

Documentation Fonctionnelle — Intégration Sweego Mail

Sommaire

1. Présentation du module	2
2. Prérequis	3
2.1. Côté Sweego	3
2.2. Côté Odoo	3
3. Installation	3
4. Configuration	4
4.1. Paramètres généraux Sweego	4
4.2. Suivi du quota email	4
4.3. Domaine de réception	5
4.4. Serveur d'envoi (SMTP Sweego)	5
4.4.1. Champs spécifiques Sweego	6
4.4.2. Plusieurs serveurs Sweego	6
4.5. Configuration des webhooks dans Sweego	7
4.5.1. Webhook Sortant (Outbound)	7
4.5.2. Webhook Entrant (Inbound)	7
5. Fonctionnement	7
5.1. Envoi d'un email	7
5.2. Résolution du domaine Reply-To	8
5.3. Suppression différée des emails	8
5.4. Gestion des bounces et emails en erreur	9
5.5. Réception des réponses (Inbound)	10
5.5.1. Chaîne de résolution du thread inbound	10
5.6. Pièces jointes inbound	11
5.7. Détection des boucles et réponses automatiques	11
5.8. Vérification des signatures webhook	12
5.9. Architecture de traitement outbound	12
6. Tâche planifiée — Synchronisation des logs	13
6.1. Rôle	13
6.2. Comportement	13
6.3. Paramétrage	13
7. Mécanisme du token Reply-To	13
7.1. Format	13
7.2. Décodage	14
8. Champs techniques ajoutés	14

8.1. Sur <code>ir.mail_server</code>	14
8.2. Sur <code>mail.mail</code>	14
8.2.1. Méthodes sur <code>mail.mail</code>	15
8.3. Sur <code>mail.message</code>	15
8.4. Sur <code>mail.notification</code>	15
8.5. Dans <code>ir.config_parameter</code>	16
9. Endpoints webhook	16
10. Dépannage	16
10.1. Les webhooks sont rejetés (401)	16
10.2. Les réponses n'apparaissent pas dans le chatter	16
10.3. Les pièces jointes inbound n'apparaissent pas	16
10.4. Les emails ne sont pas supprimés après livraison	17
10.5. L'enveloppe reste grise après un bounce	17
10.6. Les réponses automatiques (OOO) n'apparaissent pas dans le chatter	17
10.7. La tâche planifiée ne traite aucun email	17
10.8. Le serveur refuse de se créer	18
11. Glossaire	18

Module Odoo 14 — `mangono_sweego_mail`

Version	1.0
Date	Septembre 2026
Auteur	Mangono
Compatibilité	Odoo 14

1. Présentation du module

Le module `mangono_sweego_mail` est une extension Odoo 14 qui intègre [Sweego](#), plateforme française d'envoi de messages transactionnels, au cœur du système de messagerie d'Odoo.

Il apporte cinq fonctionnalités principales :

- Envoi des emails Odoo via l'infrastructure SMTP de Sweego
- Traitement des réponses email de vos clients directement dans le chatter des documents Odoo (commandes, factures, etc.)
- Routage optionnel des réponses automatiques (OOO, règles Exchange) dans le chatter du document concerné
- Suppression fiable des emails de la file d'attente uniquement après confirmation de livraison par Sweego, via webhook ou tâche planifiée

- Signalement des emails en erreur (bounce) avec mise à jour du statut dans le chatter (enveloppe rouge)

Sweego est une plateforme d'envoi email française offrant un relais SMTP performant, des statistiques de délivrabilité et des webhooks pour le suivi des envois.

2. Prérequis

2.1. Côté Sweego

- Un compte Sweego actif (<https://app.sweego.io>)
- Un domaine d'envoi vérifié et configuré dans Sweego
- Un domaine inbound configuré avec les MX pointant vers inbound.sweego.co
- Les identifiants SMTP fournis par Sweego
- Les secrets de signature webhook générés depuis le tableau de bord Sweego
- La clé API Sweego (section **API** du tableau de bord)
- L'UUID client Sweego (visible dans les paramètres du tableau de bord)

2.2. Côté Odoo

- Odoo 14 avec le module [mail](#) installé
- Le module [mangono_sweego_base](#) installé (fournit la clé API et l'UUID client partagés)
- Le domaine de réception renseigné dans le paramètre système [mail.catchall.domain](#) (**Paramètres** → **Technique** → **Paramètres** → **Paramètres système**)
- Accès administrateur à Odoo
- L'URL publique de votre instance Odoo (accessible par Sweego pour les webhooks)

3. Installation

Copiez les dossiers [mangono_sweego_base](#) et [mangono_sweego_mail](#) dans votre répertoire d'addons Odoo, puis installez [mangono_sweego_mail](#) depuis le menu **Paramètres** → **Activer le mode développeur** → **Applications**. [mangono_sweego_base](#) sera installé automatiquement en tant que dépendance.

Structure du module

```
mangono_sweego_mail/
├── models/
│   ├── ir_mail_server.py    ← Serveur sortant, logique token, cron
│   ├── mail_mail.py        ← Suppression différée, gestion bounces
│   ├── mail_message.py     ← Champ sweego_message_id
│   ├── mail_notification.py ← Confirmation de livraison par destinataire
│   └── res_config_settings.py ← Suivi du quota email (Paramètres → Sweego)
```

```

├── controllers/
│   └── webhook.py                ← Endpoints outbound + inbound
├── data/
│   └── ir_cron.xml              ← Tâche planifiée de synchronisation
├── views/
│   ├── ir_mail_server.xml      ← Formulaire étendu
│   └── res_config_settings.xml ← Section quota email dans les paramètres
└── i18n/
    └── fr.po                   ← Traductions françaises

mangono_sweego_base/           ← Module de base partagé (dépendance)
├── models/
│   └── res_config_settings.py  ← Clé API et UUID client
├── views/
│   └── res_config_settings.xml
└── i18n/
    └── fr.po

```

4. Configuration

4.1. Paramètres généraux Sweego

Les paramètres de connexion à l'API Sweego sont gérés par le module `mangono_sweego_base` et partagés entre tous les modules Sweego installés.

Accédez à **Paramètres** → **Sweego** pour renseigner ces paramètres.

Champ	Description
Clé API Sweego	Clé API disponible dans votre tableau de bord Sweego (section API). Utilisée pour les appels à l'API REST : téléchargement des pièces jointes inbound et synchronisation des logs de livraison.
UUID Client Sweego	UUID client visible dans votre tableau de bord Sweego. Requis pour construire les URLs d'appel à l'API REST (téléchargement des pièces jointes inbound).



Ces paramètres sont stockés dans `ir.config_parameter` sous les clés `mangono_sweego.api_key` et `mangono_sweego.client_uuid`. Ils sont globaux à l'instance Odoo et partagés entre tous les modules Sweego installés.

4.2. Suivi du quota email

La section **Paramètres** → **Sweego** affiche le quota d'emails de votre compte client, récupéré en temps réel via **GET** https://api.sweego.io/clients/{uuid_client}/billing/quotas à l'ouverture des paramètres.

Sont affichés le nombre d'emails consommés sur la période, le quota total, le quota de burst

(dépassement toléré) ainsi que la période de comptage (par exemple `month`). En cas d'échec de l'appel (clé API ou UUID client manquant, erreur réseau), un message d'avertissement est affiché à la place.

4.3. Domaine de réception

Le domaine inbound utilisé pour le routage des réponses email est lu dans le paramètre système `mail.catchall.domain`. Il n'est pas à saisir sur le serveur d'envoi.

Pour le renseigner : **Paramètres** → **Technique** → **Paramètres** → **Paramètres système**, clé `mail.catchall.domain`.



Sans ce paramètre, aucune adresse `reply+` n'est posée sur les emails sortants : les réponses ne reviennent pas dans le chatter. Un warning est alors écrit dans les logs à chaque envoi.



Odoo 14 ne connaît qu'un seul domaine de réception par base — le modèle `mail.alias.domain` et le champ `company_id.alias_domain_id`, qui permettent un domaine par société, n'existent qu'à partir de la 17.0. Toutes les sociétés d'une même base partagent donc le même domaine inbound.

4.4. Serveur d'envoi (SMTP Sweego)

Accédez à **Paramètres** → **Technique** → **Email** → **Serveurs d'envoi**, puis créez un nouveau serveur.

Champ	Valeur / Description
Nom	Sweego (ou tout autre nom descriptif)
Mangono Sweego SMTP	Cocher cette case dans la section Sweego settings
Serveur SMTP	<code>smtp.sweego.io</code> — à saisir manuellement
Port	<code>2525</code> (rempli automatiquement à la coche)
Chiffrement	<code>STARTTLS</code> (rempli automatiquement)
Nom d'utilisateur	Votre identifiant SMTP Sweego (obligatoire)
Mot de passe	Votre mot de passe SMTP Sweego



En cochant **Mangono Sweego SMTP**, le port (2525) et le chiffrement (STARTTLS) sont remplis automatiquement. L'hôte SMTP (`smtp.sweego.io`) reste à saisir manuellement.



Odoo 14 n'a pas de champ **Authentication** sur le serveur d'envoi (`smtp_authentication` n'apparaît qu'en 17.0) : un serveur Sweego est désigné par cette case à cocher, `sweego_enabled`.

4.4.1. Champs spécifiques Sweego

Champ	Description
URL webhook entrant Sweego	Calculée automatiquement (<code>web.base.url + /sweego/webhook/inbound</code>), en lecture seule avec un bouton pour la copier — à coller dans le champ URL du webhook Entrant côté Sweego.
URL webhook sortant Sweego	Calculée automatiquement (<code>web.base.url + /sweego/webhook/outbound</code>), en lecture seule avec un bouton pour la copier — à coller dans le champ URL du webhook Sortant côté Sweego.
Secret webhook sortant Sweego	Secret HMAC fourni par Sweego pour vérifier la signature des webhooks outbound. Copier tel quel depuis le tableau de bord Sweego (encodé en base64). Sert également à signer les tokens Reply-To.
Secret webhook entrant Sweego	Secret HMAC fourni par Sweego pour vérifier la signature des webhooks inbound. Distinct du secret sortant.
Suppression différée	Si coché, les emails ne sont supprimés qu’après confirmation de livraison par Sweego (webhook ou tâche planifiée).
Afficher les réponses auto dans le chatter	Si coché, les réponses automatiques reçues via le webhook inbound (OOO, règles de boîte aux lettres Exchange...) sont postées dans le chatter du document concerné. Désactivé par défaut.



Les URL affichées dépendent du paramètre système **URL de base** (`web.base.url`, **Paramètres** → **Technique** → **Paramètres** → **Paramètres système**). Vérifiez qu’il correspond bien à l’URL publique de votre instance avant de les copier dans Sweego.

4.4.2. Plusieurs serveurs Sweego

Vous pouvez déclarer plusieurs serveurs Sweego actifs, chacun avec ses propres secrets webhook (inbound et outbound) — ceux fournis par Sweego pour le domaine concerné.

Odoo 14 ne dispose pas du champ **Filtrage FROM** (`from_filter`, apparu en 15.0) ni de la sélection de serveur par domaine d’expéditeur. Le serveur retenu pour un envoi est donc :

- celui renseigné sur l’email lui-même (`mail.mail.mail_server_id`), posé par exemple par un modèle de mail ;
- à défaut, le serveur actif de plus petite **Priorité** (`sequence`), tous serveurs confondus.

Le module n’ajoute son en-tête de suivi et son token Reply-To que si ce serveur-là est un serveur Sweego : un serveur non-Sweego plus prioritaire, ou imposé sur l’email, l’emporte et l’email part sans marquage Sweego (voir *Envoi d’un email*).



Avec plusieurs serveurs Sweego actifs et aucun `mail_server_id` sur les emails, c'est toujours le plus prioritaire qui envoie. Pour router un envoi vers un autre serveur Sweego, il faut le désigner explicitement sur le modèle de mail concerné.



Côté réception (webhooks et décodage du token Reply-To), aucune configuration supplémentaire n'est nécessaire : le module identifie automatiquement le bon serveur en essayant le secret de chacun — voir *Vérification des signatures webhook* et *Mécanisme du token Reply-To*.

4.5. Configuration des webhooks dans Sweego

Connectez-vous à votre tableau de bord Sweego et configurez les deux webhooks suivants.



Les URL ci-dessous sont aussi affichées directement sur le serveur d'envoi Sweego dans Odoo (champs **URL webhook entrant/sortant Sweego**, avec bouton pour les copier) — pas besoin de les reconstruire à la main.

4.5.1. Webhook Sortant (Outbound)

Paramètre	Valeur
URL	https://votre-odoo.com/sweego/webhook/outbound
Événements	<code>email_sent</code> , <code>delivered</code> , <code>soft-bounce</code> , <code>hard-bounce</code> (au minimum)
Secret	Copier ici et reporter dans Secret webhook sortant Sweego dans Odoo

4.5.2. Webhook Entrant (Inbound)

Paramètre	Valeur
URL	https://votre-odoo.com/sweego/webhook/inbound
Événements	<code>email_inbound</code>
Secret	Copier ici et reporter dans Secret webhook entrant Sweego dans Odoo



Les secrets webhook sont encodés en base64 par Sweego. Copiez-les tels quels dans les champs Odoo — le module se charge du décodage avant le calcul HMAC-SHA256. Les secrets outbound et inbound sont distincts et indépendants.

5. Fonctionnement

5.1. Envoi d'un email

Lorsqu'un utilisateur envoie un email depuis Odoo (devis, facture, bon de commande, message

depuis le chatter...), le module intercepte l'envoi et injecte automatiquement deux éléments dans l'email :

- Un en-tête **X-Message-Id** contenant l'identifiant interne Odoo du message. Cet identifiant est renvoyé par Sweego dans ses webhooks et logs, et permet de retrouver le message correspondant.
- Un **Reply-To** personnalisé au format `reply+TOKEN@domaine-de-réception`, où le token encode de façon sécurisée le modèle Odoo et l'identifiant de l'enregistrement (ex : `sale.order#23`). Le domaine est celui du paramètre `mail.catchall.domain`.



Ces deux éléments ne sont ajoutés que si le serveur effectivement choisi par Odoo pour cet envoi — `mail_server_id` de l'email, à défaut le serveur actif le plus prioritaire, cf. *Plusieurs serveurs Sweego* — est bien un serveur Sweego. Si un autre serveur (Sweego ou non) est sélectionné à la place, l'email est envoyé normalement, sans en-tête ni token Sweego.



Sweego peut ré-encoder l'en-tête **X-Message-Id** en RFC 2047 (ex : `=?utf-8?q?...?=`) lorsque sa longueur dépasse 78 caractères — ce qui arrive systématiquement avec des domaines de réception basés sur un UUID. Le module décode automatiquement cet encodage lors de la réception des webhooks et des logs de synchronisation.

5.2. Résolution du domaine Reply-To

Le domaine inbound utilisé dans l'adresse Reply-To est lu à l'envoi dans le paramètre système `mail.catchall.domain`. Le nom affiché devant l'adresse est le `display_name` de l'enregistrement source (ex : `"S00014 - Benoit" <reply+...@exemple.fr>`).

Si le paramètre est vide, un warning est loggé et le Reply-To reste celui d'Odoo par défaut — les réponses ne seront pas routées dans le chatter.



Le serveur Sweego réellement utilisé pour l'envoi et le domaine inséré dans le Reply-To sont déterminés par deux mécanismes indépendants. Ce n'est pas bloquant s'ils ne sont pas alignés : au moment du décodage du token, le module essaie le secret sortant de tous les serveurs Sweego actifs (voir *Mécanisme du token Reply-To*).

5.3. Suppression différée des emails

Par défaut dans Odoo, un email est supprimé de la table `mail.mail` dès qu'il est remis au serveur SMTP. Avec Sweego, le module retarde cette suppression jusqu'à confirmation de livraison ou signalement d'erreur.

La confirmation peut arriver par deux chemins complémentaires :

- **Webhook outbound** : Sweego notifie Odoo en temps réel lors de la livraison ou du bounce

- **Tâche planifiée** : un cron interroge périodiquement l'API Sweego pour récupérer les livraisons manquées

Table 1. Cycle de vie d'un email

Étape	État	Description
1	Envoi SMTP	L'email est transmis à Sweego via SMTP. <code>mail.notification.notification_status</code> passe à <code>sent</code> .
2	Attente confirmation	<code>sweego_pending_delete = True</code> . L'email reste en base, la suppression est suspendue.
3	Webhook <code>email_sent</code>	Sweego confirme la prise en charge — log informatif uniquement.
4a	Webhook <code>delivered</code>	Sweego notifie la livraison en temps réel. L'enregistrement <code>mail.mail</code> est supprimé.
4b	Cron synchronisation	Si le webhook <code>delivered</code> n'a pas été reçu, le cron confirme les livraisons manquées → suppression.
4c	Bounce (webhook ou cron)	<code>mail.mail.state = exception</code> . <code>mail.notification.notification_status = exception</code> → enveloppe rouge dans le chatter.

5.4. Gestion des bounces et emails en erreur

Lorsqu'un email ne peut pas être livré, Sweego signale l'erreur soit via le webhook outbound (événements `soft-bounce` / `hard-bounce`) soit via l'API de logs (statut `undelivered`).

Le module traduit ces événements en états Odoo :

Événement Sweego	<code>failure_type</code> Odoo	Cause
<code>soft-bounce</code>	<code>BOUNCE</code>	Échec temporaire (boîte pleine, serveur indisponible...)
<code>hard-bounce</code>	<code>RECIPIENT</code>	Rejet permanent (adresse invalide)
<code>undelivered</code> (bounce hard)	<code>RECIPIENT</code>	Idem via l'API de logs
<code>undelivered</code> (bounce soft)	<code>BOUNCE</code>	Idem via l'API de logs
<code>undelivered</code> (blacklist)	<code>RECIPIENT</code>	Adresse présente dans <code>x-emails-blacklisted</code>
<code>undelivered</code> (autre)	<code>UNKNOWN</code>	Raison non classifiable



Odoo 14 ne connaît que quatre `failure_type` : `SMTP`, `RECIPIENT`, `BOUNCE` et `UNKNOWN`. Les valeurs plus fines des versions suivantes (`mail_bl`, `mail_email_invalid`, `mail_smtp`)

n'existent pas encore, d'où le regroupement ci-dessus. Une adresse blacklistée et un rejet définitif tombent tous deux dans **RECIPIENT**.

Lors d'un bounce, la méthode `sweego_mark_recipient_undelivered(email, failure_type, failure_reason)` est appelée. Elle :

1. Met à jour `mail.mail : state = exception, failure_reason, sweego_pending_delete = False` — `mail.mail` n'a pas de champ `failure_type` en 14.0, celui-ci n'est porté que par la notification
2. Met à jour `mail.notification : notification_status = exception` — ce que `_postprocess_sent_message` ne peut pas faire rétroactivement car il filtre les notifications déjà à `sent`
3. Appelle `_notify_message_notification_update()` pour rafraîchir l'interface en temps réel

5.5. Réception des réponses (Inbound)

Lorsqu'un client répond à un email envoyé par Odoo :

1. La réponse arrive sur le domaine de réception configuré dans Sweego (ex : `reply+TOKEN@test.mangono.cloud`)
2. Sweego transmet la réponse au webhook inbound d'Odoo via une requête POST JSON
3. Le module résout le document Odoo cible via la chaîne de résolution décrite ci-dessous
4. Un message RFC2822 est reconstruit depuis le payload Sweego (corps, pièces jointes téléchargées via l'API Sweego)
5. Le message est injecté dans le chatter via `mail.thread.message_process`
6. La réponse apparaît directement dans le chatter de l'enregistrement Odoo



Les adresses `reply+TOKEN@domaine` ne sont jamais exposées à Odoo comme destinataires réels. Le module les filtre automatiquement lors de la reconstruction du message RFC2822. Si après filtrage il ne reste aucune adresse To, l'adresse de l'expéditeur est utilisée en fallback.

5.5.1. Chaîne de résolution du thread inbound

Le module tente de résoudre l'enregistrement Odoo cible dans l'ordre suivant :

Priorité	Mécanisme	Cas couverts
1	Token <code>reply+</code> dans l'adresse To	Réponses manuelles via le Reply-To standard. Le token encode <code>{model}#{id}</code> signé HMAC.
2	In-Reply-To → <code>mail.message.message_id</code>	Réponses clients qui ont perdu le token (répondu au From au lieu du Reply-To).
3	References → <code>mail.message.message_id</code>	Réponses automatiques OOO Outlook : le fil References contient les Message-Id Odoo originaux même si le In-Reply-To pointe vers le Message-Id Sweego.

Priorité	Mécanisme	Cas couverts
4	In-Reply-To → <code>mail.message.sweego_assigned_message_id</code>	Toutes les auto-réponses (OOO, règles Exchange) sans header References . Le Message-Id assigné par Sweego est stocké au moment du webhook <code>email_sent</code> et sert de clé de lookup.
—	Routing natif <code>message_process</code>	Fallback si aucun des mécanismes ci-dessus ne trouve de thread (routing par alias Odoo). Pour les auto-réponses, le fallback est ignoré silencieusement (<code>auto_reply_ignored</code>).



Sweego remplace le **Message-Id** de l'email sortant par son propre format (`<timestamp.client_hash.fragment@domaine>`). Les mécanismes 3 et 4 existent précisément parce que le **In-Reply-To** des auto-réponses pointe vers ce Message-Id Sweego, jamais vers le Message-Id Odoo stocké en base. Le mécanisme 4 s'appuie sur `sweego_assigned_message_id`, un champ indexé sur `mail.message` renseigné lors du webhook `email_sent`.

5.6. Pièces jointes inbound

Lorsqu'un client joint un fichier à sa réponse, le module télécharge automatiquement la pièce jointe depuis l'API REST Sweego avant de la transmettre à Odoo. La clé API et l'UUID client configurés dans **Paramètres** → **Sweego** sont requis pour cette fonctionnalité.

5.7. Détection des boucles et réponses automatiques

Le module détecte les emails en boucle et les réponses automatiques via plusieurs signaux :

En-tête détecté	Cas couverts
<code>x-zone-loop</code> (toute valeur)	Boucle détectée par Sweego — ignoré systématiquement
<code>x-mailer: Sweego</code>	Email envoyé par Sweego lui-même — ignoré systématiquement
<code>Auto-Submitted</code> ≠ <code>no</code> (RFC 3834)	Réponse automatique standard (OOO, vacation reply, DSN...)
<code>x-auto-response-suppress</code> (toute valeur)	Réponses automatiques Microsoft/Outlook (OOO, règles de boîte aux lettres)

Pour les réponses automatiques (deux dernières lignes), le comportement dépend du paramètre **Afficher les réponses auto dans le chatter** :

- **Désactivé (défaut)** : la réponse est ignorée silencieusement. Le webhook répond **200 OK** avec `{"status": "auto_reply_ignored"}`.
- **Activé** : la chaîne de résolution du thread est tentée (token → **In-Reply-To** → **References** → `sweego_assigned_message_id`). Si un thread est trouvé, la réponse est postée dans le chatter. Si aucun thread n'est résolu, la réponse est abandonnée silencieusement.



Même avec **Afficher les réponses auto** activé, le webhook répond toujours **200 OK**

— Sweego ne retentera jamais un message déjà reçu.

5.8. Vérification des signatures webhook

Tous les webhooks reçus de Sweego sont vérifiés via le protocole [Standard Webhooks](#) (HMAC-SHA256). La signature est calculée sur la chaîne `{Webhook-Id}.{Webhook-Timestamp}.{body JSON brut}`. Si invalide, la requête est rejetée avec HTTP 401.

Si plusieurs sociétés utilisent chacune leur propre serveur Sweego (donc leur propre secret inbound et/ou outbound), Sweego appelle la **même URL de webhook** pour toutes — la signature est alors le seul moyen d'identifier quelle société a émis l'appel. Le module essaie le secret de **chaque serveur Sweego actif** jusqu'à trouver celui qui valide la signature, puis utilise ce serveur pour la suite du traitement (ex : vérifier si **Suppression différée** ou **Afficher les réponses auto** est activé pour cette société). La requête n'est rejetée (401) que si **aucun** secret configuré ne correspond.

5.9. Architecture de traitement outbound

Le traitement des événements Sweego est géré à deux niveaux :

Webhook outbound (`/sweego/webhook/outbound`) — traitement en temps réel par événement :

event_type	Action	Notes
email_sent	Stocke <code>sweego_assigned_message_id</code> sur <code>mail.message</code>	Le Message-Id Sweego (<code><x-swg-bid>@<domain_from></code>) est extrait du payload et indexé pour le routage des auto-réponses
delivered	<code>sweego_confirm_and_delete()</code> sur <code>mail.mail</code> + mise à jour <code>sweego_message_id</code>	
soft-bounce	<code>sweego_mark_recipient_undelivered(..., BOUNCE, ...)</code>	Enveloppe rouge dans le chatter
hard_bounce	<code>sweego_mark_recipient_undelivered(..., RECIPIENT, ...)</code>	Enveloppe rouge dans le chatter
list_unsub	Log informatif	
complaint	Log warning	
email_clicked	Log informatif	
email_opened	Log informatif	

Cron de synchronisation — traitement via l'API de logs, unifié par `_sweego_process_log_entry` :

status	Handler	Action
email_sent	<code>_sweego_on_email_sent</code>	Log informatif
delivered	<code>_sweego_on_delivered</code>	<code>sweego_confirm_and_delete()</code> + mise à jour <code>sweego_message_id</code>

status	Handler	Action
undelivered	_sweego_on_undelivered	sweego_mark_recipient_undelivered(...) → enveloppe rouge dans le chatter

6. Tâche planifiée — Synchronisation des logs

6.1. Rôle

Filet de sécurité pour les livraisons et bounces dont le webhook outbound n'a pas été reçu. Interroge périodiquement l'API Sweego `/logs/` et traite les `mail.mail` en attente.

6.2. Comportement

1. Recherche les `mail.mail` avec `sweego_pending_delete = True` et `sweego_confirmed = False`
2. Si aucun email en attente, termine immédiatement sans appel API
3. Récupère le domaine de réception depuis `mail.catchall.domain`
4. Interroge l'API Sweego, pagine les résultats par tranches de 500 jusqu'à épuisement
5. Filtre côté client sur les `Message-Id` des emails en attente
6. Délègue chaque entrée à `_sweego_process_log_entry`



Le filtre par domaine garantit qu'en cas de compte Sweego partagé entre plusieurs instances Odoo, chaque instance ne traite que ses propres emails.

6.3. Paramétrage

Créée automatiquement à l'installation, s'exécute toutes les 30 minutes par défaut. Pour modifier : **Paramètres** → **Technique** → **Automatisation** → **Actions planifiées** → **Sweego : Synchronisation des logs de livraison**.

7. Mécanisme du token Reply-To

7.1. Format

```
reply+{payload_hex}.{signature}@domaine-de-réception
```

Exemple :

```
reply+32332d73616c652e6f72646572.cb8e0be9@test.mangono.cloud
```

Décodé :

```
payload_hex → 32332d73616c652e6f72646572 → 23-sale.order
```

signature → cb8e0be9 (8 premiers caractères HMAC-SHA256)

- Le payload `{id}-{model}` est encodé en hexadécimal (résistant à la normalisation minuscules par les serveurs mail)
- La signature HMAC utilise le **secret webhook sortant** (`sweego_outbound_webhook_secret`) configuré sur le serveur d'envoi

7.2. Décodage

À la réception, le module extrait le token et tente de le décoder avec le secret sortant de **chaque serveur Sweego actif**, dans l'ordre, jusqu'à ce qu'une signature soit valide — le serveur ayant signé le token à l'envoi n'est pas forcément celui identifié à la réception (cf. *Plusieurs serveurs Sweego*). Si aucun secret ne valide la signature, le message est ignoré.

8. Champs techniques ajoutés

8.1. Sur `ir.mail_server`

Champ	Description
<code>sweego_enabled</code>	Désigne ce serveur comme serveur Sweego (Boolean, défaut : <code>False</code>). Remplace la valeur <code>smtp_authentication = 'sweego'</code> des versions 17.0 et suivantes, ce champ natif n'existant pas en 14.0.
<code>sweego_outbound_webhook_secret</code>	Secret HMAC webhooks outbound (base64, accès admin). Utilisé aussi pour signer les tokens Reply-To.
<code>sweego_inbound_webhook_secret</code>	Secret HMAC webhooks inbound (base64, accès admin)
<code>sweego_inbound_webhook_url</code>	Calculé, lecture seule — <code>web.base.url + /sweego/webhook/inbound</code> . Affiché avec un bouton copier pour la configuration côté Sweego.
<code>sweego_outbound_webhook_url</code>	Calculé, lecture seule — <code>web.base.url + /sweego/webhook/outbound</code> . Affiché avec un bouton copier pour la configuration côté Sweego.
<code>sweego_delete_on_sent</code>	Activer la suppression différée (Boolean, défaut : <code>False</code>)
<code>sweego_show_auto_replies</code>	Poster les réponses automatiques dans le chatter (Boolean, défaut : <code>False</code>). Voir la section <i>Détection des boucles et réponses automatiques</i> .



Le domaine inbound n'est pas porté par le serveur d'envoi : il est lu dans le paramètre système `mail.catchall.domain`, aussi bien à l'envoi que dans le cron de synchronisation.

8.2. Sur `mail.mail`

Champ	Description
<code>sweego_pending_delete</code>	<code>True</code> après envoi — suppression suspendue en attente de confirmation Sweego
<code>sweego_confirmed</code>	<code>True</code> après confirmation de livraison — autorise la suppression définitive
<code>sweego_auto_delete</code>	Intention <code>auto_delete</code> capturée au moment du report. La suppression après confirmation n'a lieu que si ce champ est <code>True</code> — c.-à-d. le modèle de mail avait « Suppression auto » activé, ou l'email n'avait pas de modèle (ex. notifications). Sinon l'email est conservé, la livraison étant simplement confirmée. Le suivi livraison/échec reste actif pour tous les emails en attente.

8.2.1. Méthodes sur `mail.mail`

Méthode	Description
<code>sweego_confirm_and_delete()</code>	Appelée par le webhook <code>delivered</code> ou le cron. Marque l'email comme confirmé et le supprime uniquement si <code>sweego_auto_delete = True</code> (sinon l'email est conservé).
<code>sweego_mark_undelivered(type, msg)</code>	Appelée lors d'un bounce. Met à jour <code>mail.mail</code> et <code>mail.notification</code> pour afficher l'enveloppe rouge dans le chatter.

8.3. Sur `mail.message`

Champ	Description
<code>sweego_message_id</code>	<code>transaction_id</code> Sweego enregistré lors du <code>delivered</code> (indexé)
<code>sweego_assigned_message_id</code>	Message-Id assigné par Sweego lors du relayage sortant, au format <code><x-swg-bid@domain_from></code> . Renseigné par le webhook <code>email_sent</code> . Indexé — utilisé comme dernière option de résolution du thread pour les auto-réponses.

8.4. Sur `mail.notification`

Champ	Description
<code>sweego_delivery_confirmed</code>	<code>True</code> lorsque Sweego confirme la livraison à ce destinataire précis. Un même <code>mail.mail</code> peut couvrir plusieurs destinataires, et Sweego rapporte la livraison destinataire par destinataire — ce drapeau permet de savoir quand tous les destinataires d'un <code>mail.mail</code> ont été résolus avant de supprimer l'enregistrement différé (Boolean, défaut : <code>False</code>).

8.5. Dans `ir.config_parameter`

Clé	Description
<code>mangono_sweego.api_key</code>	Clé API Sweego (géré par <code>mangono_sweego_base</code>)
<code>mangono_sweego.client_uuid</code>	UUID client Sweego (géré par <code>mangono_sweego_base</code>)

9. Endpoints webhook

Endpoint	Rôle
<code>POST /sweego/webhook/outbound</code>	Événements outbound : livraison, bounces, ouvertures, clics...
<code>POST /sweego/webhook/inbound</code>	Emails entrants (réponses clients)

10. Dépannage

10.1. Les webhooks sont rejetés (401)

- Secret webhook mal copié — copier tel quel (base64) depuis le tableau de bord Sweego
- Vérifier que outbound et inbound sont dans leurs champs respectifs
- Vérifier que l'URL webhook est accessible publiquement
- Avec plusieurs serveurs Sweego actifs, le module essaie automatiquement le secret de chacun — un **401** signifie qu'**aucun** des secrets configurés ne correspond à la signature reçue pour cet événement, pas nécessairement celui du premier serveur

10.2. Les réponses n'apparaissent pas dans le chatter

- Vérifier que le MX du domaine de réception pointe vers `inbound.sweego.co`
- Vérifier la configuration du webhook inbound dans Sweego
- Consulter les logs : `invalid inbound token signature`
- Vérifier que le paramètre système `mail.catchall.domain` est renseigné
- Consulter les logs : `no catchall domain configured` — aucun domaine résolu, le Reply-To n'a pas été posé

10.3. Les pièces jointes inbound n'apparaissent pas

- Vérifier la Clé API Sweego et l'UUID Client Sweego dans Paramètres → Sweego
- Consulter les logs : `failed to download attachment`

10.4. Les emails ne sont pas supprimés après livraison

- Vérifier que le webhook outbound est configuré avec l'événement `delivered`
- Vérifier que **Suppression différée** est activée sur le serveur
- Le cron prend le relais sous 30 minutes si le webhook n'est pas reçu
- Consulter les logs : `mail.mail X confirmed by Sweego → deleting`

10.5. L'enveloppe reste grise après un bounce

- Vérifier que le webhook outbound est configuré avec les événements `soft-bounce` et `hard_bounce`
- Si le webhook n'est pas disponible, le cron traite les statuts `undelivered` sous 30 minutes
- Consulter les logs : `mail.mail X marked undelivered by Sweego`
- L'enveloppe rouge n'est visible que pour les emails avec **Suppression différée** activée (sinon `mail.mail` est déjà supprimé au moment du bounce)

10.6. Les réponses automatiques (OOO) n'apparaissent pas dans le chatter

1. Vérifier que **Afficher les réponses auto dans le chatter** est activé sur le serveur d'envoi Sweego
2. Vérifier que le webhook `email_sent` outbound est bien configuré dans Sweego — il est nécessaire pour stocker le `sweego_assigned_message_id` qui permet le routage
3. Consulter les logs : `Sweego: stored assigned Message-Id` doit apparaître après chaque envoi d'email (confirme que le mapping est en place)
4. Consulter les logs inbound : `Thread resolved via sweego_assigned_message_id` confirme que le routage fonctionne
5. Si le log indique `auto-reply thread not resolvable`, aucun des quatre mécanismes de résolution n'a trouvé de thread — l'auto-réponse est abandonnée sans erreur
6. Vérifier que le module a été mis à jour (`odoo-bin -u mangono_sweego_mail`) après déploiement — la colonne `sweego_assigned_message_id` doit exister en base

10.7. La tâche planifiée ne traite aucun email

- Vérifier la **Clé API Sweego** dans **Paramètres** → **Sweego**
- Vérifier que le paramètre système `mail.catchall.domain` est renseigné
- Vérifier que la tâche est active dans **Paramètres** → **Technique** → **Automatisation** → **Actions planifiées**

10.8. Le serveur refuse de se créer

- Le champ **Nom d'utilisateur** est obligatoire quand **Mangono Sweego SMTP** est coché
- Le chiffrement doit rester sur **TLS (STARTTLS)**

11. Glossaire

Terme	Définition
SMTP	Simple Mail Transfer Protocol — protocole standard d'envoi d'emails
Webhook	Notification HTTP envoyée par Sweego vers Odoo pour signaler un événement
HMAC-SHA256	Algorithme de signature cryptographique utilisé pour authentifier les webhooks
Token Reply-To	Jeton encodé dans l'adresse Reply-To permettant de router les réponses vers le bon objet Odoo
Domaine de réception	Domaine email de la base Odoo (paramètre système <code>mail.catchall.domain</code>), utilisé pour la réception des réponses (MX → Sweego)
RFC 2047	Standard d'encodage des en-têtes email non-ASCII (ex : <code>=?utf-8?q?...?=</code>), appliqué par certains MTA lors du repli de lignes longues
<code>mail.mail</code>	Table Odoo contenant les emails en attente d'envoi ou récemment envoyés
<code>mail.notification</code>	Table Odoo liant un message à un destinataire, portant le statut de notification (<code>sent</code> , <code>exception...</code>) affiché dans le chatter
<code>mail.message</code>	Table Odoo contenant les messages du chatter (emails, notes internes...)
chatter	Interface de messagerie intégrée aux fiches Odoo
Standard Webhooks	Protocole ouvert de signature des webhooks basé sur HMAC-SHA256
API REST Sweego	Interface HTTP Sweego pour le téléchargement des pièces jointes et la synchronisation des logs
<code>mangono_sweego_base</code>	Module Odoo partagé gérant la configuration de connexion Sweego (clé API, UUID client)