PBtoWS - Procesos: Guía Rápida de implementación del Componente Proxy Backend

Este capitulo contiene información relacionada con el proceso de creación de componentes proxy aplicando la arquitectura propuesta en el desarrollo backend. El objetivo de esta sección es centrar al desarrollador en los aspectos fundamentales que debe tener presente al crear componentes Proxies que serán utilizados en los consumos internos de servicios SOAP Powerbuilder. Tener presente que sólo se explicará el proceso de implementación y se excluirán los demás procesos asociados a la utilización. Para más información favor consultar los pasos del Check List Component

Paso 0: Tipos de Componente Proxy

Antes de iniciar el proceso de creación de la clase de consumo se debe identificar el tipo de proxy el cual puede ser de 2 Tipos:

- **ComponentProcessProxy** - Componente de Proceso Estandard: Este tipo de proxy se debe implementar para el consumo de servicios que representan procesos estandares del ERP.
- **ComponenteCrudProxy** - Componente de Proceso Crud: Este tipo de proxy se debe implementar para el consumo de servicios CRUD (simples).

Paso 1: Creación del Proyecto del Componente Proxy

El primer paso consiste en crear el proyecto del componente Proxy. La información relacionada de esta actividad puede ser consultada en el siguiente link Crear Componente Proxy, además el componente debe estar previamente configurado en el catalogo de configuración de componentes.

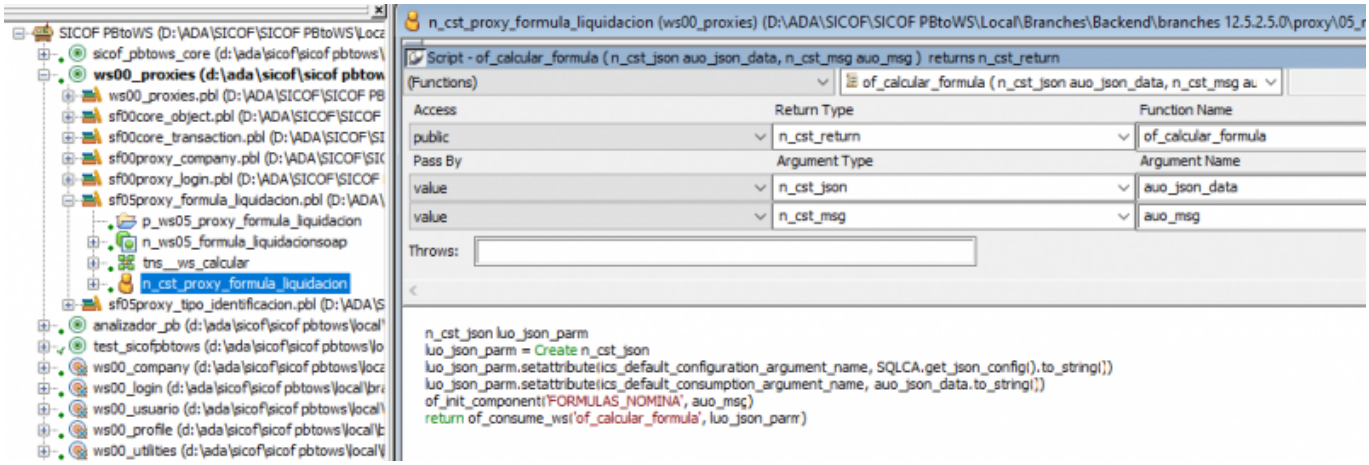
Paso 2: Crear la Clase del Componente Proxy

El siguiente paso consiste en definir la clase de operación del componente proxy de la sigueinte forma:

- Extender la clase **n_cst_proxy** para crear la clase de operación del componente proxy

Nota: Tener presente que la clase de operación debe agregarse a la libreria **sf[código de la aplicación]proxy_(nombre componente).pbl**.

A continuación se visualiza una imagen de ejemplo con la implementación de la clase de operación



1. **Clase de Operación**: Es la encarga de exponer el consumo del servicio web en funciones accesibles en el modelo de invocación de componentes. Se debe tener presente que estas clases en ningun escenario deben implementar lógica del negocio solo deben servir de intefaz de comunicación del consumo y respuesta de los servicios.

Paso 3: Modelo de implementación de invocación por capas

A continuación se explica con ejemplos en imagenes el modelo de implementación de invocación por las capas del framework aplicando la arquitectura propuesta. Se toman 2 casos en los cuales se pueden implementar Proxies.

Paso 3a: Caso ComponentProcessProxy (Ejemplo Componente Fórmulas Liquidación Nómina)

Este caso expone la forma adecuada de creación de clases de operación de proxy para procesos del ERP.

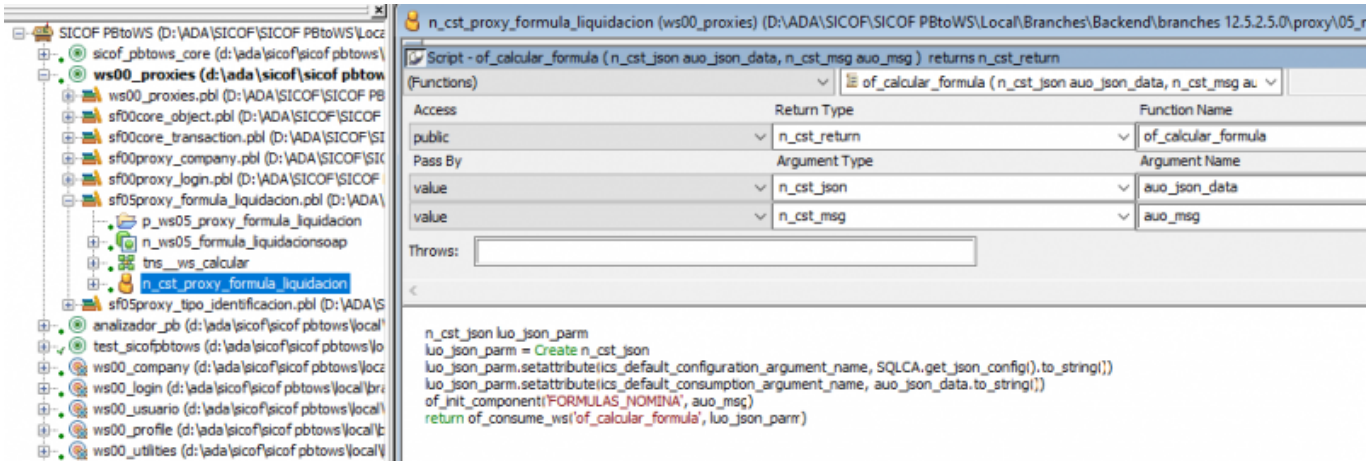
Modelo de implementación: Método Wrapper

El modelo debe implementar un método wrapper[1)] que permita:

1. Inicializar el Componente de consumo del Proxy (Previamente debe estar configurado en el catalogo de componentes. El método **of_init_component** recibe el nombre del componente y la referencia del objeto de mensajes.
2. Crear el json que contiene los parámetros que recibe el servicio el cuál en la mayoría de los casos debe recibir la cadena json del consumo actual **as_config** y una cadena en formato json que especifica los datos que requiere el consumo del proxy.
3. Por último debe consumir la interface **of_consume_ws** donde siempre se debe enviar el método wrapper actual y el json con los parametros de consumo del servicio.

De igual forma debe asegurar que el resultado del proceso debe ser devuelto en un objeto tipo **n_cst_return**

A continuación se visualiza la imagen de la implementación del Componente Fórmula Liquidación.



Consideraciones

- No implementar lógica de negocio en las clases n_cst_proxy

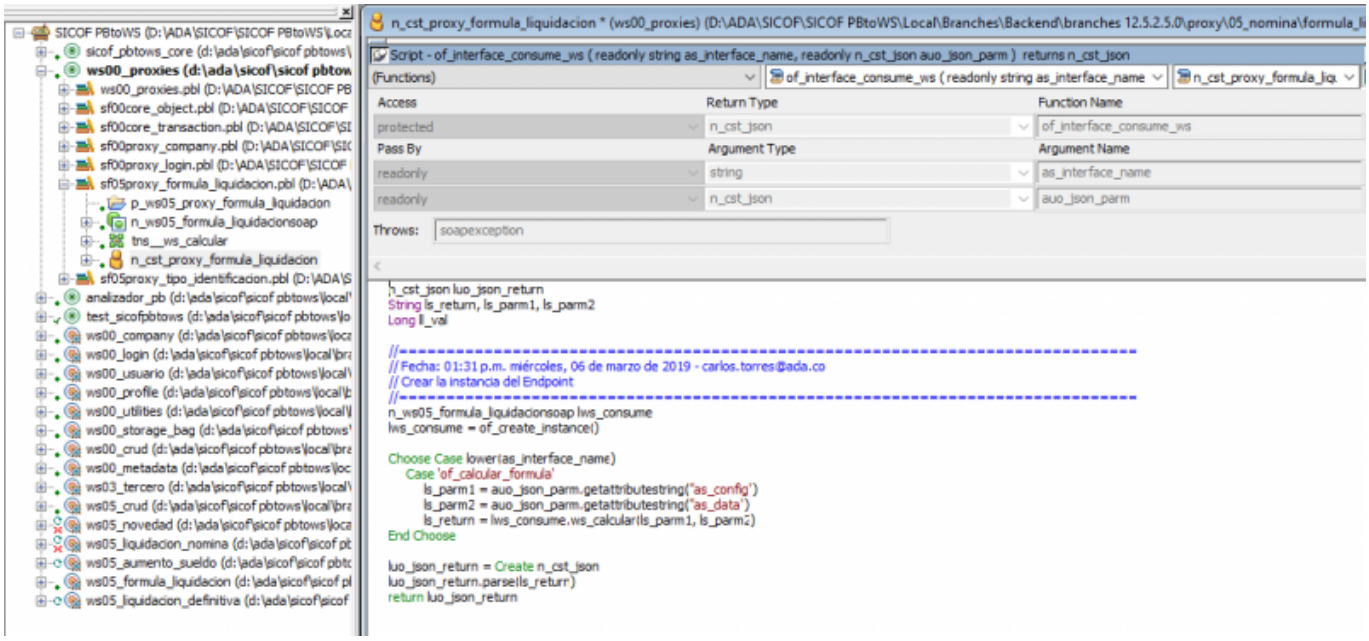
Modelo de implementación: Método of_interface_consume_ws

En este método se debe implementar la lógica de consumo de las operaciones expuestas por el servicio. Se deben seguir los siguientes pasos:

- Definir el objeto proxy del componente el cual contiene los métodos expuestos por el servicio por lo general el nombre es generado automaticamente y está definido de la siguiente forma **n_ws[código de la aplicación]_[nombre del componente]soap** Ejemplo: n_ws05_formula_liquidacionsoap
- Inicializar el objeto proxy consumiendo el método **of_create_instance()**
- Crear un Choose Case donde se puedan identifcar las interfaces de operación del servicio las cuales estan definidas en los métodos Wrapper creados. Cada opción del case debe consumir un método del proxy y debe devolver el resultado en una variable de tipo String
- Se debe encapsular el resultado y devolver en un tipo **n_cst_json** para que sea procesado en las capas de implementación posteriores.

De igual forma debe asegurar que el resultado del proceso debe ser devuelto en un objeto tipo **n_cst_json**

A continuación se visualiza la imagen de la implementación del Componente Fórmula Liquidación.



Consideraciones

- No implementar lógica de negocio en las clases n_cst_proxy

Paso 3b: Caso ComponentCrudProxy (Ejemplo Componente CRUD: Tipo Identificación)

Este caso expone la forma adecuada de creación de clases de operación de proxy para procesos CRUD (simples) del ERP.

Modelo de implementación: Métodos Wrapper con las Operaciones Soportadas por el CRUD

El modelo debe implementar los siguientes métodos wrapper[2)]:

- **of_get_ws_crud_read_only_one**: Para operaciones de consulta de un solo registro
- **of_get_ws_crud_read**: Para operaciones de listado de registro
- **of_get_ws_crud_create**: Para operaciones de inserción
- **of_get_ws_crud_update**: Para operaciones de Actualización
- **of_get_ws_crud_delete**(No Implementado): Para operaciones de eliminación

Estas funciones no pueden ser consumidas directamente por las implementaciones de operación de la clase proxy ya que son métodos protegidos por definición, por lo tanto deben encapsularse en métodos publicos para ser accedidos por la logica del negocio que lo requiera. Por estandarización los métodos deben ser encapsulados en los siguiente métodos públicos:

- **of_get_ws_crud_read_only_one → of_crud_consultar**
- **of_get_ws_crud_read → of_crud_listar**
- **of_get_ws_crud_create → of_crud_insertar**
- **of_get_ws_crud_update → of_crud_actualizar**

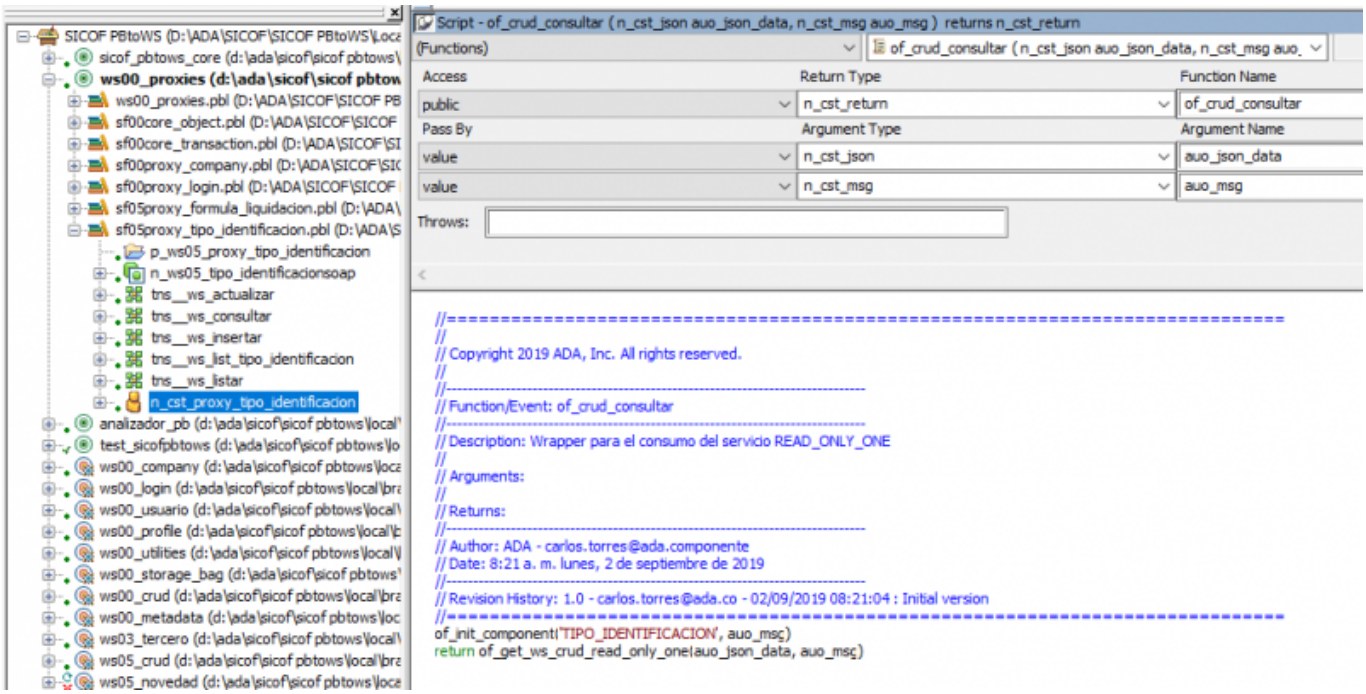
- **of_get_ws_crud_delete**(No Implementado) → **of_crud_eliminar**

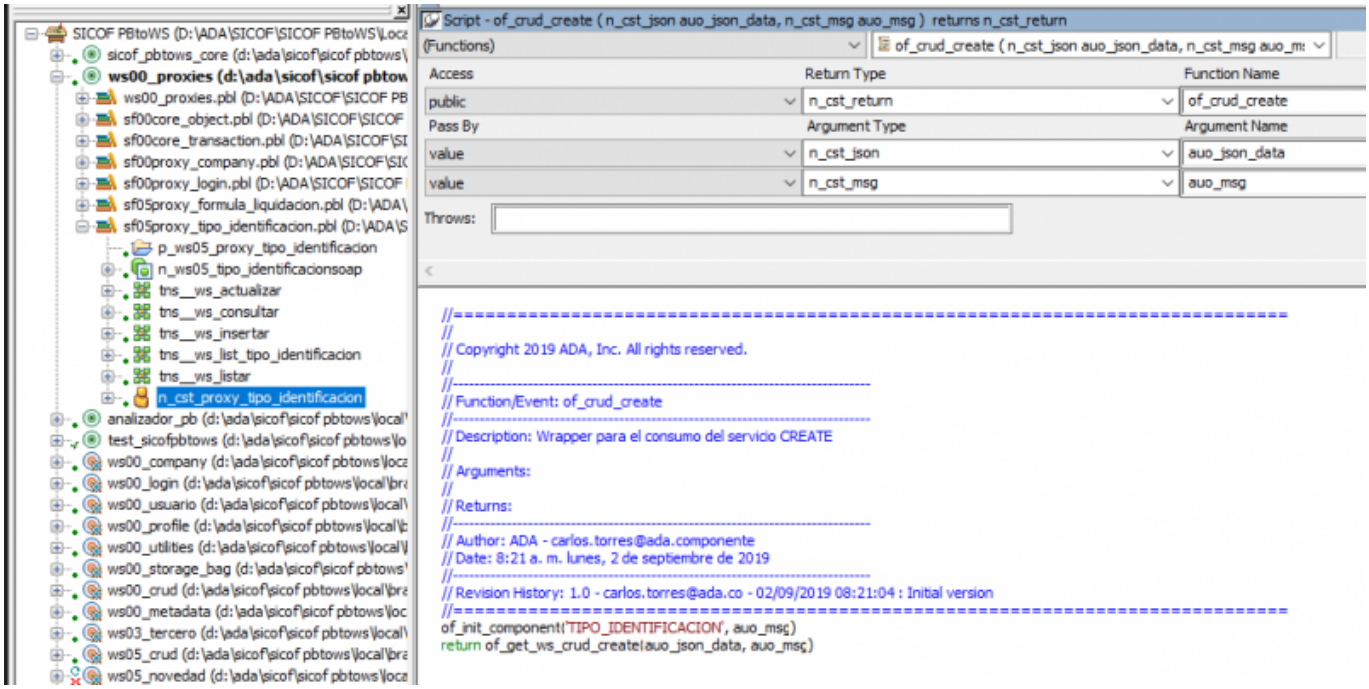
Estos métodos debe que permitir:

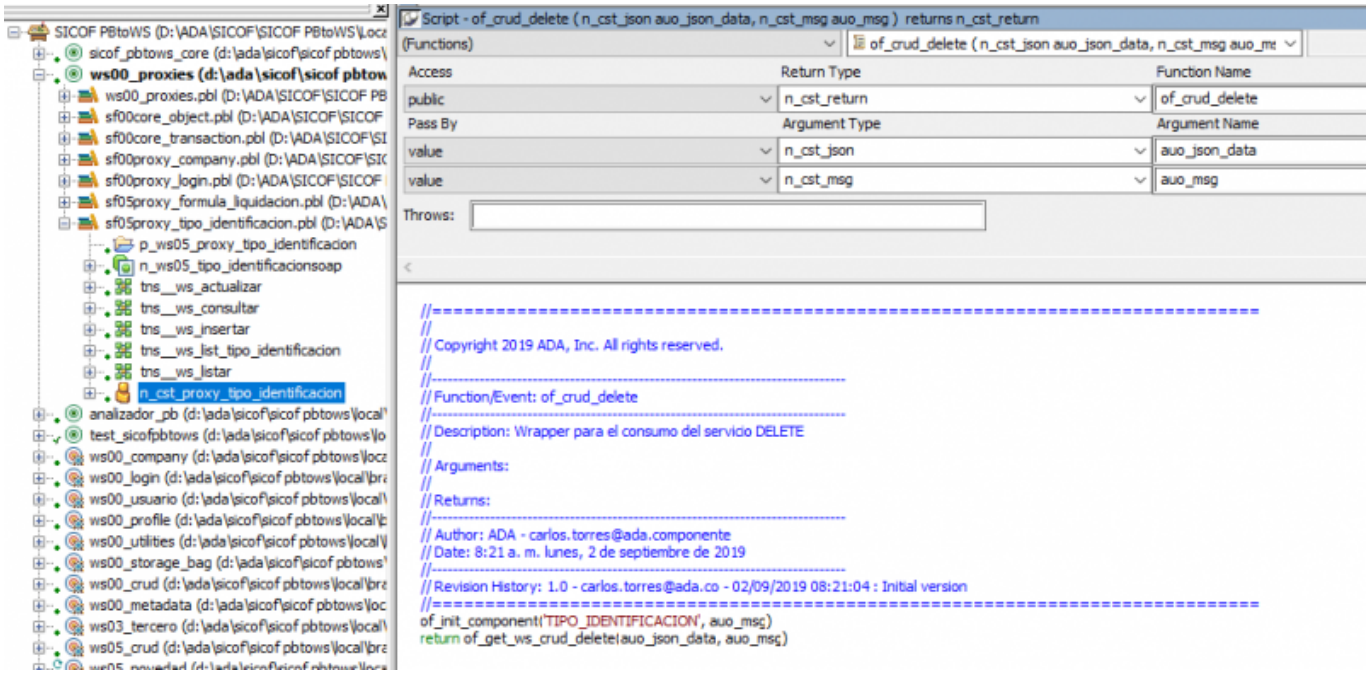
1. Inicializar el Componente de consumo del Proxy (Previamente debe estar configurado en el catalogo de componentes. El método **of_init_component** recibe el nombre del componente y la referencia del objeto de mensajes.
2. Por último debe consumir la interface que corresponda a la operación del wrapper actual siempre se debe enviar el json con los parametros de consumo del servicio (No se debe enviar el config en el json) y el objeto de mensajes .

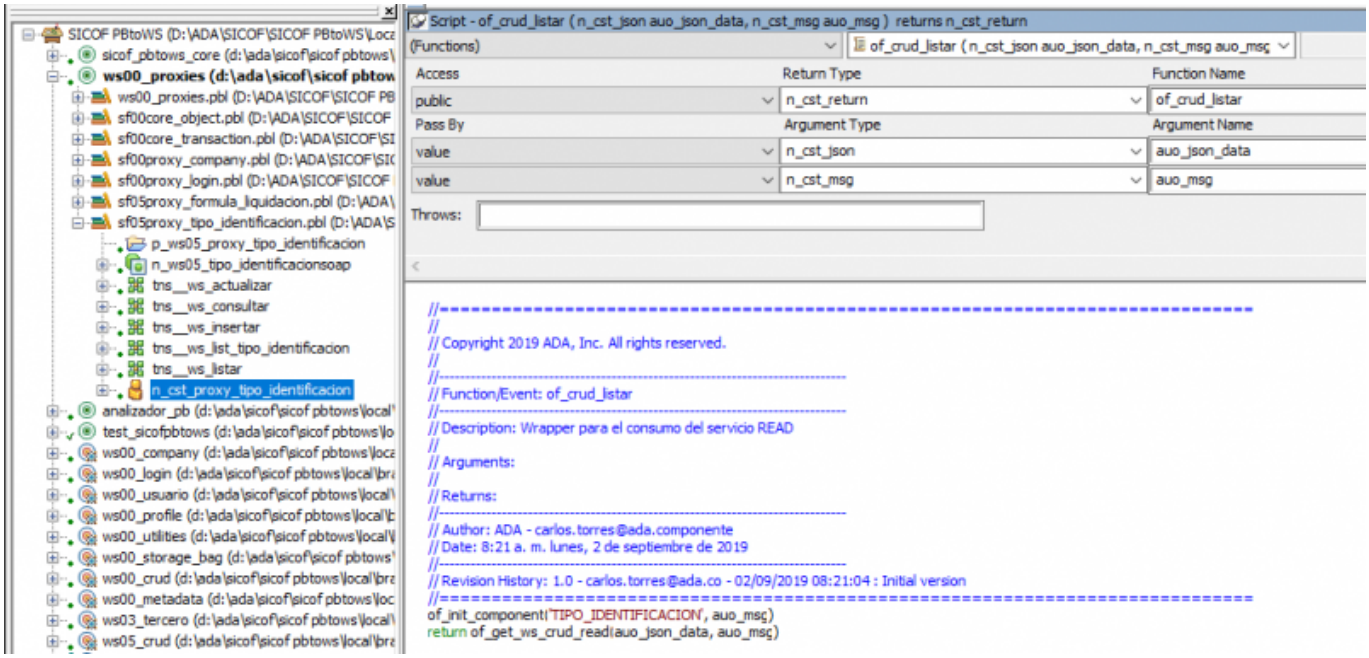
De igual forma debe asegurar que el resultado del proceso debe ser devuelto en un objeto tipo **n_cst_return**

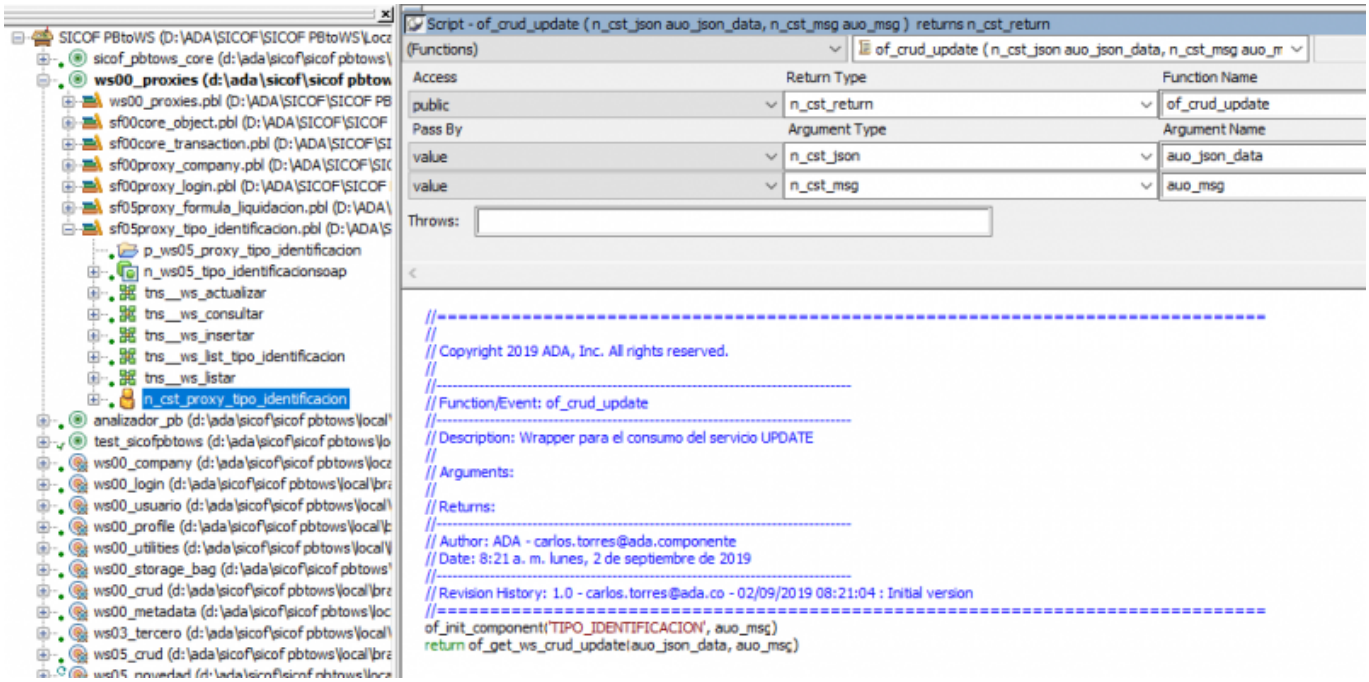
A continuación se visualiza la imagen de la implementación del Componente Fórmula Liquidación.




SICOF PBtoWS (D:\ADA\SICOF\SICOF PBtoWS\Loca
sicof_pbtows_core (d:\ada\sicof\sicof pbtows\
ws00_proxies (d:\ada\sicof\sicof pbtow
ws00_proxies.pbl (D:\ADA\SICOF\SICOF PB
sf00core_object.pbl (D:\ADA\SICOF\SICOF
sf00core_transaction.pbl (D:\ADA\SICOF\SI
sf00proxy_company.pbl (D:\ADA\SICOF\SIC
sf00proxy_login.pbl (D:\ADA\SICOF\SICOF
sf05proxy_formula_liquidacion.pbl (D:\ADA\
sf05proxy_tipo_identificacion.pbl (D:\ADA\S
p_ws05_proxy_tipo_identificacion
n_ws05_tipo_identificacionsoap
tns__ws_actualizar
tns__ws_consultar
tns__ws_insertar
tns__ws_list_tipo_identificacion
tns__ws_listar
n_cst_proxy_tipo_identificacion
analizador_pb (d:\ada\sicof\sicof pbtows\local
test_sicofpbtows (d:\ada\sicof\sicof pbtows\lo
ws00_company (d:\ada\sicof\sicof pbtows\loca
ws00_login (d:\ada\sicof\sicof pbtows\local\bra
ws00_usuario (d:\ada\sicof\sicof pbtows\local\
ws00_profile (d:\ada\sicof\sicof pbtows\local\p
ws00_utilities (d:\ada\sicof\sicof pbtows\local\
ws00_storage_bag (d:\ada\sicof\sicof pbtows'
ws00_crud (d:\ada\sicof\sicof pbtows\local\bra
ws00_metadata (d:\ada\sicof\sicof pbtows\loc
ws03_tercero (d:\ada\sicof\sicof pbtows\local\
ws05_crud (d:\ada\sicof\sicof pbtows\local\bra
ws05_novedad (d:\ada\sicof\sicof pbtows\loca
Script - of_crud_create (n_cst_json auo_json_data, n_cst_msg auo_msg) returns n_cst_return
(Functions)
of_crud_create (n_cst_json auo_json_data, n_cst_msg auo_ms
Access
Return Type
Function Name
public
n_cst_return
of_crud_create
Pass By
Argument Type
Argument Name
value
n_cst_json
auo_json_data
value
n_cst_msg
auo_msg
Throws:
//==
//
// Copyright 2019 ADA, Inc. All rights reserved.
//
//--
// Function/Event: of_crud_create
//--
// Description: Wrapper para el consumo del servicio CREATE
//
// Arguments:
//
// Returns:
//--
// Author: ADA - carlos.torres@ada.componente
// Date: 8:21 a. m. lunes, 2 de septiembre de 2019
//--
// Revision History: 1.0 - carlos.torres@ada.co - 02/09/2019 08:21:04 : Initial version
//==
of_init_component('TIPO_IDENTIFICACION', auo_msg)
return of_get_ws_crud_create(auo_json_data, auo_msg)


SICOF PBtoWS (D:\ADA\SICOF\SICOF PBtoWS\Loca
sicof_pbtows_core (d:\ada\sicof\sicof pbtows\
ws00_proxies (d:\ada\sicof\sicof pbtow
ws00_proxies.pbl (D:\ADA\SICOF\SICOF PB
sf00core_object.pbl (D:\ADA\SICOF\SICOF
sf00core_transaction.pbl (D:\ADA\SICOF\SI
sf00proxy_company.pbl (D:\ADA\SICOF\SIC
sf00proxy_login.pbl (D:\ADA\SICOF\SICOF
sf05proxy_formula_liquidacion.pbl (D:\ADA\
sf05proxy_tipo_identificacion.pbl (D:\ADA\S
p_ws05_proxy_tipo_identificacion
n_ws05_tipo_identificacionsoap
tns__ws_actualizar
tns__ws_consultar
tns__ws_insertar
tns__ws_list_tipo_identificacion
tns__ws_listar
n_cst_proxy_tipo_identificacion
analizador_pb (d:\ada\sicof\sicof pbtows\local
test_sicofpbtows (d:\ada\sicof\sicof pbtows\lo
ws00_company (d:\ada\sicof\sicof pbtows\loca
ws00_login (d:\ada\sicof\sicof pbtows\local\bra
ws00_usuario (d:\ada\sicof\sicof pbtows\local\
ws00_profile (d:\ada\sicof\sicof pbtows\local\p
ws00_utilities (d:\ada\sicof\sicof pbtows\local\
ws00_storage_bag (d:\ada\sicof\sicof pbtows'
ws00_crud (d:\ada\sicof\sicof pbtows\local\bra
ws00_metadata (d:\ada\sicof\sicof pbtows\loc
ws03_tercero (d:\ada\sicof\sicof pbtows\local\
ws05_crud (d:\ada\sicof\sicof pbtows\local\bra
Script - of_crud_delete (n_cst_json auo_json_data, n_cst_msg auo_msg) returns n_cst_return
(Functions)
of_crud_delete (n_cst_json auo_json_data, n_cst_msg auo_ms
Access
Return Type
Function Name
public
n_cst_return
of_crud_delete
Pass By
Argument Type
Argument Name
value
n_cst_json
auo_json_data
value
n_cst_msg
auo_msg
Throws:
//==
//
// Copyright 2019 ADA, Inc. All rights reserved.
//
//--
// Function/Event: of_crud_delete
//--
// Description: Wrapper para el consumo del servicio DELETE
//
// Arguments:
//
// Returns:
//--
// Author: ADA - carlos.torres@ada.componente
// Date: 8:21 a. m. lunes, 2 de septiembre de 2019
//--
// Revision History: 1.0 - carlos.torres@ada.co - 02/09/2019 08:21:04 : Initial version
//==
of_init_component('TIPO_IDENTIFICACION', auo_msg)
return of_get_ws_crud_delete(auo_json_data, auo_msg)





Consideraciones

- No implementar lógica de negocio en las clases n_cst_proxy

Modelo de implementación: Método of_interface_consume_ws

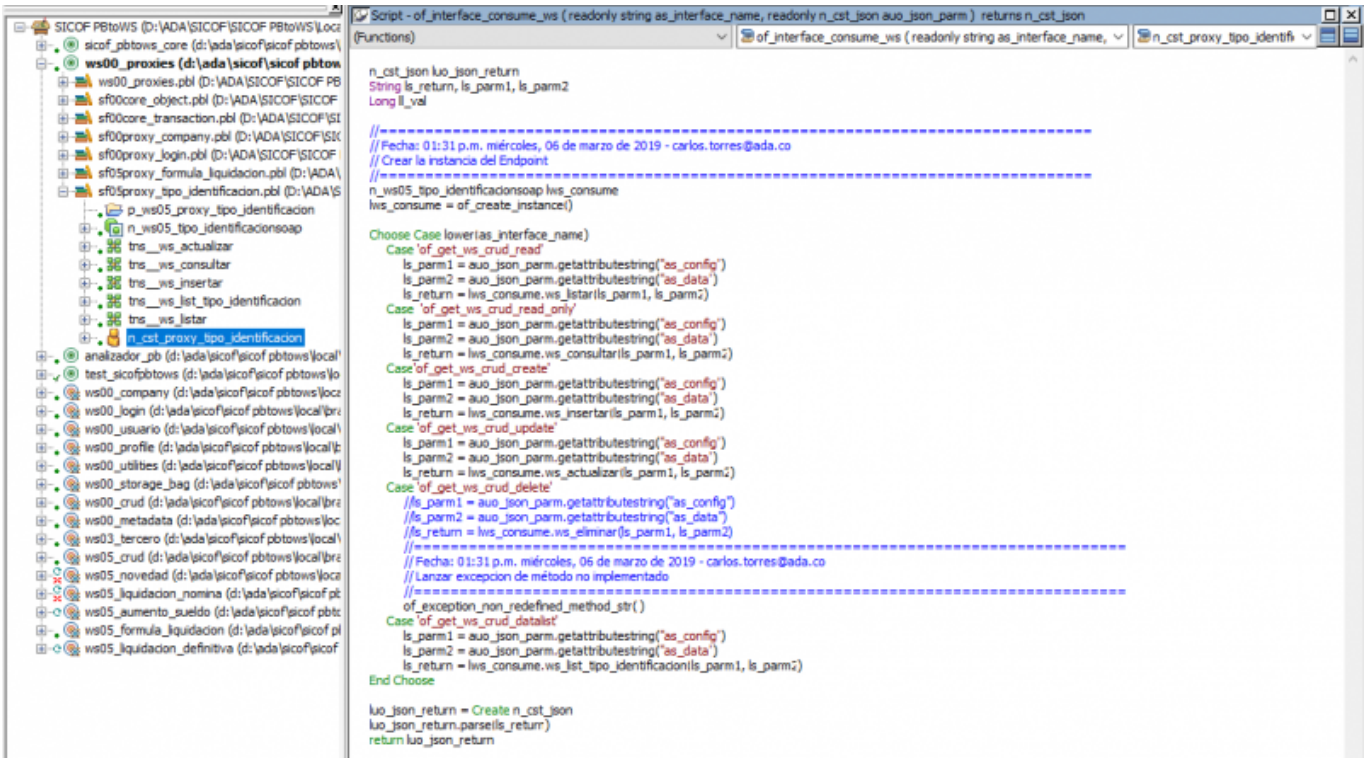
En este método se debe implementar la lógica de consumo de las operaciones expuestas por el servicio. Se deben seguir los siguientes pasos:

- Definir el objeto proxy del componente el cual contiene los métodos expuestos por el servicio por lo general el nombre es generado automaticamente y está definido de la siguiente forma **n_ws[código de la aplicación]_[nombre del componente]soap** Ejemplo: n_ws05_formula_liquidacionsoap

- Inicializar el objeto proxy consumiendo el método **of_create_instance()**
- Crear un Choose Case donde se puedan identifcar las interfaces de operación del servicio las cuales estan definidas en los métodos Wrapper creados. Cada opción del case debe consumir un método del proxy y debe devolver el resultado en una variable de tipo String
- Se debe encapsular el resultado y devolver en un tipo **n_cst_json** para que sea procesado en las capas de implementación posteriores.

De igual forma debe asegurar que el resultado del proceso debe ser devuelto en un objeto tipo **n_cst_json**

A continuación se visualiza la imagen de la implementación del Componente Fórmula Liquidación.



Consideraciones

- No implementar lógica de negocio en las clases n_cst_proxy

[1] [2]

http://consejostallerdeprogramacion.blogspot.com/2015/11/que-es-wrapper-y-donde-lo-utilizamos.html