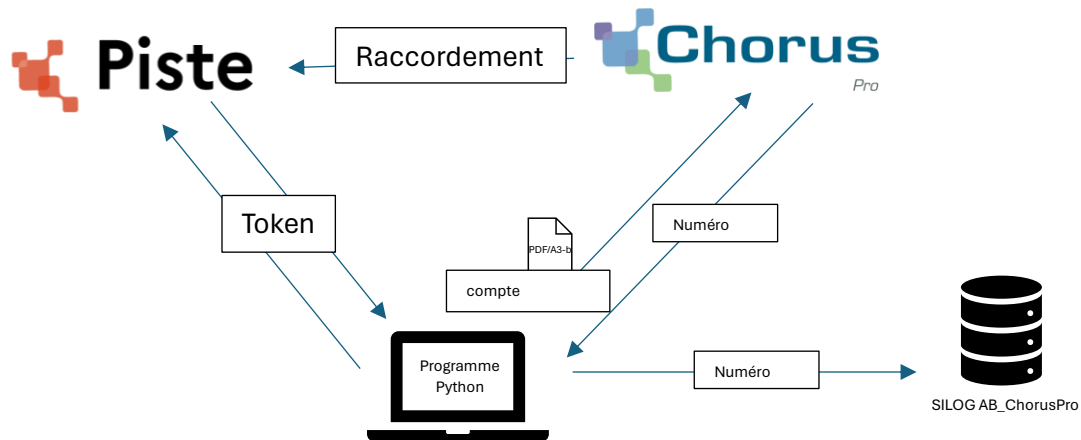
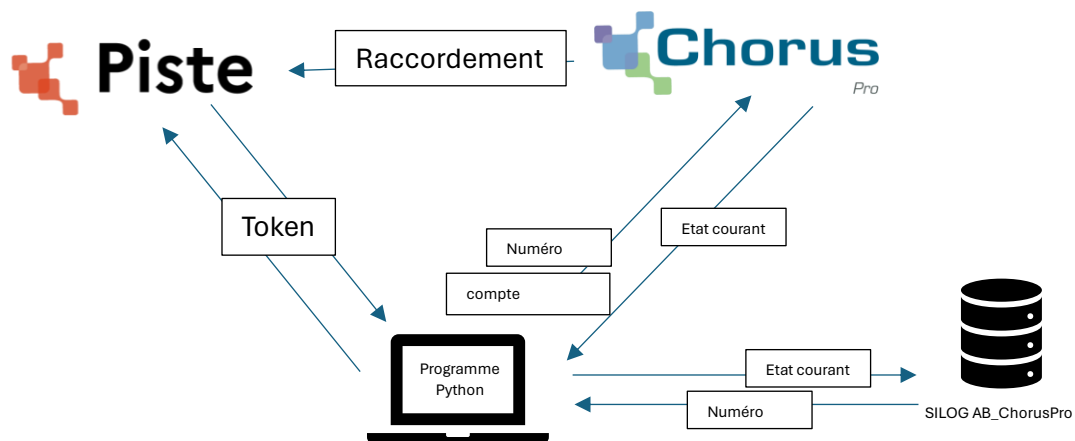


1) Mise en place d'un API

sendFact.py :



getEtatCoutant.py :



Prérequis :

`pip install requests`

`pip install logging`

`pip install dateTime`

`pip install PyPDF2`

`pip install pyodbc`

Microsoft SQL server management

1) Création de l'api sur PISTE

<https://developper.aife.economie.gouv.fr/>

- Créer un compte/se connecter
- Dans la page « applications » :

Applications

Les applications vous permettent de créer les identifiants de sécurité pour consommer les API protégées par une authentification.



Liste des applications

[Créer une application](#)

Puis Il faut remplir ce formulaire :

Général

Créer un compte Piste pour accéder aux API du service public.

Image:

Limite de 10%

Organisation:

Nom de l'application:

Description:

Téléphone:

Si le responsable d'application souhaite être joint par téléphone

Email:

Information sur la structure:

Responsable d'application:

☐ Activer l'application

[Sauvegarder l'application](#) [Annuler](#)

Dans ce cas on appelle l'API « sendFacture »

Liste des applications

[Créer une application](#)

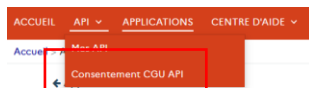
Activer Désactiver

Précédent 1

Recherche

2 Application(s)

<input type="checkbox"/>	Nom	Statut	Description	Environnement	Actions	Organisation
<input type="checkbox"/>	APP_SANDBOX_v1@allway.fr	Approved	Sandbox Application	SANDBOX	Consulter les métadonnées	Universelle
<input type="checkbox"/>	sendFacture	Approved	-	PRODUCTION	Consulter les métadonnées	Universelle



Sélectionnez les API

Valider mes choix CGU

Facture

Consentement CGU

<input type="checkbox"/>	Nom de l'API	Environnement	CGU
<input checked="" type="checkbox"/>	Factures	PROD	AIFE_ChorusPro_Factures_v1.pdf (pdf, 421.9Ko)
<input type="checkbox"/>	FacturesTravaux	PROD	AIFE_ChorusPro_FacturesTravaux_v1.pdf (pdf, 421.9Ko)
<input type="checkbox"/>	FacturesTravaux	SANDBOX	AIFE_ChorusPro_FacturesTravaux_v1.pdf (pdf, 421.9Ko)
<input type="checkbox"/>	Factures	SANDBOX	AIFE_ChorusPro_Factures_v1.pdf (pdf, 421.9Ko)

Lignes 1 à 4 sur 4 (Filtrer un maximum de 46)

Précédent 1 Suivant

10 Afficher lignes

<input checked="" type="checkbox"/>	Transverses	PROD	AIFE_ChorusPro_Transverses_v1.pdf (pdf, 421.9Ko)
-------------------------------------	-------------	------	--

On revient dans la page « Applications » :

Liste des applications

Créer une application

Activer Désactiver

Recherche

2 Application(s)

Précédent 1


<input type="checkbox"/>	Nom	Statut	Description	Environnement	Actions	Organisation
<input type="checkbox"/>	APP SANDBOX - valvalco.fr	Approved	SandBox Application	SANDBOX	Consulter les métriques	Universelle
<input type="checkbox"/>	sendFacture	Approved	-	PRODUCTION	Consulter les métriques	Universelle

Consultation de l'application :
sendFacture

Modifier l'application

Details Usage

Général



Nom:

sendFacture

Activée:

Oui

Description:

Statut:

Validée

Organisation:

Universelle

Créé par:

Service Informatique

Téléphone:

Créé le:

February 2, 2024

Email:

si@valcoy.fr

Environnement:

PRODUCTION

Information sur la structure:

Autre

Responsable d'application:

Oliver Foster

En bas de la page, cocher « factures » et « transverses » :

Sélectionner les API

Nom de l'API	Version	La description	Tag	Soumise	Soumise à validation	CTU
Capacité	1.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui
DURF Authentification	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
DURF Alternatives	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
DURF Données	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
DURF Données assemblees	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
DURF E-Contes	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
DURF Hébergement	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
DURF Opérations Economiques	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
DURF référentiels	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
E Signature	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui
EngagementUnidoc	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui
Engagements	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
Factures	1.0.0	-	PRODUCTION	<input checked="" type="checkbox"/>	Demander l'accès	Oui
FacturesPaiement	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui
Ingénierie Normes/Standards v2	2.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui
Ingénierie Normes v2	2.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui
JUDISME	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui
LogPhone	2.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui
Membrement des Organisations	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
Structures	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui
Tickets	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Non
Transverses	1.0.0	-	PRODUCTION	<input checked="" type="checkbox"/>	Demander l'accès	Oui
Utilisateurs	1.0.0	-	PRODUCTION	<input type="checkbox"/>	Demander l'accès	Oui

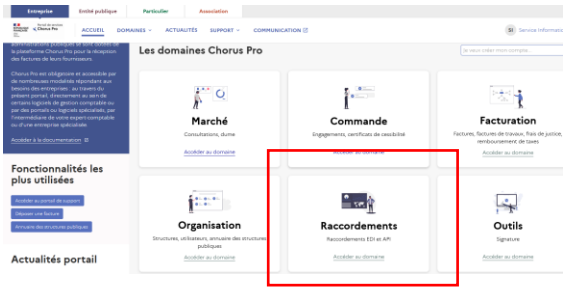
Appliquer les modifications

Annuler

2) Raccordement sur Chorus Pro

a. Raccordé l’api PISTE sur chorus pro

- Se connecter/créer un compte



Les applications du domaine Raccordements

Disponible

EDI

Accéder

Plus de détails

Disponible

API

Accéder

Plus de détails

Disponible

Suivi des flux

Accéder

Plus de détails

Compte technique

Gérer un compte technique nécessaire pour les appels en mode API.

Accéder

Accueil Recherche fiche EDI Créer Fiche EDI Emetteur Créer Fiche EDI Récepteur **Gérer raccordement API**

FIL D'ÉVÉNEMENTS

Aucun fil d'événements trouvé.

ACTUALITÉS

Aucune actualité trouvée.

Accueil Recherche fiche EDI Créer Fiche EDI Emetteur Créer Fiche EDI Récepteur **Gérer raccordement API**

RACCORDEMENTS API PISTE

Structure Etat courant

Exporter les résultats

Appliquer

Ligne(s) par page

Pagination 1 sur 1

Structure raccorder	Mode authentification	Nom de l'application PISTE	Etat courant	Actions
53744468900021 - VALO'IT	OAUTH	Test Facturation API	Supprimé	
53744468900021 - VALO'IT	OAUTH	sendFacture	Actif	

Déclarer un raccordement PISTE

Remplir ce formulaire :

DÉCLARER UN RACCORDEMENT API PISTE

INFORMATIONS GÉNÉRALES

Structure *

Nom de l'application PISTE *

Type d'utilisation *

CONTACT TECHNIQUE

Nom Prénom *

Téléphone *

Adresse électronique *

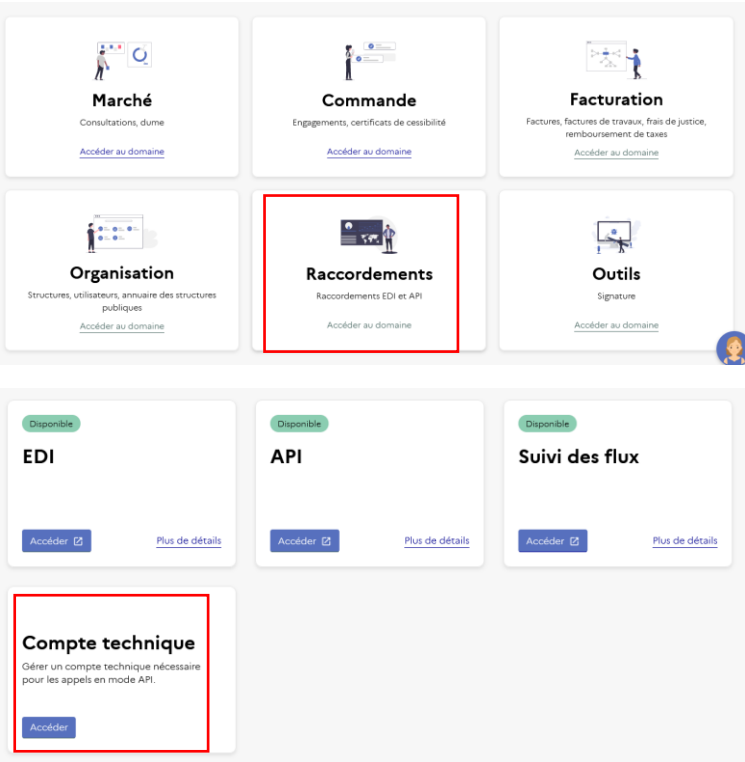
Fax

Annuler Valider

On a cela :

53744468900021 - VALO'IT	OAUTH	sendFacture	Actif	
--------------------------	-------	-------------	-------	--

b) Création d'un compte technique



Remplir ce formulaire et récupérer les données ci-dessous

Compte technique

Gérer un compte technique nécessaire pour les appels en mode API.

Attention, le compte technique ne sert que pour les services API, si vous ne les utilisez pas, la création est inutile.

Type de demande *

Création d'un compte technique

Choisir la structure *

53744468900021

Raison sociale

VAJOIT

Login

TECH_2_53744468900021@cpo.fr

E-mail du contact *

si@valoxy.fr

Mot de passe

ad@muwoGczvne5

Liste des comptes techniques

ID d'utilisateur	État	Identifiant de l'organisation	Date exp. Mot de passe	Contact	Création	Modification
TECH_1_53744468900021@cpo.fr	ACTIF	53744468900021	02/02/2025 10:56:38	si@valoxy.fr	04/07/2023 10:57:31	02/02/2024 10:56:38

Soumettre

3) Créer l'API

a. Récupérer le token (Authentication OAuth)

Sur PISTE :

On revient dans la page « Applications » :

Liste des applications Créer une application

Activer Désactiver

Recherche

Précédent 1

2 Application(s)

Nom	Statut	Description	Environnement	Actions	Organisation
APP_SANDBOX_olidenfoster@gmail.com	Approved	SandBox Application	SANDBOX	Consulter les métriques	Universelle
sendfacture	Approved	-	PRODUCTION	Consulter les métriques	Universelle

Récupérer ces données ci-dessous :

▼ Identifiants OAuth

Client ID	Type	Javascript Origins	URL de rappel	Créé	Secret key
c51e6583-e2d0-4caf-80fd-c2244c871542	Confidentiel	*		February 5, 2024 3:08 PM	Consulter le secret

De plus, dans « API » > « Mes APIs » > « Factures »

Explorer Usage

Vous devez remplir les conditions d'authentification avant de pouvoir essayer les méthodes ci-dessous.

APP_SANDBOX_olidenfoster@gmail.com - c51e6583-e2d0-4caf-80fd-c2244c871542

Schemes **https**

[Authorize](#)

Récupérer ces données ci-dessous :

Authorization URL: <https://sandbox-oauth.piste.gouv.fr/api/oauth/authorize>

Multi AuthN.OAuth (External)AccessToken (OAuth2, accessToken)

Application: APP_SANDBOX_olidenfoster@gmail.com

OAuth (External)

Token URL: <https://sandbox-oauth.piste.gouv.fr/api/oauth/token>

Flow: **accessToken**

Code python :

```
# Fonction pour obtenir le jeton d'accès
def getToken(client_id, client_secret, filePath, pathFactureOriginale, pathFichierXML):
    token_url = "https://sandbox-oauth.piste.gouv.fr/api/oauth/token"

    payload = {
        "grant_type": "client_credentials",
        "client_id": client_id,
        "client_secret": client_secret,
        "scope": "openid"
    }

    headers = {
        "Content-Type": "application/x-www-form-urlencoded"
    }

    try:
        # Demande du jeton d'accès à l'URL du jeton
        response = requests.post(token_url, data=payload, headers=headers, verify=True)

        # Lever une HTTPError pour les réponses incorrectes
        response.raise_for_status()

        # Extraction du jeton d'accès du JSON de réponse
        access_token = response.json().get("access_token")
        return access_token
    except requests.exceptions.RequestException as e:
        destination_path = os.path.join(errorPath, os.path.basename(pathFactureOriginale))
        shutil.move(pathFactureOriginale, destination_path)
        os.remove(filePath)
        os.remove(pathFichierXML)
        error_message = f"err {filePath}: \n Erreur dans la demande de jeton : {e}"
        print(error_message)
        logErr.error(error_message)
        log.error(error_message)
        return False
```

```
70 # Get access token
71 access_token = get_access_token(client_id, client_secret)
72 |
73 print(access_token)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

WJZUvb85UNoCnBcWMB2688yWgtu5CTrOSnpNL8d1HzQb6mRF8iS015

b. Envoyé le PDF à chorus pro

Dans « API » > « Mes APIs » > « Factures »

Mes API

Consulter mes API et leur documentation associée en fonction de mon organisation.

facture

2 résultat(s) trouvé(s) pour : facture

2 API affichée(s)

Factures

Published



Version : 1.0.0 Type : REST Tags : SANDBOX Environnement : SANDBOX

[Tester l'API](#) [Consulter les métriques](#)

POST `/v1/deposer/flux` La méthode `deposerFluxFacture` permet de déposer un fichier XML ou PDF/A3 permettant de renseigner les données nécessaires à la constitution d'un flux facture.

La méthode `deposerFluxFacture` permet de déposer un fichier XML ou PDF/A3 permettant de renseigner les données nécessaires à la constitution d'un flux facture.

Parameters Try it out

Name	Description
body	Example Value Model
object (body)	<pre>{ "avecSignature": true, "fichierFlux": "string", "idUtilisateurCourant": 1, "nomFichier": "string", "syntaxeFlux": "IN_DP_E1_UBL_INVOICE" }</pre>
Parameter content type	<input type="text" value="application/json;charset=utf-8"/>
CPRO-account required	Identifiant compte CPRO sous la forme 'login:password' encodé en base 64. Exemple : 'SjD6uW48cDf3c3dvcnQz'
string (header)	<input type="text" value="cpo-account"/>

Il n'est plus nécessaire de mettre le paramètre `idUtilisateur`

Tableau d'entrée

Attribut	Typage (tagger)	Format	Règles de gestion	Service fournissant la donnée	Cardinalité
idUtilisateur Courant	Nombre : identifiant technique de l'utilisateur courant dans le système CPP	integer	Cet identifiant permet d'identifier de manière unique l'utilisateur au sein du système CPP 2017		0-1
fichierFlux	String : fichier correspondant au flux de factures, encodé en base64	file			1
nomFichier	String : nom du fichier avec l'extension	varchar(200)			1
syntaxeFlux	String : choix de la syntaxe du fichier à déposer	varchar(24)	Enumération avec 6 valeurs possibles : - IN_DP_E1_UBL_INVOICE - IN_DP_E1_CII_16B - IN_DP_E2_UBL_INVOICE_MIN - IN_DP_E2_CPP_FACTURE_MIN - IN_DP_E2_CII_MIN_16B - IN_DP_E2_CII_FACTURX	recupererSyntaxeFlux	1

[https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiZub7Xg6uEAXWDU6QEHcfMCg8QFnoECBMQAQ&url=https%3A%2F%2Fcommunaute.chorus-pro.gouv.fr%2Fwp-content%2Fuploads%2F2020%2F04%2FSpecifications Externes Annexe Services API V5.00.pdf&usg=AOvVaw0lTyQrZ-69rGN6Kmtauvai&opi=89978449](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiZub7Xg6uEAXWDU6QEHcfMCg8QFnoECBMQAQ&url=https%3A%2F%2Fcommunaute.chorus-pro.gouv.fr%2Fwp-content%2Fuploads%2F2020%2F04%2FSpecifications%20Externes%20Annexe%20Services%20API%20V5.00.pdf&usg=AOvVaw0lTyQrZ-69rGN6Kmtauvai&opi=89978449)

Request URL :

`https://sandbox-api.piste.gouv.fr/cpro/factures/v1/deposer/flux`

Code python :

```
# Fonction pour appeler l'API
def sendFact(token, apiUrl, loginTechnique, password, filePath,
file_name,pathFactureOriginale,pathFichierXML):
    try:
        with open(filePath, "rb") as pdf_file:
            # Encoder le contenu du fichier PDF en base64
            encoded_string = base64.b64encode(pdf_file.read()).decode('utf-8')

            headers = {
                "Content-Type": "application/json; charset=UTF-8",
                "Authorization": "Bearer " + token,
                "cpro-account":
base64.b64encode(f"{loginTechnique}:{password}".encode()).decode()
            }

            # Construction de la charge utile JSON
            body_json = {
                "fichierFlux": encoded_string,
                "nomFichier": file_name, # <-- Le nom du fichier et pas le path
                "syntaxeFlux": "IN_DP_E2_CII_FACTURX",
                "avecSignature": False
            }

            # Effectuer une requête POST à l'API
            response = requests.post(apiUrl, headers=headers,
data=json.dumps(body_json), verify=True)

            # Lever une HTTPError pour les réponses incorrectes
            response.raise_for_status()

            # Afficher la réponse de l'API
            print(f"{file_name} \n Réponse de l'API :", response.text)
            log.info(f"{file_name} \n Réponse de l'API : {response.text}")
            return json.loads(response.text)
    except requests.exceptions.RequestException as e:
        destination_path = os.path.join(errorPath, os.path.basename(pathFactureOriginale))
        shutil.move(pathFactureOriginale, destination_path)
        os.remove(filePath)
        os.remove(pathFichierXML)
        error_message = f"err {file_name} \n Erreur dans l'envoi du fichier: {e}"
        print(error_message)
        logErr.error(error_message)
        log.error(error_message)
        return None
```

<pre>PS C:\temp\factures> & C:/Users/ofoster/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/temp/factures/test2.py Script exécuté le : 2024-02-06 11:36:15 Réponse de l'API : { "codeRetour" : 0, "libelle" : "GCU MSG 01 000", "numeroFluxDepot" : "CPP001111700000000304817", "dateDepot" : "2024-02-06", "syntaxeFlux" : "IN_DP_E2_CII_FACTURX" }</pre>	<table> <tr> <th>Code</th><th>Description</th></tr> <tr> <td>200</td><td>Succès de la requête</td></tr> <tr> <td colspan="2">Exemple Value Model</td></tr> <tr> <td colspan="2"> <pre>{ "codeRetour": 0, "dateDepot": "2024-02-06", "libelle": "TRA_M00_00000", "numeroFluxDepot": "xxxx", "syntaxeFlux": "IN_DP_E1_UBL_INVOICE" }</pre> </td></tr> </table>	Code	Description	200	Succès de la requête	Exemple Value Model		<pre>{ "codeRetour": 0, "dateDepot": "2024-02-06", "libelle": "TRA_M00_00000", "numeroFluxDepot": "xxxx", "syntaxeFlux": "IN_DP_E1_UBL_INVOICE" }</pre>	
Code	Description								
200	Succès de la requête								
Exemple Value Model									
<pre>{ "codeRetour": 0, "dateDepot": "2024-02-06", "libelle": "TRA_M00_00000", "numeroFluxDepot": "xxxx", "syntaxeFlux": "IN_DP_E1_UBL_INVOICE" }</pre>									

Sur chorus pro : Facturation > Facture émises > suivi des flux > accueil

ACCUEIL CONNECTÉ	MATÉLAS DE DONNÉES	DÉLÉGATIONS FACTURES REÇUES	DEMANDE DE REMBOURSEMENT TIC	FACTURES À VALIDER
FACTURES DE TRAVAIL	FACTURES ÉMISES	MÉMOIRES DE FRAIS DE JUSTICE	RACCORDEMENTS EDI ET API	SUIVI DES FLUX

Accueil

Rechercher

Tableau de bord

Déposer un flux

Recharger

LISTE DES FLUX ÉMIS

Exporter les résultats

5 lignes par page (14 ligne(s))

Émetteur	Code Appli Émetteur	Code Interface	N° flux	Date de dépôt	État courant	Actions
45739641818284 - MOE 45739641818284	CPP001	FSO117A	CPP001117000000000305052	13/02/2024 09:51:51	Rejeté	
45739641818284 - MOE 45739641818284	CPP001	FSO117A	CPP001117000000000305028	12/02/2024 16:19:49	Intégré	
45739641818284 - MOE 45739641818284	CPP001	FSO117A	CPP001117000000000305026	12/02/2024 16:07:31	Rejeté	
45739641818284 - MOE 45739641818284	CPP001	FSO117A	CPP001117000000000305022	12/02/2024 16:03:25	Rejeté	
45739641818284 - MOE 45739641818284	CPP001	FSO117A	CPP001117000000000305013	12/02/2024 15:57:15	Rejeté	

Pour que le code fonctionne en production, il faut changer les liens :

- <https://sandbox-oauth.piste.gouv.fr/api/oauth/token> -> <https://oauth.piste.gouv.fr/api/oauth/token>
- <https://sandbox-api.piste.gouv.fr/cpro/factures/v1/deposer/flux> -> <https://api.piste.gouv.fr/cpro/factures/v1/deposer/flux>

c. Stockées les factures envoyées dans la base de données

J'ai créé la table AB_ChorusPro :

dbo.AB_ChorusPro
Columns
Id (PK, bigint, not null)
NoFacture (varchar(30), not null)
NoFlux (varchar(30), not null)
DateEnvoi (datetime, not null)
Statut (varchar(20), null)
DateStatut (datetime, null)
FilePath (varchar(200), null)
MessageErr (varchar(200), null)

Pour récupérer le numéro de la facture :

```
def getNoFacture(pdf_path):
    text = ""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        for i in range(len(reader.pages)):
            page = reader.pages[i]
            text += page.extract_text()
    if re.findall(r"FA\d{2}/\d{5}",text):
        return re.findall(r"FA\d{2}/\d{5}",text)[0].replace(' ','')
```

```

def main():
    # Pour chaque fichier PDF dans le dossier spécifié
    for filePath in glob.glob(os.path.join(folder_path, "*.pdf")):
        # Obtenir le nom du fichier PDF
        nomFactureX = str(os.path.basename(filePath))
        noFacture = getNoFacture(filePath)
        if noFacture:
            # Construire les chemins vers les fichiers original et XML
            pathFactureOriginale = os.path.join(fichierOriginal,
            str(os.path.basename(filePath)).replace("FactX", ""))
            pathFichierXML = os.path.join(fichierOriginal,
            str(os.path.basename(filePath)).replace("FactX.pdf", ".xml"))
            # Obtenir le jeton d'accès
            token = getToken(client_id, client_secret, filePath)
            if token:
                # Envoyer le fichier à l'API
                api =
            sendFact(token, urlDeposerFlux, loginTechnique, password, filePath, nomFactureX, pathFactureOriginale, pathFichierXML)

            if api:
                # Déplacer le fichier vers le dossier "sent"
                dest_path = os.path.join(folder_path, "sent", nomFactureX)
                shutil.move(filePath, dest_path)
                # Obtenir le numéro de flux de dépôt
                numeroFluxDepot = api['numeroFluxDepot']
                # Obtenir la date de dépôt
                dateDepot = current_datetime

                # Insérer les données dans la base de données
                cursor.execute('INSERT INTO AB_ChorusPro (noFacture, noFlux, DateEnvoi,
                filePath)
                                VALUES (?, ?, ?, ?)',
                                (noFacture, numeroFluxDepot, dateDepot, dest_path))

                conn.commit()
                conn.close()

```

Résultat :

Results Messages								
	Id	NoFacture	NoFlux	DateEnvoi	Statut	DateStatut	FilePath	MessageErr
1	5	FA24/00203	CPP0011117000000000305526	2024-02-21 16:29:42.470	NULL	NULL	C:\temp\4\factures\factur-x\sent\fa24_00203Fact...	NULL

4) Récupérer l'état courant

ACCUEIL CONNECTÉ MATÉLAS DE DONNÉES DÉLÉGATIONS FACTURES REÇUES DEMANDE DE REMBOURSEMENT TIC FACTURES À VALIDER
FACTURES DE TRAVAUX FACTURES ÉMISES MÉMOIRES DE FRAIS DE JUSTICE RACCORDEMENTS EDI ET API **SUIVI DES FLUX**

Accueil Rechercher Tableau de bord Déposer un flux Recharger

LISTE DES FLUX ÉMIS Exporter les résultats

5 lignes par page (14 ligne(s))

Émetteur	Code Appli Émetteur	Code interface	N° Flux	Date de dépôt	État courant	Actions
45739641818284 - MCE 45739641818284	CPP001	FSOMITA	CPP001M7000000000305052	13/02/2024 09:51:54	Rejeté	
45739641818284 - MCE 45739641818284	CPP001	FSOMITA	CPP001M7000000000305028	12/02/2024 16:19:49	Intégré	
45739641818284 - MCE 45739641818284	CPP001	FSOMITA	CPP001M7000000000305026	12/02/2024 16:07:31	Rejeté	
45739641818284 - MCE 45739641818284	CPP001	FSOMITA	CPP001M7000000000305022	12/02/2024 16:03:25	Rejeté	
45739641818284 - MCE 45739641818284	CPP001	FSOMITA	CPP001M7000000000305013	12/02/2024 15:57:15	Rejeté	

On veut récupérer ces données et la raison du rejet.

On récupère le token :

```
# Fonction pour obtenir le jeton d'accès
def get_access_token(client_id, client_secret):
    token_url = "https://sandbox-oauth.piste.gouv.fr/api/oauth/token"

    payload = {
        "grant_type": "client_credentials",
        "client_id": client_id,
        "client_secret": client_secret,
        "scope": "openid"
    }

    headers = {
        "Content-Type": "application/x-www-form-urlencoded"
    }

    try:
        # Demande du jeton d'accès à l'URL du jeton
        response = requests.post(token_url, data=payload, headers=headers, verify=True)

        # Lever une HTTPError pour les réponses incorrectes
        response.raise_for_status()

        # Extraction du jeton d'accès du JSON de réponse
        access_token = response.json().get("access_token")
        return access_token
    except requests.exceptions.RequestException as e:
        print(f"Erreur dans la demande de jeton : {e}")
```

On va repérer les factures qui sont rejeté ou ne possède pas d'état courant :

```
def getFacturesSansEtat():
    server = 'PC-02095' #à modifier
    database = 'SILOG' #à modifier
    conn_str = f'DRIVER={{SQL
Server}};SERVER={server};DATABASE={database};Trusted_Connection=yes;'
    conn = pyodbc.connect(conn_str)
    cursor = conn.cursor()
    sql_query = "SELECT * FROM AB_ChorusPro WHERE Statut IS NULL OR Statut Like
'IN_REJETE'"

    try:
        # Execute the SQL query
        cursor.execute(sql_query)

        # Fetch all the rows returned by the query
        rows = cursor.fetchall()

        # Print the rows
        return rows
    except Exception as e:
        print("An error occurred:", e)
    finally:
        # Close the cursor and connection
        cursor.close()
        conn.close()
```

Puis on fait appel à l'api :

```
https://sandbox-api.piste.gouv.fr/cpro/transverses/v1/consulterCRDetaille
```

(enlevé « -sandbox » pour production)

```
# Fonction pour appeler l'API
def call_api(access_token, api_url, login_technique, password, numeroFluxDepot):

    headers = {
        "Content-Type": "application/json; charset=UTF-8",
        "Authorization": "Bearer " + access_token,
        "cpro-account": base64.b64encode(f"{{login_technique}}:{{password}}".encode()).decode()
    }

    # Construction de la charge utile JSON
    body_json = {
        "numeroFluxDepot": numeroFluxDepot,
        "syntaxeFlux": "IN_DP_E2_CII_FACTURX"
    }

    # Effectuer une requête POST à l'API
    response = requests.post(api_url, headers=headers,
data=json.dumps(body_json), verify=True)
    response_data = json.loads(response.text)
    return response_data
```

Puis on insère les données dans la table :

```
def setData(NoFlux,etatCourant,dateDepot,erreurs):
    server = 'PC-02095' #à modifier
    database = 'SILOG' #à modifier
    conn_str = f'DRIVER={{SQL
Server}};SERVER={server};DATABASE={database};Trusted_Connection=yes;'
    conn = pyodbc.connect(conn_str)
    cursor = conn.cursor()
    update_sql_query = "UPDATE AB_ChorusPro SET Statut = ?,DateStatut = ?, MessageErr = ?
WHERE NoFlux = ?"
    cursor.execute(update_sql_query, (etatCourant,dateDepot, erreurs, NoFlux))

    conn.commit()

    cursor.close()
    conn.close()
```

SELECT * FROM AB_ChorusPro

Results								
	Id	NoFacture	NoFlux	DateEnvoi	Statut	DateStatut	FilePath	MessageErr
1	5	FA24/00203	CPP0011117000000000305526	2024-02-21 16:29:42.470	IN_REJETE	2024-02-21 16:30:37.000	C:\temp\4\factures\factur-x\sentifa24_00203Fact...	Le numero de facture (balise : FA24/00203) existe d...
2	6	FA24/00202	CPP0011117000000000305528	2024-02-22 09:26:37.763	IN_REJETE	2024-02-22 09:27:31.000	C:\temp\4\factures\factur-x\sentifa24_00202Fact...	Le numero de facture (balise : FA24/00202) existe d...
3	7	FA24/00204	CPP0011117000000000305529	2024-02-22 11:05:22.993	IN_REJETE	2024-02-22 11:06:02.000	C:\temp\4\factures\factur-x\sentifa24_00204Fact...	Le numero de facture (balise : FA24/00204) existe d...
4	8	FA24/00205	CPP0011117000000000305530	2024-02-22 11:05:22.993	IN_REJETE	2024-02-22 11:06:03.000	C:\temp\4\factures\factur-x\sentifa24_00205Fact...	Le numero de facture (balise : FA24/00205) existe d...
5	9	FA24/00137	CPP0011117000000000305533	2024-02-22 11:16:30.370	IN_REJETE	2024-02-22 11:17:18.000	C:\temp\4\factures\factur-x\sent137FactX.pdf	Le code pays du fournisseur (FichierXml SupplyChai...
6	10	FA24/00136	CPP0011117000000000305541	2024-02-22 11:20:52.813	IN_INTEGRE	2024-02-22 11:21:25.000	C:\temp\4\factures\factur-x\sent136FactX.pdf	
7	11	FA24/00138	CPP0011117000000000305544	2024-02-22 11:22:54.500	IN_REJETE	2024-02-22 11:23:28.000	C:\temp\4\factures\factur-x\sent138FactX.pdf	Le Montant (balise FichierXml SupplyChainTradeTr...
8	12	FA24/00137	CPP0011117000000000305546	2024-02-22 11:25:54.233	IN_REJETE	2024-02-22 11:26:32.000	C:\temp\4\factures\factur-x\sent137FactX.pdf	Le code pays du fournisseur (FichierXml SupplyChai...
9	13	FA24/00137	CPP0011117000000000305547	2024-02-22 11:28:01.667	IN_REJETE	2024-02-22 11:28:35.000	C:\temp\4\factures\factur-x\sent137FactX.pdf	Le code pays du fournisseur (FichierXml SupplyChai...
10	14	FA24/00137	CPP0011117000000000305548	2024-02-22 11:29:27.963	IN_REJETE	2024-02-22 11:30:38.000	C:\temp\4\factures\factur-x\sent137FactX.pdf	Le code pays du fournisseur (FichierXml SupplyChai...
11	15	FA24/00137	CPP0011117000000000305549	2024-02-22 11:33:40.810	IN_REJETE	2024-02-22 11:34:43.000	C:\temp\4\factures\factur-x\sent137FactX.pdf	Le code pays du fournisseur (FichierXml SupplyChai...
12	16	FA24/00140	CPP0011117000000000305550	2024-02-22 11:58:35.847	IN_INTEGRE	2024-02-22 11:59:22.000	C:\temp\4\factures\factur-x\sent140FactX.pdf	
13	17	FA24/00141	CPP0011117000000000305552	2024-02-22 12:10:14.947	IN_REJETE	2024-02-22 12:10:37.000	C:\temp\4\factures\factur-x\sent141FactX.pdf	Le Montant (balise FichierXml SupplyChainTradeTr...

Code en entier :

sendFact.py

```
import requests
import base64
import json
import glob
import os
import shutil
import logging
from datetime import datetime
import re
import PyPDF2
import pyodbc

current_datetime = datetime.now()
# Formater la date et l'heure
formatted_datetime = current_datetime.strftime("%Y-%m-%d")

def getNoFacture(pdf_path):
    text = ""
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        for i in range(len(reader.pages)):
            page = reader.pages[i]
            text += page.extract_text()
    if re.findall(r"FA\d{2}/\d{5}", text):
        return re.findall(r"FA\d{2}/\d{5}", text)[0].replace(' ', '')

# Configure logging for the first log file
log = logging.getLogger('log')
log.setLevel(logging.INFO)
log_formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
file_handler = logging.FileHandler(fr'C:\temp\v4\factures\logs\{formatted_datetime}-log.log')
file_handler.setFormatter(log_formatter)
log.addHandler(file_handler)

# Configure logging for the second log file
logErr = logging.getLogger('logErr')
logErr.setLevel(logging.ERROR)
logErr_formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
fileErr_handler = logging.FileHandler(fr'C:\temp\v4\factures\logs\{formatted_datetime}-logErreur.log')
fileErr_handler.setFormatter(logErr_formatter)
logErr.addHandler(fileErr_handler)

# Fonction pour obtenir le jeton d'accès
def getToken(client_id, client_secret, filePath, pathFactureOriginale, pathFichierXML):
    token_url = "https://sandbox-oauth.piste.gouv.fr/api/oauth/token"

    payload = {
        "grant_type": "client_credentials",
        "client_id": client_id,
        "client_secret": client_secret,
        "scope": "openid"
    }

    headers = {
        "Content-Type": "application/x-www-form-urlencoded"
    }

    try:
```

```

# Demande du jeton d'accès à l'URL du jeton
response = requests.post(token_url, data=payload, headers=headers, verify=True)

# Lever une HTTPError pour les réponses incorrectes
response.raise_for_status()

# Extraction du jeton d'accès du JSON de réponse
access_token = response.json().get("access_token")
return access_token
except requests.exceptions.RequestException as e:
    destination_path = os.path.join(errorPath, os.path.basename(pathFactureOriginale))
    shutil.move(pathFactureOriginale, destination_path)
    os.remove(filePath)
    os.remove(pathFichierXML)
    error_message = f"err {filePath}: \n Erreur dans la demande de jeton : {e}"
    print(error_message)
    logErr.error(error_message)
    log.error(error_message)
    return False

# Fonction pour appeler l'API
def sendFact(token, apiUrl, loginTechnique, password, filePath,
file_name,pathFactureOriginale,pathFichierXML):
    try:
        with open(filePath, "rb") as pdf_file:
            # Encoder le contenu du fichier PDF en base64
            encoded_string = base64.b64encode(pdf_file.read()).decode('utf-8')

            headers = {
                "Content-Type": "application/json; charset=UTF-8",
                "Authorization": "Bearer " + token,
                "cpro-account":
base64.b64encode(f"{loginTechnique}:{password}".encode()).decode()
            }

            # Construction de la charge utile JSON
            body_json = {
                "fichierFlux": encoded_string,
                "nomFichier": file_name, # <-- Le nom du fichier et pas le path
                "syntaxeFlux": "IN_DP_E2_CII_FACTURX",
                "avecSignature": False
            }

            # Effectuer une requête POST à l'API
            response = requests.post(apiUrl, headers=headers,
data=json.dumps(body_json), verify=True)

            # Lever une HTTPError pour les réponses incorrectes
            response.raise_for_status()

            # Afficher la réponse de l'API
            print(f"{file_name} \n Réponse de l'API :", response.text)
            log.info(f"{file_name} \n Réponse de l'API : {response.text}")
            return json.loads(response.text)
    except requests.exceptions.RequestException as e:
        destination_path = os.path.join(errorPath, os.path.basename(pathFactureOriginale))
        shutil.move(pathFactureOriginale, destination_path)
        os.remove(filePath)
        os.remove(pathFichierXML)
        error_message = f"err {file_name} \n Erreur dans l'envoi du fichier: {e}"
        print(error_message)
        logErr.error(error_message)
        log.error(error_message)
        return None

```



```

# Identifiants et chemin du fichier
client_id = "c51e6583-e2d0-4caf-80fd-c2244c871542"
client_secret = "f3fbf9fd-939d-4ecd-8a1c-a3ffa2531305"
urlDeposerFlux = "https://sandbox-api.piste.gouv.fr/cpro/factures/v1/deposer/flux" #
enlever "sandbox-"
loginTechnique = "TECH_1_45739641818284@cpro.fr"
password = "cfa1viWvoybg)"

folder_path = r"C:\temp\v4\factures\factur-x"
errorPath = r'C:\temp\v4\factures\errorPdf'
fichierOriginal = r"C:\temp\v4\factures\archives"

server = 'PC-02095' #à modifier
database = 'SILOG' #à modifier
conn_str = f'DRIVER={{SQL
Server}};SERVER={server};DATABASE={database};Trusted_Connection=yes;'
conn = pyodbc.connect(conn_str)
cursor = conn.cursor()

def main():
    # Pour chaque fichier PDF dans le dossier spécifié
    for filePath in glob.glob(os.path.join(folder_path, "*.pdf")):
        # Obtenir le nom du fichier PDF
        nomFactureX = str(os.path.basename(filePath))
        noFacture = getNoFacture(filePath)
        if noFacture:
            # Construire les chemins vers les fichiers original et XML
            pathFactureOriginale = os.path.join(fichierOriginal,
str(os.path.basename(filePath)).replace("FactX",""))
            pathFichierXML = os.path.join(fichierOriginal,
str(os.path.basename(filePath)).replace("FactX.pdf",".xml"))
            # Obtenir le jeton d'accès
            token =
getToken(client_id,client_secret,filePath,pathFactureOriginale,pathFichierXML)
            if token:
                # Envoyer le fichier à l'API
                api =
sendFact(token,urlDeposerFlux,loginTechnique,password,filePath,nomFactureX,pathFactureOrigina
nale,pathFichierXML)
                if api:
                    # Déplacer le fichier vers le dossier "sent"
                    dest_path = os.path.join(folder_path, "sent", nomFactureX)
                    shutil.move(filePath, dest_path)
                    # Obtenir le numéro de flux de dépôt
                    numeroFluxDepot = api['numeroFluxDepot']
                    # Obtenir la date de dépôt
                    dateDepot = current_datetime

                    # Insérer les données dans la base de données
                    cursor.execute(''INSERT INTO AB_ChorusPro (noFacture, noFlux,
DateEnvoi, FilePath)
                                VALUES (?, ?, ?, ?)'',
                                (noFacture, numeroFluxDepot, dateDepot, dest_path))

                    conn.commit()
                    conn.close()

#pour exécuter le programme:
#main()

```

getEtatCourant.py

```
import requests
import base64
import json
import glob
import os
import shutil
import pyodbc
from datetime import datetime

def getFacturesSansEtat():
    server = 'PC-02095' #à modifier
    database = 'SILOG' #à modifier
    conn_str = f'DRIVER={{SQL
Server}};SERVER={server};DATABASE={database};Trusted_Connection=yes;'
    conn = pyodbc.connect(conn_str)
    cursor = conn.cursor()
    sql_query = "SELECT * FROM AB_ChorusPro WHERE Statut IS NULL OR Statut Like
'IN_REJETE'"

    try:
        # Execute the SQL query
        cursor.execute(sql_query)

        # Fetch all the rows returned by the query
        rows = cursor.fetchall()

        # Print the rows
        return rows
    except Exception as e:
        print("An error occurred:", e)
    finally:
        # Close the cursor and connection
        cursor.close()
        conn.close()

def setData(NoFlux,etatCourant,dateDepot,erreurs):
    server = 'PC-02095' #à modifier
    database = 'SILOG' #à modifier
    conn_str = f'DRIVER={{SQL
Server}};SERVER={server};DATABASE={database};Trusted_Connection=yes;'
    conn = pyodbc.connect(conn_str)
    cursor = conn.cursor()
    update_sql_query = "UPDATE AB_ChorusPro SET Statut = ?,DateStatut = ?, MessageErr = ?
WHERE NoFlux = ?"
    cursor.execute(update_sql_query, (etatCourant,dateDepot, erreurs, NoFlux))

    conn.commit()

    cursor.close()
    conn.close()

# Fonction pour obtenir le jeton d'accès
def get_access_token(client_id, client_secret):
    token_url = "https://sandbox-oauth.piste.gouv.fr/api/oauth/token"

    payload = {
        "grant_type": "client_credentials",
        "client_id": client_id,
        "client_secret": client_secret,
        "scope": "openid"
```

```

    }

    headers = {
        "Content-Type": "application/x-www-form-urlencoded"
    }

    try:
        # Demande du jeton d'accès à l'URL du jeton
        response = requests.post(token_url, data=payload, headers=headers, verify=True)

        # Lever une HTTPError pour les réponses incorrectes
        response.raise_for_status()

        # Extraction du jeton d'accès du JSON de réponse
        access_token = response.json().get("access_token")
        return access_token
    except requests.exceptions.RequestException as e:
        print(f"Erreur dans la demande de jeton : {e}")

# Fonction pour appeler l'API
def call_api(access_token, api_url, login_technique, password, numeroFluxDepot):

    headers = {
        "Content-Type": "application/json; charset=UTF-8",
        "Authorization": "Bearer " + access_token,
        "cpro-account": base64.b64encode(f"{login_technique}:{password}".encode()).decode()
    }

    # Construction de la charge utile JSON
    body_json = {
        "numeroFluxDepot": numeroFluxDepot,
        "syntaxeFlux": "IN_DP_E2_CII_FACTURX"
    }

    # Effectuer une requête POST à l'API
    response = requests.post(api_url, headers=headers,
data=json.dumps(body_json), verify=True)
    response_data = json.loads(response.text)
    return response_data

# Identifiants et chemin du fichier
client_id = "c51e6583-e2d0-4caf-80fd-c2244c871542"
client_secret = "f3fbf9fd-939d-4ecd-8a1c-a3ffa2531305"
api_url = "https://sandbox-api.piste.gouv.fr/cpro/transverses/v1/consulterCRDetaille" #
enlever "sandbox-"
login_technique = "TECH_1_45739641818284@cpro.fr"
password = "cfa1viWvoybg)"

folder_path = r"C:\temp\v4\factures\factur-x"
errorPath = r"C:\temp\v4\factures\errorPdf"
fichierOriginal = r"C:\temp\v4\factures\archives"

def main():
    token = get_access_token(client_id, client_secret)
    if token:
        lesFact = getFacturesSansEtat()
        if lesFact:
            for row in lesFact:
                api = call_api(token, api_url, login_technique, password, row[2])
                if api:
                    if "etatCourantDepotFlux" not in api:
                        continue
                    etatCourantDepotFlux = api['etatCourantDepotFlux']
                    numeroFluxDepot = api['nomFichier'][16:]

```

```

        dateHeureEtatCourantFlux =
datetime.strptime(api['dateHeureEtatCourantFlux'].replace('T','
').replace('+01:00','000'), "%Y-%m-%d %H:%M:%S.%f")
        errMessage = ""
        if "listeErreurDP" in api:
            listeErreursDp = api["listeErreurDP"]
            libelleErreursDp = [erreur.get("libelleErreurDP") for erreur in
listeErreursDp]

            for erreur in libelleErreursDp:
                errMessage += erreur + " "
            errMessage = errMessage[:200] if len(errMessage) > 200 else
errMessage

        setData(numeroFluxDepot,etatCourantDepotFlux,dateHeureEtatCourantFlux,e
rrMessage)

```