EXEMPLE DE RÉUSSITE

MISE EN ŒUVRE DE L'ARCHITECTURE WSO2 DANS UNE ENTREPRISE AUTOMOBILE

INTRODUCTION

L'intégration des systèmes est souvent l'un des principaux obstacles pour toute entreprise, quelle que soit sa dimension. Les services écrits dans des technologies ou dans des langues différentes, est l'un des problèmes auxquels de nombreuses entreprises sont confrontées chaque jour. Ça vous rappelle quelque chose, non?

Heureusement, il existe des solutions rapides et faciles à mettre en œuvre comme le WSO2, qui met fin à cette tâche informatique compliquée.

Voici un cas réussi d'intégration de systèmes réalisé avec le logiciel WSO2, dans l'une des plus prestigieuses entreprises automobiles internationales, qui compte plus de 35 000 employés.

Ce défi a été couronné d'un énorme succès, c'est pourquoi l'objectif de ce livre numérique est de partager les étapes franchies en cours de route. Découvrez les problèmes et les événements qui se sont produits lors de la mise en œuvre, les modules WSO2 installés et bien d'autres surprises.

Je suis sûr qu'il peut vous aider beaucoup à démarrer l'intégration de systèmes et vous encourager à la réaliser dans votre entreprise. Découvrez les énormes avantages que cette plateforme à logiciel libre (Open Source) peut vous offrir, à vous et à votre organisation.

N'attendez pas plus longtemps!

QUEL EST LE PROBLÈME?

Avant d'aborder le fond du problème, il convient de noter que les outils d'intégration sont souvent utilisés lorsque les différentes composantes d'un système doivent communiquer entre elles afin de résoudre un problème.

Il se peut par contre, que chacun de ces modules ait été écrit avec des technologies ou des langages de programmation différents. Face à une telle situation, les entreprises doivent évaluer les alternatives d'intégration dont elles disposent, comme l'a fait notre entreprise cliente. L'une des options à envisager est l'achat d'un outil d'intégration, qui est peu coûteux. L'autre option serait de migrer tous les éléments d'un système cadre, écrits dans la même langue, à un coût élevé pour l'organisation.

Dans ce cas, la solution était plus qu'évidente, de sorte que le constructeur automobile a opté pour un logiciel médiateur (middleware) tel que WSO2, qui simplifierait grandement l'interaction entre les composants. Incroyable, une merveille!

> L'une des options à envisager est l'achat d'un outil d'intégration, [...] l'autre option serait de migrer tous les composants d'un système cadre, écrits dans la même langue.

Il faut noter que par composante, nous entendons tout service, routine, capable de recevoir une demande et d'articuler une réponse après traitement préalable. La solution WSO2 est mise en œuvre selon les normes suivantes de l'intégration. La possibilité d'interagir entre les composants en utilisant différents protocoles de communication est également incluse.



Voici un exemple avec le protocole HTTP :

Comme vous pouvez le voir, l'utilisateur utilise la même requête pour l'utilisation de deux services mis en œuvre dans des langues différentes, grâce au logiciel médiateur (middleware). De cette façon, la demande peut être enrichie de plus d'informations, en renvoyant la réponse à l'utilisateur.

Cette solution s'est avérée idéale car elle offre une certaine souplesse à l'entreprise, tout en étant plus économique que la migration de tous les services vers un seul langage de programmation, permettant de continuer à utiliser les services par l'intermédiaire d'un médiateur : **Middleware WSO2.**

Si nous prenons l'exemple présenté comme référence, la communication entre ces derniers est produite au moyen de services web sur le protocole HTTP.

> Cette solution s'est avérée idéale car elle offre une certaine souplesse à l'entreprise, tout en étant plus économique que la migration de tous les services à un seul langage de programmation, avec la possibilité de continuer à utiliser les services par l'intermédiaire d'un médiateur : **Middleware WSO2**

WSO2 A ÉTÉ INSTALLÉE?

Avant d'aborder l'architecture réalisée pour le client en question, il convient de noter que cette décision peut être personnalisée et est orchestrée en fonction des besoins d'intégration de chaque entreprise.

Dans le cas de ce client britannique, nous avons opté pour l'installation de l'ensemble de la suite WSO2, qui se compose de différents modules répondant à différentes problématiques :

- **WSO2 API Manager:** ce module permet à l'utilisateur de créer et de publier des API pour une utilisation interne (Intranet) ou externe (Extranet). En outre, d'autres utilisateurs peuvent s'abonner aux API par le biais d'applications créées dans cet outil. Cela se fait dans le magasin (store) où vous pouvez voir toutes les API qui ont été publiées précédemment.
- **WSO2 Identity Server (IS):** c'est un serveur d'identité. Il est utilisé pour gérer la signature unique (Single Sign On) (SSO), entre les fournisseurs de services et comme gestionnaire de jetons pour l'accès aux API (protocole Oauth2).
- **WSO2 API Publisher et Store:** ce sont des outils web utilisés pour créer, publier les API et s'abonner aux API. Ce module est inclus dans le gestionnaire (API Manager).
- **WSO2 ESB:** est le mandataire (proxy) utilisé pour assurer la communication entre les composants. Il convient de noter qu'il couvre davantage de fonctionnalités puisqu'il s'agit d'un BUS commercial, telles que: enrichir les messages, les rediriger vers différents points terminaux, jouer le rôle d'équilibreur, accéder à des bases de données et à tout service, par le biais des appels RestFul. Dans le cas de ce client, json a été utilisé pour le format du message.
- **WSO2 Message Broker:** ce module est utilisé pour gérer les files d'attente créées dans l'ESB. Ces files d'attente sont utilisées pour stocker les messages pendant un certain temps jusqu'à ce que le point final, auquel ils sont destinés, soit en mesure de les traiter.

Il est important de noter que pour offrir un meilleur service, le répartiteur de charge **nginx** a été utilisé pour les passerelles API et ESB. Ce répartiteur de charge est chargé de répartir les demandes entre les travailleurs (nœuds), de la passerelle (gateway) comme de l'ESB.

SCHÉMA DE L'ARCHITECTURE WSO2

Comme on peut le voir sur l'image suivante, **API Publisher** et **API Store** ont accès à l'Intranet. Un certain nombre d'utilisateurs disposant de « privilèges » d'éditeur peuvent créer et publier les API. Le reste des utilisateurs a accès au magasin (store) pour pouvoir enregistrer les applications avec lesquelles ils s'abonnent aux API.

Les API qui ont été créées sont déployées sur les passerelles qui reçoivent les demandes de l'Intranet. Il est important de noter que ce processus peut également être réalisé en utilisant les services de repos (Restful) de **l'API Manager.**

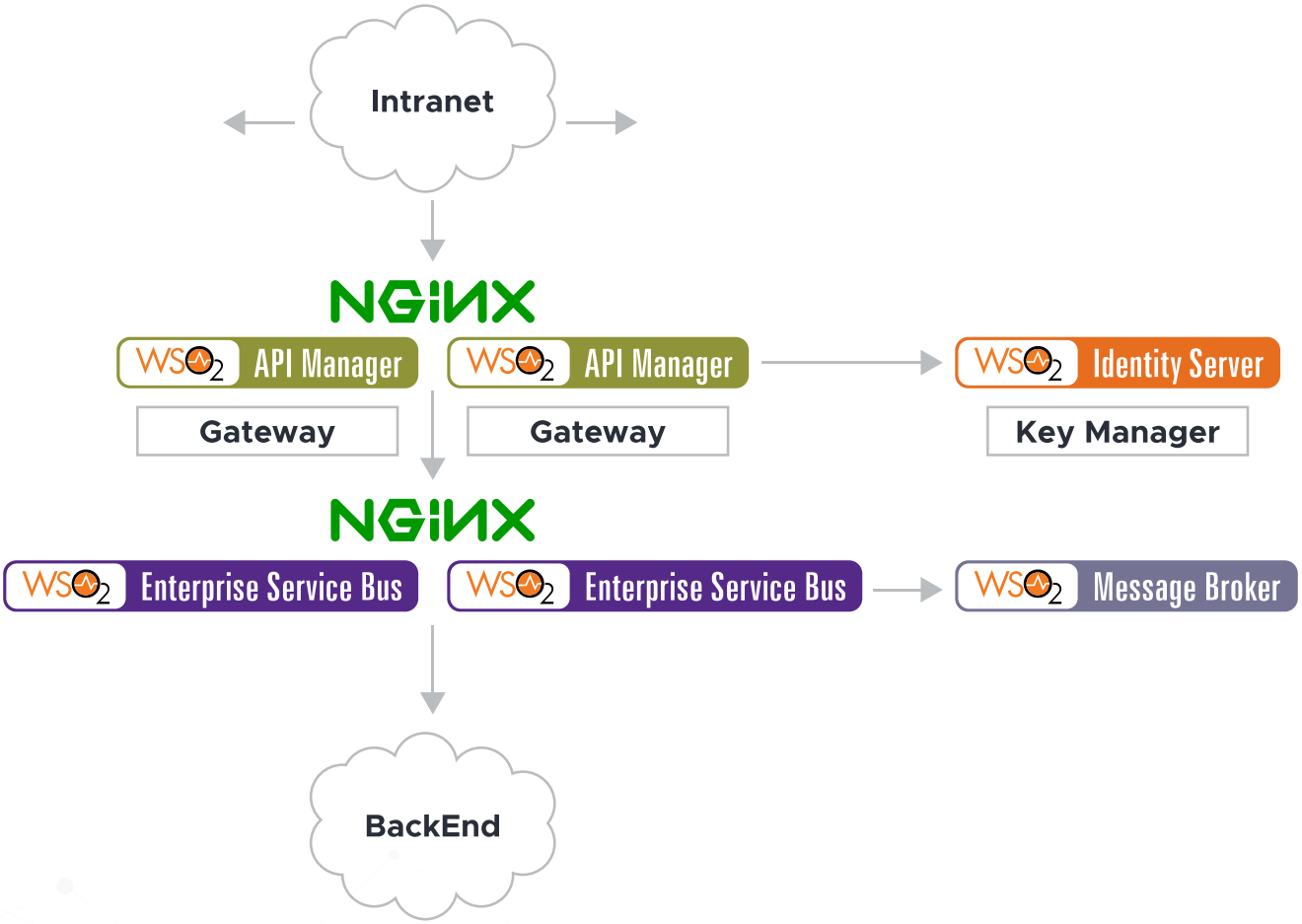


Diagramme de l'architecture du WSO2 réalisé

Toutes les API gèrent la sécurité avec le protocole Oauth2. Le service de ce protocole a été déployé dans le serveur d'identité (Identity Server) : **Responsable des clés** (Key Manager). C'est ce point final qui reçoit les demandes d'accès aux API et envoie les jetons d'accès, dans le cas où l'utilisateur et le mot de passe sont corrects pour chaque demande.

Il faut souligner que pour faciliter le déploiement des composants sur les différents serveurs, nous utilisons l'outil Puppet.

Suite à l'intégration, le gestionnaire de clé (Key Manager) a été déployé sur le serveur d'identité (Identity Server) (IS), qui se trouve à l'origine dans le gestionnaire d'API (API Manager). Il existe donc une série de tableaux qui doivent être répartis entre les deux chiffres, qui sont :

- **WSO2REG_DB:** Il contient des informations sur le journal des artéfacts, des données de configuration, des politiques et des détails sur les API.
- **WSO2UM_DB:** dans ce tableau sont stockées les autorisations et les rôles des différents utilisateurs.
- **WSO2AM_DB:** gère les données d'identité et les informations API, tels que les jetons Oauth et les clés d'accès API.

Il est important de souligner que ces programmes ont été déployés dans le cadre d'une grappe (cluster) avec MariaDB.

TYPES DE RÉPONSE : JSON OU XML

Un autre aspect à considérer dans le processus d'intégration du système avec le WSO2, était de renvoyer les messages d'erreur du gestionnaire de l'API (API Manager), dans **json**. Pour ce faire, plusieurs fichiers ont dû être modifiés :

_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>
```

Dans ce dossier, la valeur du bien **« error_message_type »** devrait être modifiée de **« application/xml »** à **« 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>
```

À la suite de cette modification, le message d'erreur généré par le gestionnaire d'API (API Manager) apparaît dans le format **json.**

À titre d'exemple, le message d'erreur au format json généré par le gestionnaire de l'API (API Manager) ressemblerait à ceci:

```
{
"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\""
}
}
```

Les changements sont effectués dans le nœud Gestionnaire et seront propagés au reste des travailleurs grâce au SVN (Subversion).

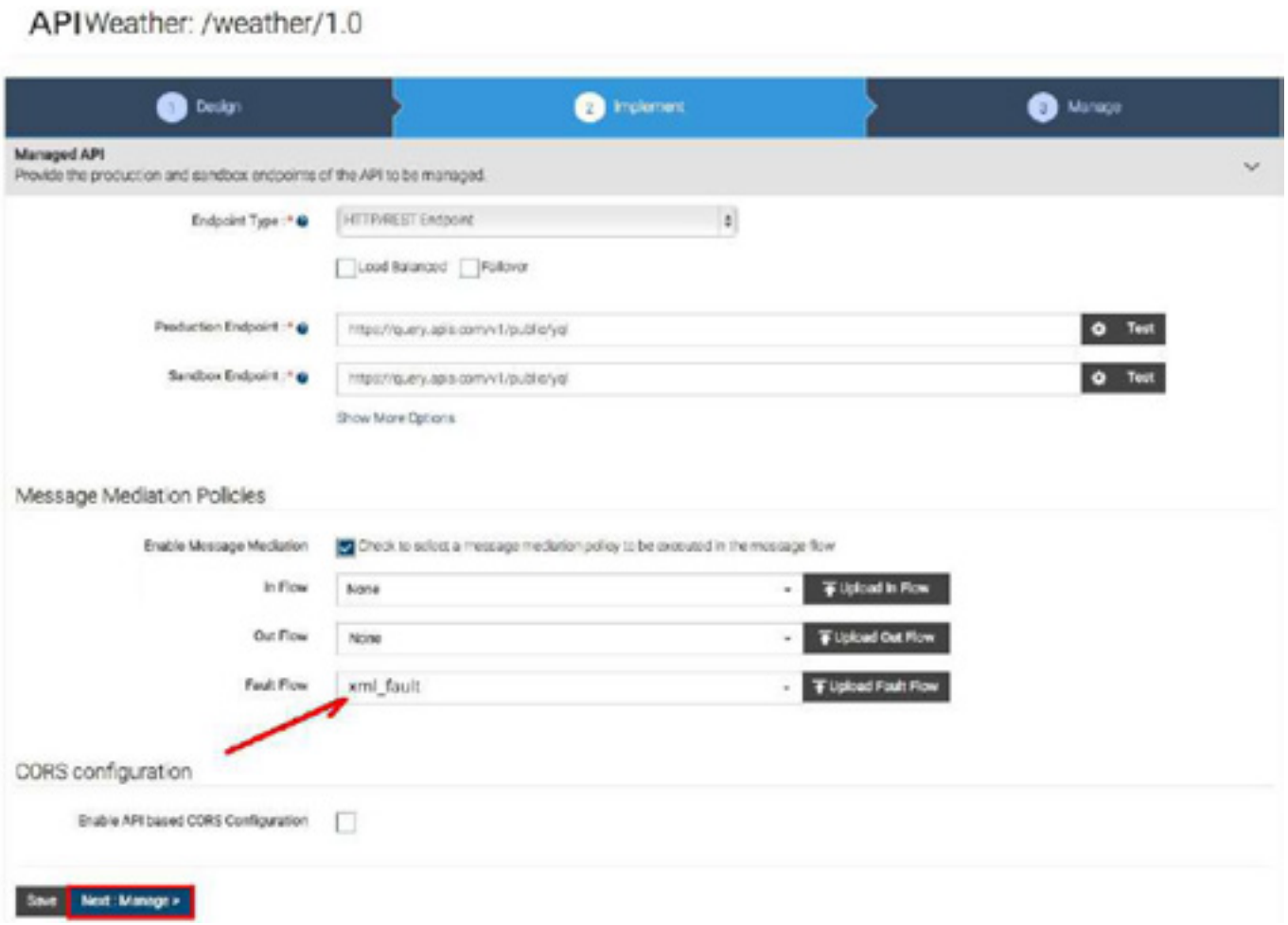
Si toute autre API souhaite envoyer des messages d'erreur en XML, il sera nécessaire de créer un flux dans le fichier **xml_fault**, comme suit :

```
<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>
```

Il est important de noter que le SVN (Subversion) n'agit pas sur ce répertoire, il est donc nécessaire d'apporter ce changement au gestionnaire et aux travailleurs manuellement. Enfin, le serveur doit être redémarré.

Lorsqu'une API est en cours de création ou en mode édition, vous pouvez sélectionner la séquence à exécuter en cas d'échec, afin de pouvoir sélectionner **xml_fault** de sorte que le message d'erreur se trouve dans **XML.**



ENVIRONNEMENT DE DÉVELOPPEMENT

En raison des caractéristiques du client particulier, Wiremock a été choisi comme simulateur des API HTTP, tandis que l'équipe de développement a réalisé tout le système principal (backend) des API. Ainsi, il a été possible de travailler en parallèle sur le développement des API dans le gestionnaire API (API Manager) et l'ESB.

Si vous voulez faire fonctionner le serveur Wiremock en écoutant sur le port 8081, et afficher le fichier journal à l'écran, ce serait : java -jar wiremock-1.57-standalone.jar --port 8081 -verbose

Comme nous savons que la meilleure façon de comprendre les choses passe par des exemples, nous proposons ici la configuration d'une API dans l'outil 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"
}
}
}
```

Comme nous l'avons vu dans le code, la requête envoyée au serveur fictif suit la méthode GET. Alors que l'URL doit répondre au patron « /Parties/Parties/[A-Za-z0-9]* », et que les paramètres lang=en et brand=any sont attendus.

La réponse qui en découle est un message ou une charge utile au format json, elle est enregistrée dans le fichier cp-Parties-en.json.

Une fois cela fait, si vous voulez tester l'API, vous pouvez utiliser la commande url, telle qu'elle apparaît dans l'exemple suivant :

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

Outre cette commande, une autre qui peut être très utile pour le fichier journal de cet outil serait :

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

Si vous pensiez avoir tout vu avec Wiremock, il présente encore bien d'autres avantages, car il permet également de cartographier l'URL de l'appel et d'envoyer directement la charge utile au format xml ou json. Dans ce cas, le système principal (backend) n'existe pas, c'est juste un serveur fictif.

Un autre exemple d'appel aux API via la passerelle API serait :

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

Là, le jeton d'accès apparaît, qui sera validé dans le serveur d'identité (Identity Server) (IS). Si l'appel est correct, il est redirigé vers l'ESB et de là, vers le système principal (backend) où il sera traité. La réponse qui est renvoyée est au format json.



RESSOURCE API NON PROTÉGÉE : PROTOCOLE OAUTH2

Dans le processus de mise en œuvre du WSO2, l'une des ressources de l'une des API devait être non protégée pour pouvoir travailler avec elle, sans la protection de Oauth2. Le problème que nous avons rencontré était que Gigya (le serveur d'identité de SAP), n'était pas en mesure d'envoyer des messages de notification en réponse à des événements utilisant ce protocole.

La façon de déprotéger une ressource dans le API Manager est très simple. Il suffit de cliquer sur la section **Manager.**

L'illustration suivante montre une liste des ressources de l'API. Le type d'authentification par défaut est « Application et Utilisateur d'application (Application User) ». Si nous sélectionnons cette valeur, une liste d'options possibles apparaît, parmi lesquelles nous devons choisir « Aucune ». Comme le type d'authentification est désormais « Aucun », l'utilisateur pourra appeler la ressource de l'API sans utiliser le jeton Oauth.

CONCLUSION

Intégrer des systèmes en différentes langues dans une entreprise de façon économique n'est plus impossible! WSO2 API Manager est le meilleur outil en source libre (Open Source) pour la gestion des API au niveau de l'entreprise.

Un autre avantage est qu'il est très facile à installer et à utiliser. En outre, il est possible de modifier son fonctionnement afin de l'adapter autant que possible aux besoins du client. Cela se fait par la configuration ou en utilisant un point d'extension comme interfaces.

La gestion des API est basée sur des rôles avec des autorisations différentes. Si nous nous concentrons spécifiquement sur cet exemple de réussite, nous avons également inclus le serveur d'identité (Identity Server) (IS), qui gère l'autorisation et l'authentification, et le WSO2 ESB, qui reçoit les demandes à partir de la passerelle de l'API (API Gateway), pour effectuer un traitement supplémentaire des messages, ce qui donne plus de flexibilité à l'outil.

Si vous avez des doutes ou si vous avez besoin de conseils, **n'hésitez pas à nous contacter**, nos conseillers se feront un plaisir de vous aider.

CONTACTEZ-NOUS!

Vous souhaitez améliorer les systèmes de votre entreprise? Consultez nos experts!

Demandez à nos consultants sans engagement. Nous vous aiderons à trouver la meilleure solution pour votre projet.

NOUS CONTACTER

ESPAGNE

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

contact@chakray.com

+34 955 252 520

ROYAUME-UNI

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

MEXIQUE

Calle Parral Nº 41
Colonia Condesa
Delegación Cuauhtémoc
CP 06140, Mexico

info.mexico@chakray.com

+52 55 5204 6581

PÉROU

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

info.peru@chakray.com

+51 1 644 9116

CANADA

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

info.canada@chakray.com

+1 (581) 700 03 75