Easy Concierge

API du moteur de réservation
Site de réservation sur mesure

Notice développeur · lecture, devis, réservation et webhooks

Destinataire	Le développeur ou l'agence web de la conciergerie
Objet	Refaire entièrement les écrans de réservation sur un site sur mesure, avec sa propre charte, en gardant le calendrier, les tarifs, le paiement Stripe et les automatisations d'Easy Concierge
Version du document	1.5 du 10 novembre 2026
Authentification	Aucune pour lire et réserver. Clé facultative pour un serveur qui réserve pour le compte des voyageurs (section 6)
Contact	Malo et Matthieu · contact@easy-concierge.fr

Ce document décrit l'API du moteur de réservation, sans clé pour les voyageurs. Elle est distincte de l'API publique protégée par clé (`/public/api/v1/`), qui sert à lire les données depuis un autre logiciel et ne permet pas de réserver.

Sommaire

1. Avant de commencer (la conciergerie)
2. Adresse et conventions
3. Codes d'erreur
4. Lire les logements et le calendrier
5. Calculer le prix (le devis)
6. Réserver
7. Ramener le voyageur sur votre site
8. Être prévenu par webhook
9. Le parcours complet en JavaScript
10. Limites à connaître

1. Avant de commencer (la conciergerie)

Ces réglages se font dans le logiciel, avec un compte administrateur, dans **Paramètres, onglet API**. Le développeur ne peut pas les faire lui-même.

Réglage	Rôle
Domaines de retour autorisés	**Indispensable.** Le domaine du site, un par ligne, HTTPS uniquement. Sans lui, le voyageur ne peut pas revenir sur votre site après le paiement (section 7), et le captcha anti-robot ne peut pas s'afficher sur votre site (section 6).
Rappel automatique de votre site	Facultatif. L'adresse HTTPS qui recevra les webhooks. Un secret de signature est créé à l'enregistrement : à transmettre au développeur (section 8).
Clé API	Facultative. Utile seulement si toutes les réservations partent du serveur du site, et non du navigateur du voyageur (section 6). Type de clé **Site internet** (proposé par défaut) : elle ne donne pas accès aux réservations de la conciergerie. À transmettre par un lien sécurisé à usage unique, jamais en clair par e-mail.
Mode test	Pour les essais : fiche du logement, onglet **Publication**, case **Mode test : essai à blanc, aucune réservation n'est créée**. Un essai ne bloque aucune date et ne crée ni réservation, ni tâche, ni paiement ; il est renvoyé avec `status: cancelled` et `test: true`. À décocher avant la mise en ligne.

2. Adresse et conventions

Élément	Valeur
Adresse unique	`https://easy-concierge.fr/guest/{dossier}/public/booking/api.php` `{dossier}` est l'identifiant de la conciergerie, communiqué par elle ou par Easy Concierge.
Authentification	Aucune pour lire ou réserver. Appelable depuis n'importe quel domaine (CORS activé).
Action	Lectures en `GET`, avec `?action=…` dans l'adresse. Réservation en `POST` : `action` peut aller indifféremment dans l'adresse (`?action=book`), dans le corps JSON (`{"action":"book", …}`) ou dans un champ de formulaire.
Fichier de description	Description OpenAPI 3 de cette API, importable dans Postman, Insomnia ou un générateur de client : `https://easy-concierge.fr/openapi-moteur-reservation.yaml`
Formats	JSON en UTF-8. Dates au format `YYYY-MM-DD`. Montants en nombres décimaux, dans la devise indiquée par `currency`.

Photos	Chemins relatifs, à préfixer par `https://easy-concierge.fr/guest/{dossier}/`

3. Codes d'erreur

Toute réponse porte un booléen `success`. En cas d'échec :

```
{ "success": false, "error": "Dates non disponibles" }
```

HTTP	Cas
400	Paramètre manquant ou invalide, dates indisponibles. `"Action non reconnue"` : `action` est absente ou mal orthographiée.
404	Logement ou réservation introuvable.
403	`book` sans jeton anti-robot valide (section 6). La réponse porte `captcha: true`.
405	`book` appelé en GET.
409	Le montant à régler a changé depuis votre devis. La réponse porte `reload: true` : refaites `calculate` et réaffichez le montant avant de renvoyer le voyageur payer.
429	Plafond de cadence atteint (section 10). Ce n'est pas une erreur de saisie : faites patienter.
500	Échec côté paiement.
503	Aucun compte d'encaissement configuré sur ce logement.

4. Lire les logements et le calendrier

`GET ?action=properties`

Paramètre	Effet
`city`	Ville.
`guests`	Capacité minimale.
`bedrooms`	Nombre de chambres.
`check_in` + `check_out`	Ajoute `is_available` à chaque logement.

Tous facultatifs. L'ordre est aléatoire à chaque appel.

```
{
  "success": true,
  "properties": [
    { "id": 22, "name": "Villa 10 pers.", "public_title": "Spacieuse villa",
      "public_slug": "spacieuse-villa", "city": "Clohars-Carnoët", "country": "France",
      "max_guests": 10, "bedrooms": 5, "beds": 5, "bathrooms": 3,
      "checkin_time_min": "17:00", "checkin_time_max": "20:00", "checkout_time": "10:00",
      "cleaning_price": 220, "linen_price": 0,
      "main_photo": "photos/gallery_1771167444.jpg", "min_price": 150 }
  ]
}
```


`GET ?action=property&id=22`

La fiche complète : tout ce qui précède, plus `public_description`, `address`, `gps_lat`, `gps_lng`, `house_rules`, `cancellation_policy`, `default_deposit_amount` (caution annoncée), `tourist_tax_type` / `tourist_tax_rate` / `tourist_tax_cap`, `meta_title`, `meta_description`, `cleaning_mandatory`, `booking_mode`, `turnover_buffer_days`, les majorations court séjour et remises long séjour, ainsi que `photos[]` et `amenities[]`.


```
"photos":    [ { "url": "photos/gallery_1771167444.jpg", "caption": "Façade",
                 "is_main": 1, "sort_order": 0 } ],
"amenities": [ { "key": "bbq", "value": "1", "label": "Barbecue", "icon": "flame" } ]
```


`GET ?action=availability&property_id=22&year=2026&month=7`

Le mois, nuit par nuit. Un mois et un logement par appel.

> **Piège fréquent :** `calendar` est un **objet indexé par date**, pas une liste.


```
{
  "success": true, "year": 2026, "month": 7,
  "calendar": {
    "2026-07-14": {
      "date": "2026-07-14", "day": 14, "day_of_week": 2, "is_weekend": false,
      "price": 160, "min_nights": 2, "rule": null,
      "is_available": true, "is_booked": false, "is_checkin_day": false,
      "is_blocked": false, "is_too_soon": false, "is_buffer": false, "is_past": false
    }
  }
}
```


`is_available` est le **seul drapeau à tester pour vendre**. Les autres expliquent un refus :

Champ	Sens
`is_booked`	Nuit déjà réservée.
`is_blocked`	Fermée à la main par la conciergerie.
`is_too_soon`	Sous le délai de préavis.
`is_buffer`	Nuit réservée au ménage après un départ.
`is_past`	Nuit passée.

`rule`	`no_checkin`, `no_checkout` ou `no_checkin_checkout`.
`min_nights`	Varie selon la date : lisez-le sur la **nuit d'arrivée**.

GET `?action=check_availability&property_id=22&check_in=…&check_out=…`

```
{ "success": true, "available": true, "rules_valid": true,
  "rules_errors": [], "nights": 5, "short_stay": null }
```

GET `?action=options&property_id=22`

Les options payantes que le voyageur peut ajouter à son séjour.

```
{ "success": true, "property_id": 22, "currency": "EUR",
  "options": [ { "id": 3, "label": "Panier d'accueil", "description": "Produits locaux",
                 "price": 25, "type": "welcome_basket", "image": "photos/upsell/…jpg" } ] }
```

GET `?action=cities · ?action=amenities`

Sans paramètre : les villes des logements, et la liste de référence des équipements.

5. Calculer le prix (le devis)

GET `?action=calculate&property_id=22&check_in=…&check_out=…&guests=4`

Paramètre	Effet
`include_cleaning`	0 ou 1, 1 par défaut.
`options`	`3,4` ou `[3,4]`. Revérifiés côté serveur : un id inconnu, appartenant à un autre logement ou non vendable est ignoré sans erreur.
`code`	Code de remise saisi par le voyageur (facultatif). Majuscules et minuscules sont équivalentes.
`guest_email`	Facultatif ici : utile seulement pour un code limité à une utilisation par voyageur.

```
{
  "success": true, "property_id": 22, "property_name": "Spacieuse villa",
  "check_in": "2026-07-14", "check_out": "2026-07-19", "nights": 5, "guests": 4,
  "currency": "EUR", "price_source": "easy_concierge",
  "include_cleaning": true, "cleaning_fee_available": 220, "linen_fee_available": 0,
  "short_stay": null,
  "guest_supplement": null,
  "long_stay_discount": null,
  "discount_code": null,
  "breakdown": {
    "nights": [ { "date": "2026-07-14", "price": 160 } ],
    "subtotal": 800, "cleaning_fee": 220, "linen_fee": 0, "tourist_tax": 22,
    "options": [ { "id": 3, "label": "Panier d'accueil", "price": 25 } ],
    "options_total": 25,
    "total": 1067
  },
  "deposit": { "amount": 533.5, "is_partial": true, "percent": 50,
               "due_lines": 1, "balance": 533.5 }
}
```


Affichez `deposit.amount`, pas seulement le total. Quand le logement a un plan de paiement (acompte puis solde), c'est ce montant que la banque demandera, et lui seul. Afficher le total puis envoyer le voyageur payer la moitié sans explication le fait renoncer. `is_partial: false` signifie que tout se règle maintenant, et `balance` vaut alors 0.

- `short_stay` et `long_stay_discount` valent `null` la plupart du temps. Quand ils sont renseignés, leur `amount` est **déjà compris** dans `subtotal` : affichez-les pour expliquer le prix, ne les ajoutez ni ne les retranchez jamais.
- `guest_supplement` (tarif à la personne) vaut `null` quand le logement n'en a pas ou qu'aucun voyageur ne dépasse le nombre inclus. Sinon : `{ "applied": true, "included_guests": 4, "fee_per_guest": 30, "extra_guests": 2, "nights": 5, "amount": 300 }`. Le supplément s'ajoute au prix de chaque nuit **avant** la majoration court séjour, la remise long séjour et le code de remise ; son `amount` est **déjà compris** dans `subtotal` et dans `breakdown.nights[].price`. Le devis dépend donc de `guests` : **recalculez à chaque changement du nombre de voyageurs**.
- `guests` au-delà de la capacité du logement (`max_guests`) est refusé : `{ "success": false, "error": "Ce logement accueille 8 voyageurs au maximum." }`. Bornez votre sélecteur à `max_guests`.
- `discount_code` vaut `null` sans paramètre `code`. Code accepté : `{ "applied": true, "code": "FIDELE10", "label": "−10 %", "amount": 80 }`, et `amount` est lui aussi **déjà déduit** de `subtotal`. Code écarté : `{ "applied": false, "reason": "…", "message": "…" }`, à afficher tel quel sous le champ. `reason` vaut `invalid` (code inconnu, expiré, épuisé ou d'un autre logement, sans distinction), `short_stay` (séjour majoré sous la durée minimale) ou `long_stay_better` (la remise long séjour, plus avantageuse, est gardée). Le devis reste valable dans les trois cas, au prix sans le code.
- **La remise d'un code porte sur les nuitées seulement**, jamais sur le ménage, le linge, les options ni la taxe de séjour. Elle ne se cumule pas avec la remise long séjour.
- `price_source` vaut `easy_concierge` ou `pricelabs` selon l'outil qui pilote les tarifs du logement.
- Séjour invendable : `{ "success": false, "errors": [...] }`.

6. Réserver

`POST action=book`

Corps en JSON (`Content-Type: application/json`) ou en formulaire. `action` se place dans le corps, comme dans l'exemple de la section 9, ou dans l'adresse (`api.php?action=book`) : les deux sont acceptés.

Champs	Liste
Obligatoires	`property_id` , `check_in` , `check_out` , `guest_name` , `guest_email`
Facultatifs	`guests` , `minors` , `guest_phone` , `guest_message` , `include_cleaning` , `options` , `return_url` , `cancel_url` , `due_lines` , `discount_code`
Anti-robot	`cf-turnstile-response` : obligatoire quand la réservation part du navigateur du voyageur (voir ci-dessous).

`minors` : nombre de voyageurs mineurs, **compris** dans `guests` et exonérés de taxe de séjour. Accepté aussi par `calculate` . Il y a toujours au moins un adulte : la valeur est plafonnée à `guests − 1` .

Le captcha anti-robot

Depuis octobre 2026, une réservation envoyée depuis un navigateur doit prouver qu'un humain est derrière, avec un captcha **Cloudflare Turnstile**. En mode « Managed », il reste invisible pour presque tous les voyageurs.

1. La conciergerie déclare le domaine de votre site dans **Paramètres, onglet API**, rubrique **Domaines de retour autorisés** (le même réglage que pour `return_url` , section 7). Votre domaine est alors autorisé auprès de Cloudflare automatiquement, en **10 minutes au plus**. Rien à nous demander.
2. Votre page lit la clé publique du widget, depuis le navigateur :

   ```
   GET ?action=turnstile
   { "success": true, "required": true, "host": "mon-site.fr",
     "sitekey": "0x4AAAAAA…" }
   ```

   La clé dépend du site qui appelle (en-tête `Origin` , envoyé par le navigateur) : appelez cette action depuis votre page, pas depuis votre serveur. `sitekey` vaut `null` avec un `message` tant que le domaine n'est pas déclaré ou pas encore prêt. `required: false` signifie que le captcha n'est pas exigé pour ce compte : affichez le widget quand même si `sitekey` est renseignée, cela ne coûte rien.
3. Affichez le widget sur votre page de réservation :

   ```
   <script src="https://challenges.cloudflare.com/turnstile/v0/api.js" async defer></script>
   <div class="cf-turnstile" data-sitekey="…clé reçue…"></div>
   ```

4. Joignez le jeton produit par le widget au corps de `book` , dans le champ `cf-turnstile-response` (section 9). Un jeton ne sert qu'une fois : après un refus, réinitialisez le widget (`turnstile.reset()`) avant de renvoyer.

Sans jeton, ou avec un jeton refusé, l'API répond `403` avec `captcha: true` et ne crée rien : affichez son message, réinitialisez le widget et laissez le voyageur réessayer.

N'envoyez dans `discount_code` qu'un code que `calculate` a renvoyé avec `applied: true` . Il est revérifié : s'il n'est plus valable (dernière utilisation partie entre-temps, voyageur qui l'a déjà utilisé), l'API refuse avec `success: false` et un message, sans rien créer. Elle ne réserve jamais au plein tarif un voyageur qui croyait payer le prix remisé.

Renvoyez le `due_lines` reçu de `calculate` : si le montant exigible a changé entre-temps (une échéance devenue due à minuit, un plan modifié), l'API refuse en `409` plutôt que d'encaisser un montant différent de celui que vous avez affiché.

Trois réponses possibles, selon le réglage du logement, lisible à l'avance dans `booking_mode` sur la fiche pour adapter le libellé du bouton :

```
// paiement en ligne : redirigez le voyageur vers stripe_url
{ "success": true, "public_booking_id": 42,
  "stripe_url": "https://checkout.stripe.com/…", "session_id": "cs_live_…" }

// demande par e-mail : la conciergerie rappellera le voyageur
{ "success": true, "public_booking_id": 42, "booking_id": 1337,
  "mode": "email_request", "redirect_url": "…" }

// mode test : essai à blanc, aucune réservation créée
{ "success": true, "public_booking_id": 42, "booking_id": null,
  "mode": "test_without_payment", "test": true, "redirect_url": "…" }
```

Réservations envoyées depuis votre serveur

Sans clé, l'API accepte un nombre limité de créations par heure et par adresse. C'est sans effet quand chaque voyageur réserve depuis son navigateur. Si toutes vos réservations partent de **votre serveur** (une seule IP), demandez à la conciergerie une clé API de type **Site internet** et envoyez-la dans l'en-tête :

```
Authorization: Bearer ek_…
```

> La clé lève ce plafond par adresse et dispense du captcha (un serveur ne peut pas en résoudre un). Seules les clés de la conciergerie ont cet effet : type **Site internet** (droit `site:read`) ou **Lecture complète** (`read`). Une clé de partenaire (lecture des disponibilités, revenue management) ne lève rien : la réservation suit le chemin public, avec captcha. Une clé inconnue, expirée ou appelée depuis une adresse non autorisée n'est pas refusée en `401` : elle est ignorée de la même façon. La clé ne doit **jamais** figurer dans le JavaScript de vos pages ni dans l'adresse (`?api_key=` est ignoré). Le plafond de trois demandes en attente par logement reste appliqué.

7. Ramener le voyageur sur votre site

> **À faire en premier.** La conciergerie déclare votre domaine dans **Paramètres, onglet API**, rubrique **Domaines de retour autorisés**. Une adresse dont le domaine n'y figure pas est refusée, et seul le HTTPS est accepté. C'est ce qui empêche un tiers de détourner les voyageurs juste après leur paiement.

Sans `return_url`, le voyageur termine sur les pages de confirmation d'Easy Concierge. Avec, il revient chez vous, et nous ajoutons `session_id` à votre adresse (ou `public_booking_id` pour les logements sans paiement en ligne). `cancel_url` reçoit les abandons.

```
GET ?action=booking_status&session_id=cs_live_…
```

Ou, pour les modes sans paiement : `public_booking_id` + `guest_email`, l'e-mail servant de preuve.

```
{ "success": true, "public_booking_id": 42, "booking_id": 1337,
  "status": "confirmed", "payment_status": "succeeded",
  "property_id": 22, "check_in": "2026-07-14", "check_out": "2026-07-19",
  "nights": 5, "total": 1067, "currency": "EUR",
  "paid_at": "2026-06-02 14:31:08", "confirmed_at": "2026-06-02 14:31:09",
  "expires_at": "2026-06-02 15:01:02" }
```

`status` vaut `pending`, `confirmed`, `cancelled` ou `expired`. `booking_id` reste `null` tant que la réservation n'est pas confirmée. Une réservation faite en **Mode test** est renvoyée avec `status` à `cancelled` et `"test": true` : ce n'est pas une vraie réservation.

> **Le voyageur revient souvent avant la confirmation.** C'est la banque qui nous prévient du paiement, par un appel séparé qui arrive parfois quelques secondes après le retour du voyageur sur votre page. Au premier appel, `status` peut donc valoir encore `pending` : affichez une attente, réinterrogez, et ne concluez **jamais** à un échec sur une seule réponse.
>
> **Mais bornez la relance** : cinq tentatives au maximum, espacées de deux à trois secondes, puis affichez « nous vérifions votre paiement » et arrêtez. Traitez un `404` comme définitif. Notre pare-feu bannit temporairement une adresse qui accumule les réponses `404` ou `403` : une relance sans limite depuis le navigateur ferait bannir le voyageur lui-même.

8. Être prévenu par webhook

Plutôt que d'interroger l'API, votre serveur peut être prévenu. Une fois l'adresse renseignée dans **Paramètres, onglet API**, rubrique **Rappel automatique de votre site**, nous y envoyons un POST JSON à chaque événement :

event	Cas
`booking.confirmed`	Le paiement a abouti, la réservation existe.
`booking.cancelled`	La réservation a été annulée.
`booking.expired`	Le voyageur n'a pas payé dans les trente minutes.

> **Jusqu'à cinq minutes de délai.** Les envois partent par vagues, pas à l'instant même. C'est sans conséquence pour enregistrer une réservation ou déclencher un e-mail, mais ne construisez pas dessus un écran que le voyageur regarde en direct : pour lui, gardez `booking_status`.

```
POST https://votre-site.fr/votre-adresse
X-EasyConcierge-Signature: t=1790076972,v1=65efbc0d…

{ "event": "booking.confirmed",
  "sent_at": "2026-07-14T09:12:33+00:00",
  "data": { "public_booking_id": 42, "booking_id": 1337, "status": "confirmed",
            "payment_status": "succeeded", "property_id": 22,
            "check_in": "2026-07-14", "check_out": "2026-07-19", "nights": 5,
            "total": 1067, "currency": "EUR",
            "guest_name": "Jean Dupont", "guest_email": "jean@exemple.fr",
            "paid_at": "…", "confirmed_at": "…" } }
```

Vérifier la signature

Même principe que Stripe : un HMAC-SHA256 calculé sur `<horodatage>.<corps brut>` avec le secret de signature. **Signez le corps brut**, pas un JSON réencodé : le moindre espace change le résultat.

```
$corps  = file_get_contents('php://input');
$entete = $_SERVER['HTTP_X_EASYCONCIERGE_SIGNATURE'] ?? '';
preg_match('/t=(\d+),v1=([a-f0-9]+)/', $entete, $m);

// rejeter ce qui est trop vieux : une signature valide rejouée reste valide
if (!$m || abs(time() - (int) $m[1]) > 300) { http_response_code(400); exit; }

$attendu = hash_hmac('sha256', $m[1] . '.' . $corps, VOTRE_SECRET);
if (!hash_equals($attendu, $m[2])) { http_response_code(401); exit; }

http_response_code(200);   // répondez 2xx, sinon nous retentons
```

Répondez vite, et en 2xx. Tout autre code déclenche de nouvelles tentatives : cinq essais en tout, le premier immédiat, puis 1, 5, 30 et 120 minutes après chaque échec, soit un peu plus de deux heures et demie, après quoi l'événement est abandonné. Accusez réception d'abord, travaillez ensuite. **Un même événement peut arriver deux fois** (si votre réponse s'est perdue, par exemple) : appuyez-vous sur `public_booking_id` pour ignorer un doublon. Gardez `booking_status` comme filet de sécurité.

9. Le parcours complet en JavaScript

```
const API = 'https://easy-concierge.fr/guest/{dossier}/public/booking/api.php';

// 0. la clé du captcha pour ce site (section 6), puis afficher le widget
const { sitekey } = await (await fetch(`${API}?action=turnstile`)).json();
if (sitekey) turnstile.render('#captcha', { sitekey });

// 1. le devis, options comprises
const devis = await (await fetch(`${API}?action=calculate&property_id=22`
  + `&check_in=2026-07-14&check_out=2026-07-19&guests=4&options=3`)).json();
afficher(devis.breakdown.total, devis.deposit.amount);   // total, et à régler maintenant

// 2. la réservation
const r = await (await fetch(API, {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    action: 'book', property_id: 22,
    check_in: '2026-07-14', check_out: '2026-07-19', guests: 4,
    guest_name: 'Jean Dupont', guest_email: 'jean@exemple.fr',
    options: [3],
    due_lines: devis.deposit.due_lines,                  // garde anti-écart
    'cf-turnstile-response': turnstile.getResponse(),    // jeton du widget (section 6)
    return_url: 'https://mon-site.fr/merci',             // domaine à déclarer
    cancel_url: 'https://mon-site.fr/reservation'
  })
})).json();
if (r.stripe_url) location.href = r.stripe_url;

// 3. sur /merci, le session_id est dans l'adresse
const sid = new URLSearchParams(location.search).get('session_id');
let etat;
for (let essai = 0; essai < 5; essai++) {                  // relance BORNÉE
  const rep = await fetch(`${API}?action=booking_status&session_id=${sid}`);
```

```
  if (rep.status === 404) break;                          // inconnu : définitif
  etat = await rep.json();
  if (etat.status !== 'pending') break;                   // confirmée, annulée ou expirée
  await new Promise(t => setTimeout(t, 2500));
}
// toujours `pending` après cinq essais ? afficher une attente, ne pas boucler.
```

10. Limites à connaître

- **Blocage de 30 minutes.** Une réservation non payée bloque les dates pendant 30 minutes, puis se libère seule.
- **Cadence.** Trois réservations en attente au maximum par logement, et un nombre limité de créations par heure et par adresse (sauf clé Site internet, section 6). Au-delà, l'API répond `429`.
- **Codes de remise.** Dix codes refusés par heure et par adresse, au-delà le code est écarté avec un message d'attente. Limite levée pour un appel portant une clé Site internet valide (section 6).
- **Pas de panier.** Une réservation porte un seul logement.
- **Ni annulation ni modification** par l'API : c'est la conciergerie qui le fait dans le logiciel.
- **Webhooks limités aux réservations.** Rien n'est envoyé sur un changement de tarif ou de disponibilité.
- **Pas d'environnement de test séparé.** Utilisez le **Mode test : essai à blanc** d'un logement (section 1).
- **Caution.** Elle n'est pas prise par cette API : `default_deposit_amount` sert seulement à l'annoncer.

Les exemples de ce document sont fictifs. Pour une question technique, écrivez à contact@easy-concierge.fr en précisant le dossier de la conciergerie, l'action appelée, l'heure de l'appel et la réponse reçue.