

AVANT IT

AvantPOS · Manual de integración por API

Documento confidencial para socios de Avant IT

Versión del documento	1.0
Fecha	14/10/2026
Versión mínima de AvantPOS	2.58.3 (16/09/2026)
Soporte	soporte@avant-it.net · 93 694 05 02

Confidencial · Avant IT. Documento privado para socios que lo hayan solicitado. No publicar ni redistribuir sin autorización de Avant IT.

ÍNDICE

- 1** Qué permite y qué no
- 2** Requisitos y alcance
- 3** Alta: generar la clave
- 4** Autenticación
- 5** Referencia de endpoints
- 6** Eventos en tiempo real (SSE)
- 7** Avisos al local (POST /avisos)
- 8** Tokens de un solo uso (acceso desde el backoffice)
- 9** Límites y buenas prácticas
- 10** Protección de datos
- 11** Versionado y compromiso de estabilidad
- 12** Ejemplos completos

1. Qué permite y qué no

La API de integración de AvantPOS permite que un sistema externo del local (videovigilancia con IA, cartelera digital, business intelligence, un ERP, control de presencia...) **consulte en tiempo real** lo que ocurre en el punto de venta y **avise** al personal del local. Es una API de solo lectura, con una única excepción: el envío de avisos.

Lo que permite

- Consultar el estado del equipo: versión, reloj, zona horaria, nombre comercial y licencia.
- Listar los dispositivos (TPV, comanderas y tablets) y los usuarios del local.
- Consultar el estado de las mesas: libres u ocupadas, con su ticket, importe y camarero.
- Consultar el historial de operaciones y eventos (auditoría), paginado por id y filtrable por fechas y tipo.
- Recibir esas mismas operaciones **al segundo** por un flujo de eventos (SSE) con reanudación sin pérdidas.
- Enviar avisos al local: pantallas del TPV y comanderas, Telegram y correo.
- Validar tokens de un solo uso para abrir el servicio del integrador desde el backoffice.

Lo que NO permite

- Crear ni modificar tickets, líneas, cobros, productos, precios, mesas ni usuarios.
- Crear pedidos ni enviar comandas a cocina.
- Imprimir ni abrir el cajón.

Si su proyecto necesita escribir en AvantPOS (por ejemplo, volcar pedidos de otra plataforma), no está cubierto por esta API: contacte con Avant IT en soporte@avant-it.net para valorarlo.

Módulo de licencia. El código de la API no exige ningún módulo de pago: con una clave válida, todos los endpoints responden. El campo `licencia.modulos.cameras` de `GET /estado` informa de si el local tiene contratado el módulo «Cámaras con IA», pero AvantPOS no bloquea la API por ello; si su servicio lo requiere, es usted quien decide qué hacer. Sí depende de ese módulo la pestaña «Cámaras» del backoffice (sección 8). No confunda este módulo con el módulo «Integración Glovo / Just Eat», que es otra cosa y no afecta a esta API.

2. Requisitos y alcance

Dónde funciona. La API se sirve dentro de la red del local, en el equipo donde corre AvantPOS (el servidor del local, al que se conectan TPV, comanderas y tablets). La dirección base es:

```
http://IP-del-TPV:8085/api/integracion
```

El puerto por defecto es el 8085, el de red local que deja configurado el instalador (que también crea la regla de firewall «AvantPOS LAN» en el equipo). La IP es la del equipo donde corre AvantPOS; el backoffice la muestra al final de la pestaña «Sistemas externos» (por ejemplo

`http://192.168.0.15:8085/api/integracion/eventos`). Se recomienda reservar una IP fija o una reserva DHCP para ese equipo.

Dónde NO funciona

- **Acceso remoto (túnel del backoffice).** El acceso remoto solo permite administrar el local desde fuera, y por diseño bloquea `/api/integracion`: devuelve `403` con el código `remoto_no_permitido`. La razón es de seguridad: esta API da nombres de empleados y operaciones sensibles, y solo debe estar al alcance de la red física del local.
- **Instancias AvantPOS en la nube** (`{local}.tpv.avantpos.es`). Una instancia en la nube no está en la red del local y protege toda su API con la clave de acceso del local (responde `401` con el código `nube_acceso` a quien no la presente), de modo que un sistema externo no puede integrarse con ella.

Sin cifrar. La API va por **HTTP sin cifrar** dentro de la red del local. Recomendaciones:

- Use una red de confianza y **no exponga el puerto 8085 a internet** (ni con redirección de puertos en el router ni con VPN abiertas a terceros).
- Si el router lo permite, ponga el sistema del integrador y el servidor de AvantPOS en una VLAN propia, con reglas que solo dejen pasar el tráfico necesario.
- Trate la clave como una contraseña: quien la capture en la red podría leerla, por eso conviene limitar quién tiene acceso a esa red.

Un solo punto de acceso. Cada equipo con base de datos propia expone su propia API. Lo habitual (un servidor y comanderas) es un único punto de acceso.

3. Alta: generar la clave

La clave la genera el propio local (o Avant IT durante la puesta en marcha) desde el backoffice.

1. Abra el **backoffice** de AvantPOS con un usuario administrador.
2. Vaya a **Configuración > Integraciones** y abra la pestaña «**Sistemas externos**». (Hasta la versión 2.61.5 esta opción estaba en Sistema > Integraciones; el menú se rediseñó en la 2.61.6.)
3. Pulse «**Generar clave**».
4. Se muestra la clave completa **una sola vez**, con el formato `avp_` seguido de 48 caracteres hexadecimales (52 caracteres en total). Pulse «Copiar» y entréguela al integrador por un canal seguro. Cuando salga de esa pantalla no se puede volver a ver: AvantPOS solo guarda un resumen irreversible (hash) y el prefijo (los 10 primeros caracteres) para que el local reconozca cuál es.
5. La pantalla muestra además la fecha de creación y el último acceso de la clave actual, útil para comprobar que el sistema del integrador se está conectando.

Rotar la clave. Pulse «Generar clave nueva» (el botón cambia de nombre cuando ya hay una). El backoffice pide confirmación: la clave anterior **deja de funcionar en el acto** y cualquier sistema que la usara recibirá `401` hasta que se le configure la nueva. Los flujos SSE ya abiertos con la clave anterior no se cortan hasta que se reconecten, así que reinicie el cliente tras el cambio. No hay periodo de convivencia entre claves: coordine el cambio con el integrador.

Retirar la clave. Pulse «Quitar clave». La integración queda desactivada: todas las peticiones reciben `401`. Se puede volver a activar generando una clave nueva (la anterior no se recupera).

Solo hay una clave por local. Si trabajan varios integradores en el mismo local, comparten la clave; en ese caso conviene que el local la gestione y la rote cuando termine la relación con alguno de ellos.

4. Autenticación

Todas las peticiones a `/api/integracion/*` deben llevar la clave en una de estas dos cabeceras:

```
Authorization: Bearer avp_0123456789abcdef0123456789abcdef0123456789abcdef
```

```
X-API-Key: avp_0123456789abcdef0123456789abcdef0123456789abcdef
```

Si se envían las dos, tiene prioridad `Authorization: Bearer`. La clave no se admite en la URL (query string).

Errores

Código	Cuándo	Cuerpo
401	Clave ausente, incorrecta, retirada, o integración sin clave generada	<code>{"error":"Clave de integración ausente o incorrecta.", "codigo":"clave_integracion"}</code>
401	<code>POST /tokens/validar</code> con un token inválido, caducado o ya usado	Sin cuerpo
400	Datos no válidos en <code>POST /avisos</code>	<code>{"error":"El aviso necesita un título."}</code> (el texto cambia según el caso, ver sección 7)
400	Parámetro mal formado (por ejemplo <code>desdeId=abc</code> o una fecha ilegible) o JSON no válido	Normalmente sin cuerpo; en todo caso, sin el campo <code>codigo</code>
403	La petición llega por el acceso remoto	<code>{"error":"Esta acción no está disponible en el acceso remoto.", "codigo":"remoto_no_permitido"}</code>
401	La petición va a una instancia en la nube sin la clave de acceso del local	<code>{"error":"Introduce la clave de acceso del local.", "codigo":"nube_acceso"}</code>
500	Error interno	<code>{"error":"Ha ocurrido un error inesperado. Vuelve a intentarlo; si sigue pasando, avisa a soporte."}</code>

No existe el código 429. La API no limita las consultas por ritmo; el único límite es el de avisos (60 por hora, que se rechaza con 400). Un 402 no se produce en esta API.

La comprobación de la clave es en tiempo constante y AvantPOS anota el último acceso como mucho una vez por minuto.

5. Referencia de endpoints

Todas las rutas cuelgan de `http://IP-del-TPV:8085/api/integracion` . Los cuerpos de petición y respuesta son JSON en UTF-8 con nombres de campo en camelCase. Los ejemplos usan estas variables:

```
BASE="http://192.168.0.15:8085/api/integracion"
CLAVE="avp_0123456789abcdef0123456789abcdef0123456789abcdef"
```

Método	Ruta	Para qué
GET	<code>/estado</code>	Comprobar la clave, el reloj y datos del local
GET	<code>/dispositivos</code>	Puntos de venta: TPV, comanderas y tablets
GET	<code>/usuarios</code>	Usuarios del local
GET	<code>/mesas</code>	Estado de la sala en este momento
GET	<code>/auditoria</code>	Historial de eventos, paginado por id
GET	<code>/eventos</code>	Flujo en tiempo real (SSE) con reanudación
POST	<code>/avisos</code>	Enviar un aviso al local
POST	<code>/tokens/validar</code>	Validar un token de un solo uso del backoffice

5.1 GET /estado

Sirve para comprobar que la clave funciona y para conocer la hora del equipo, el nombre del local y su zona horaria. Sin parámetros.

```
curl -s "$BASE/estado" -H "Authorization: Bearer $CLAVE"
```

```
{
  "ok": true,
  "version": "2.61.8",
  "machineId": "3F9A1C7E52B0D846",
  "ahoraUtc": "2026-10-14T09:12:33.4567891Z",
  "nombreComercial": "Bar La Esquina",
  "zonaHoraria": "Europe/Madrid",
  "licencia": {
    "modo": "Completa",
    "caducidad": "2027-03-31",
    "modulos": { "camaras": false }
  }
}
```

Campo	Significado
<code>version</code>	Versión de AvantPOS (formato <code>2.61.8</code>)
<code>machineId</code>	Identificador del equipo, el mismo de la licencia

Campo	Significado
ahoraUtc	Hora del equipo en UTC; compárela con la suya para detectar desfases de reloj
nombreComercial	Nombre comercial de la empresa configurada (puede ser <code>null</code>)
zonaHoraria	Zona horaria del equipo en formato IANA (por ejemplo <code>Europe/Madrid</code>)
licencia.modo	Demo , Completa , Caducada o DemoCaducada
licencia.caducidad	Fecha <code>yyyy-MM-dd</code> , o <code>null</code> si no aplica
licencia.modulos	<code>null</code> en licencias anteriores a los módulos (sin restricciones); en otro caso <code>{"camaras": true false}</code>

5.2 GET /dispositivos

Lista todos los puntos de venta, activos o no, ordenados por id. Sirve, por ejemplo, para asociar cada aparato con una cámara, una pantalla o una ubicación.

```
curl -s "$BASE/dispositivos" -H "Authorization: Bearer $CLAVE"
```

```
[
  { "id": 1, "nombre": "TPV 1", "tipo": "tpv", "activo": true, "cajaCobroId": null,
    "ip": "192.168.0.21", "ultimaVistaUtc": "2026-10-14T09:10:02.118Z" },
  { "id": 3, "nombre": "Comandera Sala", "tipo": "comandera", "activo": true, "cajaCobroId": 1,
    "ip": "192.168.0.34", "ultimaVistaUtc": "2026-10-14T09:11:47.502Z" }
]
```

Campo	Significado
id	Id del dispositivo. Es el que aparece como <code>dispositivoId</code> en los eventos
tipo	tpv , comandera o tablet
cajaCobroId	Id de la caja a la que imputa sus ventas (una comandera vende en la caja de cobro de su TPV); <code>null</code> si no tiene
ip	Última IP desde la que actuó el aparato (<code>null</code> si aún no se ha visto)
ultimaVistaUtc	Última vez que se vio actividad del aparato (se anota como mucho una vez por minuto)

5.3 GET /usuarios

Lista todos los usuarios, activos o no, ordenados por id.

```
curl -s "$BASE/usuarios" -H "Authorization: Bearer $CLAVE"
```

```
[
  { "id": 1, "nombre": "Stefano", "rol": "Admin", "activo": true },
  { "id": 4, "nombre": "Marta", "rol": "Camarero", "activo": true }
]
```

`rol` es `Camarero`, `Encargado` o `Admin`. La API no devuelve PIN ni ningún otro dato personal que no sea el nombre.

5.4 GET /mesas

Estado de la sala en este instante: **una entrada por mesa**, incluidas las libres, ordenadas por sala y por id. Los campos de plano (`forma`, `x`, `y`, `ancho`, `alto`) son los mismos con los que el TPV dibuja la mesa.

```
curl -s "$BASE/mesas" -H "Authorization: Bearer $CLAVE"
```

```
[
  { "mesaId": 12, "mesa": "7", "salonId": 1, "salon": "Sala", "forma": "rect",
    "x": 320, "y": 140, "ancho": 100, "alto": 100, "comensales": 4,
    "ticketId": 214, "abiertoUtc": "2026-10-14T08:02:11.000Z", "cuentaPedida": false,
    "importeCents": 4850, "usuarioId": 4, "usuario": "Marta",
    "ultimaActividadUtc": "2026-10-14T08:40:07.000Z", "comensalesTicket": 4 },
  { "mesaId": 13, "mesa": "8", "salonId": 1, "salon": "Sala", "forma": "rect",
    "x": 460, "y": 140, "ancho": 100, "alto": 100, "comensales": 4,
    "ticketId": null, "abiertoUtc": null, "cuentaPedida": false,
    "importeCents": null, "usuarioId": null, "usuario": null,
    "ultimaActividadUtc": null, "comensalesTicket": null }
]
```

Campo	Significado
<code>mesaId</code> , <code>mesa</code>	Id y nombre (o número) de la mesa
<code>salonId</code> , <code>salon</code>	Sala a la que pertenece
<code>comensales</code>	Capacidad de la mesa
<code>ticketId</code>	Ticket abierto en la mesa; <code>null</code> si está libre (y entonces el resto de campos del ticket también son <code>null</code>)
<code>abiertoUtc</code>	Cuándo se abrió el ticket
<code>cuentaPedida</code>	<code>true</code> si ya se imprimió la cuenta
<code>importeCents</code>	Total actual del ticket en céntimos
<code>usuarioId</code> , <code>usuario</code>	Camarero responsable del ticket
<code>ultimaActividadUtc</code>	Última línea añadida, o la apertura si aún no hay líneas
<code>comensalesTicket</code>	Comensales declarados en el ticket

Si hubiera más de un ticket abierto en la misma mesa, se devuelve el más antiguo. Los cambios de sala se conocen en el momento por los eventos de operativa de la sección 6; esta consulta sirve para la foto inicial y para reconciliar.

5.5 GET /auditoria

Historial de eventos en **orden ascendente por id**. Es el mismo objeto que viaja por el flujo en tiempo real (sección 6.1).

Parámetro	Descripción
desdeId	Devuelve eventos con id mayor que este (exclusivo). Es el cursor de paginación
limite	Máximo de eventos. Por defecto 500; mínimo 1 y máximo 1000 (un valor fuera de rango se ajusta, no da error)
desde	Fecha y hora mínima (inclusiva), ISO 8601. Use UTC con Z, por ejemplo 2026-10-14T00:00:00Z
hasta	Fecha y hora máxima (exclusiva), ISO 8601
tipo	Solo eventos de ese tipo (anulacion_linea , ticket_cobrado ...). Un único tipo por consulta

```
curl -s "$BASE/auditoria?desdeId=48210&limite=2" -H "Authorization: Bearer $CLAVE"
```

```
[
  {
    "id": 48211,
    "utc": "2026-10-14T13:42:07.318Z",
    "tipo": "invitacion",
    "puntoVentaId": 1, "puntoVenta": "TPV 1",
    "dispositivoId": 3, "dispositivo": "Comandera Sala", "dispositivoTipo": "comandera",
    "usuarioId": 4, "usuario": "Marta",
    "autorizadoPorId": 1, "autorizadoPor": "Stefano",
    "ticketId": 214, "mesa": "7", "ticketNombre": null,
    "importeCents": 1150,
    "detalle": "Diavola 33 cm x1",
    "datos": null
  },
  {
    "id": 48212,
    "utc": "2026-10-14T13:44:51.904Z",
    "tipo": "ticket_cobrado",
    "puntoVentaId": 1, "puntoVenta": "TPV 1",
    "dispositivoId": 3, "dispositivo": "Comandera Sala", "dispositivoTipo": "comandera",
    "usuarioId": 4, "usuario": "Marta",
    "autorizadoPorId": null, "autorizadoPor": null,
    "ticketId": 214, "mesa": "7", "ticketNombre": null,
    "importeCents": 4850,
    "detalle": "Efectivo 48,50 (entregado 50,00, cambio 1,50)",
    "datos": {
      "formasPago": [
        { "formaPagoId": 1, "nombre": "Efectivo", "importeCents": 4850,
          "entregadoCents": 5000, "cambioCents": 150, "efectivo": true }
      ],
      "propinaCents": 0,
      "dispositivoId": 3
    }
  }
]
```

Paginación. Para recorrer un histórico largo, repita la consulta con `desdeId` igual al `id` del último evento recibido hasta que devuelva una lista vacía (o menos de `limite` elementos). Los `id` son crecientes y sirven de cursor. Si no pasa `desdeId`, la consulta empieza en el primer evento que cumpla los demás filtros.

Los valores de los ejemplos son ilustrativos; la estructura y los nombres de campo son los reales.

5.6 GET /eventos

Flujo en tiempo real (Server-Sent Events). Se describe en la sección 6.

5.7 POST /avisos

Envía un aviso al local. Se describe en la sección 7.

5.8 POST /tokens/validar

Valida un token de un solo uso emitido por el backoffice. Se describe en la sección 8.

6. Eventos en tiempo real (SSE)

6.1 El objeto evento

Es el mismo en `GET /auditoria` (como elemento de la lista) y en `GET /eventos` (como contenido de `data`):

Campo	Tipo	Significado
<code>id</code>	entero	Identificador creciente del evento. Sirve de cursor y de clave de deduplicación
<code>utc</code>	fecha ISO 8601	Momento de la operación, en UTC, con fracción de segundo (por ejemplo <code>2026-10-14T13:42:07.318Z</code>)
<code>tipo</code>	texto	Tipo de evento (catálogo en 6.4)
<code>puntoVentaId</code> , <code>puntoVenta</code>	entero, texto	Caja a la que se imputa la operación (la del ticket). Las ventas de las comanderas se imputan a su caja de cobro
<code>dispositivoId</code> , <code>dispositivo</code> , <code>dispositivoTipo</code>	entero, texto, texto	Aparato desde el que se hizo (<code>tpv</code> , <code>comandera</code> o <code>tablet</code>). Es el dato que hay que cruzar con una cámara o una ubicación. Puede ser <code>null</code> en operaciones internas o muy antiguas
<code>usuarioId</code> , <code>usuario</code>	entero, texto	Usuario que hizo la operación
<code>autorizadoPorId</code> , <code>autorizadoPor</code>	entero, texto	Encargado que metió su PIN para autorizar la operación; <code>null</code> si el propio usuario tenía permiso
<code>ticketId</code>	entero	Ticket afectado (<code>null</code> si no aplica)
<code>mesa</code>	texto	Número o nombre de la mesa del ticket (<code>null</code> si no tiene)
<code>ticketNombre</code>	texto	Nombre del pedido para llevar o a domicilio (<code>null</code> si no tiene)
<code>importeCents</code>	entero	Importe afectado, en céntimos (<code>null</code> si no aplica; su significado depende del tipo)
<code>detalle</code>	texto	Texto libre para personas. No debe interpretarse automáticamente , salvo la marca (ya enviada a cocina) al final del detalle de <code>anulacion_linea</code> , que es estable
<code>datos</code>	objeto o <code>null</code>	Datos estructurados, solo en los tipos que los tienen (ver 6.4)

6.2 Formato del flujo

```
curl -N "$BASE/eventos" \  
  -H "Authorization: Bearer $CLAVE" \  
  -H "Last-Event-ID: 48210"
```

La respuesta es `text/event-stream`. Cada mensaje tiene tres líneas y termina con una línea en blanco:

```
: conectado
```

```
id: 48211
event: invitacion
data: {"id":48211,"utc":"2026-10-14T13:42:07.318Z","tipo":"invitacion","puntoVentaId":1,"puntoVenta":"TPV
1","dispositivoId":3,"dispositivo":"Comandera
Sala","dispositivoTipo":"comandera","usuarioId":4,"usuario":"Marta","autorizadoPorId":1,"autori
zadoPor":"Stefano","ticketId":214,"mesa":"7","ticketNombre":null,"importeCents":1150,"detalle":
"Diavola 33 cm x1","datos":null}
```

```
: ping
```

- `id`: es el id del evento.
- `event`: es el **tipo** del evento. Atención: como los mensajes llevan nombre, un `EventSource` de navegador **no** los entrega a `onmessage`; habría que registrar un `addEventListener` por cada tipo. Por eso, y porque `EventSource` no permite cabeceras de autenticación, recomendamos leer el flujo con `fetch` o con un cliente HTTP (ejemplos en la sección 12).
- `data`: es el objeto evento en una sola línea de JSON.
- Las líneas que empiezan por `:` son comentarios: `: conectado` al abrir y `: ping` cada **25 segundos** sin actividad. Sirven para mantener viva la conexión; si deja de recibir nada (ni eventos ni pings) durante más de 60 segundos, considere la conexión caída y reconecte.

6.3 Reanudación y deduplicación

- Con `Last-Event-ID: <id>` (cabecera) o `?desdeId=<id>` (parámetro), AvantPOS envía primero **todo lo posterior a ese id**, leído de la base de datos, y después lo que vaya ocurriendo. Así no se pierde nada aunque su sistema haya estado caído. Si se envían ambos, manda la cabecera; si la cabecera no es un número, se usa `desdeId`.
- **Sin ninguno de los dos**, solo recibirá lo que ocurra a partir de ese momento. Para empezar desde el principio del histórico use `desdeId=0`.
- Un mismo `id` **no llega dos veces dentro de la misma conexión**. Entre conexiones distintas sí puede repetirse (si usted reconecta con un id anterior al último que procesó), por lo que debe deduplicar por `id` (sección 9).
- Los eventos se emiten **después** de guardarse en la base de datos, y siempre con `id` creciente.
- Cada cliente tiene un búfer de 1024 avisos pendientes. Si su lector es tan lento que lo desborda, se descartan los más antiguos de esa conexión. Por eso, además de reconectar con el último `id`, recomendamos **reconciliar de vez en cuando** consultando `GET /auditoria?desdeId=<último id procesado>`: devuelve cualquier hueco.
- Al reiniciar AvantPOS (por ejemplo, tras una actualización) la conexión se corta: reconecte con el último `id`.

6.4 Catálogo de tipos de evento

Los tipos de la tabla son los que genera la versión 2.61.8. `importeCents` indica qué representa el importe del evento. Un guion (-) significa que va a `null`.

Operativa de sala

Estos seis tipos describen el movimiento de la sala. No aparecen en la pantalla de Auditoría del backoffice, solo en la API.

tipo	Cuándo	importeCents	datos
ticket_abierto	Se abre un ticket (mesa ocupada o venta en barra). detalle : «Mesa N» o «Barra»	-	{mesaId, comensales} (mesaId es null en barra)
comensales	Cambian los comensales declarados. detalle : el número nuevo	-	{antes, despues}
enviado_cocina	Se envía una comanda. detalle : «N artículos en M líneas»	-	{lineas, articulos, retenidas, numeroCocina}
cuenta_pedida	Se imprime la cuenta	total del ticket	-
mesa_movida	El ticket cambia de mesa. detalle : «origen → destino»	-	{mesaOrigenId, mesaDestinoId}
mesa_liberada	La mesa queda libre. Solo se emite si el ticket tenía mesa	-	{mesaId, motivo} con motivo = cobro, anulacion, consumo_propio o union

Cobros, cajón y facturas

tipo	Cuándo	importeCents	datos
ticket_cobrado	Se cierra un ticket cobrado. detalle : formas de pago, entregado y cambio	total del ticket	{formasPago: [{formaPagoId, nombre, importeCents, entregadoCents, cambioCents, efectivo}], propinaCents, dispositivoId}
cajon_abierto	El cajón se abre al cobrar	-	-
cajon_manual	Apertura manual del cajón desde el TPV	-	-
devolucion	Se emite una devolución de un ticket. detalle : «Devolución del ticket #serie+número»	total devuelto	-
reapertura	Se reabre un ticket ya cobrado	total	-
factura_emitida	Se emite factura completa de un ticket. detalle : número, cliente y NIF	total	-
factura_rectificada	Se rectifica una factura. detalle : número, tipo, factura rectificada y motivo	total	-

tipo	Cuándo	importeCents	datos
consumo_propio	Ticket cerrado como consumo propio	importe	-
venta_credito	Venta a crédito a un cliente. detalle : nombre del cliente	importe	-
liquidacion_credito	Cobro posterior de una venta a crédito. detalle : cliente y forma de pago	importe	-

Anulaciones, descuentos y cambios en un ticket

tipo	Cuándo	importeCents	datos
anulacion_linea	Se anula una línea con motivo. detalle termina en (ya enviada a cocina) si ya estaba en preparación	importe de la línea (más sus añadidos)	-
eliminacion_linea	Se quita una línea que aún no se había enviado a cocina. detalle termina en «sin enviar a cocina»	importe de la línea	-
anulacion_ticket	Se anula un ticket entero. detalle : el motivo	total	-
invitacion	Una línea queda invitada (a 0 €). detalle : línea y motivo	importe invitado	-
invitacion_ticket	Un ticket completo queda invitado	importe invitado	-
descuento_linea	Descuento manual en una línea. detalle : descripción de la línea	importe descontado	-
descuento_ticket	Descuento manual en todo el ticket. detalle : porcentaje, por ejemplo «10%»	-	-
cambioPrecio	Precio de una línea cambiado a mano. detalle : «descripción: precio anterior → nuevo»	nuevo precio unitario	-
cambio_cantidad_enviada	Se cambia la cantidad de una línea ya enviada a cocina. detalle : «descripción: antes → después»	precio unitario por la diferencia (negativo si se reduce)	-
cambio_camarero	Se asigna el ticket a otro camarero. detalle : «id anterior → id nuevo (nombre)»	-	-
traspaso_lineas	Líneas movidas a otro ticket. detalle : número de líneas y ticket destino	importe de las líneas	-
urgente_cocina	Una comanda (o líneas concretas) se marca como urgente	-	-

Caja y jornada

tipo	Cuándo	importeCents	datos
login	Un usuario entra por PIN. detalle : «Acceso por PIN»	-	-
apertura_caja	Apertura de turno con fondo declarado	fondo declarado	-
descuadre_caja	Cierre de caja con descuadre de efectivo. detalle : el motivo indicado	descuadre (puede ser negativo)	-
fichaje_manual	Apunte manual en el registro de jornada. detalle : trabajador, tipo, fecha y motivo	-	-
fichaje_anulado	Se anula un apunte del registro de jornada	-	-

Notas sobre estos tipos:

- En `login`, `puntoVentaId` y `dispositivoId` identifican el aparato por el que se entra (si el aparato no se identificó, `puntoVentaId` es 0).
- El cierre Z de caja no genera un evento propio: sí lo hace su descuadre (`descuadre_caja`), si lo hay.
- `fichaje_manual` y `fichaje_anulado` llevan siempre `puntoVentaId` 1 y no tienen ticket.

Administración

Estos tipos se generan por operaciones de administración del backoffice. Llevan siempre `puntoVentaId` 1 y no tienen ticket. Su **detalle** es un resumen para personas.

tipo	Cuándo
precios_masivos	Cambio masivo de precios
edicion_masiva_productos	Edición masiva de productos
borrado_masivo_productos	Baja masiva de productos
asistente_ia	Cambios en el catálogo aplicados por el asistente de IA
asistente_ia_deshacer	Cambios del asistente de IA deshechos
reset_operaciones	Borrado de los datos de operaciones (formación)
nube_vincular	Se generó un código para conectar otro aparato

Tipos nuevos. Pueden añadirse tipos de evento en versiones futuras. Su consumidor debe **ignorar sin error** los tipos que no conozca.

7. Avisos al local (POST /avisos)

Permite a un sistema externo avisar al personal del local, donde ya mira: pantallas del TPV y comanderas, Telegram y correo. Es el único endpoint que escribe, y solo envía mensajes: no modifica datos de ventas.

```
curl -s -X POST "$BASE/avisos" \
  -H "Authorization: Bearer $CLAVE" \
  -H "Content-Type: application/json" \
  -d '{
    "nivel": "alerta",
    "titulo": "Persona en zona de caja fuera de horario",
    "texto": "Cámara Caja 1, 02:14. Sin operación en el TPV.",
    "url": "http://192.168.0.15:8090/casos/318",
    "puntoVentaId": 1,
    "soloEncargados": true,
    "canales": ["tpv", "telegram", "correo"]
  }'
```

Campo	Obligatorio	Descripción
titulo	Sí	Título del aviso. Sin título: 400 «El aviso necesita un título.»
nivel	No	aviso (por defecto) o alerta . Otro valor: 400 «nivel debe ser «aviso» o «alerta».»
texto	No	Detalle del aviso
url	No	Enlace «Ver» para abrir más información. Debe empezar por http:// o https:// ; si no: 400 «url debe empezar por http:// o https://.»
puntoVentaId	No	Caja o aparato al que se refiere; se muestra su nombre junto al título
soloEncargados	No	Por defecto true
canales	No	Lista con tpv , telegram y/o correo . Por defecto, solo tpv . Otro valor: 400 «canales admite tpv, telegram y correo.»

Respuesta (200): los canales por los que el aviso **ha salido de verdad**.

```
{ "enviado": ["tpv"] }
```

Un canal no configurado se ignora sin error y simplemente no aparece en `enviado` . Compruebe siempre esa lista.

Dónde se ve cada canal

- `tpv` : aviso en las pantallas de venta y comanderas conectadas, con un botón «Ver» que abre la `url` , y con sonido. Con `soloEncargados: true` solo lo muestran las pantallas donde hay un usuario con acceso al backoffice. Siempre sale por este canal si se pide (no depende de configuración).
- `telegram` : mensaje al chat de Telegram que el local tenga configurado (token del bot y chat, en la misma pestaña «Sistemas externos»). Las alertas y los avisos se distinguen por un icono al principio del

mensaje. Si no está configurado, se ignora.

- `correo` : correo a los destinatarios del correo de cierres, con asunto «ALERTA: título» o «Aviso: título». Si el correo del local no está configurado, se ignora.

Privacidad del aviso en el TPV. El aviso del canal `tpv` viaja por el canal de eventos interno de la red del local, que lo reciben todos los equipos conectados; `soloEncargados` solo decide qué pantallas lo enseñan, no cifra nada. No ponga en `título` ni en `texto` nada que no deba ver un camarero: el detalle sensible debe ir detrás de la `url`.

Límite: 60 avisos por hora (ventana deslizante, para todo el local). Pasado el límite se responde `400` con «Demasiados avisos: como mucho 60 por hora.» El límite protege al local de un fallo del sistema externo que inunde las pantallas. Agrupe avisos repetidos en el lado del integrador.

8. Tokens de un solo uso (acceso desde el backoffice)

Para qué sirve. Si su sistema tiene una interfaz web propia (un visor de cámaras, un cuadro de mandos), el local puede abrirla **dentro del backoffice** sin una segunda contraseña. El flujo es:

1. En **Configuración > Integraciones > Sistemas externos** el local guarda la **dirección de su servicio** (por ejemplo `http://192.168.0.15:8090` ; debe empezar por `http://` o `https://`).
2. Aparece la pestaña **«Cámaras»** del backoffice (requiere el módulo «Cámaras con IA» contratado), que abre `<dirección>/?token=<token>` en un marco integrado (o en una pestaña nueva, a elección del usuario).
3. El token lo emite AvantPOS para el usuario que está dentro del backoffice: es **de un solo uso y caduca a los 60 segundos**. Solo puede emitirse para usuarios con acceso al backoffice.
4. Su servicio, al recibir la petición con `?token=` , lo valida contra AvantPOS (con la clave de integración) y obtiene quién es el usuario y qué permisos tiene. Si todo es correcto, abre su sesión sin pedir credenciales.

Es una función opcional. Si su sistema no tiene interfaz propia, puede ignorarla.

```
curl -s -X POST "$BASE/tokens/validar" \
-H "Authorization: Bearer $CLAVE" \
-H "Content-Type: application/json" \
-d '{"token":"9c1e7a4f0b2d58e3a6f14c07b9d2e85a31c6f7d04b8e92a1"}'
```

Respuesta 200 :

```
{
  "usuarioId": 1,
  "nombre": "Stefano",
  "rol": "Admin",
  "permisos": {
    "puedeAnular": true, "puedeDescuentos": true, "puedeInvitar": true, "puedeReabrir": true,
    "puedeVerInformes": true, "puedeAbrirCajon": true, "puedeBackoffice": true,
    "puedeCerrarCaja": true, "puedeAutorizar": true
  }
}
```

Respuesta 401 (sin cuerpo) si el token no existe, ya se usó, caducó o su usuario ya no está activo. Notas:

- El token se **consume** al validarlo: el segundo intento falla.
- Los tokens se guardan en la memoria de AvantPOS: si el programa se reinicia, los que hubiera pendientes dejan de valer.
- Valide el token **en el servidor** de su servicio, nunca desde el navegador del usuario (la API no admite CORS y la clave no debe llegar al navegador).
- Si su servicio se muestra dentro de un marco en el backoffice, debe permitir ser incrustado desde el origen del TPV (`http://IP-del-TPV:8085`).

- **Convención opcional de los sistemas de cámaras:** en la pantalla «Vigilante de caja» del backoffice, el botón «Ver vídeo» abre `<dirección>/casos/por-auditoria/<id>`, y se pinta solo donde el servicio responde que existe caso, consultando `GET <dirección>/api/casos/existen?ids=1,2,3` (que debe devolver la lista de ids con caso, por ejemplo `[1,3]`). Para eso, el servicio debe permitir CORS desde el origen del TPV. Si no se implementa, simplemente no aparecen los botones.

9. Límites y buenas prácticas

Límites

- 1 a 1000 eventos por consulta a `/auditoria` (500 por defecto).
- 60 avisos por hora en total.
- Una clave por local.
- Un búfer de 1024 notificaciones por conexión SSE (ver 6.3).
- Ningún límite de ritmo en el resto de consultas, pero la API corre en el mismo equipo que el TPV: no abuse del sondeo. Para seguir la actividad use el flujo SSE y no consultas repetidas.

Buenas prácticas

- **Reconexión con espera exponencial.** Si el flujo se corta, reintente tras 1 s, 2 s, 4 s... hasta un máximo (por ejemplo 30 s), con un poco de variación aleatoria, y vuelva a 1 s tras una conexión estable. Reconecte siempre con el último `id` procesado.
- **Idempotencia por `id`.** Trate el `id` como clave única: guarde el último procesado de forma duradera (fichero, base de datos) **después** de procesar el evento, y descarte los que ya vio. Así un reinicio o una reconexión no duplican efectos.
- **Reconciliación.** Al arrancar y cada cierto tiempo (por ejemplo, cada pocos minutos) consulte `GET /auditoria?desdeId=<último id>` por si faltara algo, y `GET /mesas` para la foto actual de la sala.
- **Fechas.** Todas las fechas de la API son **UTC** en ISO 8601, con `Z`. Conviértalas a la hora local solo para mostrarlas, usando la zona IANA que devuelve `GET /estado (zonaHoraria)`: los horarios (por ejemplo, «fuera de horario») se calculan en hora local del local, con cambio de horario de verano incluido. Las fracciones de segundo pueden tener longitud variable; use un analizador ISO 8601 estándar.
- **Reloj.** Sincronice su servidor con NTP y compare `ahoraUtc` de `GET /estado` con su hora para detectar desfases (importante al cruzar eventos con grabaciones de vídeo).
- **Ignorar lo desconocido.** Ignore los campos nuevos y los tipos de evento que no conozca (sección 11).
- **Cursor por `id`, no por hora.** Dos eventos pueden tener la misma hora; el orden fiable es el `id`.
- **No interprete `detalle`.** Use los campos estructurados y `datos`. El texto del `detalle` puede cambiar de redacción.
- **Servidor a servidor.** La API no admite CORS: llámela desde un servicio, no desde el navegador de un usuario.
- **Reserve una IP fija** (o una reserva DHCP) para el equipo de AvantPOS y para su sistema, y limite en el firewall quién puede llegar al puerto 8085.
- **Cabecera `X-Dispositivo`.** No la envíe: es de uso interno de las pantallas de AvantPOS.

Errores frecuentes

Síntoma	Causa probable
401 <code>clave_integracion</code> en todo	Clave mal copiada, clave rotada o retirada, o la integración nunca se activó

Síntoma	Causa probable
No se conecta (tiempo de espera agotado)	IP o puerto incorrectos, firewall, o el equipo no está en la misma red
403 remoto_no_permitido	Intenta entrar por el acceso remoto; solo desde la red del local
401 nube_acceso	Es una instancia en la nube; la API de integración no se usa con ellas
EventSource no recibe nada	EventSource no envía cabeceras ni recibe eventos con nombre en onmessage ; use fetch o un cliente HTTP
El flujo se corta cada cierto tiempo	Un proxy intermedio corta las conexiones inactivas o hace de búfer; el servidor envía : ping cada 25 s
Eventos duplicados tras reconectar	Es normal si reconecta con un id anterior; deduplique por id
dispositivoId es null	Operación interna o anterior a la versión 2.58.3
400 al enviar un aviso	Falta el título, nivel o canales no válidos, url sin http(s):// o se superó el límite horario
enviado no incluye telegram o correo	Ese canal no está configurado en el local

10. Protección de datos

La API devuelve **nombres de empleados y lo que hace cada uno** (accesos, anulaciones, descuentos, cobros, fichajes...) junto con la hora y el aparato. Son datos personales de los trabajadores del local.

- **El local es el responsable del tratamiento.** El integrador que recibe estos datos actúa como **encargado del tratamiento** (art. 28 del RGPD) y necesita un **contrato de encargado del tratamiento** (o cláusulas equivalentes) firmado con el local, antes de empezar a recibirlos.
- **Minimización.** Recoja solo los campos que su servicio necesita. Si no necesita nombres, no los guarde (por ejemplo, conserve solo `usuarioId`).
- **Conservación.** Defina y documente con el local cuánto tiempo se guardan los datos y borre o anonimice los que superen ese plazo.
- **Seguridad.** Proteja la clave como una contraseña (gestor de secretos, acceso restringido), guarde los datos con las medidas técnicas adecuadas, limite quién puede verlos y deje constancia de los accesos. Notifique al local sin demora cualquier incidente que afecte a la clave o a los datos.
- **Finalidad.** Use los datos únicamente para el servicio contratado por el local, no para otros fines propios ni de terceros.
- **Transparencia.** El local debe informar a sus trabajadores del sistema y de su finalidad, especialmente si cruza la actividad con videovigilancia.

Este apartado es solo orientativo y no constituye asesoramiento jurídico; cada integrador y cada local deben revisar sus obligaciones con su asesor.

11. Versionado y compromiso de estabilidad

- La API de integración, en su versión actual (v2), existe desde AvantPOS **2.58.3** (16/09/2026): auditoría y flujo en tiempo real, estado de sala (`/mesas`), eventos de operativa, avisos (`/avisos`) y tokens (`/tokens/validar`) llegaron juntos en esa versión. Es la versión mínima que necesita para todo lo descrito en este manual.
- La ruta no lleva el número de versión (`/api/integracion/...`). Para saber con qué versión de AvantPOS está hablando, consulte `version` en `GET /estado` .
- **Cambios compatibles**, que pueden llegar en cualquier actualización de AvantPOS: campos nuevos en las respuestas y en `datos` , tipos de evento nuevos, endpoints nuevos y parámetros opcionales nuevos. Su integración debe tolerarlos: ignore lo que no conozca y no dependa del orden de los campos ni de que `datos` tenga exactamente los campos actuales.
- **Cambios incompatibles** (quitar o renombrar campos, cambiar el significado de un valor, retirar un endpoint): Avant IT los avisará a los integradores con antelación razonable y los documentará. No se asumen plazos concretos de aviso.
- El texto de `detalle` no forma parte del contrato salvo la marca (`ya enviada a cocina`) de `anulacion_linea` .
- Mantenga actualizado su contacto con Avant IT para recibir los avisos de cambios.

12. Ejemplos completos

Ambos ejemplos hacen lo mismo: leen la clave del entorno, cargan los dispositivos, abren el flujo con reanudación desde el último id guardado, procesan cada evento una sola vez, y reconectan con espera exponencial. El cursor se guarda en un fichero.

12.1 Python (requests, sin librerías de SSE)

```
import json
import os
import time

import requests # pip install requests

BASE = os.environ.get("AVANTPOS_BASE", "http://192.168.0.15:8085/api/integracion")
CLAVE = os.environ["AVANTPOS_CLAVE"] # avp...
CABECERAS = {"Authorization": f"Bearer {CLAVE}"}
FICHERO_CURSOR = "cursor.txt"

def leer_cursor():
    try:
        with open(FICHERO_CURSOR) as f:
            return int(f.read().strip())
    except (FileNotFoundError, ValueError):
        return None

def guardar_cursor(evento_id):
    with open(FICHERO_CURSOR, "w") as f:
        f.write(str(evento_id))

def eventos_sse(respuesta):
    """Genera (id, tipo, datos) por cada mensaje del flujo SSE."""
    eid, tipo, datos = None, None, []
    for linea in respuesta.iter_lines(decode_unicode=True):
        if linea is None:
            continue
        if linea == "":
            # fin de mensaje
            if datos:
                yield eid, tipo, json.loads("\n".join(datos))
            eid, tipo, datos = None, None, []
        elif linea.startswith(":"):
            # comentario: ": conectado", ": ping"
            continue
        elif linea.startswith("id:"):
            eid = int(linea[3:].strip())
        elif linea.startswith("event:"):
            tipo = linea[6:].strip()
        elif linea.startswith("data:"):
            datos.append(linea[5:].lstrip())

def procesar(evento):
    # Ignore los tipos que no conozca.
```

```

    if evento["tipo"] in ("anulacion_linea", "anulacion_ticket", "invitacion", "cajon_manual"):
        print(evento["utc"], evento["tipo"], evento["usuario"], evento["dispositivo"],
evento["detalle"])

def main():
    estado = requests.get(f"{BASE}/estado", headers=CABECERAS, timeout=10).json()
    print("Local:", estado["nombreComercial"], "zona:", estado["zonaHoraria"])
    dispositivos = {d["id"]: d for d in requests.get(f"{BASE}/dispositivos", headers=CABECERAS,
timeout=10).json()}

    espera = 1
    while True:
        cursor = leer_cursor()
        cab = dict(CABECERAS)
        if cursor is not None:
            cab["Last-Event-ID"] = str(cursor)
        try:
            # timeout de lectura 60 s: el servidor envía un ping cada 25 s
            with requests.get(f"{BASE}/eventos", headers=cab, stream=True, timeout=(10, 60)) as
r:
                r.raise_for_status()
                r.encoding = "utf-8"                    # el flujo es UTF-8 y requests supondría
ISO-8859-1
                espera = 1
                for eid, tipo, evento in eventos_sse(r):
                    if cursor is not None and eid <= cursor:
                        continue                        # ya procesado
                    procesar(evento)
                    guardar_cursor(eid)                 # solo después de procesar
                    cursor = eid
            except requests.RequestException as e:
                print("Flujo cortado:", e)
            time.sleep(espera)
            espera = min(espera * 2, 30)                # espera exponencial, máximo 30 s

if __name__ == "__main__":
    main()

```

Enviar un aviso desde Python:

```

r = requests.post(f"{BASE}/avisos", headers=CABECERAS, timeout=10, json={
    "nivel": "alerta",
    "titulo": "Persona en zona de caja fuera de horario",
    "texto": "Cámara Caja 1, 02:14. Sin operación en el TPV.",
    "url": "http://192.168.0.15:8090/casos/318",
    "puntoVentaId": 1,
    "soloEncargados": True,
    "canales": ["tpv", "telegram"],
})
print(r.status_code, r.json())    # 200 {'enviado': ['tpv', 'telegram']}

```

12.2 JavaScript (Node.js 18 o superior, fetch)

```
import { readFileSync, writeFileSync } from "node:fs";

const BASE = process.env.AVANTPOS_BASE ?? "http://192.168.0.15:8085/api/integracion";
const CLAVE = process.env.AVANTPOS_CLAVE; // avp...
const FICHERO_CURSOR = "cursor.txt";

const leerCursor = () => {
  try { return parseInt(readFileSync(FICHERO_CURSOR, "utf8"), 10) || null; }
  catch { return null; }
};

const guardarCursor = (id) => writeFileSync(FICHERO_CURSOR, String(id));

function procesar(evento) {
  // Ignore los tipos que no conozca.
  if (["anulacion_linea", "anulacion_ticket", "invitacion",
"cajon_manual"].includes(evento.tipo)) {
    console.log(evento.utc, evento.tipo, evento.usuario, evento.dispositivo, evento.detalle);
  }
}

// Lee el flujo SSE y llama a alMensaje({id, tipo, datos}) por cada mensaje.
async function leerFlujo(respuesta, alMensaje) {
  const decodificador = new TextDecoder();
  let pendiente = "";
  for await (const trozo of respuesta.body) {
    pendiente += decodificador.decode(trozo, { stream: true });
    let corte;
    while ((corte = pendiente.indexOf("\n\n")) !== -1) {
      const bloque = pendiente.slice(0, corte);
      pendiente = pendiente.slice(corte + 2);
      let id = null, tipo = null;
      const datos = [];
      for (const linea of bloque.split("\n")) {
        if (linea.startsWith(":")) continue; // comentario / ping
        if (linea.startsWith("id:")) id = parseInt(linea.slice(3).trim(), 10);
        else if (linea.startsWith("event:")) tipo = linea.slice(6).trim();
        else if (linea.startsWith("data:")) datos.push(linea.slice(5).trimStart());
      }
      if (datos.length) await alMensaje({ id, tipo, datos: JSON.parse(datos.join("\n")) });
    }
  }
}

async function main() {
  const cab = { Authorization: `Bearer ${CLAVE}` };
  const estado = await (await fetch(`${BASE}/estado`, { headers: cab })).json();
  console.log("Local:", estado.nombreComercial, "zona:", estado.zonaHoraria);

  let espera = 1000;
  for (;;) {
    let cursor = leerCursor();
    const headers = { ...cab, ...(cursor !== null ? { "Last-Event-ID": String(cursor) } : {}) };
    try {
      const r = await fetch(`${BASE}/eventos`, { headers });
      if (!r.ok) throw new Error(`HTTP ${r.status}`);
      espera = 1000;
      await leerFlujo(r, async ({ id, datos }) => {
```

```
        if (cursor !== null && id <= cursor) return; // ya procesado
        procesar(datos);
        guardarCursor(id); // solo después de procesar
        cursor = id;
    });
} catch (e) {
    console.error("Flujo cortado:", e.message);
}
await new Promise((res) => setTimeout(res, espera));
espera = Math.min(espera * 2, 30000); // espera exponencial, máximo
30 s
}
}

main();
```

Validar un token en Node (en el servidor de su servicio):

```
const r = await fetch(`${BASE}/tokens/validar`, {
    method: "POST",
    headers: { Authorization: `Bearer ${CLAVE}`, "Content-Type": "application/json" },
    body: JSON.stringify({ token }),
});
if (r.status === 401) { /* token inválido, caducado o ya usado */ }
else { const usuario = await r.json(); /* {usuarioId, nombre, rol, permisos} */ }
```

Avant IT · soporte@avant-it.net · 93 694 05 02