OTEP — Open Tracking Event Protocol

翻譯僅供參考 — 以英文版本為準。

> 狀態：草案 v0.1 · 一份開放、廠商中立的規範 · 授權：開放（見 §10）

OTEP 是一個適用於任何可追蹤對象生命週期的開放、廠商中立協定。 它定義單一事件模型與單一狀態詞彙表，讓來自任何來源的追蹤事件——自有車隊配送、第三方快遞、承運商標籤等——皆能被 交換、理解並投影至國際標準，而無需為每一方重新整合。

本文件即規範本身：結構、欄位、狀態表、狀態機、外部標準映射，以及建構 合規實作的一致性規則。它與實作無關——它描述的是協定，而非任何特定廠商的內部機制。RFC-2119 關鍵字（MUST / SHOULD / MAY）具規範效力。

1. OTEP 解決的問題

追蹤是碎片化的：每家承運商對欄位與狀態碼的命名各不相同、每個 配送通路以自己的格式回報，而連接全球標準意味著一次又一次地重新整合。OTEP 為生產者與消費者提供單一共同 語言：生產者只需發送一次 OTEP 事件，每個 OTEP 消費者都能理解它們，並能將其投影至所需的標準。

2. 設計原則

1. 事件溯源（Event-sourced）。 事件時間軸是真實來源；「目前狀態」永遠 是一種投影——即最新事件的狀態。
2. What / When / Where / Why。 每個事件都圍繞這四個維度塑形——與 GS1 EPCIS、IATA ONE Record 及 UN/CEFACT 共享的相同維度——因此這些標準 是 OTEP 事件的輸出投影，而非平行的模型。
3. 無清單來源不是障礙。 一個僅暴露目前狀態（無歷史）的來源， 可透過為每個觀察到的變化合成一個事件來處理（§5）。
4. 附加式（Additive）。 OTEP 與任何既有追蹤 API 並存暴露；採用它絕不會 要求對消費者已使用的內容做出破壞性變更。

3. OTEP 時間軸

時間軸是一個信封，承載被追蹤的對象與一份有序的事件清單。

```
{
  "otep_version": "0.1",
  "profile": "parcel",                  // domain profile (§8)
  "subject": {
    "tracking_number": "SR123...",      // ?1 identifier required
    "order_id": 12345,                  // optional
    "package_id": 67890,                // optional
    "external_tracking_number": "1Z...",// optional
    "gs1_sscc": "00...",                // optional — enables EPCIS epcList
    "piece_id": "..."                   // optional — enables ONE Record linkage
  },
  "current_status": "delivered",        // projection of the latest event
  "delivered": true,
  "events": [ /* OTEP events, §4 */ ]
}
```

OTEP 事件

```
{
  // WHAT — carried once at the timeline level (subject above)

  // WHEN
  "occurred_at": "2026-06-10T09:30:00-04:00",  // event instant, ISO-8601 with offset
  "recorded_at": "2026-06-10T09:45:23-04:00",  // ingestion instant (optional)
  "time_type": "actual",                        // actual | estimated | scheduled

  // WHERE
  "location": {
    "name": "Toronto Hub",
```

```
    "code": "YYZ-2",
    "gln": "0614141000005",                      // GS1 GLN ? EPCIS SGLN
    "lat": 43.6777, "lng": -79.6248,
    "country": "CA"                              // ISO 3166-1 alpha-2
  },

  // WHY / WHAT HAPPENED
  "status_code": "out_for_delivery",             // an OTEP status code (§4)
  "phase": "out_for_delivery",                   // derived from status_code
  "incident_reason": null,                        // an OTEP incident reason (§4) when exception

  // WHO
  "actor": { "type": "driver", "name": "Jane D.", "phone": "+1..." },

  // PROVENANCE
  "source": {
    "type": "self_delivery",                     // self_delivery | third_party_delivery | carrier_label
    "provider_id": null,
    "carrier_code": null,
    "external_event_code": null,                  // your raw status code (preserve it)
    "raw": { /* original payload */ }
  },

  // PROOF
  "pod": {
    "photos": [ { "url": "...", "content_base64": null } ],
    "signature": [ { "url": "...", "content_base64": null } ],
    "recipient": "John Smith"
  }
}
```

除了 `subject`、`occurred_at`、`status_code` 與 `source` 之外，每個欄位皆為選填——部分來源僅填入它們所擁有的內容。節點 / 樞紐掃描會將 `location` 設為掃描設施。 高頻 GPS 遙測不是 OTEP 事件；事件在狀態變化或節點掃描時發送。

4. 詞彙表

4.1 狀態碼

20 個標準生命週期碼。 `phase` 永遠可從碼推導而出。

code	phase	終態	POD	含義
`information_submitted`	pre_shipment			已接收訂單資訊
`booking_confirmed`	pre_shipment			承運商 / 預約已確認
`awaiting_pickup`	pre_shipment			已就緒待取件
`out_for_pickup`	pickup			正前往取件途中
`picked_up`	pickup		✓	已自寄件方取件
`pickup_failed`	exception			取件嘗試失敗
`pickup_rescheduled`	exception			取件將重試
`received`	inbound			已於設施收件
`arrival_scan`	inbound			於節點的到達掃描
`in_transit`	transit			運送中
`package_outbound`	transit			已離開設施
`removed_from_route`	exception			已自路線移除
`route_cancelled`	exception			路線已取消
`out_for_delivery`	out_for_delivery			已在車上
`delivered`	delivered	✓	✓	已送達收件人
`delivery_failed`	exception			配送嘗試失敗
`delivery_rescheduled`	exception			將重試 / 重新配送

code	phase	終態	POD	含義
`return_to_sender`	return	✓		退回原點
`rejected_by_recipient`	return	✓		收件人拒收
`cancelled`	return	✓		訂單已取消

4.2 階段（Phases）

`pre_shipment` · `preparing` · `pickup` · `inbound` · `in_custody` · `transit` · `out_for_delivery` · `delivered` · `exception` · `return`。（`preparing` 與 `in_custody` 保留供非包裹設定檔使用——§8。）

4.3 來源類型與時間類型

`source.type`：`self_delivery` · `third_party_delivery` · `carrier_label`。`time_type`：`actual` · `estimated` · `scheduled`（預設 `actual`）。

4.4 事件原因（Incident reasons）

當事件處於 `exception` 階段時，它 SHOULD 攜帶一個來自此 標準化詞彙表的 `incident_reason`：

- 承運商：`carrier_damaged_parcel`、`carrier_sorting_error`、`carrier_address_not_found`、`carrier_parcel_lost`、`carrier_not_enough_time`、`carrier_vehicle_issue`、`carrier_capacity_exceeded`、`carrier_mechanical_delay`
- 零售商 / 寄件方：`retailer_cancelled`、`retailer_incorrect_data`、`retailer_not_ready`、`retailer_incorrect_parcel`、`retailer_incorrect_dimensions`、`retailer_packaging_issue`
- 收件人：`consignee_refused`、`consignee_business_closed`、`consignee_not_available`、`consignee_not_home`、`consignee_cancelled`、`consignee_verification_failed`、`consignee_incorrect_address`、`consignee_access_restricted`、`consignee_safe_place_unavailable`
- 海關：`customs_delay`、`customs_documentation`、`customs_duties_unpaid`、`customs_prohibited`、`customs_inspection`
- 不可抗力：`weather_delay`、`natural_disaster`、`force_majeure`
- 其他：`parcel_being_researched`、`security_issue`、`regulatory_hold`、`unknown`

5. 狀態機與僅狀態來源

階段向前推進；例外狀況會中斷並回復解決。終態（`delivered`、`return_to_sender`、`rejected_by_recipient`、`cancelled`）關閉該對象。

```
pre_shipment ? preparing ? pickup ? inbound ? in_custody ? transit ? out_for_delivery ? delivered ?
                                    ???????????? exception ??????????
                                    ???????????? return / cancelled ?
```

規則：

- 消費者 MUST 依 `occurred_at` 排序事件，並 MUST 容忍亂序到達。
- 進入終態的轉換具冪等性；重複者會被去重。
- 在終態狀態之後，生產者 MUST NOT 發送進一步事件，除非是有文件記載的 RMA / 重新開啟流程。
- 僅暴露目前狀態的來源 MUST 為每個觀察到的變化合成一個事件（搭配穩定的去重鍵），而非省略歷史。隨著時間推移，這些快照會累積 成一條時間軸。

6. 外部標準映射

OTEP 的四個維度與各主要標準逐欄對齊，因此每一者皆為 OTEP 事件的 輸出投影。

OTEP	GS1 EPCIS 2.0	IATA ONE Record	UN/CEFACT
`occurred_at` (+offset)	`eventTime` + `eventTimeZoneOffset`	`eventDate` + `eventTimeType=Actual`	Event Date/Time
`recorded_at`	`recordTime`	recordedAt	—
subject (sscc/piece)	`epcList` (SSCC URN)	`linkedObject` → Piece/Shipment	Consignment
`location` (gln/lat/lng)	`readPoint` / `bizLocation` (SGLN)	`recordedAtLocation` → Location	Location
`status_code`	`bizStep` + `disposition`	`eventCode`	Transport status code
`actor`	sourceList / extension	Actor / Party	Party
`incident_reason`	disposition / ErrorDeclaration	event remark	Status reason code

各碼對應的值 (EPCIS CBV `bizStep` / `disposition` 、ONE Record `eventCode` 、UN/CEFACT code)
發佈於機器可讀的代碼簿中。除這些國際標準外，一條 OTEP 時間軸也能投影至 OpenTelemetry traces、OGC SensorThings observations，以及常見的商務平台 (AfterShip、Shopify、Amazon、Walmart、BigCommerce、Magento、 WooCommerce、Etsy) 。

> 信賴度：EPCIS CBV 值是穩定的標準 URN。ONE Record 與 UN/CEFACT 的代碼值 是針對最後一哩的最佳契合，並 SHOULD 在外部使用前對照官方代碼清單 進行驗證。「送達收件人」並無完全對應的 CBV bizStep——採用最接近的契合 (`receiving` + `received`)，或使用一個使用者詞彙擴充 URN。

7. OTEP API

OTEP 透過一個小型的唯讀 HTTP 介面被消費；所有端點皆為公開。

動詞	路徑	回傳
GET	`/api/v1/otep/trackings/{tracking_number}`	某追蹤號的時間軸
GET	`/api/v1/otep/trackings/{tracking_number}/events`	僅事件
POST	`/api/v1/otep/trackings/batch`	一次呼叫處理多個追蹤號
POST	`/api/v1/otep/validate`	對所提交時間軸的一致性檢查 (§9)

另提供一個暴露相同時間軸的 GraphQL 查詢。

內容協商

相同的時間軸會被序列化為您所請求的任一表示形式，透過 `?format=` 查詢參數或一個 `Accept` 設定檔：

請求	表示形式
`?format=otep` (預設)	原生 OTEP 時間軸
`?format=epcis`	GS1 EPCIS 2.0 (JSON-LD `ObjectEvent` s)
`?format=onerecord`	IATA ONE Record (`LogisticsEvent` s)
`?format=uncefact`	UN/CEFACT transport status
`?format=otlp`	OpenTelemetry traces
`?format=sensorthings`	OGC SensorThings observations
`?format=aftership` | `shopify` | `amazon` | `walmart` | `bigcommerce` | `magento` | `woocommerce` | `etsy`	該平台的追蹤 / 履約格式

在某投影中無法被指派代碼的事件會被略過並計數 (絕不會被靜默 丟棄) 。

8. 領域設定檔 (Domain profiles)

OTEP 是一個通用協定，而非僅限包裹。協定層（事件信封、階段 主幹、狀態機）是通用的；具體的狀態碼則屬於一個透過 `profile` 宣告於時間軸上的設定檔（profile）。§4 中的代碼是 **`parcel`** 設定檔。其他領域——
搬家、餐飲外送、倉儲等——在其設定檔之下加入自己的代碼集， 命名空間為 `otep:<profile>:<code>`
，每一者皆向上映射至相同的階段主幹。新增設定檔 是一項擴充，而非協定變更。

9. 一致性規範（規範性）

當生產者或消費者所發送或接受的每個事件都滿足 這些表格與規則時，它即為 OTEP 合規（OTEP-conformant）。

9.1 時間軸信封

欄位	型別	必要性	約束
`otep_version`	string	MUST	semver，例如 `0.1`
`profile`	string	MUST	一個已註冊的設定檔
`subject`	object	MUST	§9.2
`current_status`	string|null	SHOULD	一個狀態碼（§4）
`current_phase`	string|null	SHOULD	若兩者皆存在，MUST 等於 `current_status` 的階段
`delivered`	boolean	SHOULD	當且僅當 `current_status` = `delivered` 時為 `true`
`events`	array	MUST	事件物件（§9.3），可依 `occurred_at` 排序

9.2 **`subject`**

`tracking_number` / `order_id` / `package_id` 中 MUST 至少存在其一。

欄位	型別	約束
`tracking_number`	string	非空
`order_id` / `package_id`	integer|null	
`external_tracking_number`	string|null	
`gs1_sscc`	string|null	18 位數字
`piece_id`	string|null	

9.3 事件物件

#	欄位	型別	必要性	約束
1	`occurred_at`	string	MUST	含偏移量的 ISO-8601
2	`recorded_at`	string|null	SHOULD	ISO-8601
3	`time_type`	string	MAY（預設 `actual`）	`actual` | `estimated` | `scheduled`
4	`status_code`	string|null	MUST[1]	§4.1 中的一個代碼
5	`phase`	string|null	SHOULD	MUST 等於 `status_code` 的階段
6	`incident_reason`	string|null	SHOULD[2]	§4.4 中的一個原因
7	`description`	string|null	MAY	人類可讀
8	`location`	object|null	MAY	§9.4
9	`actor`	object|null	MAY	`{ type, name?, phone? }`；type ∈ {driver, operator, carrier, system}
10	`source`	object	MUST	§9.5
11	`pod`	object|null	MAY[3]	`{ photos[], signature[], recipient? }`

[1] 一個已編碼的事件 MUST 攜帶來自 §4.1 的狀態碼。一個您尚無法分類的原始掃描 MAY 設定 `status_code = null`，但 MUST 將原生代碼保存於 `source.external_event_code`，並 MUST 被計數，絕不丟棄。 [2] 一個 `exception` 階段的事件 SHOULD 攜帶一個 `incident_reason`。 [3] `status_code` ∈ {`delivered`, `picked_up`} 的事件 SHOULD 攜帶一個 `pod`。

9.4 `location`

`name`（string）· `code`（string）· `gln`（GS1 GLN，13 位數字）· `lat` / `lng`（WGS-84）· `country`（ISO 3166-1 alpha-2）。皆為選填。

9.5 `source`

欄位	型別	必要性	約束
`type`	string	MUST	`self_delivery` | `third_party_delivery` | `carrier_label`
`provider_id`	integer|null	MAY	
`carrier_code`	string|null	MAY	
`external_event_code`	string|null	SHOULD[4]	您的原始狀態碼
`raw`	object|null	MAY	原始酬載

[4] 當 `status_code` 為 null 時 MUST 存在，以確保原生代碼絕不遺失。

9.6 規則

1. 時間。 `occurred_at` MUST 可解析為 ISO-8601。於匯入時正規化數值紀元與 `.NET /Date(ms)/`； 不要發送那些形式。
2. 排序 / 去重。 消費者 MUST 依 `occurred_at` 排序、容忍亂序到達， 並對（ `subject`、 `status_code`、 `occurred_at` ）去重。
3. 狀態機。 在終態狀態之後，不要發送進一步事件，除非是有文件記載的 RMA / 重新開啟。
4. 無靜默遺失。 無法被編碼的事件 MUST 被計數，絕不丟棄。
5. 推導。 `phase`、 `current_status`、 `current_phase`、 `delivered` 皆為投影——若 存在，它們 MUST 與事件時間軸一致。

9.7 一致性層級與驗證

- 層級 1（Level 1） — 發送信封（§9.1）、含必要欄位的事件（§9.3）、有效的狀態碼（§4.1）、有效的階段（§4.2），並遵守狀態機（§5）。
- 層級 2（Level 2） — 額外發送至少一個外部標準投影（§6），並在適用時， 發送設定檔專屬的代碼（§8）。

驗證您的輸出：將一條時間軸 POST 至 `POST /api/v1/otep/validate`。將任何 `errors` 視為阻斷；處理 `warnings`。機器可讀的 JSON Schema 與完整的 代碼簿（每個狀態碼、階段與外部映射）皆已發佈供離線驗證。

10. 開放性與治理

OTEP 是一份開放規範，任何一方皆可免費實作。

- 規範性 vs 資訊性。 規範性：事件信封、階段主幹、狀態詞彙表、 狀態機與外部標準欄位映射。實作者如何將 OTEP 綁定至其自身的內部系統是其自身的事務，不在此處範圍內。
- 穩定識別碼。 代碼以 `otep:<profile>:<code>` 定址，階段以 `otep:phase:<name>` 定址。一旦在某已發佈版本中發佈，識別碼的含義即不可變更。
- 版本控制。 語意化版本控制。新增代碼 / 設定檔屬於 MINOR（向後相容） 變更；變更既有代碼的含義屬於 MAJOR 變更，並 SHOULD 避免。協定 版本隨每條時間軸傳遞（ `otep_version` ）。
- 擴充。 新的設定檔與代碼是針對本規範提出，而非分叉，因此 獨立實作者得以收斂。實驗性代碼 MAY 使用 `x-` 前綴（ `otep:parcel:x-my_code` ）直到註冊為止。
- 授權。 本規範意圖以一個開放授權發佈——待定， 尚待簽核。

11. 廠商互通性 — 自帶你的標準

OTEP 歡迎其他廠商帶來他們自己的追蹤事件標準，使 OTEP 能與其 互通，雙向皆可：

- 將 OTEP 投影 → 你的標準。 定義一個從 OTEP 時間軸到你格式的映射， 重用 OTEP 狀態詞彙表。它是一個純轉換——時間軸進、你的結構 出——因此映射只需編寫一次，每個 OTEP 生產者便都能發送你的格式。
- 將你的標準映射 → OTEP。 提供一個從你的狀態詞彙表到 OTEP 代碼（§4）的對照表，並在需要時加上一個設定檔（§8）。你的原始代碼會保存於 `source.external_event_code`；未映射的代碼會被計數，絕不丟棄。

即使你無法直接採用 OTEP，仍歡迎在你自己的 API 回應中加入單一正規化的 `otep_status`，並分享你的追蹤事件代碼以供對照映射。針對本規範提出 映射（而非分叉），使實作者收斂；新格式接入 相同的 `?format=` 內容協商，且絕不破壞既有消費者。