TwelveLabs

PEGASUS 1.5 · TBM · INTEGRATION GUIDE

World Cup Video Analysis Integration Guide

Pegasus 1.5 Time-Based Metadata API · Request/Response Format & Integration Guide

This document is an integration guide for the TwelveLabs Pegasus 1.5 Time-Based Metadata (TBM) API applied to World Cup broadcast video analysis. Sending the same request as the demo returns a JSON response in the same format, which the broadcaster can parse and apply to any UI.

DEMO Demo: tbm-demo-alpha.vercel.app · Download the exact request and response JSON from the Request Format and Response Format sections below (also via the demo's top-right buttons). All samples are generated from the same World Cup match clip used in the live demo.

01 API Overview

ITEM	VALUE
Base URL	`https://api.twelvelabs.io/v1.3`
Analysis Request (async)	`POST /analyze/tasks`
Result Retrieval	`GET /analyze/tasks/{task_id}` — poll until `status: ready`
Authentication	`x-api-key: <API_KEY>`
Model	`pegasus1.5`
Analysis Mode	`analysis_mode: "time_based_metadata"`
Official Docs	`docs.twelvelabs.io`

The analysis result is returned as a JSON string in `result.data` of the task. Parse it once more on the client to obtain the response structure below.

02 Request Format

Define the items to extract using `segment_definitions`. Each definition has a segmentation instruction (`description`) and extraction fields (`fields`), written in natural language. The payload below is the exact request used to generate this demo — 3 layers: shot boundaries / soccer events / broadcast structure.

```
{
  "model_name": "pegasus1.5",
  "video": {
    "type": "asset_id",
    "asset_id": "6a170c7dc6d6bda7c4c9b08d"
  },
  "analysis_mode": "time_based_metadata",
  "response_format": {
    "type": "segment_definitions",
    "segment_definitions": [
      {
        "id": "L0_shot_boundaries",
        "description": "Detect all shot boundaries in this soccer broadcast.",
        "fields": [
          { "name": "shot_type", "type": "string",
            "enum": ["WIDE_PITCH","PLAYER_CLOSE_UP","CROWD","BENCH","OTHER"] },
          { "name": "description", "type": "string" },
          { "name": "main_actions", "type": "array", "items": {"type":"string"} },
          { "name": "emotional_elements", "type": "string" },
          { "name": "player_names", "type": "array", "items": {"type":"string"} }
        ]
      },
      {
        "id": "H16_soccer_events",
        "description": "Identify soccer gameplay events, live or replayed.",
        "fields": [
          { "name": "event_type", "type": "string",
            "enum": ["GOAL","SHOT_ON_TARGET","SHOT_OFF_TARGET","HEADER",
              "SAVE","FREE_KICK","CORNER_KICK","PENALTY_KICK",
              "FOUL","YELLOW_CARD","RED_CARD","VAR_REVIEW","CELEBRATION"] },
          { "name": "playback_status", "type": "string",
            "enum": ["LIVE_GAMEPLAY","REPLAY"] },
          { "name": "match_start_time", "type": "string" },
          { "name": "match_end_time",   "type": "string" },
          { "name": "description", "type": "string" },
          { "name": "player_names", "type": "array", "items": {"type":"string"} }
        ]
      },
      {
        "id": "H16_broadcast_structure",
        "description": "Classify non-live inserts: replays, halftime, interviews.",
        "fields": [
          { "name": "segment_type", "type": "string",
            "enum": ["REPLAY_ANALYSIS","HALFTIME","POST_MATCH",
              "PLAYER_SPOTLIGHT","CROWD_REACTION","INTERVIEW"] },
          { "name": "description", "type": "string" },
          { "name": "player_names",  "type": "array", "items": {"type":"string"} },
          { "name": "emotional_state", "type": "string" }
        ]
      }
    ]
  }
}
```

The full description text (with per-field instructions) is in the file downloaded via the API Request button on the demo.

03 Response Format

The response is an array of segments keyed by each `segment_definition` id. Each segment contains `start_time` / `end_time` (seconds) and a `metadata` object with the defined fields.


```
{
  "H16_soccer_events": [
    {
      "start_time": 53.0,
      "end_time": 84.0,
      "metadata": {
        "event_type": "GOAL",
        "playback_status": "LIVE_GAMEPLAY",
        "match_start_time": "00:53",
        "match_end_time": "00:58",
        "description": "Pereira scores for Fulham with a chip
                        over the goalkeeper.",
        "player_names": ["Andreas Pereira", "Aaron Ramsdale"]
      }
    }
  ]
}
```


Fields by Layer

LAYER (ID)	KEY FIELDS
L0_shot_boundaries	shot_type, description, main_actions[], emotional_elements, player_names[]
H16_soccer_events	event_type, playback_status (LIVE/REPLAY), match_start_time, match_end_time, description, player_names[]
H16_broadcast_structure	segment_type, description, player_names[], emotional_state

The full response sample (all three layers) is available via the Response JSON button in the demo.

04 Analysis Configurations

Given the on-device / offline-first requirements of the broadcaster, starting with option A or B is recommended.

OPTION	CONFIGURATION	INTERNET	BEST FOR
A — First	Pegasus standalone: extract event JSON, then text search	`Cloud for extraction only`	Live broadcast (offline)
B — Next	Pegasus + Bedrock Claude: JSON-based event chain analysis / natural language queries	`Bedrock (pre-computable)`	Deep analysis (pre-goal buildup)
C — Later	Jockey + Bedrock Claude: multimodal agent search	`Required`	Open-ended pre-show search

Option B passes the match JSON directly to the Claude prompt — no separate vector DB required.

05 Integration Pipeline

Basic integration flow for option A. Pipeline implementation and final UI decisions are handled by the broadcaster. AWS infrastructure (S3, signed URLs) can be guided as needed.

1. Broadcaster: Upload media file to S3
2. Broadcaster: Generate signed URL
3. Call Pegasus 1.5 TBM API — `POST /analyze/tasks`
4. Save result JSON to S3
5. Broadcaster: Parse JSON and apply to frontend / UI

Option B adds an "Analyze JSON with Bedrock Claude" step after step 4.