

OUM

TA.070-6 DISEÑO TÉCNICO DE EXTENSIONES E INTERFACES



ER12_GL_Importación de Póliza

Autor:	Oracle Consulting
Fecha de creación:	enero 15, 2018
Última actualización:	enero 15, 2018
Código de referencia OUM:	DS.140 Design Specification
Versión:	1.0

Aprobadores:

Ivan Muñoz

Ángel Flores

1 Control de Documento

1.1 Bitácora de Cambios

Fecha	Autor	Versión	Referencia del cambio
02-feb-2018	Efrain Arellanes	1.0	No Previous Document

1.2 Revisores

Nombre	Posición

Contenido

1	Control de Documento	ii
1.1	Bitácora de Cambios	ii
1.2	Revisores.....	ii
2	Resumen técnico	1
3	Diagrama de la integración	3
4	Lógica de implementación	4
4.1	Calendarización	4
4.2	Implementación de servicio web JournalENT	5
4.3	Implementación de servicio web JournalBiz	9
4.4	Implementación de servicio web FinancialsTec	13
4.5	Diseño de Datos	17
5	Origen de Datos	19
6	Lógica de Validación	20
7	Diseño SQL	21
7.1	Sentencias SQL.....	21
8	Reglas de Negocio	26
8.1	Diseño del Servicio	26
9	Consideraciones de Rendimiento	27
9.1	Estrategia de Reinicio.....	27
9.2	Seguridad	27
9.3	Personalización	27
10	Catálogo de Errores	28
11	Consideraciones de Instalación	29
12	URL de Acceso y seguridad	30
13	Temas abiertos y cerrados	31
13.1	Temas Abiertos	31
13.2	Temas Cerrados.....	31

2 Resumen técnico

Este documento presenta el complemento para la integración que conforma la importación de pólizas contables hacia el ERP Cloud. Esta integración es referida por el documento funcional TA020-3_ER12_GL_Importación de Pólizas Contables_131217.docx.

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:

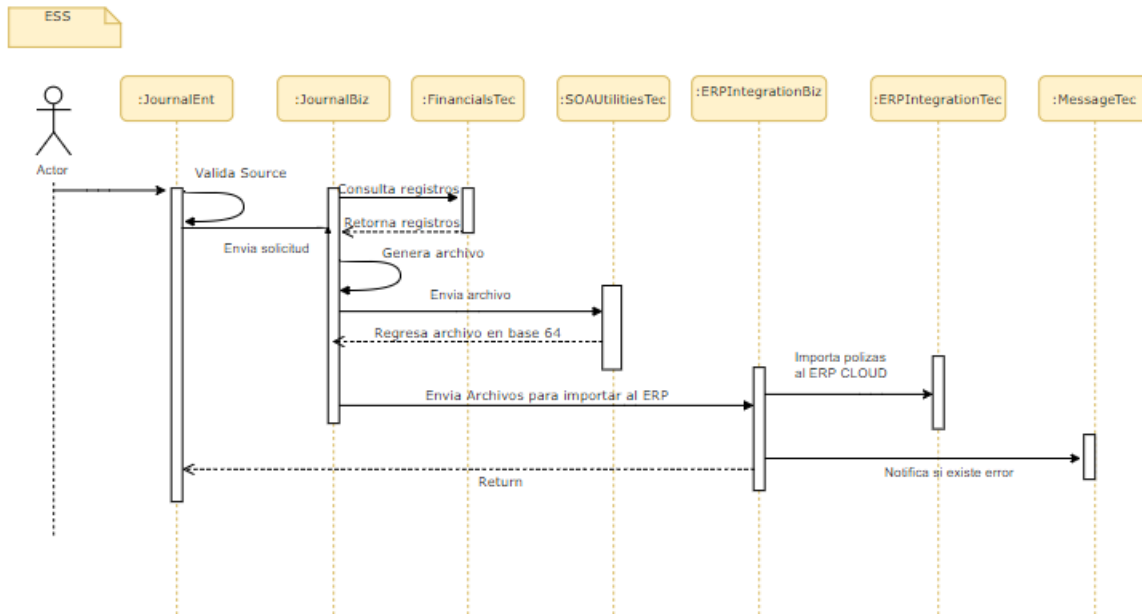
- **Inbound:** En este tipo de integraciones, las aplicaciones legadas llamarán a servicios web SOAP expuestos de lado de los servidores internos de la empresa, enviando la información hacia el ERP Cloud, es decir, son aquellas integraciones que nos permiten gestionar información referente a Clientes, 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.

3 Diagrama de la integración



4 Lógica de implementación

4.1 Calendarización

Componente	Propiedad	Valor
Trabajo	Nombre	SendJournalNEJobDef
	Nombre mostrado	SendJournalNEJobDefJob
	Paquete	/soa
	Descripción	Send journal 'Nómina de Empleados' to erp cloud
	Tipo de trabajo	SyncWebserviceJobType
	Operación	SendFileJournal
	Solicitud	<ns1:SendFileJournalRq xmlns:ns1="http://soa.estrellaroja.com.mx/JournalEnt"> <ns1:Source>Nómina de Empleados</ns1:Source> <ns1:Status>NEW</ns1:Status> </ns1:SendFileJournalRq>
Planificación	Nombre	SendJournalsNESch
	Nombre mostrado	SendJournalsNESch
	Paquete	/soa
	Descripción	SendJournalsNESch
	Frecuencia	0 Hours, 10 Minutes
	Zona horaria	UTC-06:00 Mexico City

Componente	Propiedad	Valor
Trabajo	Nombre	SendJournalTMSJobDef
	Nombre mostrado	SendJournalTMSJobDefJob
	Paquete	/soa
	Descripción	Send journal 'Viajado' to erp cloud
	Tipo de trabajo	SyncWebserviceJobType
	Operación	SendFileJournal
	Solicitud	<ns1:SendFileJournalRq xmlns:ns1="http://soa.estrellaroja.com.mx/JournalEnt"> <ns1:Source>TMS</ns1:Source> <ns1:Status>NEW</ns1:Status> </ns1:SendFileJournalRq>
Planificación	Nombre	SendJournalsTMSSch
	Nombre mostrado	SendJournalsTMSSch
	Paquete	/soa
	Descripción	SendJournalsTMSSch
	Frecuencia	0 Hours, 10 Minutes
	Zona horaria	UTC-06:00 Mexico City

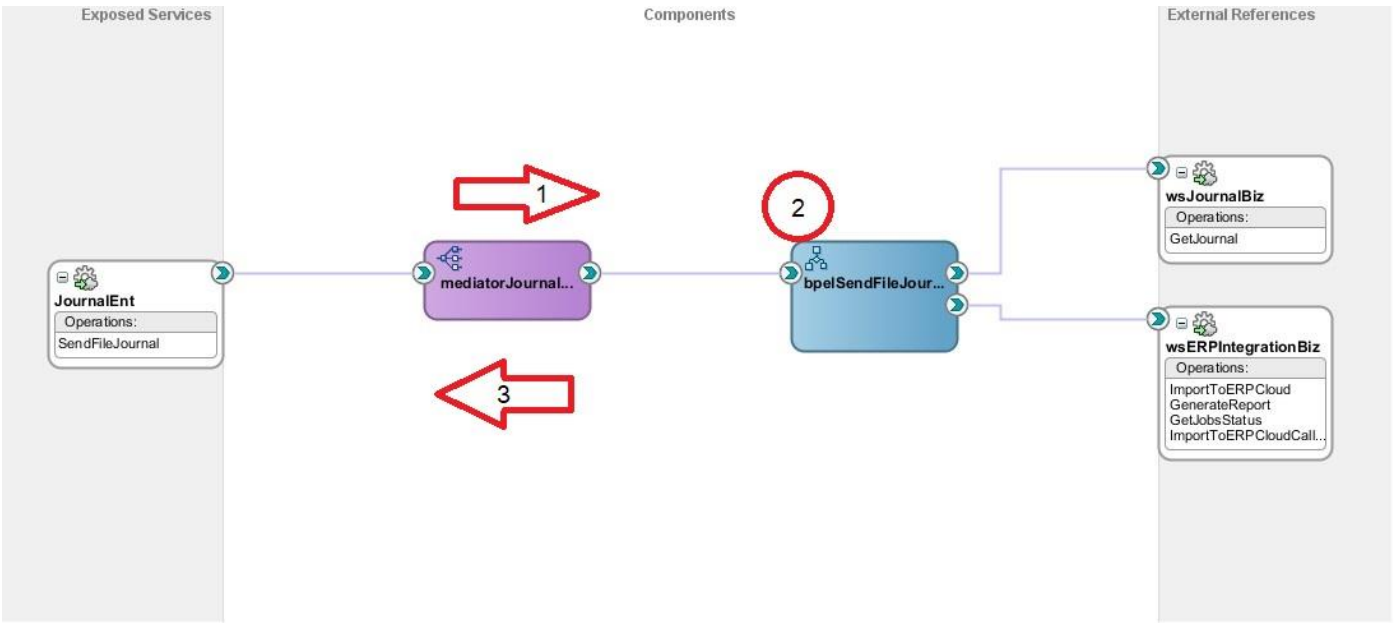
4.2 Implementación de servicio web JournalENT

El servicio web *JournalEnt* tiene las operaciones de:

- 1. SendFileJournal

Este servicio web utiliza una lógica de proceso de varios componentes de SOA que se encargan de enrutar las peticiones hacia los servicios de la capa de negocio. A continuación, se muestra su lógica de implementación:

- 1. Petición del cliente dependiendo de la operación SendFileJournal .
- 2. Invocación de BPEL bpelSendFileJournal.
- 3. Después de haber ejecutado una de las operaciones, el servicio responde la petición



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	Web Service
Mediador	Mediator (SOA.SOA)
BPEL	BPEL Process

El módulo principal se denomina mediador Mediator , 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.

4.2.1 Lista de Objetos

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

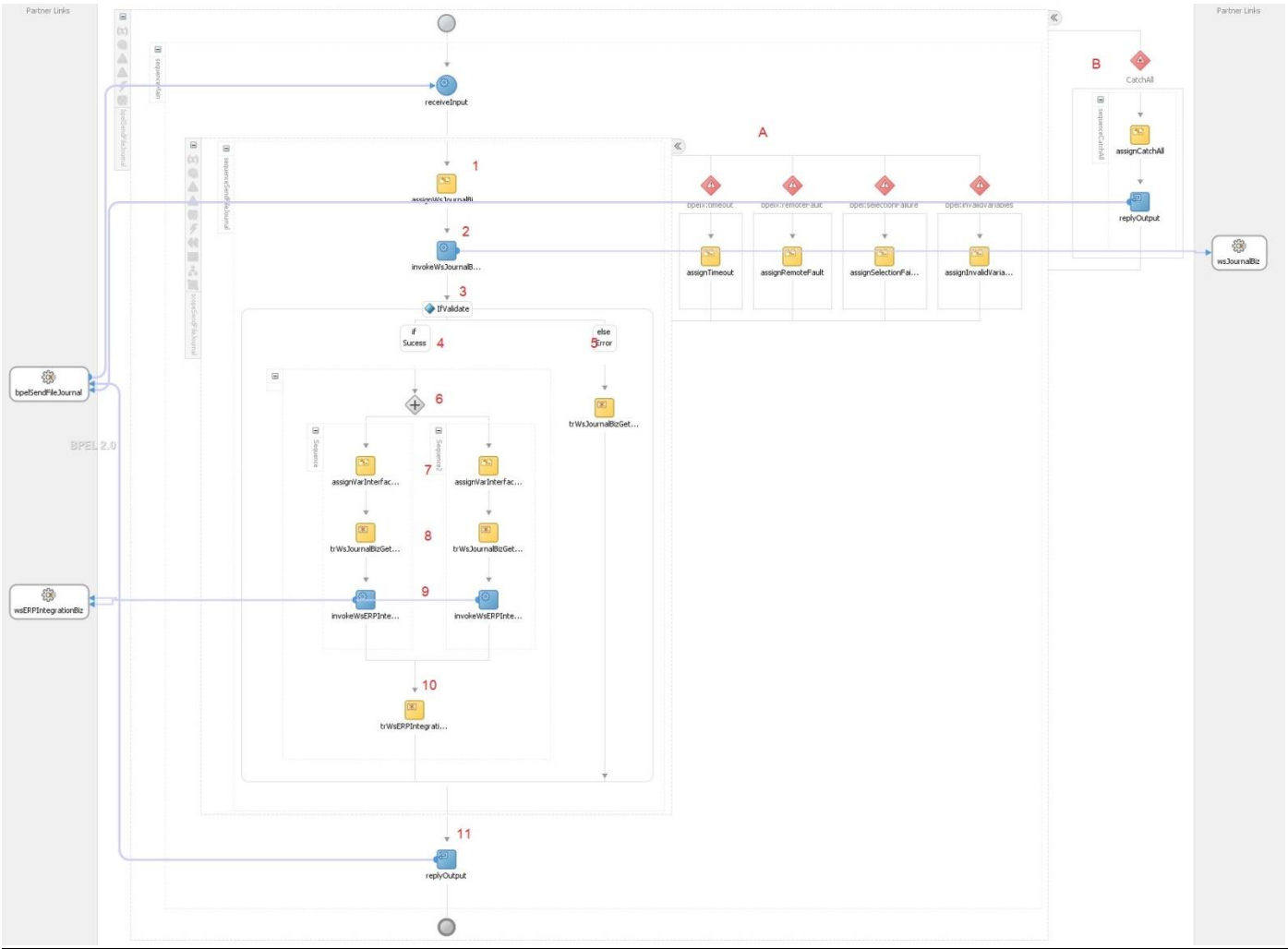
Elemento	Tipo	Descripción
JournalEnt	Web Service	Servicio Expuesto
mediatorJournalEnt	Mediator	Componente de SOA utilizado para integrar y enrutar los mensajes entre los servicios web
bpelSendFileJournal	Bpel	Componente de Orquestación
wsJournalBiz	Web Service	Servicio externo
wsERPIntegrationBiz	Web Service	Servicio externo

4.2.2 Send File Journal

La operación *SendFileJournal* realiza una serie de pasos basados dentro de un bpel.

El proceso BPEL realiza las siguientes acciones:

1. Se procede a asignar los valores de entrada del bpel al servicio JournalBiz-SendFileJournal.
2. Se invoca al servicio JournalBiz-SendFileJournal.
3. Se valida si existen datos recuperados del servicio.
4. Si existen datos, continua con el proceso de guardado de información.
5. Si no existen datos, se manda el error con datos no encontrados.
6. Se manda en paralelo para dos libros.
7. Se asigna la variable de la interface.
8. Se transforma para la entrada del servicio ERPIntegrationBiz-ImportToERPcloud.
9. Se invoca al servicio ERPIntegrationBiz-ImportToERPcloud.
10. Se da la respuesta del servicio
11. Salida y respuesta del bpel.
12. A y B Manejo de errores
13. Fin del Flujo.



4.2.3 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.

4.2.4 Pruebas

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

<http://ersoacsprod-wls-1:9073/soa-infra/services/default/JournalEnt/JournalEnt?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, van adjuntos en la misma, para Nomina de empleados o viajado:

[2018/02/15 11:18:56]

Received "SendFileJournal" call from partner "bpelSendFileJournal"

```
- <inputVariable>
- <part name="SendFileJournalRq" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <SendFileJournalRq xmlns:ns1="http://soa.estrellaroja.com.mx/JournalEnt"
  xmlns="http://soa.estrellaroja.com.mx/JournalEnt">
  <ns1:Source>TMS</ns1:Source>
  <ns1:Status>NEW</ns1:Status>
</SendFileJournalRq>
</part>
</inputVariable>
```

Response: El servicio retorna un id, con el cual se puede consultar la operación JobStatus para checar el estatus de los jobs del flujo.

Reply to partner "bpelSendFileJournal".

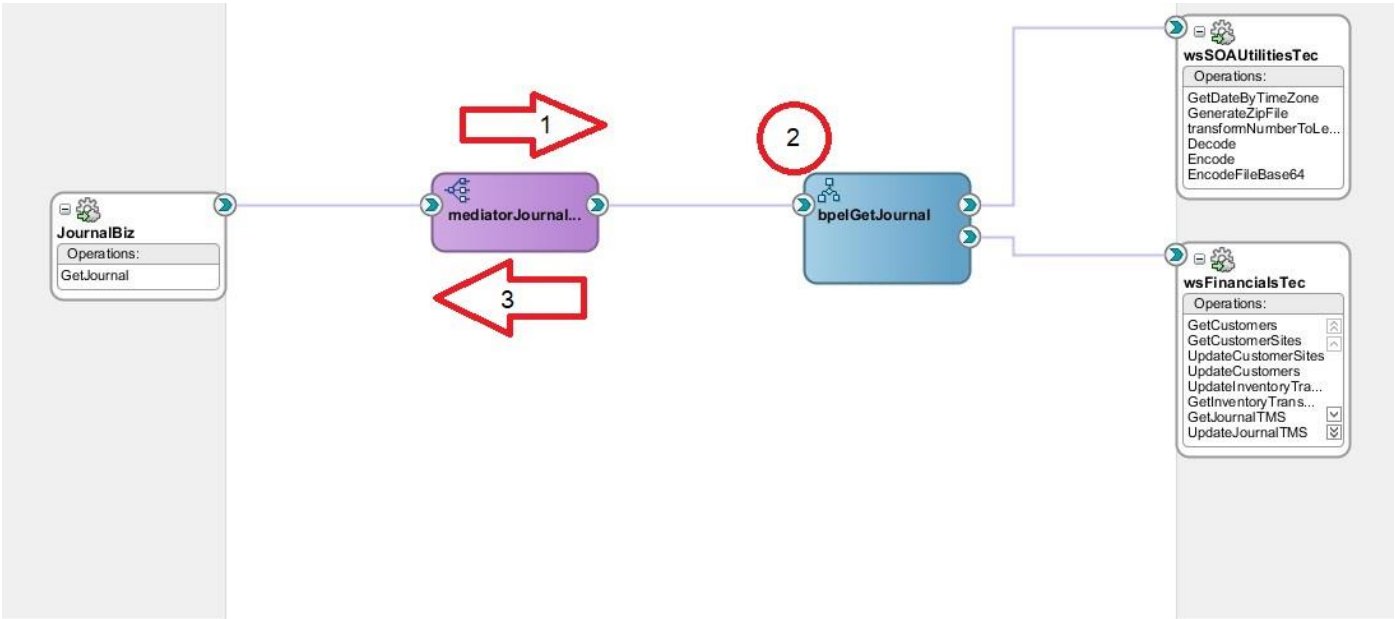
```
- <outputVariable>
- <part name="SendFileJournalRs" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <SendFileJournalRs xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:r
  /EstrellaRojaCommons" xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktyp
  xmlns:client="http://soa.estrellaroja.com.mx/JournalEnt/bpelSendFileJournal" xmlns:
  <ns1:Success>Success</ns1:Success>
- <tns:Return>
- <tns:Process>
  <tns:FileLedgerName>LIBRO ER</tns:FileLedgerName>
  <tns:Id>1388944</tns:Id>
  <tns:ProcessName>ImportToErpCloud</tns:ProcessName>
  <tns:Status>SUCCESS</tns:Status>
</tns:Process>
</tns:Return>
- <tns:Return>
- <tns:Process>
  <tns:FileLedgerName>LIBRO SECUNDARIO ER</tns:FileLedgerName>
  <tns:Id>1389010</tns:Id>
  <tns:ProcessName>ImportToErpCloud</tns:ProcessName>
  <tns:Status>SUCCESS</tns:Status>
</tns:Process>
</tns:Return>
</SendFileJournalRs>
</part>
</outputVariable>
```

4.3 Implementación de servicio web JournalBiz




El servicio web *JournalBiz* tiene las operaciones de:


1. GetJournal

- Este servicio web utiliza una lógica de proceso de varios componentes de SOA que se encargan de enrutar las peticiones hacía los servicios de la capa de negocio. A continuación, se muestra su lógica de implementación:
- 4. Petición del cliente dependiendo de la operación que desee ejecutar.
 - 5. Invocación de BPEL bpelGetJournal.
 - 6. Después de haber ejecutado una de las operaciones, el servicio responde la petición



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	 Web Service
Mediador	 Mediator (SOA.SOA)
BPEL	 BPEL Process

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.

4.3.1 Lista de Objetos

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

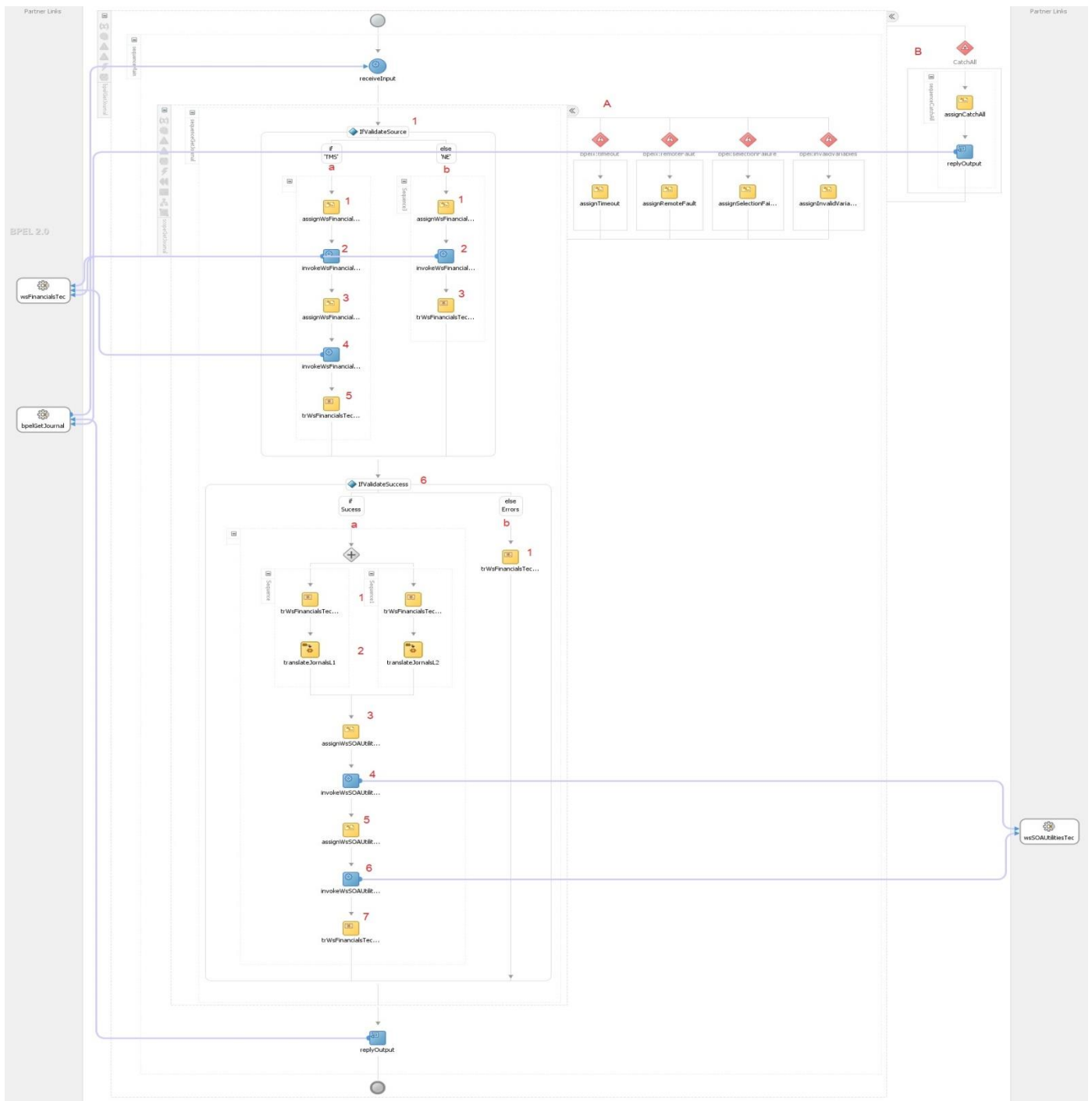
Elemento	Tipo	Descripción
JournalBiz	Web Service	Servicio Expuesto
bpelGetJournal	Bpel	Componente de Orquestación
mediatorJournalBiz	Mediator	Componente de SOA utilizado para integrar y enrutar los mensajes entre los servicios web
wsSOAUtilitiesTec	Web Service	Servicio externo
wsFinancialsTec	Web Service	Servicio externo

4.3.2 Send File Journal

La operación *GetJournal* realiza una serie de pasos basados dentro de un bpel.

El proceso BPEL realiza las siguientes acciones:

1. Valida source.
 - a) Viajado
 1. Se procede a asignar los valores de entrada del bpel al servicio FinancialsTec-UpdateJournalTMS para actualizar bach_id
 2. Se invoca al servicio FinancialsTec-UpdateJournalTMS
 3. Se procede a asignar los valores de entrada del bpel al servicio FinancialsTec-GetJournalTMS.
 4. Se invoca al servicio FinancialsTec-GetJournalTMS
 5. Se asigna la respuesta de salida.
 - b) Nómina
 1. Se procede a asignar los valores al servicio FinancialsTec-GetJournalNE
 2. Se invoca al servicio FinancialsTec-GetJournalNE.
 3. Se asigna la respuesta de salida.
6. Se valida éxito o error.
 - a) Success
 1. Se procede a asignar los valores a las variables para entrada del servicio.
 2. Se hace un translate para armar el archivo en una cadena.
 3. Se procede a asignar los valores al servicio SOAUtilitiesTec-Encode
 4. Se invoca al servicio SOAUtilitiesTec-Encode
 5. Se procede a asignar los valores al servicio SOAUtilitiesTec-EncodeFileBase64.
 6. Se asigna la respuesta de salida.
 - b) Error
 1. Se asigna salida con errores
7. A y B Manejo de errores
8. Fin del Flujo.



4.3.3 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.

4.3.4 Pruebas

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

<http://ersoacsprod-wls-1:9073/soa-infra/services/default/JournalBiz/JournalBiz?WSDL>

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

Request: Status y source:

```
[2018/02/15 11:24:57]
Received "GetJournal" call from partner "bpelGetJournal"
- <inputVariable>
  - <part name="GetJournalRq" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    - <GetJournalRq xmlns="http://soa.estrellaroja.com.mx/JournalBiz"
      xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <Status>NEW</Status>
      <Source>TMS</Source>
    </GetJournalRq>
  </part>
</inputVariable>
```

Response: El servicio retorna cadenas de base 64 con el contenido de la consulta de información de pólizas.

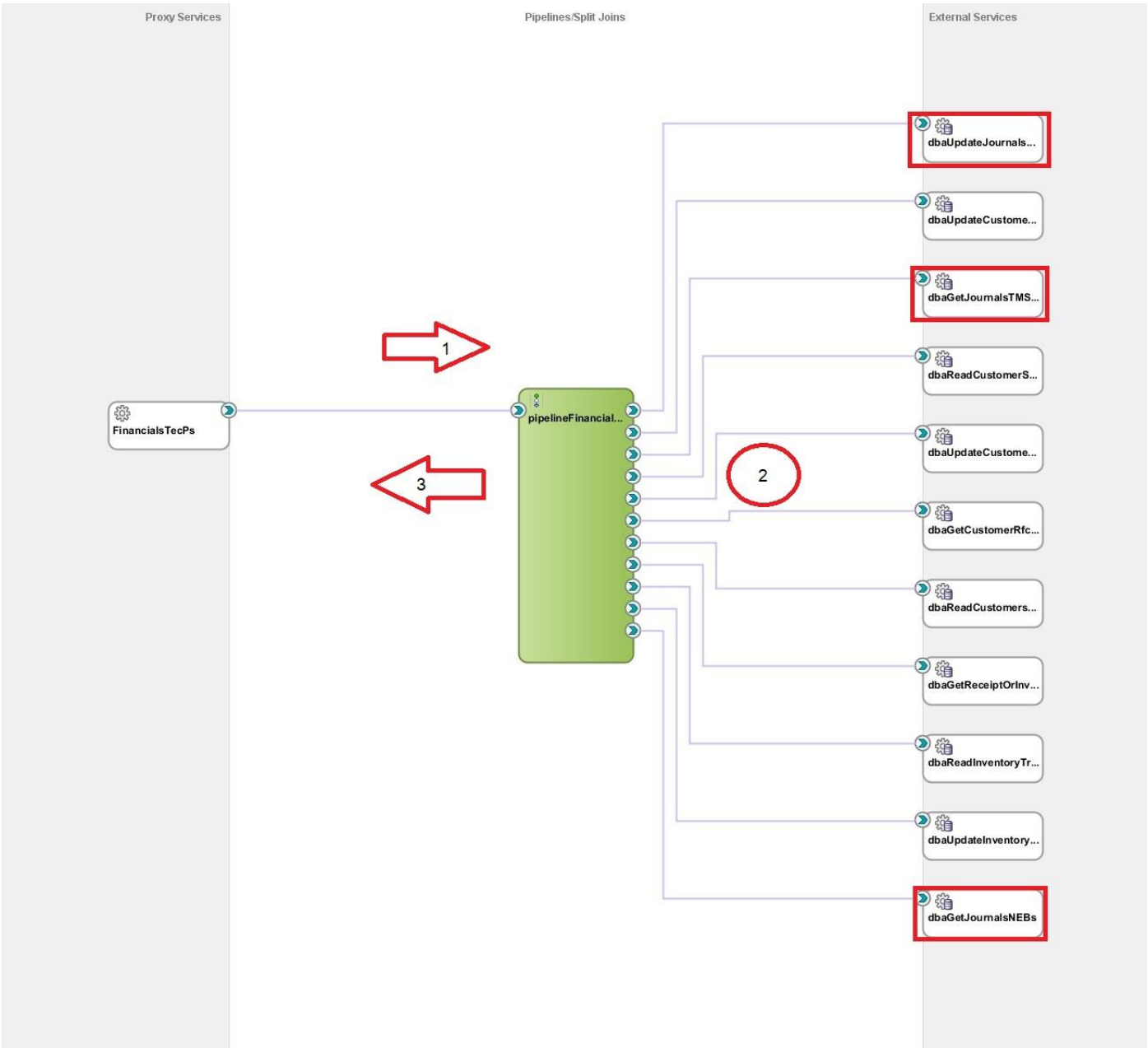
:

```
Reply to partner "bpelGetJournal".
- <outputVariable>
- <part name="GetJournalRs" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <GetJournalRs xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:ns2="http://soa.estrellaroja.com.mx/SOAUtilitiesTec/types" xmlns:ns1="http://soa.estrellaroja.com.mx/EstrellarojaCommons" xmlns:plink="http://docs.oasis-open.org/wsbpel/2.0/plinktype" xmlns:ns4="http://TargetNamespace.com/ServiceName" xmlns:ns3="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:client="http://soa.estrellaroja.com.mx/JournalBiz/bpelGetJournal" xmlns:tns="http://soa.estrellaroja.com.mx/JournalBiz">
- <ns1:Success/>
- <ns1:Errors>
- <ns1:Error>
- <ns1:ErrorCode/>
- <ns1:ShortDescription/>
- <ns1:Description/>
- <ns1:BusinessProcess/>
- <ns1:FailedService/>
- <ns1:TimeStamp/>
- </ns1:Error>
- </ns1:Errors>
- <tns:Return>
- <tns:Result>
- <tns:FileName>LEDGER1.csv</tns:FileName>
- <tns:FileContent>
- <tns:FileContent>
- </tns:FileContent>
- </tns:Result>
- <tns:Result>
- <tns:FileName>LEDGER2.csv</tns:FileName>
- <tns:FileContent>
- <tns:FileContent>
- </tns:FileContent>
- </tns:Result>
- </tns:Return>
- </GetJournalRs>
- </part>
- </outputVariable>
```



4.4 Implementación de servicio web FinancialsTec


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)



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

Componente	Icono del componente
Adaptador de base	 Database
Tubería	 Pipeline

El módulo principal se denomina pipeline  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:

4.4.5 Lista de objetos

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

#	Nombre	Tipo	Descripción	GIT
1	FinancialsTecPs	Web Service	Servicio Expuesto	
2	dbaGetJournalsNEBs	Adaptador	Adaptador de base	
3	dbaGetJournalsTMSBs	Adaptador	Adaptador de base	
4	dbaUpdateJournalsNEBs	Adaptador	Adaptador de base	
5	pipelineFinancialsTec	Tubería	Flujo de tubería	

4.4.6 GetJournalTMS

La operación **GetJournalTMS** 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 “varGetJournalTMSRq”.
2. Se procede a enrutar hacia el servicio de negocios que deseamos consumir “dbaGetJournalsTMSBs”.
3. Se asignan los valores de la variable “varGetJournalTMSRq” al servicio de negocios “dbaGetJournalsTMSBs” y se consume dicho servicio.
4. Se recupera el resultado del servicio consumido, transformando y alojando estos dentro de una variable “GetJournalTMSRs”.
5. Se rutea al servicio saliente correspondiente a esta operación.
6. Se asignan los valores de la variable “varGetJournalTMSRs” hacia la salida del servicio.

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

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

7. Se asigna la entrada del servicio a una variable “varUpdateJournalTMSRq”.
8. Se procede a enrutar hacia el servicio de negocios que deseamos consumir “dbaUpdateJournalsTMSBs”.

9. Se asignan los valores de la variable “varUpdateJournalTMSRq” al servicio de negocios “dbaUpdateJournalsTMSBs” y se consume dicho servicio.
10. Se recupera el resultado del servicio consumido, transformando y alojando estos dentro de una variable “UpdateJournalTMSRs”.
11. Se rutea al servicio saliente correspondiente a esta operación.
12. Se asignan los valores de la variable “varUpdateJournalTMSRs” hacia la salida del servicio.

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

4.4.7 GetJournalNE

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

13. Se asigna la entrada del servicio a una variable “varGetJournalNERq”.
14. Se procede a enrutar hacia el servicio de negocios que deseamos consumir “dbaGetJournalsNEBs”.
15. Se asignan los valores de la variable “varGetJournalNERq” al servicio de negocios “dbaGetJournalsNEBs” y se consume dicho servicio.
16. Se recupera el resultado del servicio consumido, transformando y alojando estos dentro de una variable “GetJournalNERS”.
17. Se rutea al servicio saliente correspondiente a esta operación.
18. Se asignan los valores de la variable “varGetJournalNERS” hacia la salida del servicio.

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

4.4.8 Pruebas

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

<http://ersoacsprod-wls-1:9073/soa-infra/services/default/JournalEnt/JournalEnt?WSDL>

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

Request: Se require el source, status y batch_id:

```

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <fin:GetJournalTMSRq xmlns:fin="http://soa.estrellaroja.com.mx/FinancialsTec">
      <fin:Status>NEW</fin:Status>
      <fin:Source>TMS</fin:Source>
      <fin:BatchId>201801251922</fin:BatchId>
    </fin:GetJournalTMSRq>
  </soap:Body>
</soap:Envelope>

```

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

```

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header/>
  <soap:Body xmlns="http://schemas.xmlsoap.org/soap/envelope/">
    <tns:GetJournalTMSRs xmlns:cmn="http://soa.estrellaroja.com.mx/EstrellaRojaCommons" xmlns:tns="http://soa.estrellaroja.com.mx/EstrellaRojaCommons">
      <cmn:Success/>
      <tns:Return>
        <tns:Journals>
          <tns:StatusCode>NEW</tns:StatusCode>
          <tns:LedgerID>2021</tns:LedgerID>
          <tns:EffectiveDateOfTransaction>2018-01-01</tns:EffectiveDateOfTransaction>
          <tns:JournalSource>TMS</tns:JournalSource>
          <tns:JournalCategory>Ingresos</tns:JournalCategory>
          <tns:CurrencyCode>MXN</tns:CurrencyCode>
          <tns:JournalEntryCreationDate>2018-01-23</tns:JournalEntryCreationDate>
          <tns:ActualFlag>A</tns:ActualFlag>
          <tns:Segment1>02</tns:Segment1>
          <tns:Segment2>00</tns:Segment2>
          <tns:Segment3>0000</tns:Segment3>
          <tns:Segment4>00</tns:Segment4>
          <tns:Segment5>2231</tns:Segment5>
          <tns:Segment6/>
          <tns:Segment7/>
          <tns:Segment8/>
          <tns:Segment9/>
          <tns:Segment10/>
          <tns:Segment11/>
          <tns:Segment12/>
          <tns:Segment13/>
        </tns:Journals>
      </tns:Return>
    </tns:GetJournalTMSRs>
  </soap:Body>
</soap:Envelope>

```

4.5 Diseño de Datos

A continuación, se listan los componentes SOA utilizados en las capas Ent y Biz.

4.5.1 Capa Empresarial

#	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.

4.5.2 Capa de Negocio

#	Nombre	Tipo	Descripción
1	IfValidateSource	if	Validación de datos.
2	IfValidateSuccess	if	Validación de datos.
3	assignWsFinancialsTecGetJournalNERq	assign	Asignación de valores.
4	invokeWsFinancialsTecGetJournalNE	invoke	Invocación de Servicio.
5	trWsFinancialsTecGetJournalNERSToVarGetJournals	transformation	Transformación de datos.
6	trWsFinancialsTecGetJournalRsToVarJournalsL1	transformation	Transformación de datos.
7	translateJournalsL1	transformation	Transformación de datos.
8	trWsFinancialsTecGetJournalRsToVarJournalsL2	transformation	Transformación de datos.
9	translateJournalsL2	transformation	Transformación de datos.

5 Origen de Datos

Los orígenes de datos se encuentran dentro del adaptador de base, por lo cual en este apartado se mencionará aquellos servicios web junto con las operaciones ocupadas para poder realizar dicha integración.

La siguiente tabla muestra los distintos orígenes de datos correspondientes a esta integración dentro de la capa de técnica.

Servicio	Operación	Descripción
FinancialsTec	GetJournalTMS	Extrae pólizas de viajado.
	GetJournalNE	Extrae pólizas de Nómina de Empleados.
	UpdateJournalTMS	Actualiza pólizas de viajado.

6 Lógica de Validación

Esta lógica hace referencia a las validaciones que se emplean en los orígenes para poder tener la información congruente y poder procesarla para finalmente extraer pólizas, la forma de realizar este proceso lo podemos encontrar en el documento de capas técnicas.

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

- Las pólizas deberán tener un batch_id null.

7 Diseño SQL

Este es mediante los adaptadores de base que se encuentran en el servicio FINANCIALSTEC el cual invoca procedimientos y/o funciones del paquete XXER_UTILITIES_PKG.

7.1 Sentencias SQL

Nómina de Empleados

Actualiza je_batch_id.

```
UPDATE GL_INTERFACE
  SET JE_BATCH_ID = P_BATCH_ID
  WHERE JE_BATCH_ID IS NULL
  AND STATUS=P_STATUS
  AND USER_JE_SOURCE_NAME=P_SOURCE
  ;
```

Consulta líneas de pólizas.

```
SELECT XXER_GL_INTERFACE_TYPE (STATUS,
  LEDGER_ID,
  ACCOUNTING_DATE,
  CURRENCY_CODE,
  DATE_CREATED,
  CREATED_BY,
  ACTUAL_FLAG,
  USER_JE_CATEGORY_NAME,
  USER_JE_SOURCE_NAME,
  CURRENCY_CONVERSION_DATE,
  ENCUMBRANCE_TYPE_ID,
  BUDGET_VERSION_ID,
  USER_CURRENCY_CONVERSION_TYPE,
  CURRENCY_CONVERSION_RATE,
  AVERAGE_JOURNAL_FLAG,
  ORIGINATING_BAL_SEG_VALUE,
  SEGMENT1,
  SEGMENT2,
  SEGMENT3,
  SEGMENT4,
  SEGMENT5,
  SEGMENT6,
  SEGMENT7,
  SEGMENT8,
  SEGMENT9,
  SEGMENT10,
  SEGMENT11,
  SEGMENT12,
  SEGMENT13,
  SEGMENT14,
  SEGMENT15,
  SEGMENT16,
  SEGMENT17,
  SEGMENT18,
```


SEGMENT19,
SEGMENT20,
SEGMENT21,
SEGMENT22,
SEGMENT23,
SEGMENT24,
SEGMENT25,
SEGMENT26,
SEGMENT27,
SEGMENT28,
SEGMENT29,
SEGMENT30,
ENTERED_DR,
ENTERED_CR,
ACCOUNTED_DR,
ACCOUNTED_CR,
TRANSACTION_DATE,
REFERENCE1,
REFERENCE2,
REFERENCE3,
REFERENCE4,
REFERENCE5,
REFERENCE6,
REFERENCE7,
REFERENCE8,
REFERENCE9,
REFERENCE10,
REFERENCE11,
REFERENCE12,
REFERENCE13,
REFERENCE14,
REFERENCE15,
REFERENCE16,
REFERENCE17,
REFERENCE18,
REFERENCE19,
REFERENCE20,
REFERENCE21,
REFERENCE22,
REFERENCE23,
REFERENCE24,
REFERENCE25,
REFERENCE26,
REFERENCE27,
REFERENCE28,
REFERENCE29,
REFERENCE30,
JE_BATCH_ID,
PERIOD_NAME,
JE_HEADER_ID,
JE_LINE_NUM,
CHART_OF_ACCOUNTS_ID,
FUNCTIONAL_CURRENCY_CODE,
CODE_COMBINATION_ID,
DATE_CREATED_IN_GL,
WARNING_CODE,
STATUS_DESCRIPTION,
STAT_AMOUNT,

```

GROUP_ID,
REQUEST_ID,
SUBLEDGER_DOC_SEQUENCE_ID,
SUBLEDGER_DOC_SEQUENCE_VALUE,
ATTRIBUTE1,
ATTRIBUTE2,
GL_SL_LINK_ID,
GL_SL_LINK_TABLE,
ATTRIBUTE3,
ATTRIBUTE4,
ATTRIBUTE5,
ATTRIBUTE6,
ATTRIBUTE7,
ATTRIBUTE8,
ATTRIBUTE9,
ATTRIBUTE10,
ATTRIBUTE11,
ATTRIBUTE12,
ATTRIBUTE13,
ATTRIBUTE14,
ATTRIBUTE15,
ATTRIBUTE16,
ATTRIBUTE17,
ATTRIBUTE18,
ATTRIBUTE19,
ATTRIBUTE20,
CONTEXT,
CONTEXT2,
INVOICE_DATE,
TAX_CODE,
INVOICE_IDENTIFIER,
INVOICE_AMOUNT,
CONTEXT3,
USSGL_TRANSACTION_CODE,
DESCR_FLEX_ERROR_MESSAGE,
JGZZ_RECON_REF,
REFERENCE_DATE,
SET_OF_BOOKS_ID,
BALANCING_SEGMENT_VALUE,
MANAGEMENT_SEGMENT_VALUE,
FUNDS_RESERVED_FLAG,
CODE_COMBINATION_ID_INTERIM)
BULK COLLECT INTO C_GL_INTERFACE FROM XXER_GL_INTERFACE_V
  WHERE STATUS=P_STATUS
  AND JE_BATCH_ID=P_BATCH_ID
  AND USER_JE_SOURCE_NAME=P_SOURCE;

```

Viajado

Actualiza je_batch_id.

```

UPDATE XXER_GL_INTERFACE
  SET JE_BATCH_ID = P_BATCH_ID
  WHERE JE_BATCH_ID IS NULL
  AND STATUS=P_STATUS
  AND USER_JE_SOURCE_NAME=P_SOURCE
;

```

Consulta líneas de pólizas.

```
SELECT
    reference4,
    Status,
    ledger_id,
    accounting_date,
    user_je_source_name,
    user_je_category_name,
    currency_code,
    date_created,
    actual_flag,
    segment1,
    segment2,
    segment3,
    segment4,
    segment5,
    segment6,
    segment7,
    segment8,
    segment9,
    segment10,
    segment11,
    segment12,
    segment13,
    segment14,
    segment15,
    segment16,
    segment17,
    segment18,
    segment19,
    segment20,
    segment21,
    segment22,
    segment23,
    segment24,
    segment25,
    segment26,
    segment27,
    segment28,
    segment29,
    segment30,
    Sum (entered_dr) entered_dr,
    Sum (entered_cr) entered_cr,
    Sum (accounted_dr) accounted_dr,
    Sum (accounted_cr) accounted_cr,
    user_currency_conversion_type,
    currency_conversion_date,
    currency_conversion_rate,
    average_journal_flag,
    group_id
FROM xxer_gl_interface_v
WHERE STATUS=P_STATUS
    AND JE_BATCH_ID=P_BATCH_ID
    AND USER_JE_SOURCE_NAME=P_SOURCE
GROUP BY
    reference4,
    status,
```

```
        ledger_id,  
        accounting_date,  
        user_je_source_name,  
        user_je_category_name,  
        currency_code,  
        date_created,  
        actual_flag,  
        segment1,  
        segment2,  
        segment3,  
        segment4,  
        segment5,  
        segment6,  
        segment7,  
        segment8,  
        segment9,  
        segment10,  
        segment11,  
        segment12,  
        segment13,  
        segment14,  
        segment15,  
        segment16,  
        segment17,  
        segment18,  
        segment19,  
        segment20,  
        segment21,  
        segment22,  
        segment23,  
        segment24,  
        segment25,  
        segment26,  
        segment27,  
        segment28,  
        segment29,  
        segment30,  
        user_currency_conversion_type,  
        currency_conversion_date,  
        currency_conversion_rate,  
        average_journal_flag,  
        group_id  
ORDER BY reference4;
```

8 Reglas de Negocio

8.1 Diseño del Servicio

A continuación, se describe la entrada y salida correspondiente del JournalEnt

Para la operación SendFileJournal, el mensaje de entrada se denomina SendFileJournalRq

Elemento	Sub-Elemento	Tipo	Requerido
SendFileJournalRq		tns:SendFileJournalRqType	Sí
SendFileJournalRs		tns:SendFileJournalRsType	Sí
	varInterfaceL1	string	Sí
	varInterfaceL2	string	Sí
SendFileJournalRqType			Sí
	Source	string	Sí
	Status	string	Sí
SendFileJournalRsType			Sí
Return		tns:SendFileJournalReturnTypes	No
SendFileJournalReturnTypes			Sí
Process		tns:ProcessType	1 o Más
ProcessType			Sí
	FileLedgerName	string	Sí
	Id	long	Sí
	ProcessName	string	Sí
	Status	string	Sí

A continuación, se describe la entrada y salida correspondiente del JournalBiz para la operación GetJournal, el mensaje de entrada se denomina GetJournalRq

Elemento	Sub-Elemento	Tipo	Requerido
GetJournalRq		tns:GetJournalRqType	Sí
GetJournalRs		tns:GetJournalRsType	Sí
GetJournalRqType			Sí
	Status	string	Sí
	Source	string	Sí
GetJournalRsType			Sí
Return		tns:GetJournalReturnTypes	No
GetJournalReturnTypes			Sí
Result		tns:JournalType	Opcional-Lista
JournalType			Sí
	FileName	string	Sí
	FileContent	base64Binary	Sí

9 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.

9.1 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.

9.2 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.

9.3 Personalización

En caso de querer 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.

10 Catálogo de Errores

La lista de errores de SOA que se podrían suscitar, se encuentran dentro del archivo "CatalogosErroresEstrellaRoja.xlsx".

11 Consideraciones de Instalación

A continuación, se listan los pasos a seguir para la instalación de esta integración.

1. Creación de objetos de base de datos, estos deben haberse creado antes de instalar los compuestos de la capa técnica.

Objetos de base para Nómina de empleados (Base en On -premise):



XXER_UTILITIES_PKG.pks



XXER_GL_EQV_TBL.sql



XXER_GL_INTERFACE_T.sql



XXER_GL_INTERFACE_TYPE.sql



XXER_GL_INTERFACE_V.sql



XXER_UTILITIES_PKG.pkb

Objetos de base para viajado (Base en Cloud):



XXER_UTILITIES_PKG.pks



XXER_GL_INTERFACE_T.sql



XXER_GL_INTERFACE_TYPE.sql



XXER_GL_INTERFACE_TYPE_T.sql



XXER_GL_INTERFACE_V.sql



XXER_UTILITIES_PKG.pkb

2. Los servicios de la capa técnica y de negocio deben de estar desplegados en SOA.
3. Instalar la aplicación compuesta de SOA JournalBiz (Este archivo se encuentra en los entregables, el cual tiene el mismo nombre con extensión Jar).
4. Instalación de la aplicación compuesta de SOA JournalEnt (Este archivo se encuentra en los entregables, el cual tiene el mismo nombre con extensión Jar).
5. Crear la calendarización de la operación "SendFileJournal" del compuesto "JournalEnt", esta se muestra en el punto 3.1.

12 URL de Acceso y seguridad

A continuación se anexa el WSDL correspondiente al servicio JournalEnt, el cual dentro de la infraestructura de la empresa no requiere seguridad.

URL para el acceso al servicio web en el ambiente de desarrollo:

Servicio	URL
JournalEnt	http://<hostname>:<puerto>/soa-infra/services/default/JournalEnt/JournalEnt?WSDL
JournalBiz	http://<hostname>:<puerto>/soa-infra/services/default/JournalBiz/JournalBiz?WSDL
FinancialsTec	sb://<hostname>:<puerto>/FinancialsTecPs

13 Temas abiertos y cerrados

13.1 Temas Abiertos

ID	Tema	Solución	Responsabilidad	Fecha Objetivo	Fecha impacto

13.2 Temas Cerrados

ID	Tema	Solución	Responsabilidad	Fecha Objetivo	Fecha impacto