Android Punch In / Punch Out API

Trynet CRM - Android integration documentation

Current behavior

Yes. The Android Punch In and Punch Out APIs now follow the same core business rules as the website Punch In / Punch Out.

Both now validate the staff member's approved work location, GPS latitude/longitude/accuracy, allowed radius and distance. Punch In also uses the staff member's configured starting time plus a 5-minute grace period for late attendance. Punch Out validates location again and requires a Punch In record for the current day.

Intentional API-format differences: the website receives the logged-in staff through the web session and uses a base64 image field. The Android API receives the staff ID from the URL and continues using a multipart image file named **img**. These differences do not change the attendance/work-location business rules.

API routes

Purpose	Method	Endpoint
Punch In	POST	/api/staff-punch/{staff_id}
Punch Out	POST	/api/staff-punchout/{staff_id}
Check active punch	GET	/api/punchCheck/{staff_id}

1. Punch In API

Method: POST

Endpoint: `/api/staff-punch/{staff_id}`

Content-Type: multipart/form-data

Request fields

Field	Type	Required	Description
img	Image file	Yes	Punch-in camera image. jpg/jpeg/png/webp, max 10 MB.
location	String	Yes	Readable location/address text from the Android app.
latitude	Number	Yes	Current GPS latitude, -90 to 90.
longitude	Number	Yes	Current GPS longitude, -180 to 180.
accuracy	Number	Yes	GPS accuracy in metres; must be 0 or greater.

Example request

```
POST /api/staff-punch/92
multipart/form-data

img = <image file>
location = Kakkanad, Kochi
latitude = 10.015860
longitude = 76.341870
```

```
accuracy = 12
```

Punch In server logic

1. Find the staff member using **{staff_id}**.

2. Validate location, latitude, longitude, accuracy and image.

3. Validate current GPS position using the same **WorkLocationService** used by the website.

4. If the staff member is outside the approved work-location radius, reject Punch In.

5. Save the punch photo.

6. Store punch-in latitude, longitude, GPS accuracy and calculated distance.

7. Set the current server time as **punch_in**.

8. Read the staff member's configured **starting_time**.

9. Apply the website rule: **starting_time + 5 minutes** grace period.

10. If Punch In is after the grace time, update today's Attendance remark to **late**.

11. Create/update today's Attendance record for an on-time Punch In as well.

Success response - 200


```
{
  "success": true,
  "message": "Punch-in recorded successfully",
  "punch_id": 123,
  "punch_in": "2026-08-19 09:58:10",
  "distance": 18.42,
  "radius": 50
}
```


Outside work location - 422


```
{
  "success": false,
  "message": "Work location validation message",
  "distance": 126.5,
  "radius": 50
}
```


Staff not found - 404


```
{
  "success": false,
  "message": "Staff not found."
}
```

2. Punch Out API

Method: POST

Endpoint: `/api/staff-punchout/{staff_id}`

Content-Type: multipart/form-data

Request fields

Punch Out uses the same fields as Punch In: **img, location, latitude, longitude, accuracy**.

Field	Type	Required	Description
img	Image file	Yes	Punch-in camera image. jpg/jpeg/png/webp, max 10 MB.
location	String	Yes	Readable location/address text from the Android app.
latitude	Number	Yes	Current GPS latitude, -90 to 90.
longitude	Number	Yes	Current GPS longitude, -180 to 180.
accuracy	Number	Yes	GPS accuracy in metres; must be 0 or greater.

Example request

```
POST /api/staff-punchout/92
multipart/form-data

img = <image file>
location = Kakkanad, Kochi
latitude = 10.015860
longitude = 76.341870
accuracy = 10
```

Punch Out server logic

1. Find the staff member using **{staff_id}**.

2. Validate location, latitude, longitude, accuracy and image.

3. Validate the current GPS position against the approved work location using **WorkLocationService**.

4. Reject Punch Out when the current position is outside the allowed radius.

5. Find today's latest Punch record for the staff member.

6. If no Punch In exists for today, return a 422 response instead of failing.

7. Save the Punch Out photo.

8. Store punch-out latitude, longitude, GPS accuracy and calculated distance.

9. Set the current server time as **punch_out**.

Success response - 200

```
{
  "success": true,
  "message": "Punch-out recorded successfully",
  "punch_id": 123,
  "punch_out": "2026-08-19 18:05:41",
  "distance": 16.18,
  "radius": 50
}
```

No Punch In today - 422

```
{
  "success": false,
  "message": "No Punch In record found for today."
}
```

3. Check current punch status

Method: GET

Endpoint: `/api/punchCheck/{staff_id}`

The current API returns **1** when the staff member has an active Punch In today with no Punch Out yet, otherwise it returns **0**.

```
GET /api/punchCheck/92

Response when punched in: 1
Response when not punched in / already punched out: 0
```

Android implementation flow

App state	Recommended action
Before Punch In	Call GET /api/punchCheck/{staff_id}. If 0, show Punch In.
Punch In	Capture photo + current GPS + accuracy, then POST multipart/form-data.
After successful Punch In	Show Punch Out state.
Punch Out	Capture a fresh photo + fresh GPS + accuracy, then POST multipart/form-data.
After successful Punch Out	Call punchCheck again; it should return 0.

Important integration notes

• Use the staff ID returned by the staff login API as **{staff_id}**.

• Send Punch In and Punch Out as **multipart/form-data**, not JSON.

• The Android app must request location permission and obtain latitude, longitude and GPS accuracy immediately before each punch action.

• The backend decides whether the location is allowed. The Android app should display the backend error message directly when a 422 response is returned.

• Do not calculate late status in Android. The backend uses the staff member's configured starting time plus the 5-minute grace period.

• Do not calculate distance/radius acceptance in Android as the final authority. The backend WorkLocationService performs the authoritative check.

Backend routes already exist; no route changes are required for these APIs.