Part 2F — Agent Phone (devices and their own API)

Ringmaster Agent Phone is the desktop app agents keep running to pick up calls (AGENT-PHONE.md). This part has two sides:

- **K. For the client app (API key):** connect an agent's Agent Phone with a one-time code, list their devices, remove one. This is the "Connect Agent Phone" button on your dashboard.
- **L. Agent Phone's own API (device token):** what the app itself calls. You only need it to build your own app; Ringmaster's Agent Phone already does all of it.

Who does what:

Job	Who
Who the person is (password, 2FA, staff accounts)	**Your app.** Ask for a pairing code only after your own login
May this computer take this agent's calls (keys, tokens, revoke)	**Ringmaster.** The engine checks every accept, decline and hang-up

```
Agent signed in to your dashboard → "Connect Agent Phone"
  your backend ── POST /v1/accounts/{acc}/agents/{ref}/devices/pairings ──► engine
              ◄── { code: "482913", link: "ringmaster-agent://pair?code=482913&host=…",
expiresAt } ──
  show the code (and the link) → the agent types the code into Agent Phone
Agent Phone: makes a P-256 key (kept in the OS password vault)
  POST /v1/agent/pair { code, deviceName, platform } + DPoP proof (signed with that key)
  ◄── { deviceId, accessToken (15 min), refreshToken }
```

K. Devices (client app, API key)

K1. A pairing code

Purpose: A one-time code for one agent. It works **once**, for **5 minutes**. Pairing tries are limited per IP.

Endpoint: `POST` `{base_url}/accounts/{accountId}/agents/{ref}/devices/pairings`

Success Response JSON Sample (`201`):


```
{
  "success": true,
  "httpStatus": "201 CREATED",
  "message": "Pairing code",
  "action": null,
  "action_time": "2026-11-14T14:20:02.118",
  "data": {
    "code": "078183",
    "link": "ringmaster-agent://pair?code=078183&host=https%3A%2F%2Fapi.ringmaster.example",
    "expiresAt": "2026-11-14T11:25:02.101Z"
  }
}
```


Show both: the **code** to type, and the **link** as a button (it opens Agent Phone where links are set up).

Errors: `404` "Agent not found: ...".

K2. List an agent's devices

Endpoint: `GET` `{base_url}/accounts/{accountId}/agents/{ref}/devices`


```
"data": [
  { "id": "1400d27e-8e21-4f9c-be7d-8e7c77c5f660", "name": "Agent Phone on reception-pc",
"platform": "linux",
    "appVersion": "0.1.0", "pairedAt": "2026-11-14T11:24:05Z", "lastSeenAt": "2026-11-
14T11:24:13Z", "lastIp": "192.168.100.215" }
]
```


At most **5** devices per agent; pairing a sixth answers `409 TOO_MANY_DEVICES`.

K3. Remove a device

Endpoint: `DELETE` `{base_url}/accounts/{accountId}/agents/{ref}/devices/{deviceId}`

Success Response: `200`, `message` "Device removed". The device is cut off **at once**: its tokens stop working, it is taken out of the lobby, it receives `device_revoked` and goes back to its pairing screen. You get `agent.device_revoked`.

A device is also revoked when: the agent is removed, its old refresh token is used again (copied: reason `TOKEN_REUSED`), it was not used for 30 days (`UNUSED`), or the agent signs out on it (`SIGNED_OUT`).

K4. Shared computers (stations)

A **station** is a computer the business shares (the reception PC). It belongs to no agent and rings for nobody until an agent starts a shift on it **from your dashboard**, after signing in to your app: your login is the proof of who they are. The computer switches by itself (it waits in its own room, `station-{id}`). One agent at a time; starting a shift while someone else is on replaces them (never during a call: `409 SHIFT_BUSY`).

Method	Path	Does
POST	`/accounts/{accountId}/stations/pairings`	A one-time code for a new shared computer (same shape as K1)
GET	`/accounts/{accountId}/stations`	Stations and who is on shift: `id`, `name`, `platform`, `onShiftRef`, `onShiftName`, `shiftSince`, `idleEndMin`, `endAtTime`, `lastSeenAt`
PATCH	`/accounts/{accountId}/stations/{stationId}`	`{"idleEndMin": 120, "endAtTime": "17:30"}`: end the shift after that many minutes without calls (0 = never), and at that time of day (`""` = never)
POST	`/accounts/{accountId}/stations/{stationId}/shift`	`{"agentRef": "asha"}`: start the shift ("Start my shift on Reception PC")
DELETE	`/accounts/{accountId}/stations/{stationId}/shift`	End the shift now
DELETE	`/accounts/{accountId}/stations/{stationId}`	Remove the station (signed out at once)
PUT	`/accounts/{accountId}/agents/{ref}/pin`	Optional: `{"pin": "4321"}`, 4–6 digits, stored hashed (`""` removes it)

PINs are optional and Ringmaster's Agent Phone doesn't ask for them: a shift starts from the dashboard. They exist for a client app that builds its own station screen (L7); five wrong PINs lock the agent out for 15 minutes (`409 PIN_LOCKED`). Webhooks: `station.paired`, `station.removed`, `agent.shift_started` · `agent.shift_ended`.

L. Agent Phone's API (device token)

Base: `{base_url}/agent`. Every call except L1 and L2 sends `Authorization: Bearer <accessToken>`. A device token works **only** here (never on `/v1/accounts/…`), and only for its own agent: there is no `agentRef` in these requests.

`401 DEVICE_UNAUTHORIZED` means the token expired, or the device was removed: refresh once; if that fails too, go back to the pairing screen.

DPoP proofs

L1 and L2 carry a header `DPoP: <proof>` (RFC 9449): a compact JWS signed with the device's key.

Part	Content
header	`{"typ": "dpop+jwt", "alg": "ES256", "jwk": {"kty": "EC", "crv": "P-256", "x": "…", "y": "…"}}`
claims	`{"htm": "POST", "htu": "<full URL of this request>", "iat": <unix seconds>, "jti": "<random, ≥ 12 chars>"}`
signature	ES256 over `base64url(header).base64url(claims)`, as raw R‖S (64 bytes), base64url

Proofs older than 60 s, or seen before (same `jti`), are refused. On refresh the key must be the one used at pairing.

L1. Pair

Endpoint: **POST** `{base_url}/agent/pair` (no token; DPoP proof)

Request:

```
{ "code": "078183", "deviceName": "Agent Phone on reception-pc", "platform": "linux",
"appVersion": "0.1.0" }
```

Success Response (`201`):

```
"data": { "deviceId": "1400d27e-8e21-4f9c-be7d-8e7c77c5f660", "accessToken": "rma_…",
"expiresIn": 900,
```

```
          "refreshToken": "rmr_…", "agentRef": "asha", "agentName": "Asha", "business":
 "Ringmaster Lab" }
```


Keep the private key and the refresh token in the OS password vault; keep the access token in memory only.

Errors: `401` "This code is wrong or has expired…" · `401` DPoP errors · `409 TOO_MANY_DEVICES` · `429` too many tries.

L2. Refresh

Endpoint: **POST** `{base_url}/agent/token` (no token; DPoP proof from the same key)

Request: `{"refreshToken": "rmr_…"}`. **Success Response**: the same shape as L1, with a **new** refresh token. Save it before anything else: the old one is dead, and using it again revokes the device.

L3. The desk (everything the app shows)

Endpoint: **GET** `{base_url}/agent/me`

Same answer as I14 for this device's agent.

L4. Search the dial pad

Endpoint: **GET** `{base_url}/agent/search?digits=0745`

Same as I15: digits match numbers; letters match the client app's contacts and earlier callers' names.

L5. Lobby token

Endpoint: **POST** `{base_url}/agent/lobby-token`

Like I12, with the identity `agent-{agentId}:dev-{deviceId}`.

L6. Calls

Method	Path	Does	Same as
POST	`/agent/calls/{callId}/accept`	Accept the ringing call; returns the call room token	J1
POST	`/agent/calls/{callId}/decline`	Body `{"reason": "WITH_CUSTOMER"}` (optional)	J2
POST	`/agent/calls/{callId}/take`	Answer a waiting caller now, even while away	J9
POST	`/agent/available`	"I'm back": `AVAILABLE` again (out of wrap-up: ready). Going away stays on the dashboard	I10
POST	`/agent/calls/{callId}/hangup`	Hang up (cancels the dial if the other phone still rings)	J6
POST	`/agent/calls/{callId}/join`	Join an outgoing call that is this agent's (after `outgoing`)	J7
POST	`/agent/calls/{callId}/call-back`	Call the other person of an earlier call back, from the same line	J8
POST	`/agent/calls`	Dial pad: `{"numberId": "…", "to": "0745051250"}`	Click-to-call with `meta.via: "dial-pad"`

Pairing with a **station** code (K4) gives `"kind": "STATION"` and `agentRef: null` in L1's answer.

L7. Shifts (stations only)

A station's token acts for the agent on shift; with nobody on shift, agent endpoints answer `409 NO_SHIFT`.

Method	Path	Does
GET	`/agent/station`	`{name, business, onShift: {ref, name, since} | null, agents: [{ref, name}]}` (agents with a PIN)
POST	`/agent/station-token`	Token for the station's own room `station-{id}`, where `shift {state: "started", agentRef, name}` arrives while nobody is on shift

Method	Path	Does
POST	`/agent/shift`	Optional PIN start: `{"agentRef": "asha", "pin": "4321"}` (`401` wrong PIN, `409 PIN_LOCKED`, `409 NO_PIN`, `409 SHIFT_BUSY`)
DELETE	`/agent/shift`	End my shift

When a shift ends elsewhere (dashboard, idle time, end of day), the station gets a `shift` signal `{deviceId, state: "ended", reason}` in the agent's lobby and is taken out of it.

L8. Sign out

Endpoint: **DELETE** `{base_url}/agent/device`

The device is revoked (`SIGNED_OUT`); delete the key and refresh token from the vault.

Revision #1
Created 14 November 2026 12:23:18 by Admin Qbit
Updated 14 November 2026 12:23:30 by Admin Qbit