CASO DE ÉXITO

IMPLANTACIÓN DE LA ARQUITECTURA WSO2 EN UNA EMPRESA AUTOMOVILÍSTICA

INTRODUCCIÓN

La integración de sistemas suele ser uno de los principales obstáculos para cualquier compañía, independientemente de su tamaño. Servicios escritos en distintas tecnologías o en distintos lenguajes, es uno de los problemas a los que se enfrentan día a día muchas empresas. Te suena, ¿verdad?

Afortunadamente, existen soluciones rápidas y fáciles de implementar como WSO2, que ponen fin a esta complicada tarea de IT.

A continuación se presenta un exitoso caso de integración de sistemas, llevado a cabo con el software de WSO2, en una en una de las compañías automovilísticas más prestigiosas del panorama internacional, la cual cuenta con más de 35000 empleados.

El reto culminó en un tremendo éxito, por ello el objetivo de este ebook es compartir los pasos llevados a cabo a lo largo del proceso. Conoce en primera persona los problemas y acontecimientos que surgieron durante la implantación, los módulos de WSO2 instalados y muchas sorpresas más.

Seguro que puede ayudarte mucho a la hora de iniciarte en la integración de sistemas, así como animarte a llevarla a cabo en tu compañía. Descubre las grandes ventajas que esta plataforma Open Source puede ofrecerte a ti, y a tu organización.

¡No esperes más!

¿CUÁL ES EL PROBLEMA QUE SE PLANTEA?

Antes de abordar el quid de la cuestión es necesario destacar que las herramientas de integración suelen emplearse cuando se desea comunicar diferentes componentes de un sistema entre sí, con el fin de solucionar un problema.

No obstante, puede suceder que cada uno de estos módulos haya sido escrito con distintas tecnologías o lenguajes de programación. Ante tal situación las empresas deben valorar qué alternativas de integración disponen, como así lo hizo nuestra empresa cliente. Una de las opciones a barajar es comprar una herramienta de integración, lo cual resulta económico. Mientras que la otra opción, sería migrar todos los componentes de un sistema framework, escrito en el mismo lenguaje, suponiendo un elevado coste para la organización.

En tal caso, la solución era más que obvia, por lo que la compañía automovilística optó por un middleware como WSO2, el cual simplificaría en gran medida la interacción entre componentes. ¡Una maravilla, vaya!

> Una de las opciones a barajar es comprar una herramienta de integración, [...] la otra opción, sería migrar todos los componentes de un sistema framework, escrito en el mismo lenguaje

Cabe destacar que por componente entendemos que se trata de cualquier servicio, rutina, capaz de recibir una petición y articular una respuesta tras un procesamiento previo. La solución de WSO2 se implementa siguiendo unos estándares de integración. Además se incluye la capacidad de interacción entre componentes que usan diferentes protocolos de comunicaciones.



A continuación se presenta un ejemplo con el protocolo HTTP:

Tal y como se puede observar, el usuario emplea una misma petición para usar dos servicios implementados en distintos lenguajes, gracias al middleware. De este modo, la petición puede enriquecerse con más información, retornando la respuesta al usuario.

Esta solución resultó idónea ya que aporta flexibilidad a la compañía, además de resultar más económica que migrar todos los servicios a un único lenguaje de programación, pudiendo seguir utilizando los servicios a través de un mediador: **WSO2 middleware.**

Si tomamos de referencia el ejemplo presentado, la comunicación entre estos se produce mediante servicios web sobre protocolo HTTP.

> Esta solución resultó idónea ya que aporta flexibilidad a la compañía, además de resultar más económica que migrar todos los servicios a un único lenguaje de programación, pudiendo seguir utilizando los servicios a través de un mediador: **WSO2 middleware**

¿QUÉ ARQUITECTURA DE WSO2 SE INSTALÓ?

Antes de adentrarnos en la arquitectura realizada al cliente en cuestión, cabe destacar que esta decisión es personalizable y se orquesta en función de las necesidades de integración de cada compañía.

Respecto al caso de este cliente británico, se optó por la instalación de toda la suite de WSO2, la cual consta de diversos módulos que dan respuesta a distintos problemas:

- **WSO2 API Manager:** este módulo le permite al usuario crear y publicar APIs para uso interno (Intranet) o externo (Extranet). Además, otros usuarios pueden suscribirse a las APIs a través de aplicaciones creadas en esta herramienta. Esto se realiza en el Store donde se pueden ver todas las APIs que fueron previamente publicadas.
- **WSO2 Identity Server (IS):** se trata de un servidor de identidad. Se emplea para gestionar el Single Sign On (SSO), entre proveedores de servicio y como gestor de tokens de acceso a las APIs (Oauth2 protocol).
- **WSO2 API Publisher y Store:** se trata de herramientas web utilizadas para crear, publicar las APIs y suscribir a las APIs. Este módulo se encuentra incluido en el API Manager.
- **WSO2 ESB:** es el proxy empleado para asegurar la comunicación entre componentes. Cabe destacar que abarca más funcionalidades puesto que se trata de un BUS empresarial, como por ejemplo: enriquecer mensajes, redireccionarlos a distintos puntos finales, hacer de balanceador, acceder a bases de datos y a cualquier servicio, a través de llamadas RestFul. En el caso de este cliente, se empleó json para el formato de mensajes.
- **WSO2 Message Broker:** este módulo se emplea para gestionar las colas creadas en el ESB. Dichas colas se utilizan para guardar mensajes durante algún tiempo hasta que el punto final, al cual van destinadas, es capaz de procesarlos.

Resulta importante destacar, que para ofrecer un mejor servicio se utilizó el balanceador **nginx** para las gateways de las APIs y los ESBs. Este balanceador se encarga de repartir las peticiones entre los workers (nodos), tanto de la gateway como del ESB.

DIAGRAMA DE LA ARQUITECTURA WSO2

Tal y como se puede observar en la siguiente imagen, el **API Publisher** y el **API Store** tienen acceso desde la Intranet. Existe un número de usuarios con "privilegios" de publicadores, los cuales pueden crear y publicar las APIs. El resto de usuarios tienen acceso al Store para poder dar de alta aplicaciones con las que se suscriben a las APIs.

Las APIs que se han creado se despliegan a los gateways que reciben las peticiones desde la Intranet. Resulta importante destacar, que este proceso también se puede realizar haciendo uso de los servicios Restful del **API Manager.**

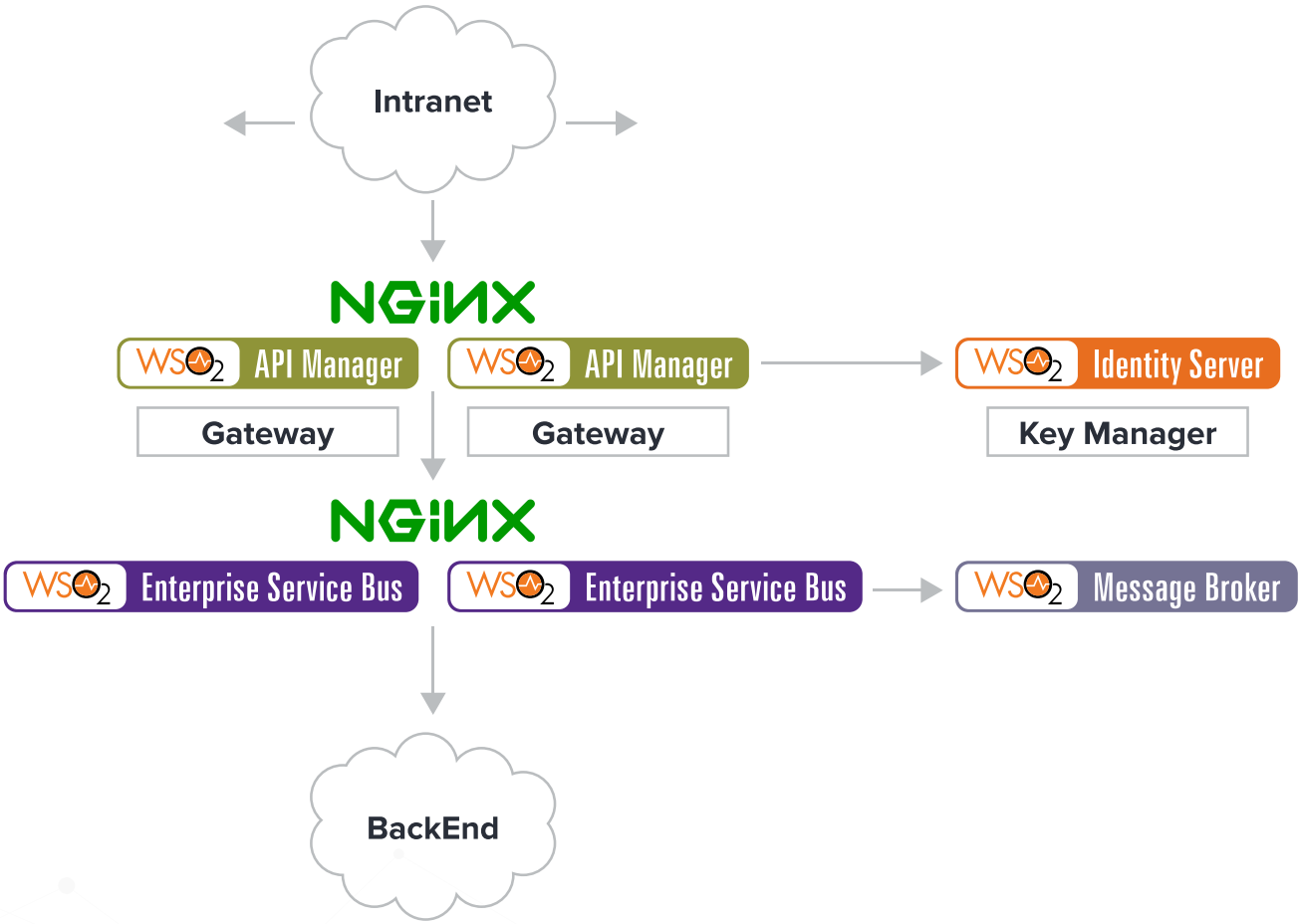


Diagrama de la arquitectura de WSO2 realizada

Todas las APIs gestionan la seguridad con el protocolo Oauth2. El servicio de dicho protocolo se ha desplegado en el Identity Server: **Key Manager**. Es este punto final quien recibe las solicitudes de acceso a las APIs, y quien envía los token de acceso, en el caso de que el usuario y clave, sean correctos en cada petición.

Cade destacar que para facilitar el despliegue de los componentes en distintos servidores empleamos la herramienta Puppet.

Siguiendo con la integración, el Key Manager se ha desplegado en el Identity Server (IS), que originalmente está en el API Manager. Por tanto, existen una serie de tablas que tienen que ser compartidas entre ambas figuras, las cuales son:

- **WSO2REG_DB:** Contiene información del registro de artefactos, datos de configuración, políticas y detalles de las APIs.
- **WSO2UM_DB:** en esta tabla se almacenan los permisos y roles de los distintos usuarios.
- **WSO2AM_DB:** Mantiene datos de identidad e información de las APIs, como los token Oauth y las claves de acceso a las APIs.

Resulta importante subrayar que estos esquemas se han desplegado en un clúster con MariaDB.

TIPOS DE RESPUESTA: JSON O XML

Otro de los aspectos a considerar en el proceso de integración de sistemas con WSO2, era retornar los mensajes de error del API Manager, en formato **json**. Para ello se debían modificar diversos archivos:

_auth_failure_handler.xml

```
<AM_HOME>/repository/deployment/server/synapse-configs/default/sequences/_auth_failure_
handler.xml

<sequence name="_auth_failure_handler_" xmlns="http://ws.apache.org/ns/synapse">
<property name="error_message_type" value="application/json"/>
<property name="TRANSPORT_HEADERS" action="remove" scope="axis2"/>
<sequence key="_cors_request_handler_"/>
</sequence>
```

En este archivo el valor de la propiedad **"error_message_type"** se debe modificar de **"application/xml"** a **"application/json"**

fault.xml

```
<AM_HOME>/repository/deployment/server/synapse-configs/default/sequences/fault.xml

<sequence xmlns="http://ws.apache.org/ns/synapse" name="fault">
...
<filter source="$axis2:HTTP_METHOD" regex="^(?!.*(POST|PUT)).*$">
<property name="messageType" value="application/json " scope="axis2"/>
</filter>
...
</sequence>
```

Gracias a esta modificación, el mensaje de error generado por el API Manager aparece en formato **json**.

A modo de ejemplo, el mensaje de error en formato json generado por el API Manager sería de la siguiente forma:

```
{
"fault":{
"code":900902,
"message":"Missing Credentials",
"description":"Required OAuth credentials not provided. Make sure your API invocation call
has a
header: \"Authorization: Bearer ACCESS_TOKEN\""
}
}
```

Las modificaciones se realizan en el nodo Manager y será propagado al resto de los workers gracias al SVN (Subversion).

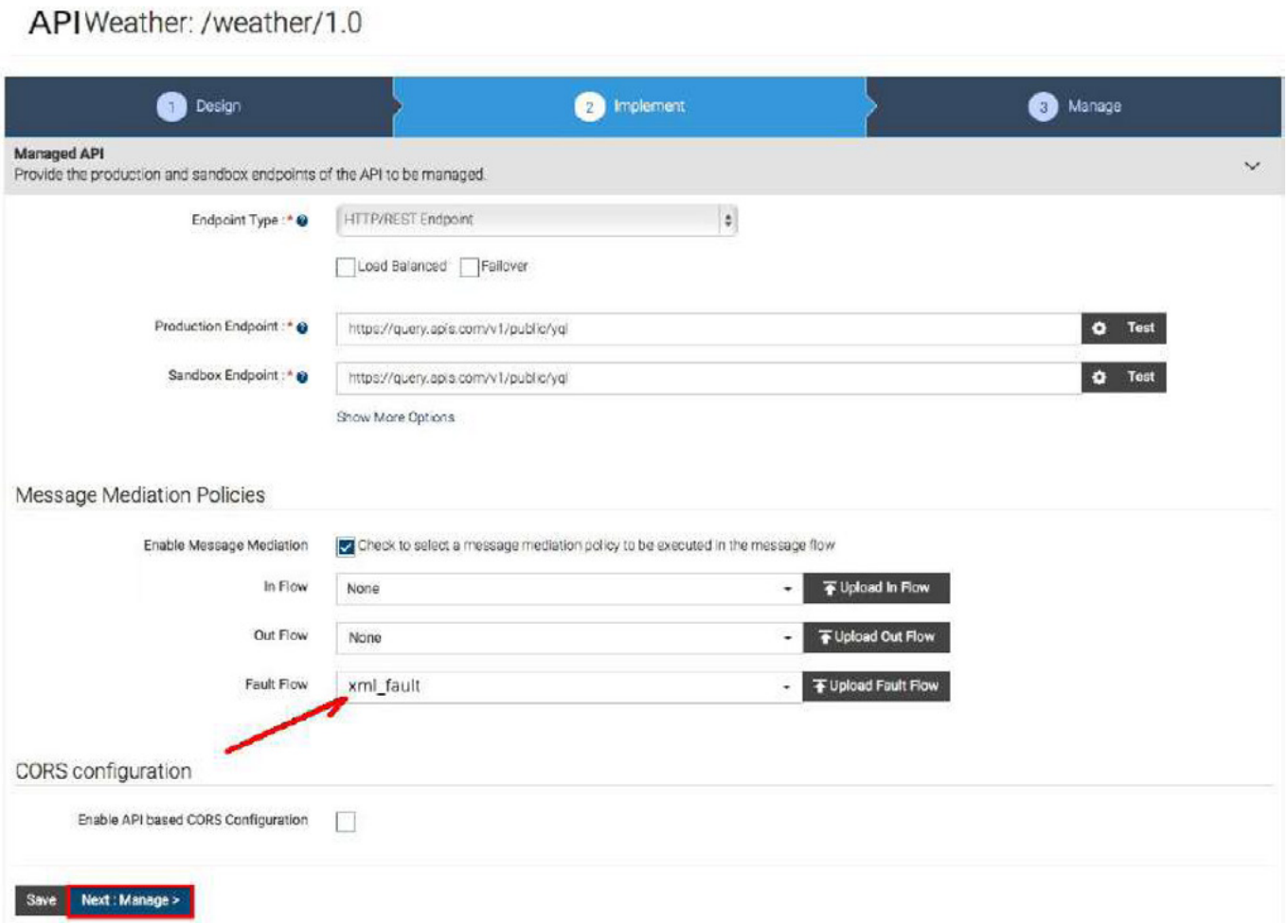
En el caso de que cualquier otra API quiera enviar mensajes de error en XML, resultará necesario crear una secuencia en el archivo **xml_fault**, de la siguiente forma:

```
<AM_HOME>/repository/deployment/server/synapse-configs/default/sequences/xml_fault

<?xml version="1.0" encoding="ISO-8859-1"?>
<sequence xmlns="http://ws.apache.org/ns/synapse" name="xml_fault">
<log level="custom">
<property name="STATUS" value="Executing custom 'fault' sequence"/>
<property name="ERROR_CODE" expression="get-property('ERROR_CODE')"/>
<property name="ERROR_MESSAGE" expression="get-property('ERROR_MESSAGE')"/>
</log>
<payloadFactory>
<format>
<am:fault xmlns:am="http://wso2.org/apimanager">
<am:code>$1</am:code>
<am:type>Status report</am:type>
<am:message>Runtime Error</am:message>
<am:description>$2</am:description>
</am:fault>
</format>
<args>
www.chakray.com 11
<arg expression="$ctx:ERROR_CODE"/>
<arg expression="$ctx:ERROR_MESSAGE"/>
</args>
</payloadFactory>
<filter xpath="$ctx:CUSTOM_HTTP_SC">
<then>
<property name="HTTP_SC" expression="$ctx:CUSTOM_HTTP_SC" scope="axis2"/>
</then>
<else>
<property name="HTTP_SC" value="500" scope="axis2"/>
</else>
</filter>
<class name="org.wso2.carbon.apimgt.usage.publisher.APIMgtFaultHandler"/>
<property name="RESPONSE" value="true"/>
<header name="To" action="remove"/>
<property name="NO_ENTITY_BODY" scope="axis2" action="remove"/>
<property name="ContentType" scope="axis2" action="remove"/>
<property name="Authorization" scope="transport" action="remove"/>
<property name="Host" scope="transport" action="remove"/>
<property name="Accept" scope="transport" action="remove"/>
<property name="X-JWT-Assertion" scope="transport" action="remove"/>
<property name="messageType" value="application/xml" scope="axis2"/>
<send/>
<drop/>
</sequence>
```

Es importante destacar que SVN (Subversion) no actúa en este directorio, por lo que resulta necesario realizar este cambio en el manager y los workers de manera manual. Finalmente, se debe reiniciar el servidor.

En el momento que se está creando una API o se encuentra en modo de edición, se puede seleccionar la secuencia que se va a ejecutar cuando se produzca un fallo, de tal manera que podremos seleccionar **xml_fault** para que el mensaje de error sea en **XML.**



ENTORNO DE DESARROLLO

Debido a las características del cliente en particular, se optó por emplear Wiremock como simulador de APIs HTTP, mientras que el equipo de desarrollo llevaba a cabo todo el backend de las APIs. Así, se podía trabajar paralelamente en el desarrollo de las APIs en el API Manager y en el ESB.

En el caso de que se deseara ejecutar el servidor Wiremock escuchando por elpuerto 8081, y mostrar el archivo de logs por pantalla, sería: java -jar wiremock-1.57-standalone.jar --port 8081 –verbose

Como sabemos que la mejor forma de comprender las cosas es mediante ejemplos, a continuación te proponemos la configuración de una API en la herramienta WireMock:

```
{
"priority": 1,
"request": {
"method": "GET",
"urlPathPattern": "/Parties/Parties/[A-Za-z0-9]*",
"queryParameters" : {
"lang": {
"equalTo" : "en"
},
"brand": {
"equalTo" : "any"
}
}
},
"response": {
"status": 200,
"bodyFileName": "../__files/JsonFiles/cp-Parties-en.json",
"headers": {
"Content-Type": "application/json",
"charset": "utf-8"
}
}
}
```

Como hemos podido comprobar en el código, la request que se envía al servidor mock sigue el método GET. Mientras que la url debe responder al patrón "/Parties/Parties/[A-Za-z0-9]*", y como parámetros se espera lang=en y brand=any.

La respuesta que se origina es un mensaje o payload en formato json, se guarda en el archivo cp-Parties-en.json.

Una vez realizado esto, si se desea probar la API se puede usar el comando url, tal y como aparece en el ejemplo siguiente:

```
curl -X POST http://host-wiremock-0:8081/apiexample/1/item?lang=en
```

Además de este comando, otro que puede resultarte muy útil para el archivo de logs de esta herramienta sería:

```
tail -f /var/log/wiremock.log
```

Si creías haberlo visto todo con Wiremock, aún guarda muchas ventajas más, ya que también es capaz de mapear la url de la llamada y enviar directamente la carga útil en formato xml o json. En este caso, el backend no existe, tan solo se trata de un servidor mock.

Otro ejemplo de una llamada a las APIs a través de la API Gateway, sería:

```
curl --header "Accept: application/json" --header "Authorization: Bearer b642d7d1a04971badd-
31a97607b6dd8" "https://apigateway.com/v1/ReferenceData/ReferenceDataItems?lang=en"
```

Allí aparece el token de acceso, el cual será validado en el Identity Server (IS). Si es correcto, la llamada se redirige hacia el ESB y de este al backend donde será procesada. La respuesta que se devuelve es en formato json.



RECURSO API SIN PROTECCIÓN: OAUTH2 PROTOCOL

En el proceso de implantación de WSO2, uno de los recursos de una de las APIs tuvo que ser desprotegido para poder trabajar con él, sin protección Oauth2. El problema que nos encontramos residía en que Gigya (Identity Server de SAP), no era capaz de enviar mensajes de notificación en respuesta a eventos utilizando este protocolo.

La manera de desproteger un recurso en el API Manager es muy sencilla. Tan solo se debe pulsar la sección **Manager.**

En la siguiente ilustración se muestra un listado de los recursos de la API. El tipo de autenticación por defecto es la "Application & Application User". Si seleccionamos dicho valor aparece un listado de posibles opciones, entre las cuales debemos seleccionar "None". Como ahora el tipo de autenticación es "None", el usuario podrá llamar al recurso de la API sin necesidad de usar el token Oauth.

CONCLUSIÓN

Integrar en una compañía sistemas en distintos lenguajes de forma económica ¡ya no es imposible! WSO2 API Manager es la mejor herramienta Open Source para gestionar las APIs a nivel empresarial.

Otra de sus ventajas, es que es muy sencilla de instalar y de usar. Además, es posible modificar su funcionamiento con el fin de adaptarlo al máximo a las necesidades del cliente. Esto se realiza por medio de la configuración o usando punto de extensión como interfaces.

La gestión de las APIs está basada en roles con distintos permisos. Si nos centramos en concreto en este caso de éxito, también se ha incluido el Identity Server (IS), que gestiona la autorización y autenticación; y el WSO2 ESB, que recibe las peticiones desde la API Gateway, para realizar un procesamiento extra a los mensajes, dotando de mayor flexibilidad a la herramienta.

Si tienes dudas o necesitas asesoramiento, **no dudes en contactar con nosotros**, nuestros asesores estarán encantados de ayudarte.

CONTÁCTANOS

¿Quieres mejorar los sistemas de tu empresa? ¡Consulta a nuestros expertos!

Pregunta sin compromiso a nuestros consultores. Te ayudaremos a encontrar la mejor solución para tu proyecto.

CONTACTA CON NOSOTROS

ESPAÑA

C/ Gonzalo Jiménez de Quesada, 2, Torre Sevilla, planta 4, 41092, Sevilla

contact@chakray.com

+34 955 252 520

REINO UNIDO

3 High Street, Warwick, Warwickshire, CV34 4AP

info@chakray.co.uk

+44 (0) 1926 298 195

SRI LANKA

104 1/1, Pagoda Road, Kotte, Pita Kotte. Sri Lanka

apac-info@chakray.com

+94 11 580 0887

MÉXICO

Calle Parral Nº 41
Colonia Condesa
Delegación Cuauhtémoc
CP 06140, Ciudad de México

info.mexico@chakray.com

+52 55 8311 4415

PERÚ

Los Ibis 165, Dpto. 101, San Isidro, CP 15036, Lima. Peru

info.peru@chakray.com

+51 1 644 9116

CANADÁ

40 rue François-De Lauzon
La Prairie (Québec) J5R6W6
Canada

info.canada@chakray.com

+1 (581) 700 03 75