Dokument pól – APM .NET Core (Elastic APM / ECS)

# 1) Zakres i strumienie danych

**• Typy zdarzeń (processor.event):** transaction, span, error, metric.

**• Standard ECS + APM:** Pola wspólne (service.\*, host.\*, cloud.\*, labels.\*, user.\*) oraz specyficzne dla APM (transaction.\*, span.\*, error.\*).

**• Typowe data streams / indeksy:** traces-apm-\*, logs-apm.error-\*, metrics-apm.\* (praca głównie z aplikacją APM w Kibanie).

# 2) Kontekst wspólny (dla wszystkich typów zdarzeń)

|  |  |  |
| --- | --- | --- |
| Pole | Przykład | Opis / Zastosowanie |
| @timestamp | 2025-08-22T13:41:05.123Z | Czas zdarzenia. Oś czasu, okna czasowe, korelacje. |
| event.dataset | apm | Nazwa zestawu danych. Ułatwia filtrowanie danych APM. |
| event.outcome | success / failure / unknown | Szybkie rozróżnienie powodzenia operacji. Agregacje „error rate”. |
| trace.id | 8f3…c2a | ID śladu – łączy transakcje i spany w jeden przepływ. Podstawa korelacji. |
| service.name | Orders.Api | Nazwa usługi z agenta .NET. Segmentacja dashboardów i SLO. |
| service.version | 1.7.3 | Wersja wdrożenia; porównania „przed/po release”. |
| service.environment | prod / dev | Środowisko; filtry i RBAC. |
| agent.name | dotnet | Agent APM (tu: .NET). Diagnozy kompatybilności. |
| agent.version | X.Y.Z | Wersja agenta; analiza regresji po aktualizacji. |
| host.name | api-01 | Nazwa hosta; korelacja z Metricbeat/Infra. |
| container.id | … | ID kontenera; analizy w Kubernetes/Docker. |
| kubernetes.pod.name | orders-api-7c9… | Mapowanie do Poda; porównania replik. |
| cloud.provider/region | azure / westeurope | Segmentacje kosztów/wydajności per region. |
| labels.\* | labels.tenant: acme | Tagowanie domenowe (klient, funkcja, feature flag). |
| user.id / user.email / user.name | 42 / a@b | Kontekst użytkownika biznesowego (jeśli zbierany). |
| user\_agent.\* | Chrome 124 | Przeglądarka/agent; przydatne w RUM/edge. |
| client.ip / source.ip | 203.0.113.10 | IP klienta; triage incydentów i rate limiting. |

# 3) Transakcje (processor.event: transaction)

|  |  |  |
| --- | --- | --- |
| Pole | Przykład | Opis / Zastosowanie |
| transaction.id | a1b2… | ID transakcji (korzeń śladu lub wewnętrzna). |
| transaction.name | GET /api/orders/{id} | Nazwa logiczna (route). Podstawa grupowania. |
| transaction.type | request / background | Rodzaj pracy (HTTP, zadanie, kolejka). |
| transaction.result | HTTP 2xx | Wynik biznesowo-techniczny (np. statusowy). |
| transaction.duration.us | 154321 | Czas trwania w µs. p95/p99, SLO, alerty latency. |
| transaction.sampled | true/false | Czy transakcja była próbkowana (wpływ na wolumen). |
| http.request.method | GET | Filtrowanie po metodach, reguły limitów. |
| http.response.status\_code | 200 / 500 | Error rate, 4xx/5xx heatmapy. |
| http.version | 1.1 / 2 | Diagnostyka protokołu. |
| url.original / url.path / url.query | /api/orders/1 | Grupowanie po ścieżkach, analiza query. |
| destination.service.resource | sqlserver / redis | Cel transakcji (zależność główna). |
| faas.\* |  | Jeśli serverless – identyfikacja invokacji FaaS. |

Użycie:

* • SLO/SLI: budowa wskaźników na transaction.duration.us i event.outcome.
* • Observability end-to-end: wejście do śladu (trace.id) i zejście do spanów zależności.

# 4) Spany (processor.event: span)

|  |  |  |
| --- | --- | --- |
| Pole | Przykład | Opis / Zastosowanie |
| span.id | c3d4… | ID spanu (odcinka pracy w transakcji). |
| span.name | SELECT Orders WHERE … | Nazwa akcji (DB, HTTP, cache). |
| span.type | db / external / messaging / cache | Kategoria spanu. |
| span.subtype | mssql / http / rabbitmq / redis | Podkategoria/technologia. |
| span.action | query / request / publish | Rodzaj działania. |
| span.duration.us | 83120 | Czas spanu; analiza wąskich gardeł. |
| span.destination.service.resource | mssql://ordersdb | Cel zależności do wykresu usług. |
| db.type | sql | Klasa bazy danych. |
| db.statement | SELECT … | Treść zapytania (jeśli włączone). |
| db.instance / db.user | OrdersDb / sa | Kontekst połączenia (ostrożnie z PII). |
| http.request.method | POST | Dla spanów HTTP (wywołania zewn.). |
| http.response.status\_code | 201 | Szybka diagnoza błędów zależności. |
| message.queue.name | orders-created | Kolejki (RabbitMQ/Kafka/Azure SB). |
| message.routing\_key / message.age.ms |  | Opóźnienia i routingi w messagingu. |

Użycie:

* • „Top slow spans” → priorytety optymalizacji (DB/HTTP/cache).
* • Mapy zależności: kto do kogo woła i z jaką latencją.

# 5) Błędy (processor.event: error)

|  |  |  |
| --- | --- | --- |
| Pole | Przykład | Opis / Zastosowanie |
| error.id | e9f… | ID błędu. |
| error.exception.type | SqlException | Typ wyjątku .NET. |
| error.exception.message | Timeout expired | Komunikat błędu (maskować PII). |
| error.exception.stacktrace[] | ramek stosu | Diagnostyka miejsca awarii. |
| error.culprit | OrdersController.Get | Miejsce wskazane jako winowajca. |
| error.log.level | Error | Poziom z loggera (jeśli mapowany). |
| error.grouping\_key | … | Stabilne grupowanie podobnych błędów. |
| transaction.id / trace.id |  | Powiązanie błędu z transakcją/śladem. |
| service.\* / host.\* / labels.\* |  | Taki sam kontekst jak w innych typach. |

Użycie:

* • „Top error groups” – priorytety poprawek.
* • Korelacja z transakcją i spanem celu (np. błąd DB → wolny span DB).

# 6) Metryki APM (processor.event: metric)

|  |  |  |
| --- | --- | --- |
| Pole | Przykład | Opis / Zastosowanie |
| metricset.name | transaction / service | Nazwa zestawu metryk. |
| transaction.duration.count | 1234 | Licznik transakcji w oknie metryki. |
| transaction.duration.sum.us | … | Łączny czas transakcji; średnie/p50/p95. |
| span.self\_time.count/sum.us | … | Czas własny spanów (jeśli dostępne). |
| faas.duration.\* |  | Metryki FaaS (jeśli dotyczy). |
| system.\* |  | Rzadziej przez APM; zależnie od konfiguracji/źródeł. |

Użycie:

* • Ekonomiczne agregacje dla SLI/SLO bez pełnego próbkowania.
* • Wykresy obciążenia usług w czasie.

# 7) HTTP, URL, sieć – pola wspólne

|  |  |  |
| --- | --- | --- |
| Pole | Przykład | Opis / Zastosowanie |
| http.request.method | GET | Segmentacje i kontrola ruchu. |
| http.request.body.bytes | 512 | Wielkość żądania (jeśli włączone). |
| http.response.status\_code | 200 | Error rate. |
| http.response.body.bytes | 4096 | Wielkość odpowiedzi. |
| url.original / url.path / url.query |  | Grupowanie po ścieżkach, analiza parametrów. |
| url.scheme / url.domain / url.port |  | Routing, LB, certyfikaty. |
| client.ip / client.port |  | Klient końcowy (edge). |
| destination.address / destination.port |  | Cel połączenia wychodzącego. |
| server.address / server.port |  | Serwer obsługujący żądanie. |

# 8) Użyte pola domenowe (DB, messaging, cache, runtime)

|  |  |  |
| --- | --- | --- |
| Domenowe | Kluczowe pola | Zastosowanie |
| Baza danych | db.type, db.statement, db.instance, db.user | Optymalizacja zapytań, indeksów, timeoutów. |
| Messaging | message.queue.name, message.routing\_key, message.age.ms | Lag kolejki, retry, dead letters. |
| Cache | span.type: cache, span.subtype: redis/memory | Skuteczność cache vs. źródło prawdy. |
| .NET runtime | service.runtime.name/version (jeśli dostępne) | Rewizje frameworka a wydajność. |

# 9) KQL – przykładowe kwerendy (Kibana)

1. processor.event: "transaction" and service.environment: "prod" and http.response.status\_code >= 500

2. processor.event: "transaction" and transaction.duration.us >= 2000000

3. processor.event: "span" and span.type: "db" and span.subtype: "mssql" and span.duration.us >= 200000

4. processor.event: "error" and not error.exception.type: "TimeoutException"

5. trace.id: "8f3c…"

6. service.name: "Orders.Api"

7. service.environment: "staging" and http.response.status\_code >= 400 and http.response.status\_code < 500

8. labels.tenant: "acme"

9. processor.event: "span" and span.type: "external" and span.subtype: "http"

10. processor.event: "error" and error.culprit: "\*Orders\*"

# 10) Dobre praktyki zbierania z agenta .NET

* Nazwy transakcji: używaj nazw opartych o route (np. GET /api/orders/{id}), nie o ścieżkę fizyczną.
* Maskowanie PII: nie loguj pełnych payloadów w db.statement, http.request.body, ani danych osobowych w labels.\*.
* Sampling i SLO: łącz pełne metryki (metric) z próbkowaniem śladów, aby utrzymać koszty przy wiarygodnych SLI.
* Wersjonowanie i env: konsekwentnie ustawiaj service.version i service.environment – to podstawa analizy „przed/po”.
* Mapa zależności: dopilnuj, by spany zewnętrzne miały poprawne span.type/subtype/action – lepsza wizualizacja i alerty.

# 11) Minimalny „cheat-sheet” pól krytycznych

* Identyfikacja: service.name, service.environment, service.version, agent.name/version
* Korelacja: trace.id, transaction.id, span.id
* Jakość: event.outcome, http.response.status\_code, error.\*
* Wydajność: transaction.duration.us, span.duration.us
* Zależności: span.type/subtype/action, destination.service.resource
* Kontekst: host.\*, container.\*, kubernetes.\*, labels.\*

# 12) Notatki zgodności/uwagi

* Pola APM są zgodne z ECS (Elastic Common Schema). Dostępność pól zależy od wersji agenta i konfiguracji (np. redakcja payloadów, sampling).
* Nazwy pól trwania są w mikrosekundach (\*.us). W dashboardach Kibany widzisz je w ms/s – to tylko prezentacja.