

Skor Tabelası (ScoreSay) rezervasyon entegrasyonu

API sürümü: 2026-10-01 · **Okuyucu:** kulübün rezervasyon sistemini geliştiren ekip

Bu entegrasyon kulübün rezervasyon sistemini Skor Tabelası kort tabletlerine bağlar:

- Rezervasyonları siz gönderirsiniz:** kort, başlangıç ve bitiş saati, oyuncu adları. Her değişikliği ve iptali olduğu anda gönderirsiniz.
- Kort tableti adları saatinde gösterir.** Rezervasyon başlamadan 10 dakika önce, o kortun tableti oyuncuları ve rezervasyon saatini bir kartta gösterir. Oyuncular **MAÇI BAŞLAT**'a basar ve skoru tablette tutar.
- Sonucu size yollarız.** Maç bitince webhook adresinize imzalı bir `match.finished` olayı `POST` ederiz: kazanan ve set skorları.

Rezervasyonları API'ye göndermek yerine **kendi servisinizden okumamızı** da isteyebilirsiniz: bkz. [10. Çekme modu](#).

Entegrasyon her kulüp ve her rezervasyon sistemi için aynıdır. Birden çok kulübe hizmet veren bir rezervasyon sistemi her kulüp için ayrı API anahtarı kullanır.

1. Kurulum (kulüp başına bir kez)

Kulüp bunu Skor Tabelası kulüp panelinde yapar ([skortabelasi.com/panel/](#) → **Entegrasyon** sekmesi). Kulüp yöneticisi ya da sahibi size şunları verir:

Ne	Nereden	Ne yaparsınız
API adresi	Panel, Entegrasyon sekmesinin üstü	Her isteğin öneki
API anahtarı (<code>ssk_live_...</code>)	Panel → <i>API anahtarı</i> → <i>Anahtar üret</i> (yalnız bir kez gösterilir)	<code>Authorization: Bearer <anahtar></code> olarak gönderirsiniz
Kort numaraları	Panel → <i>Kort numaraları</i> ya da <code>GET /v1/club</code>	Kendi kort kimliklerinizi bu numaralara eşlersiniz
Webhook imza anahtarı (<code>whsec_...</code>)	Panel → <i>Sonuç adresi</i> → <i>İmza anahtarını göster</i>	<code>ScoreSay-Signature</code> başlığını doğrularsınız

Siz de kulübe **webhook adresinizi** (https) verirsiniz; yönetici panele girer. **Test olayı gönder** düğmesi adresinize hemen örnek bir sonuç yollar.

API adresi `https://skortabelasi.com/api/club/v1` 'dir. Aynı yol `https://scoresay.com/api/club/v1` adresinde de çalışır.

Kontrol listesi

- ☐ `GET /v1/club` kulübü ve kortlarını döndürüyor (anahtar çalışıyor).
- ☐ Kort kimlikleriniz Skor Tabelası kort numaralarına eşlendi.
- ☐ Sisteminizde rezervasyon oluşturmak, değiştirmek ve iptal etmek `POST /v1/bookings` çağırıyor.
- ☐ Webhook'unuz imzayı doğruluyor, 10 saniye içinde `2xx` dönüyor ve tekrar gelen olay kimliklerini yok sayıyor.
- ☐ Panelden (ya da `POST /v1/webhooks/test` ile) gönderilen test olayı geliyor ve doğrulanıyor.

Önerilen kurulum

Sorunsuz ve güvenli bir entegrasyon için bunu öneriyoruz:

- Takımları gönderin:** kim kiminle oynuyor belliyse `teams` alanını kullanın (`[[A1, A2], [B1, B2]]`). Ad olarak "Ad Soyad" ya da "Ad S." gönderin; her oyuncuya kendi üye kimliğinizi (`id`) ekleyin.
- Her değişiklikte tam hâli gönderin:** rezervasyon oluşturulduğunda, değiştiğinde (kort, saat, oyuncu) ve iptal edildiğinde `POST /v1/bookings` ile rezervasyonun **tamamını** gönderin. Rezervasyonun sizdeki son değişme zamanını `updatedAt` olarak ekleyin; sırası karışan eski istek böylece kendiliğinden yok sayılır. İstek başarısız olursa (ağ hatası, `5xx`, `429`) aynı gövdeyle yeniden deneyin.
- Sonucu webhook ile alın, olay akışıyla tamamlayın:** sonuç için imzalı `https` webhook birincil yoldur (maç bittikten yaklaşık 2 dakika sonra gelir). Ayrıca saatte bir `GET /v1/events?after=<son imleç>` çağırın: adresiniz bir süre erişilemez kalsa bile hiçbir sonuç kaybolmaz. İki yoldan gelen aynı olayı `id` ile tekilleştirin.
- İlk kurulumda toplu yükleme:** canlıya geçerken önümüzdeki günlerin rezervasyonlarını `POST /v1/bookings/batch` ile bir kez yükleyin.

2. Kurallar

- Kimlik doğrulama:** her istek `Authorization: Bearer ssk_live_...` taşır. Anahtar tek bir kulübe aittir. Aynı anda en çok iki anahtar etkin olabilir; anahtar kesintisiz yenilenir.
- Biçim:** giden ve gelen JSON, UTF-8. Başarılı yanıtlarda `"ok": true` vardır.
- Hatalar:** `{"ok": false, "error": "<kod>", ...ayrıntı}` ve HTTP durumu (bkz. [Hatalar](#)).
- Saatler:** açık ofsetli ISO 8601, örneğin `2026-10-07T18:00:00+03:00` ya da `2026-10-07T15:00:00Z` . Ofsetsiz saat reddedilir (`bad_time`), çünkü hangi saat diliminde olduğu belirsizdir. Yanıtlar saati sizin gönderdiğiniz ofsetle döndürür.
- Hız sınırı:** anahtar başına dakikada 120 istek. Fazlası `429 rate_limited` ve `retryAfter` (saniye) döner. Aynı IP'den tekrarlanan başarısız kimlik denemeleri de sınırlıdır.
- Bilinmeyen alanlar:** yanıtlarımızda ve olaylarımızda ileriki sürümlerde yeni alanlar olabilir. Bilmediğiniz alanı yok sayın.

3. Rezervasyonlar

3.1 Oluştur ya da güncelle — `POST /v1/bookings`

Oluşturma, güncelleme ve iptal için tek uç: her zaman rezervasyonun **tam ve güncel hâlini** gönderin. Rezervasyon **sizin** `id` 'nizle tanınır. Aynı hâli tekrar göndermek zararsızdır (`"result": "unchanged"`).

```
POST /api/club/v1/bookings
Authorization: Bearer ssk_live_...
Content-Type: application/json
```

```
{
  "id": "R-2026-10-07-0042",
  "court": 2,
  "startsAt": "2026-10-07T18:00:00+03:00",
  "endsAt": "2026-10-07T19:30:00+03:00",
  "status": "confirmed",
  "label": "Pazar ligi",
  "teams": [
    [ { "name": "Ali Kaya", "id": "uye-118" }, { "name": "Veli Demir", "id": "uye-204" } ],
    [ { "name": "Can Yıldız", "id": "uye-77" }, { "name": "Ece Ak" } ]
  ],
  "updatedAt": "2026-10-07T11:12:40+03:00"
}
```

Alan	Zorunlu	Açıklama
id	evet	Sizin rezervasyon kimliğiniz. [A-Za-z0-9._:-] karakterlerinden 1-100 uzunlukta dize (sayı da kabul edilir). Kulüp içinde tekil.
court	evet	Skor Tabelası kort numarası (bkz. GET /v1/club).
startsAt , endsAt	evet	Ofsetli ISO 8601. Bitiş başlangıçtan sonra; en çok 12 saat; başlangıç 7 gün öncesi ile 180 gün sonrası arasında.
status	hayır	confirmed (varsayılan) ya da cancelled.
teams	hayır	İki taraf, her biri 0-2 oyuncu: [[oyuncu, oyuncu], [oyuncu, oyuncu]]. Yanıttaki [{ "side": "A", "players": [...] }, ...] biçimi de kabul edilir.
players	hayır	teams yerine: tarafları bilmiyorsanız en çok 4 oyunculuk düz liste. 2 oyuncu → 1'e 1; 3 → ilk ikisi üçüncüye karşı; 4 → ilk ikisi son ikisine karşı. Oyuncular başlatmadan önce tablette adları düzenleyebilir.
label	hayır	Kartta görünen kısa metin (en çok 60 karakter), örneğin Ders ya da Lig.
sport	hayır	padel, tennis, pickleball ya da padbol. Kortun tanımlı sporu önceliklidir; bu alan yalnız sporu tanımsız kortta kullanılır.
version	hayır	Artan tamsayı (ya da updatedAt, ofsetli ISO). Saklanandan eski sürümle gelen istek yok sayılır ("result": "stale"); sırası karışan yeniden denemelere karşı korur. İkisi de yoksa son yazan kazanır.

Oyuncu { "name": "Ali Kaya", "id": "uye-118" } ya da yalnız ad dizesidir. name 1-40 karakter. id isteğe bağlıdır (sizin üye kimliğiniz, en çok 100 karakter). Rezervasyon yanıtlarında aynen geri döner.

Tablette adlar: kart ad başına en çok 24 karakter gösterir. Uzun adlar ilk ad + soyadının baş harfine kısalır ("Ahmet Mehmet Yılmazoğlu" → "Ahmet Y."). Aynı rezervasyonda iki aynı ad "Ali" ve "Ali 2" olur. Oyuncuların tanıyacağı adları gönderin; nüfus kaydındaki tam ad gerekmez.

Yanıt

Durum	result	Anlamı
201	created	Yeni rezervasyon.
200	updated	Bir şey değişti.
200	unchanged	Saklananla aynı.
200	stale	version / updatedAt saklanandan eski; yok sayıldı. Saklanan rezervasyon döner.

```
{
  "ok": true,
  "result": "created",
  "booking": {
    "id": "R-2026-10-07-0042",
    "court": 2,
    "startsAt": "2026-10-07T18:00:00+03:00",
    "endsAt": "2026-10-07T19:30:00+03:00",
    "status": "confirmed",
    "sport": null,
    "label": "Pazar ligi",
    "teams": [
      { "side": "A", "players": [ { "name": "Ali Kaya", "id": "uye-118" }, { "name": "Veli Demir", "id": "uye-204" } ] },
      { "side": "B", "players": [ { "name": "Can Yıldız", "id": "uye-77" }, { "name": "Ece Ak", "id": null } ] }
    ],
    "version": null,
    "createdAt": "2026-10-07T08:12:40.118Z",
    "updatedAt": "2026-10-07T08:12:40.118Z"
  }
}
```

Kort kulübün mevcut Skor Tabelası paketinin dışındaysa rezervasyon yine saklanır, yanıtta "warnings": ["court_locked"] olur. Kulübün paketi o kortu kapsayana dek kortun tableti maç başlatamaz.

3.2 İptal — DELETE /v1/bookings/{id}

"status": "cancelled" göndermekle aynıdır. İdempotenttir: iki kez iptal "result": "unchanged" döner; bilinmeyen kimlik 200 ve "existed": false döner. Kart tabletten saniyeler içinde kalkar.

3.3 İlk yükleme — POST /v1/bookings/batch

{ "bookings": [...en çok 200 rezervasyon...] }. Her kayıt POST /v1/bookings gibi işlenir; geçersiz bir kayıt ötekileri durdurmaz.

```
{ "ok": true, "results": [
  { "id": "R-1", "ok": true, "result": "created" },
  { "id": "R-2", "ok": false, "error": "unknown_court", "court": 7 }
] }
```

3.4 Okuma — GET /v1/bookings ve GET /v1/bookings/{id}

- GET /v1/bookings?from=...&to=...&court=2&status=confirmed : aralıklı kesişen rezervasyonlar. from / to ofsetli ISO; varsayılan son 24 saatten 7 gün sonrasına; aralık en çok 31 gün.
- GET /v1/bookings/{id} : rezervasyon ve matches — size gönderilen sonuç, webhook'taki data.match biçiminde (sonuç henüz gönderilmediyse boş). Kaçan bir webhook'u tamamlamak için kullanın.

3.5 Kulüp ve kortlar — GET /v1/club

```
{
  "ok": true,
  "apiVersion": "2026-10-01",
  "club": { "id": "cl_...", "slug": "sports-international", "name": "Sports International" },
  "courts": [
    { "court": 1, "name": "Cam Kort", "sport": "padel", "locked": false },
    { "court": 2, "name": "Merkez Kort", "sport": "tennis", "locked": false }
  ],
  "webhook": { "url": "https://rezervasyon.ornek.com/scoresay/webhook", "enabled": true }
}
```

Kulübü club.id ile tanıyın; slug değişebilir.

4. Kortta ne olur?

- Tablet kendi kortunun sonraki 24 saatlik rezervasyonlarını alır ve saklar; kart internet yokken de çalışır.
- **** startsAt 'tan 10 dakika önceden endsAt 'a kadar****, kortu bir maç ya da turnuva tutmuyorsa tablet rezervasyon kartını gösterir. İki rezervasyon çakışırsa (art arda saatler) geç başlayanı gösterilir.
- **MAÇI BAŞLAT** adları dolu maç kurulumunu açar. Oyuncular adları ve kuralları (set sayısı, altın sayısı...) değiştirip başlatabilir.
- Karttan başlatılan maç rezervasyona bağlanır. Rezervasyonun saat penceresinde o kortta kart kullanılmadan başlatılan maç da (kortun QR koduyla telefonda ya da eski bir tablet uygulamasından) saatiyle bağlanır.
- Bir rezervasyon için sonuç **bir kez** gönderilir: maç bittikten yaklaşık 2 dakika sonra, son hâliyle. Oyuncular bu sürede tablette skoru düzeltebilir; düzeltilmiş skor gider.
- Gönderildikten sonraki düzeltmeler ve aynı rezervasyonda sonradan oynanan başka maçlar gönderilmez.
- Kimse oynamazsa hiçbir şey gönderilmez.
- Sonuç, maç kurallarıyla bittiğinde ya da oyuncular **Maçı bitir** dediğinde gider. Bitirilmeden bırakılan maç (tablette **Maçı iptal et** ya da açık maçın üstüne yeni maç) gönderilmez; skor tutmak istemeyen oyuncular için de böyledir.
- Değişiklik ve iptaller tablete saniyeler içinde (en geç bir dakikada) ulaşır.
- Skor Tabelası'nda düzenlenen turnuvalar önceliklidir: kortu bir turnuva tutarken kart gösterilmez.

5. Sonuçlar — webhook

Rezervasyona bağlı bir maç bitince, yaklaşık 2 dakika sonra webhook adresinize bir olay `POST` ederiz. Bir rezervasyon için sonuç bir kez gönderilir.

5.1 İstek

```
POST /scoresay/webhook
Content-Type: application/json
User-Agent: ScoreSay-Webhooks/1
ScoreSay-Event-Id: evt_3J9xkq2Lw8Q
ScoreSay-Event-Type: match.finished
ScoreSay-Signature: t=1791399000,v1=5b1f0d...e2a9
```

```
{
  "id": "evt_3J9xkq2Lw8Q",
  "type": "match.finished",
  "test": false,
  "data": {
    "booking": { "id": "R-2026-10-07-0042" },
    "match": {
      "id": "m_Q2b8Xk1dPzE",
      "revision": 1,
      "court": 2,
      "winner": "A",
      "score": {
        "sets": [ { "a": 6, "b": 4 }, { "a": 6, "b": 7 }, { "a": 10, "b": 8 } ],
        "setsWon": { "a": 2, "b": 1 }
      }
    }
  }
}
```

Sonuç bilerek sade tutulur; ileride yeni alanlar eklenebilir. Bilmediğiniz alanı yok sayın.

Alan	Açıklama
id	Olay kimliği (= ScoreSay-Event-Id). Aynı olay birden çok kez gelebilir; bununla tekillleştirin.
type	Şimdilik yalnız match.finished .
test	Panelden ya da POST /v1/webhooks/test ile gönderilen deneme olayında true ; kaydetmeyin.
data.booking.id	Sizin rezervasyon kimliğiniz. Rezervasyonsuz deneme olayında data.booking null olur.
data.match.id	Skor Tabelası maç kimliği.
data.match.revision	Şimdilik her zaman 1. Aynı rezervasyon için sonuç yeniden gönderilirse artar; büyük olan geçerlidir.
data.match.court	Maçın oynandığı kort (Skor Tabelası kort numarası).
data.match.winner	"A" , "B" ya da null (kazanan yok). Kazananı yalnız tamamlanan setler belirler: oyuncular Maçı bitir dediğinde kazanan set sayısı eşitse — hiç set tamamlanmadıysa da — null gelir; yarım kalan setin oyunları sayılmaz (3-6, 6-0 → 1-1, kazanan yok). A tarafı her zaman rezervasyonun birinci takımıdır : oyuncular tablette takımların yerini değiştirirse de skor ve winner sizin sıranıza göre gelir.
data.match.score.sets	Set başına bir kayıt: a A tarafının, b B tarafının skoru. Oyun sayısıdır; süper tie-break ve ralli sayımlı sporlarda (pickleball, padbol) sayıdır. Oyuncular set ortasında Maçı bitir dediyse son set o anki skoruyla gelir.
data.match.score.setswon	A ve B tarafının kazandığı set sayısı.

5.2 İmzayı doğrulama

ScoreSay-Signature: t=<unix saniye>,v1=<hex>[,v1=<hex>]

1. **Ham istek gövdesini** geldiği gibi alın (JSON'u yeniden serileştirmeyin).
2. HMAC-SHA256(anahtar = imza anahtarı, mesaj = t + "." + hamGövde) değerini küçük harf hex olarak hesaplayın.
3. **Herhangi bir** v1 değerine eşitse kabul edin (sabit zamanlı karşılaştırma).
4. t saatinizden 5 dakikadan fazla uzaksa reddedin (tekrar oynatma koruması).

Kulüp imza anahtarını yenilerse 24 saat boyunca her olay iki v1 taşır: biri yeni, biri eski anahtarla. Node.js, PHP ve C# örnekleri [examples/](examples/) klasöründe.

5.3 Teslim kuralları

- **10 saniye içinde** herhangi bir 2xx ile yanıt verin. Ağır işi yanıtta sonra yapın.
- Başka her şey (2xx dışı, zaman aşımı, bağlantı hatası, yönlendirme) başarısızdır. **Yönlendirme izlenmez.**
- Başarısız olay 30 sn, 2 dk, 10 dk, 30 dk, 1 sa, 2 sa, 4 sa, 8 sa, 12 sa, 24 sa ve 24 sa sonra yeniden denenir (yaklaşık 3 günde 12 deneme). Sonra başarısız işaretlenir; kulüp panelden yeniden gönderebilir.
- Teslim **en az bir kezdir**: aynı olay birden çok kez gelebilir. ScoreSay-Event-Id (= id) ile tekrarları ayıklayın.
- Webhook adresi https olmalı ve internetten erişilebilmeli.

5.4 Webhook olmadan — GET /v1/events

Webhook alamıyorsanız ya da boşlukları doldurmak için aynı olayları API'den okuyun:

```
GET /v1/events?after=<imleç>&limit=100
→ { "ok": true, "events": [ { "cursor": "41", ...webhook gövdesiyle aynı JSON... } ], "next": "57" }
```

next değerini saklayıp sonraki çağrıda after olarak verin. Olaylar 30 gün tutulur.

5.5 Test olayı — POST /v1/webhooks/test

Tanımlı webhook'a "test": true işaretli örnek bir match.finished yollar ve denemeyi bekler: { "ok": true, "eventId": "evt_...", "delivered": true, "status": 200 }. { "bookingId": "<sizin rezervasyon kimliğiniz>" } ile örnek o rezervasyonun numarasını ve kortunu kullanır. Test olayları yeniden denenmez. Saatte en çok 10.

6. Sınır durumlar

Durum	Davranış
Rezervasyon maç oynandıktan sonra geldi	Rezervasyon gelince (8 gün içinde) maç bağlanır ve sonuç gönderilir.
Rezervasyon başka korta ya da saate taşındı	Tabletler saniyeler içinde güncellenir. Zaten bağlanmış maçlar bağlı kalır.
Aynı kortta art arda rezervasyonlar	Her maç, başlangıcını penceresinde içeren rezervasyona aittir; ikisi de içeriyorsa (10 dakikalık erken pencere) geç başlayan kazanır.
Oyuncular erken geldi	Kart startsAt 'tan 10 dakika önce açılır; bu pencerede başlatılan maç rezervasyona aittir.
Kortta internet yok	Kart saklanan listeden çalışır; sonuç bağlantı gelince yüklenip gönderilir (7 güne kadar sonra).
Entegrasyon kurulmadan önce biten maçlar	Gönderilmez.
Maç bitirilmeden bırakıldı	Gönderilmez; GET /v1/bookings/{id} yanıtında da görünmez.

7. Hatalar

HTTP	error	Anlamı
400	bad_json	Gövde geçerli JSON değil.
400	bad_request	Eksik ya da bozuk alan (bkz. field).
400	bad_id	id yok ya da [A-Za-z0-9._:-]{1,100} değil.
400	bad_time	Ofsetsiz saat, bitiş başlangıçtan önce, 12 saatten uzun ya da aralık dışı (bkz. field, reason).
400	bad_players	Taraf başına 2'den, toplam 4'ten fazla oyuncu; boş ya da 40 karakterden uzun ad; teams ve players birlikte.
400	bad_status	status confirmed / cancelled değil.
400	bad_sport	Bilinmeyen sport.
401	unauthorized	Anahtar yok, yanlış ya da iptal edilmiş.
404	not_found	Bilinmeyen rezervasyon (GET /v1/bookings/{id} , bookingId 'li test).
409	too_many_bookings	Kulübün 5.000'den fazla gelecek rezervasyonu.
409	no_webhook	Test olayı istendi ama webhook adresi tanımlı değil.
422	unknown_court	Kulüpte bu numaralı kort yok.
429	rate_limited	Çok fazla istek; retryAfter saniye sonra tekrar deneyin.

5xx ve ağ hataları güvenle yeniden denenebilir: bu API'deki her yazım idempotenttir.

8. Kişisel veriler (KVKK)

- Rezervasyon verilerinin veri sorumlusu kulüptür; Skor Tabelası oyuncu adlarını yalnız kulübün kort tabletinde göstermek ve sonucu kulübün rezervasyon sistemine döndürmek için işler.
- Yalnız görünen ad ve isteğe bağlı üye kimliği gönderin. Telefon, e-posta gibi diğer kişisel veriler gerekmez ve gönderilmemelidir.
- Rezervasyonlar bitişten 30 gün sonra, olaylar 30 gün sonra silinir. Maç sonuçları, kulübün kortlarında oynanan her maç gibi Skor Tabelası'ndaki kulüp maç geçmişinde kalır.
- Rezervasyon adları oyundan önce kulübün herkese açık ekranlarında yayınlanmaz; yalnız rezerve edilen kortun tabletine ve o tabletteki kareyi (QR) okutup maçı kendi telefonundan başlatan oyuncunun telefonuna gider. Oyuncular maçı başlatınca maç, diğer maçlar gibi kulübün ekranlarında görünür.

9. Sürümler

API yolda sürümlenir (/v1). v1 içinde yalnız alan, uç ve olay türü ekleriz; hiçbirini kaldırmaz ya da yeniden adlandırmayız. Bilmediğiniz alanları ve olay türlerini yok sayın.

10. Çekme modu

Rezervasyonları `POST /v1/bookings` ile göndermek yerine kendi servisinizden okumamızı isteyebilirsiniz. Kulüp, servisinizin adresini ve anahtarını panelde **Entegrasyon → Rezervasyon sisteminden çek** kartına girer (biçim: *Standart*). Sonuçlar yine 5. bölümdeki gibi gelir; `data.booking.id` sizin rezervasyon numaranızdır.

10.1 İsteğimiz

```
GET https://rezervasyon.ornek.com/scoresay/bookings?from=2026-10-09T21:00:00Z&to=2026-10-11T21:00:00Z
Authorization: Bearer <kulübe özel anahtar>
Accept: application/json
User-Agent: ScoreSay-BookingPull/1
```

- Adresi siz belirlersiniz; kendi parametrelerinizi (örneğin `?kulup=1`) içerebilir. Anahtar adres satırında olamaz: `Authorization` başlığında gider.
- `from` ve `to` UTC'dir (`Z` ile biter): kulübün bugününün 00:00'ından 48 saat sonrasına. İkisini yok sayıp önümüzdeki 48 saatin rezervasyonlarını döndürmeniz de yeterlidir; aralık dışını biz eleriz.
- Yanıtı 10 saniye bekleriz; yanıt en çok 256 KB olabilir. Yönlendirme izlenmez.

10.2 Yanıtınız

200, `application/json`, UTF-8. Rezervasyon nesnesi `POST /v1/bookings` ile aynıdır.

```
{
  "bookings": [
    {
      "id": "R-2026-10-10-0042",
      "court": 1,
      "startsAt": "2026-10-10T23:00:00+03:00",
      "endsAt": "2026-10-11T00:00:00+03:00",
      "status": "confirmed",
      "teams": [
        [ { "name": "Ali Kaya", "id": "uye-118" }, { "name": "Veli Demir", "id": "uye-204" } ],
        [ { "name": "Can Yıldız", "id": "uye-77" }, { "name": "Ece Ak" } ]
      ]
    }
  ]
}
```

Alan	Zorunlu	Açıklama
id	evet	Sizin rezervasyon numaranız (metin ya da sayı; harf, rakam, ., _, - ; en çok 64 karakter). Sonuçta data.booking.id olarak aynen geri gelir.
court	evet	Sizin kort numaranız (1–99). Kulüp panelde kendi kortuna eşler; aynıysa eşleme gerekmez.
startsAt	evet	Ofsetli ISO 8601. Ofsetsiz saat kabul edilmez.
endsAt	hayır	Yoksa kulübün paneldeki maç süresi kullanılır (varsayılan 60 dakika).
status	hayır	confirmed (varsayılan) ya da cancelled .
teams	evet	İki takım, her takımda en çok 2 oyuncu: { "name": ..., "id": ... } ya da yalnız ad. İlk takım A, ikinci takım B. id isteğe bağlıdır (en çok 100 karakter).
label , sport	hayır	POST /v1/bookings ile aynı.

- **Kart, takımlar tamamlanınca gösterilir:** iki takım da dolu ve eşit olmalı (1'e 1 ya da 2'ye 2); padel ve padbol kortunda 2'ye 2. Oyuncusu eksik rezervasyon listede kalabilir; adlar dolunca aynı id ile güncel hâlini döndürün.
- **İptal:** status: "cancelled" gönderin ya da kaydı listeden çıkarın. Henüz başlamamış bir rezervasyon listede yoksa iptal sayılır.
- **Değişiklik:** kort, saat ya da oyuncu değişirse aynı id ile yeni hâlini döndürün; kart kendiliğinden güncellenir.
- **Hata:** 2xx dışı yanıtta ya da zaman aşımında mevcut kartlara dokunmayız; 5 dakika sonra yeniden sorarız.
- **Adlar:** tamamı büyük harfle yazılmış ad düzeltilir; tablette ad "Ad S." biçiminde görünür.

10.3 Ne sıklıkla sorarız

Ne zaman	Sıklık
Düzenli	15 dakikada bir
Her maçtan önce	Başlangıçtan 5 dakika önce bir kez
Oyuncusu eksik rezervasyon	Başlangıçtan 5 dakika önce başlar, 10 dakika sonrasına dek dakikada bir
Kortta maç bitince	Hemen bir kez (sıradaki maçın kartı için)
Hata aldıktan sonra	5 dakika beklenir

11. Güvenlik özeti

Konu	Kural
Taşıma	İki yönde de yalnız HTTPS (443). Düz <code>http</code> ve yönlendirme kabul edilmez.
Sunucumuzun IP adresi	Çekme isteklerimiz ve sonuç bildirimlerimiz tek sabit adresten çıkar: <code>147.93.84.183</code> . Güvenlik duvarınızda ya da hız sınırınızda bu adrese izin verin.
API anahtarı (<code>ssk_live_...</code>)	Kulübe özeldir, yalnız özeti saklanır; aynı anda en çok iki anahtar etkin olur (kesintisiz yenileme).
Çekme anahtarı	Kulübe özel olmalı; en az 16 karakter kabul edilir, 32 rastgele karakter öneririz. <code>Authorization</code> başlığında gider (adres satırındaki anahtar sunucu kayıtlarına düşer). Yalnız kaydedildiği alan adına gönderilir.
Sonuç imzası	Her sonuç isteği imzalıdır (5.2). İmzası tutmayan isteği işlemeyin.
Tekrar oynatma	İmzadaki <code>t</code> 5 dakikadan eskiyse reddedin; aynı olay <code>id</code> değerini ikinci kez işlemeyin.
Kişisel veri	Yalnız görünen ad ve isteğe bağlı üye kimliği (8. bölüm).

Destek: info@skortabelasi.com

Ek — İmza doğrulama örnekleri

Node.js · examples/verify-signature.mjs

```
// ScoreSay webhook signature check - Node.js (no dependencies).
//
// Header: ScoreSay-Signature: t=<unix seconds>,v1=<hex>[,v1=<hex>]
// v1 = HMAC-SHA256(signing secret, t + "." + raw request body), lowercase hex.
// During a secret rotation the header carries two v1 values; accept if any matches.
import crypto from 'node:crypto';

/**
 * @param {string} header      value of the ScoreSay-Signature header
 * @param {string|Buffer} rawBody the request body exactly as received (do not re-serialise the JSON)
 * @param {string} secret      the club's signing secret (whsec_...)
 * @param {number} [toleranceSec=300] maximum clock difference (replay protection)
 * @param {number} [nowSec]      current unix time in seconds (for tests)
 * @returns {boolean}
 */
export function verifyScoreSaySignature(header, rawBody, secret, toleranceSec = 300, nowSec =
Math.floor(Date.now() / 1000)) {
  if (typeof header !== 'string' || !secret) return false;
  let t = null;
  const sigs = [];
  for (const part of header.split(',')) {
    const [k, v] = part.split('=', 2).map((s) => (s || '').trim());
    if (k === 't' && /^d+$/i.test(v)) t = Number(v);
    else if (k === 'v1' && /^[0-9a-f]{64}$/i.test(v)) sigs.push(v);
  }
  if (t === null || !sigs.length) return false;
  if (Math.abs(nowSec - t) > toleranceSec) return false;
  const body = Buffer.isBuffer(rawBody) ? rawBody.toString('utf8') : String(rawBody);
  const expected = Buffer.from(crypto.createHmac('sha256', secret).update(t + '.' + body).digest('hex'), 'hex');
  return sigs.some((s) => {
    const got = Buffer.from(s, 'hex');
    return got.length === expected.length && crypto.timingSafeEqual(got, expected);
  });
}
```

```

<?php
// ScoreSay webhook signature check – PHP 7.3+ (no dependencies).
//
// Header: ScoreSay-Signature: t=<unix seconds>,v1=<hex>[,v1=<hex>]
// v1 = HMAC-SHA256(signing secret, t + "." + raw request body), lowercase hex.
// During a secret rotation the header carries two v1 values; accept if any matches.

function scoresay_verify_signature(string $header, string $rawBody, string $secret, int $toleranceSec = 300):
bool
{
    $t = null;
    $sigs = [];
    foreach (explode(',', $header) as $part) {
        $kv = explode('=', $part, 2);
        if (count($kv) !== 2) continue;
        $k = trim($kv[0]);
        $v = trim($kv[1]);
        if ($k === 't' && ctype_digit($v)) $t = (int) $v;
        elseif ($k === 'v1' && preg_match('/^[0-9a-f]{64}$/', $v)) $sigs[] = $v;
    }
    if ($t === null || !$sigs || $secret === '') return false;
    if (abs(time() - $t) > $toleranceSec) return false;
    $expected = hash_hmac('sha256', $t . '.' . $rawBody, $secret);
    foreach ($sigs as $s) {
        if (hash_equals($expected, $s)) return true;
    }
    return false;
}

// Usage (plain PHP endpoint):
//
// $raw = file_get_contents('php://input');           // raw body – do not json_decode first
// $sig = $_SERVER['HTTP_SCORESAY_SIGNATURE'] ?? '';
// if (!scoresay_verify_signature($sig, $raw, getenv('SCORESAY_WEBHOOK_SECRET'))) {
//     http_response_code(400);
//     exit;
// }
// $event = json_decode($raw, true);
// // de-duplicate by $event['id'], store the result keyed by $event['data']['match']['id']
// // (keep the highest $event['data']['match']['revision'])
// http_response_code(200);

```

C# / .NET · examples/VerifySignature.cs

```
// ScoreSay webhook signature check - C# / .NET 6+ (no dependencies).
//
// Header: ScoreSay-Signature: t=<unix seconds>,v1=<hex>[,v1=<hex>]
// v1 = HMAC-SHA256(signing secret, t + "." + raw request body), lowercase hex.
// During a secret rotation the header carries two v1 values; accept if any matches.
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

public static class ScoreSaySignature
{
    public static bool Verify(string header, string rawBody, string secret, int toleranceSec = 300)
    {
        if (string.IsNullOrEmpty(header) || string.IsNullOrEmpty(secret)) return false;
        long? t = null;
        var sigs = new List<string>();
        foreach (var part in header.Split(','))
        {
            var i = part.IndexOf('=');
            if (i < 0) continue;
            var k = part.Substring(0, i).Trim();
            var v = part.Substring(i + 1).Trim();
            if (k == "t" && long.TryParse(v, out var ts)) t = ts;
            else if (k == "v1" && v.Length == 64) sigs.Add(v);
        }
        if (t == null || sigs.Count == 0) return false;
        if (Math.Abs(DateTimeOffset.UtcNow.ToUnixTimeSeconds() - t.Value) > toleranceSec) return false;

        using var hmac = new HMACSHA256(Encoding.UTF8.GetBytes(secret));
        var expected = hmac.ComputeHash(Encoding.UTF8.GetBytes(t.Value + "." + rawBody));
        foreach (var s in sigs)
        {
            byte[] got;
            try { got = Convert.FromHexString(s); } catch (FormatException) { continue; }
            if (CryptographicOperations.FixedTimeEquals(got, expected)) return true;
        }
        return false;
    }
}

// ASP.NET Core minimal API usage:
//
// app.MapPost("/scoresay/webhook", async (HttpRequest req) => {
//     using var reader = new StreamReader(req.Body, Encoding.UTF8);
//     var raw = await reader.ReadToEndAsync(); // raw body - do not deserialize first
//     var sig = req.Headers["ScoreSay-Signature"].ToString();
//     if (!ScoreSaySignature.Verify(sig, raw, Environment.GetEnvironmentVariable("SCORESAY_WEBHOOK_SECRET")))
//         return Results.BadRequest();
//     // de-duplicate by event id; store the result keyed by data.match.id (keep the highest revision)
//     return Results.Ok();
// });
```

Makine tarafından okunur tanım: openapi.yaml (aynı klasörde).