carta porte |
| Estatus Timbrado CP | VARCHAR2(150 CHAR) | Estatus actual del timbrado carta porte |
| URL XML CP | VARCHAR2(150 CHAR) | URL de descarga de XML de factura de carta porte |
| URL PDF CP | VARCHAR2(150 CHAR) | URL de descarga de PDF de factura de carta porte |
| Fecha CP | VARCHAR2(150 CHAR) | Fecha de de timbrado de factura de carta porte |

## 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 los datos de forma correcta.

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

* Solo procesa Facturas que existan en APOLO.
* Valida que tenga información la consulta de lo contrario manda un archivo vacío.

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 información de la Transacción .



### Query de paquetes encargado de generar la consulta de información de CFDIs a TMS.



# Reglas de Negocio

Las reglas de negocio se describen a continuación

* Se requiere tener los UUID y documentos XML Y PDF de las facturas de ingreso ligadas al documento creado en APOLO.

* Se requiere tener los URL y documentos XML Y PDF de las facturas de carta porte ligadas al documento creado en APOLO.
* Los documentos carta porte se generan en distintos momentos una vez generada su factura de ingreso
* La consulta de información debe ser en tiempo real.

## Diseño del Servicio

Para llevar a cabo el inicio de la integración se tiene que ejecutar el compuesto de LetterPorteBiz el cual cuenta con dos operaciones:

GenerateURLRelationCFDIs:Servicio que recibe el numero de documento donde va a crear la url donde se generan el CSV de realiciones.

GenerateRelationCFDIs:Servicio que recibe un numero de documento y retorna CSV de relaciones en B64 con su nombre de docuemento.

# 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

* No hay riesgos ya que el servicio es a demanda por interfaz OSB.

## 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\_ 12.2.1.4.0.190828
* La integración cuenta con parámetros que pueden ser personalizables, los cuales se describen a continuación.

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

|  |  |  |
| --- | --- | --- |
| Component | Code | Value |
| UploadURLTransactions | PreFijoName | Relations\_CFDIs\_ |
| UploadURLTransactions | DuplicateAttachments | no |
| UploadURLTransactions | Path | /Custom/ER\_INTEGRATION/ER27\_AR\_Carta\_Porte/Data Models/XXER\_AR\_INVOICES\_TRX.xdm |
| RelationCFDIs | URL | http://150.136.16.44/LetterPorteTec/RelationshipCFDIsPS?transaction= |
| UploadURLTransactions | NotifiactionsSuccess | si |
| UploadURLTransactions | NotifiactionsError | si |
| RelationCFDIs | NotifiactionsSuccess | si |

## 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.

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

* sca\_LetterPorteBiz.jar
* LetterPorteTec.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.

|  |  |
| --- | --- |
| Servicio | URL |
| LetterPorteBiz | **http://<hostname>:<puerto>/soa-infra/services/PROD/LetterPorteBiz/LetterPorteBiz?WSDL** |
| LetterPorteTec | **sb://<hostname>:<puerto>/LetterPorteTecPS**  **http://<hostname>:<puerto>//LetterPorteTec/RelationshipCFDIsPS?transaction=<NumeroTransaccion>** |

# Temas abiertos y cerrados

## Temas Abiertos

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

## Temas Cerrados

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