

Documentation Technique - Module WhatsApp Business API pour Odoo 16

Table des matières

1. [Vue d'ensemble](#)
 2. [Architecture](#)
 3. [Modèles de données](#)
 4. [Fonctionnalités principales](#)
 5. [Intégration API WhatsApp](#)
 6. [Webhooks](#)
 7. [Configuration](#)
 8. [Utilisation](#)
 9. [Exemples de code](#)
 10. [Dépannage](#)
-

Vue d'ensemble

Description

Le module **WhatsApp b-2-b** est une intégration complète de l'API WhatsApp Business Cloud pour Odoo 16. Il permet d'envoyer et de recevoir des messages WhatsApp directement depuis Odoo, avec support des messages texte, images, documents, templates, et messages interactifs avec boutons.

Version

- **Version du module :** 16.0.1.0.0
- **Version Odoo requise :** 16.0
- **Auteur :** Al Hussein
- **Licence :** LGPL-3

Dépendances

- **Modules Odoo :** `base`, `contacts`, `sale`, `account`
- **Bibliothèques Python :** `requests`

Fonctionnalités principales

- Configuration de compte WhatsApp Business
- Envoi de messages texte, images, documents
- Envoi de templates WhatsApp avec paramètres dynamiques
- Messages interactifs avec boutons (reply, URL)

- Réception de messages via webhook
 - Journalisation complète des messages
 - Gestion des conversations
 - Intégration avec les commandes (sale.order)
 - Intégration avec les factures (account.move)
 - Scénarios interactifs multi-étapes
 - Actions personnalisées sur boutons
 - Envoi automatique de notifications
 - Rappels automatiques de factures impayées
-

Architecture

Structure du module

```

api_whatsapp/
├── __init__.py
├── __manifest__.py
└── controllers/
    ├── __init__.py
    └── whatsapp_webhook.py      # Gestion des webhooks Meta
└── data/
    ├── whatsapp_button_action_examples.xml
    ├── whatsapp_cron_data.xml
    ├── whatsapp_invoice_download_action.xml
    ├── whatsapp_invoice_payment_actions.xml
    ├── whatsapp_invoice_validation_actions.xml
    ├── whatsapp_order_details_actions.xml
    ├── whatsapp_order_download_action.xml
    ├── whatsapp_order_validation_actions.xml
    └── whatsapp_order_validation_actions_v2.xml
    └── whatsapp_template_examples.xml
└── models/
    ├── __init__.py
    ├── whatsapp_config.py      # Configuration principale
    ├── whatsapp_message.py     # Journal des messages
    ├── whatsapp_conversation.py # Conversations
    ├── whatsapp_template.py    # Templates WhatsApp
    ├── whatsapp_send_message.py # Wizards d'envoi
    ├── whatsapp_send_partner_message.py
    ├── whatsapp_button_action.py # Actions sur boutons
    ├── whatsapp_interactive_scenario.py # Scénarios interactifs
    ├── whatsapp_cron.py        # Tâches planifiées
    ├── account_move_whatsapp.py # Intégration factures
    ├── sale_order_whatsapp.py  # Intégration commandes
    ├── res_partner_whatsapp.py # Extension partenaires
    └── res_config_settings.py
└── security/
    └── ir.model.access.csv      # Droits d'accès
└── views/
    └── whatsapp_config_views.xml

```

```
    └── whatsapp_message_views.xml
    └── whatsapp_conversation_views.xml
    └── whatsapp_template_views.xml
    └── whatsapp_send_message_views.xml
    └── whatsapp_button_action_views.xml
    └── whatsapp_interactive_scenario_views.xml
    └── account_move_whatsapp_views.xml
    └── sale_order_whatsapp_views.xml
    └── res_partner_whatsapp_views.xml
```

Flux de données



Modèles de données

1. whatsapp.config

Description : Configuration principale du compte WhatsApp Business.

Champs principaux :

- `name` : Nom de la configuration
- `is_active` : Configuration active (une seule active à la fois)
- `phone_number_id` : ID du numéro de téléphone WhatsApp (Meta)
- `access_token` : Token d'accès API
- `facebook_app_secret` : Secret de l'application Facebook (pour validation webhook)
- `verify_token` : Token de vérification webhook
- `auto_send_order_creation` : Envoi automatique à la création de commande
- `auto_send_unpaid_invoices` : Envoi automatique de rappels factures impayées
- `unpaid_invoice_days` : Nombre de jours avant envoi rappel
- `show_button_in_invoice` : Afficher bouton WhatsApp sur factures
- `show_button_in_order` : Afficher bouton WhatsApp sur commandes
- `show_button_in_partner` : Afficher bouton WhatsApp sur partenaires

Méthodes principales :

- `send_text_message()` : Envoi message texte
- `send_image_message()` : Envoi image
- `send_document_message()` : Envoi document

- `send_template_message()` : Envoi template
- `send_interactive_message()` : Envoi message interactif
- `send_text_to_partner()` : Envoi texte à un partenaire
- `get_active_config()` : Récupère la config active

2. whatsapp.message

Description : Journal de tous les messages WhatsApp (entrants et sortants).

Champs principaux :

- `direction` : `in` (entrant) ou `out` (sortant)
- `config_id` : Configuration utilisée
- `conversation_id` : Conversation associée
- `wa_message_id` : ID message WhatsApp (Meta)
- `phone` : Numéro de téléphone
- `contact_id` : Partenaire Odoo associé
- `content` : Contenu du message
- `message_type` : Type (text, image, document, interactive, etc.)
- `status` : Statut (sent, delivered, read, error, etc.)
- `raw_payload` : Payload JSON brut
- `raw_response` : Réponse API brute

Relations :

- Many2one → `whatsapp.config`
- Many2one → `whatsapp.conversation`
- Many2one → `res.partner`

Méthodes principales :

- `create_from_webhook()` : Création depuis webhook
- `action_reply_message()` : Répondre à un message

3. whatsapp.conversation

Description : Conversations WhatsApp avec les contacts.

Champs principaux :

- `name` : Identifiant de la conversation
- `phone` : Numéro de téléphone
- `contact_id` : Partenaire Odoo
- `contact_name` : Nom du contact
- `message_ids` : Messages de la conversation

- `message_count` : Nombre de messages (calculé)

Relations :

- One2many → `whatsapp.message`

4. whatsapp.template

Description : Métadonnées des templates WhatsApp approuvés.

Champs principaux :

- `name` : Nom du template (Meta)
- `language_code` : Code langue (fr, en, etc.)
- `category` : Catégorie (MARKETING, UTILITY, AUTHENTICATION)
- `status` : Statut (APPROVED, PENDING, REJECTED)
- `parameter_structure` : Structure JSON des paramètres
- `has_parameters` : A des paramètres (calculé)

Méthodes principales :

- `get_parameter_structure()` : Parse la structure JSON

5. whatsapp.button.action

Description : Actions personnalisées exécutées lors du clic sur un bouton.

Champs principaux :

- `name` : Nom de l'action
- `button_id` : ID du bouton (ex: `btn_validate_order`)
- `action_type` : Type (send_message, custom_python)
- `python_code` : Code Python personnalisé
- `response_message` : Message de réponse
- `active` : Actif

Méthodes principales :

- `execute_action()` : Exécute l'action

6. whatsapp.interactive.scenario

Description : Scénarios de messages interactifs avec réponses automatiques.

Champs principaux :

- `name` : Nom du scénario
- `initial_message` : Message initial avec boutons
- `button_1_id, button_1_title, button_1_response` : Bouton 1
- `button_2_id, button_2_title, button_2_response` : Bouton 2

- `button_3_id`, `button_3_title`, `button_3_response` : Bouton 3
- `button_X_next_scenario_id` : Scénario suivant (multi-étapes)
- `active` : Actif

Méthodes principales :

- `handle_button_click()` : Gère le clic sur un bouton
- `send_test_scenario()` : Envoie un test

7. Extension sale.order

Champs ajoutés :

- `x_whatsapp_creation_sent` : Message création envoyé
- `x_whatsapp_creation_sent_date` : Date envoi création
- `x_whatsapp_details_sent` : Détails envoyés
- `x_whatsapp_state_sent` : Message état envoyé
- `x_whatsapp_validated` : Validée via WhatsApp
- `x_whatsapp_rejected` : Rejetée via WhatsApp
- `x_show_whatsapp_button` : Afficher bouton (calculé)
- `x_has_phone` : A un téléphone (calculé)

Méthodes principales :

- `_send_whatsapp_creation_notification()` : Notification création
- `_send_whatsapp_state_notification()` : Notification changement état
- `action_send_order_details_whatsapp()` : Envoie détails commande

8. Extension account.move

Champs ajoutés :

- `x_whatsapp_invoice_sent` : Facture envoyée
- `x_whatsapp_invoice_sent_date` : Date envoi facture
- `x_whatsapp_residual_sent` : Message résiduel envoyé
- `x_whatsapp_unpaid_reminder_sent` : Rappel impayé envoyé
- `x_show_whatsapp_button` : Afficher bouton (calculé)

Méthodes principales :

- `_send_whatsapp_invoice()` : Envoie facture PDF
 - `_send_whatsapp_residual_notification()` : Notification montant résiduel
 - `_send_unpaid_invoice_reminder()` : Rappel facture impayée
 - `action_send_invoice_details_whatsapp()` : Envoie détails facture
-

Fonctionnalités principales

1. Envoi de messages

Messages texte simples

```
whatsapp_config = self.env['whatsapp.config'].get_active_config()
result = whatsapp_config.send_text_to_partner(
    partner_id=partner.id,
    message_text="Bonjour, votre commande est prête !"
)
```

Messages avec images

```
result = whatsapp_config.send_image_message(
    to_phone="+221781234567",
    image_link="https://example.com/image.jpg",
    caption="Photo du produit"
)
```

Messages avec documents (PDF)

```
result = whatsapp_config.send_document_message(
    to_phone="+221781234567",
    document_link="https://example.com/invoice.pdf",
    filename="facture_001.pdf",
    caption="Votre facture"
)
```

2. Templates WhatsApp

Les templates doivent être approuvés dans Meta Business Suite.

```
components = [
    {
        "type": "body",
        "parameters": [
            {"type": "text", "text": "John Doe"},
            {"type": "text", "text": "10000"}
        ]
    }
]

result = whatsapp_config.send_template_message(
    to_phone="+221781234567",
    template_name="order_confirmation",
    language_code="fr",
    components=components
)
```

3. Messages interactifs

Messages avec boutons (1 à 3 boutons maximum).

```
buttons = [
    {
        "type": "reply",
        "reply": {
            "id": "btn_validate_order_123",
            "title": "Valider"
        }
    }
]
```

```

        },
        {
            "type": "reply",
            "reply": {
                "id": "btn_cancel_order_123",
                "title": "Annuler"
            }
        }
    ]
}

result = whatsapp_config.send_interactive_message(
    to_phone="+221781234567",
    body_text="Voulez-vous valider cette commande ?",
    buttons=buttons
)

```

4. Scénarios interactifs

Création de conversations multi-étapes avec réponses automatiques.

Exemple : Validation de commande

1. Message initial : "Votre commande est prête. Valider ?"
2. Bouton "Valider" → Message : "Commande validée !"
3. Bouton "Annuler" → Message : "Commande annulée."

5. Actions personnalisées sur boutons

Exécution de code Python lors du clic sur un bouton.

Variables disponibles dans le code Python :

- **env** : Environnement Odoo
- **message** : Message WhatsApp reçu
- **contact** : Partenaire associé
- **button_id** : ID du bouton cliqué
- **_logger** : Logger Python
- **ValidationError** : Exception Odoo

Exemple :

```

# Dans whatsapp.button.action
order = env['sale.order'].search([...], limit=1)
if order:
    order.write({'state': 'sale'})
    message.config_id.send_text_to_partner(
        partner_id=contact.id,
        message_text="Commande validée !"
    )

```

6. Envoi automatique

Notifications de commande

- **Création** : Message avec boutons "Valider", "Annuler", "Voir détail"

- **Changement d'état** : Notification lors de la confirmation
- **Détails** : Envoi des détails avec produits et montants

Notifications de facture

- **Validation** : Envoi automatique du PDF facture
- **Montant résiduel** : Notification lors du changement
- **Rappels impayés** : Envoi automatique après X jours (configurable)

7. Rappels automatiques (Cron)

Tâche planifiée quotidienne pour envoyer des rappels de factures impayées.

Configuration :

- `auto_send_unpaid_invoices` : Activer/désactiver
 - `unpaid_invoice_days` : Nombre de jours avant rappel
-

Intégration API WhatsApp

Endpoint utilisé

```
POST https://graph.facebook.com/v21.0/{phone_number_id}/messages
```

Headers requis

```
{
  "Authorization": "Bearer {access_token}",
  "Content-Type": "application/json"
}
```

Format des requêtes

Message texte

```
{
  "messaging_product": "whatsapp",
  "recipient_type": "individual",
  "to": "+221781234567",
  "type": "text",
  "text": {
    "body": "Message texte"
  }
}
```

Message interactif

```
{
  "messaging_product": "whatsapp",
  "recipient_type": "individual",
  "to": "+221781234567",
  "type": "interactive",
  "interactive": {
    "type": "button",
    "body": {
      "text": "Répondre à ce message"
    }
  }
}
```

```

        "text": "Message avec boutons"
    },
    "action": {
        "buttons": [
            {
                "type": "reply",
                "reply": {
                    "id": "btn_yes",
                    "title": "Oui"
                }
            }
        ]
    }
}

```

Template

```
{
    "messaging_product": "whatsapp",
    "recipient_type": "individual",
    "to": "+221781234567",
    "type": "template",
    "template": {
        "name": "order_confirmation",
        "language": {
            "code": "fr"
        },
        "components": [
            {
                "type": "body",
                "parameters": [
                    {"type": "text", "text": "John Doe"}
                ]
            }
        ]
    }
}
```

Gestion des erreurs

Le module gère automatiquement les erreurs courantes :

- **131047** : Numéro non WhatsApp
- **131026** : Fenêtre 24h expirée (utiliser template)
- **131031** : Numéro non autorisé (mode test)
- **190** : Token invalide
- **131053** : Erreur upload média (URL localhost)

Webhooks

Configuration

1. **URL du webhook** : <https://votre-domaine.com/whatsapp/webhook>

2. **Token de vérification** : Défini dans la configuration
3. **Champs à écouter** : `messages`, `message_status`

Validation (GET)

Meta envoie une requête GET pour valider le webhook :

```
GET
/whatsapp/webhook?hub.mode=subscribe&hub.verify_token=TOKEN&hub.challenge=CHALLENGE
```

Le contrôleur retourne le `hub.challenge` si le token est valide.

Réception (POST)

Meta envoie les événements en POST :

```
{
  "object": "whatsapp_business_account",
  "entry": [
    {
      "id": "...",
      "changes": [
        {
          "value": {
            "messaging_product": "whatsapp",
            "metadata": {...},
            "contacts": [...],
            "messages": [...],
            "statuses": [...]
          }
        }
      ]
    }
  ]
}
```

Validation de signature (SHA256)

Le webhook valide la signature SHA256 pour sécuriser les requêtes :

```
import hmac
import hashlib

signature = request.headers.get('X-Hub-Signature-256', '').replace('sha256=', '')
expected_signature = hmac.new(
    app_secret.encode(),
    request.data,
    hashlib.sha256
).hexdigest()

if not hmac.compare_digest(signature, expected_signature):
    return "Invalid signature", 403
```

Types d'événements traités

1. **Messages entrants** :
 - Texte
 - Images
 - Documents
 - Messages interactifs (boutons)

- Templates
2. **Statuts de messages :**
- **sent** : Envoyé
 - **delivered** : Livré
 - **read** : Lu
 - **failed** : Échec
-

Configuration

1. Configuration WhatsApp

Menu : WhatsApp > Configuration

Paramètres requis :

- **Nom** : Nom de la configuration
- **Phone Number ID** : ID du numéro (Meta Business Suite)
- **Access Token** : Token d'accès API
- **Facebook App Secret** : Secret de l'application (validation webhook)
- **Verify Token** : Token de vérification webhook

Paramètres optionnels :

- **Envoi automatique commandes** : Activer notifications création
- **Envoi automatique factures impayées** : Activer rappels
- **Nombre de jours avant rappel** : Délai pour rappels
- **Affichage boutons** : Contrôler visibilité boutons WhatsApp

2. Configuration webhook Meta

1. Aller dans **Meta Business Suite > WhatsApp > Configuration**
2. Configurer l'URL : <https://votre-domaine.com/whatsapp/webhook>
3. Définir le **Verify Token** (identique à la config Odoo)
4. Sélectionner les champs : **messages, message_status**

3. Configuration Odoo

Paramètre système : `web.base.url`

⚠️ Important : Pour l'envoi de documents, l'URL doit être publique (pas `localhost`).

Utilisation

Envoi manuel de message

1. **Menu :** WhatsApp > Envoyer un message WhatsApp
2. Saisir le numéro ou sélectionner un partenaire
3. Saisir le message
4. Cliquer sur "Envoyer"

Envoi de template

1. **Menu :** WhatsApp > Envoyer un template WhatsApp
2. Sélectionner le template
3. Saisir les paramètres (JSON ou via interface)
4. Cliquer sur "Envoyer"

Envoi de message interactif

1. **Menu :** WhatsApp > Envoyer un message avec boutons
2. Sélectionner un scénario ou créer les boutons manuellement
3. Saisir le message
4. Cliquer sur "Envoyer"

Création de scénario interactif

1. **Menu :** WhatsApp > Scénarios interactifs
2. Créer un nouveau scénario
3. Définir le message initial
4. Configurer les boutons (1 à 3)
5. Définir les réponses automatiques
6. Optionnel : Lier à un scénario suivant

Création d'action de bouton

1. **Menu :** WhatsApp > Actions de boutons
2. Créer une nouvelle action
3. Définir l'ID du bouton (ex: `btn_validate_order`)
4. Choisir le type d'action :
 - **Envoyer message** : Message simple
 - **Code Python** : Code personnalisé
5. Saisir le code Python si nécessaire

Consultation des messages

1. **Menu :** WhatsApp > Messages
2. Filtrer par direction, type, statut
3. Cliquer sur un message pour voir les détails
4. Utiliser le bouton "Répondre" pour répondre

Consultation des conversations

1. **Menu :** WhatsApp > Conversations
 2. Voir la liste des conversations
 3. Cliquer pour voir les messages
-

Exemples de code

Exemple 1 : Envoi notification commande

```
def _send_order_notification(self):
    self.ensure_one()

    whatsapp_config = self.env['whatsapp.config'].get_active_config()
    if not whatsapp_config:
        return

    phone = self.partner_id.phone or self.partner_id.mobile
    if not phone:
        return

    message = f"Bonjour {self.partner_id.name},\n\n"
    message += f"Votre commande {self.name} a été créée.\n\n"
    message += f"Montant : {self.amount_total:.0f} F CFA"

    buttons = [
        {
            "type": "reply",
            "reply": {
                "id": f"btn_validate_order_{self.id}",
                "title": "Valider"
            }
        }
    ]

    result = whatsapp_config.send_interactive_message(
        to_phone=phone,
        body_text=message,
        buttons=buttons
    )
```

Exemple 2 : Envoi facture PDF

```
def _send_invoice_pdf(self):
    self.ensure_one()

    # Générer le PDF
```

```

    report =
self.env['ir.actions.report']._get_report_from_name('account.report_invoice')
pdf_content, _ = report._render_qweb_pdf(self.id)

# Créer attachment public
attachment = self.env['ir.attachment'].create({
    'name': f"{self.name}.pdf",
    'type': 'binary',
    'datas': base64.b64encode(pdf_content),
    'res_model': 'account.move',
    'res_id': self.id,
    'public': True,
})

# Générer URL publique
base_url = self.env['ir.config_parameter'].sudo().get_param('web.base.url')
pdf_url = f"{base_url}/web/content/{attachment.id}?download=true"

# Envoyer
whatsapp_config = self.env['whatsapp.config'].get_active_config()
result = whatsapp_config.send_document_message(
    to_phone=self.partner_id.phone,
    document_link=pdf_url,
    filename=f"{self.name}.pdf",
    caption=f"Votre facture {self.name}"
)

```

Exemple 3 : Action personnalisée sur bouton

```

# Dans whatsapp.button.action (python_code)
order_id = int(button_id.replace('btn_validate_order_', ''))
order = env['sale.order'].browse(order_id)

if order.exists():
    order.write({'state': 'sale'})

    response_msg = f"Commande {order.name} validée avec succès !"
    message.config_id.send_text_to_partner(
        partner_id=order.partner_id.id,
        message_text=response_msg
    )

    _logger.info("Commande %s validée via WhatsApp", order.name)

```

Exemple 4 : Scénario multi-étapes

```

# Scénario 1 : Demande validation
scenario_1 = {
    'name': 'Validation commande',
    'initial_message': 'Voulez-vous valider cette commande ?',
    'button_1_id': 'btn_yes',
    'button_1_title': 'Oui',
    'button_1_next_scenario_id': scenario_2.id, # Lien vers scénario 2
    'button_2_id': 'btn_no',
    'button_2_title': 'Non',
}

# Scénario 2 : Confirmation
scenario_2 = {
    'name': 'Confirmation paiement',

```

```
'initial_message': 'Choisissez votre méthode de paiement',
'button_1_id': 'btn_pay_wave',
'button_1_title': 'Payer avec Wave',
'button_2_id': 'btn_pay_orange',
'button_2_title': 'Payer avec Orange Money',
}
```

Dépannage

Problèmes courants

1. Messages non envoyés

Symptômes : Aucun message envoyé, pas d'erreur visible.

Solutions :

- Vérifier que la configuration est active (`is_active = True`)
- Vérifier le `phone_number_id` et `access_token`
- Vérifier les logs Odoo pour les erreurs API
- Vérifier que le numéro est autorisé (mode test)

2. Erreur 131026 (Fenêtre 24h)

Symptôme : "Fenêtre de 24h expirée"

Solution : Utiliser un template WhatsApp au lieu d'un message texte.

3. Erreur 131053 (Upload média)

Symptôme : "Media upload error", "localhost resolved address is private"

Solution :

- Configurer `web.base.url` avec une URL publique
- Le module détecte automatiquement localhost et utilise un bouton de téléchargement

4. Webhook non reçu

Symptômes : Messages entrants non enregistrés.

Solutions :

- Vérifier l'URL du webhook dans Meta Business Suite
- Vérifier le `verify_token`
- Vérifier que le serveur est accessible depuis Internet
- Vérifier les logs du contrôleur webhook

5. Signature webhook invalide

Symptôme : Erreur 403 sur webhook.

Solution : Vérifier que `facebook_app_secret` est correct dans la configuration.

6. PDF non envoyé

Symptômes : Message texte envoyé au lieu du PDF.

Solutions :

- Vérifier que le rapport de facture existe
- Vérifier les permissions sur `ir.attachment`
- Vérifier que `public=True` sur l'attachment
- Vérifier l'URL générée (pas localhost)

Logs utiles

Les logs Odoo contiennent des informations détaillées :

```
# Rechercher dans les logs
grep "WhatsApp" /var/log/odoo/odoo.log
grep "whatsapp" /var/log/odoo/odoo.log
```

Niveaux de log :

- **INFO** : Opérations normales
- **WARNING** : Problèmes non bloquants
- **ERROR** : Erreurs nécessitant attention

Tests

Test d'envoi simple

```
# Dans la console Odoo
config = env['whatsapp.config'].get_active_config()
result = config.send_text_message(
    to_phone="+221781234567",
    message_text="Test message"
)
print(result)
```

Test webhook

```
# Simuler un webhook
curl -X POST https://votre-domaine.com/whatsapp/webhook \
-H "Content-Type: application/json" \
-H "X-Hub-Signature-256: sha256=..." \
-d '{"object":"whatsapp_business_account","entry":[]}'
```

Sécurité

Bonnes pratiques

1. **Tokens** : Ne jamais exposer les tokens dans les logs
2. **Webhook** : Toujours valider la signature SHA256
3. **Permissions** : Limiter les droits d'accès aux modèles

4. **URLs publiques** : Utiliser HTTPS pour les URLs publiques
5. **Validation** : Valider tous les inputs utilisateur

Droits d'accès

Le fichier `security/ir.model.access.csv` définit les droits :

- **Administrateur** : Accès complet
 - **Utilisateur** : Accès lecture/écriture limité
 - **Public** : Aucun accès
-

Évolutions futures

Améliorations prévues

- [] Support des messages vocaux
 - [] Support des listes interactives
 - [] Support des catalogues produits
 - [] Interface de chat en temps réel
 - [] Statistiques et analytics
 - [] Support multi-comptes WhatsApp
 - [] Intégration avec d'autres modules Odoo
-

Support

Contact

- **Auteur** : Al Hussein
- **Module** : WhatsApp b-2-b
- **Version** : 16.0.1.0.0

Ressources

- [Documentation Meta WhatsApp Business API](#)
 - [Documentation Odoo](#)
-

Dernière mise à jour : 2025-11-20