

# Package: PhysioCompliance (via r-universe)

October 11, 2026

**Type** Package

**Title** Evidence, Privacy, and Lifecycle Controls for Physiological Records

**Version** 0.4.2

**Description** Provides deterministic, tamper-evident audit trails, externally authenticated electronic signatures, explicit de-identification policies, project-owned lifecycle and risk-document templates, normalized requirement-risk-control-test traceability, objective-evidence verification, and read-only ecosystem engineering-readiness checks for PhysioExperiment records and packages. Reports expose structural, linkage, record-state, signature, privacy, traceability, and package-readiness findings without repairing stored evidence. These technical controls and documentation aids do not determine device status, assign risk acceptability, authorize release, or establish conformity with a regulatory framework or standard.

**Depends** R (>= 4.2.0)

**Imports** digest, jsonlite, methods, openssl, PhysioExperiment (>= 2.0.0), S4Vectors, SummarizedExperiment

**Suggests** PhysioCore, BiocCheck, callr, covr, knitr, rmarkdown, testthat (>= 3.0.0)

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**VignetteBuilder** knitr

**RoxygenNote** 7.3.3

**Maintainer** Yusuke Matsui <mail.to.matsui@gmail.com>

**URL** <https://github.com/x-biosignal/PhysioCompliance>,  
<https://x-biosignal.r-universe.dev/PhysioCompliance>,  
<https://x-biosignal.github.io/PhysioCompliance/>

**BugReports** <https://github.com/x-biosignal/PhysioCompliance/issues>

**Config/testthat/edition** 3

**Collate** 'audit-trail.R' 'e-signature.R' 'deidentify.R'  
'pseudonymization.R' 'header-scrub.R' 'data-subject.R'  
'templates.R' 'risk-matrix.R' 'risk-trace.R' 'conformance.R'  
'conformance-write.R' 'package.R'

**Config/pak/sysreqs** libssl-dev zlib1g-dev

**Repository** <https://x-biosignal.r-universe.dev>

**Date/Publication** 2026-10-03 13:11:41 UTC

**RemoteUrl** <https://github.com/x-biosignal/PhysioCompliance>

**RemoteRef** HEAD

**RemoteSha** 508084907afa00eeb862ff05c304ebaba76c0c5

## Contents

|                                  |    |
|----------------------------------|----|
| appendAuditEvent . . . . .       | 3  |
| auditDeidentification . . . . .  | 4  |
| auditTrail . . . . .             | 4  |
| conformanceCheck . . . . .       | 5  |
| conformanceCriteria . . . . .    | 6  |
| dataSubjectErase . . . . .       | 6  |
| dataSubjectExport . . . . .      | 8  |
| dateShift . . . . .              | 8  |
| deidentify . . . . .             | 9  |
| eSign . . . . .                  | 10 |
| headerScrub . . . . .            | 12 |
| initializeAuditTrail . . . . .   | 13 |
| lifecycleTemplate . . . . .      | 14 |
| pseudonymize . . . . .           | 15 |
| reidentify . . . . .             | 16 |
| riskManagementTemplate . . . . . | 16 |
| riskMatrix . . . . .             | 18 |
| safeHarborPolicy . . . . .       | 19 |
| standardsSources . . . . .       | 20 |
| traceabilityMatrix . . . . .     | 20 |
| validateTraceability . . . . .   | 21 |
| verifyAuditTrail . . . . .       | 21 |
| verifyESignatures . . . . .      | 22 |
| writeConformanceReport . . . . . | 23 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>24</b> |
|--------------|-----------|

---

|                  |                              |
|------------------|------------------------------|
| appendAuditEvent | <i>Append an audit event</i> |
|------------------|------------------------------|

---

## Description

Appends an event to a valid audit chain. A changed record may be appended when its historical chain remains valid; the new event then establishes the changed record as the current audited state.

## Usage

```
appendAuditEvent(
  x,
  action,
  actor,
  reason,
  details = list(),
  timestamp = Sys.time()
)
```

## Arguments

|           |                                                                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------|
| x         | An initialized <a href="#">PhysioExperiment::PhysioExperiment</a> object.                                 |
| action    | Non-empty domain action. Names beginning with physio_compliance. are reserved.                            |
| actor     | Non-empty identifier for the responsible actor.                                                           |
| reason    | Non-empty reason for the action.                                                                          |
| details   | A recursively named plain list containing supported atomic values. Named lists are sorted before hashing. |
| timestamp | One finite POSIXct value.                                                                                 |

## Value

A modified copy of x with one additional event.

## Examples

```
x <- PhysioExperiment::PhysioExperiment(
  assays = list(raw = matrix(1:6, nrow = 3)),
  samplingRate = 100
)
x <- initializeAuditTrail(x, actor = "operator-01")
x <- appendAuditEvent(
  x,
  action = "quality_control",
  actor = "operator-01",
  reason = "reviewed acquisition quality"
)
verifyAuditTrail(x)
```

---

`auditDeidentification`    *Audit a stored de-identification transformation*

---

### Description

Rescans supported metadata and validates the stored report. A passing result means that this checker found no configured residual identifier; it is not a HIPAA, GDPR, Safe Harbor, or file-format compliance determination.

### Usage

```
auditDeidentification(x, policy = NULL)
```

### Arguments

|                     |                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------|
| <code>x</code>      | A supported de-identified PhysioCore record.                                                  |
| <code>policy</code> | Optional effective policy. When omitted, a built-in policy matching the stored label is used. |

### Value

A list containing status, findings, severity counts, and check time.

### Examples

```
x <- PhysioExperiment::PhysioExperiment(
  assays = list(raw = matrix(1:6, nrow = 3)),
  colData = S4Vectors::DataFrame(patient_name = c("A", "B")),
  samplingRate = 100
)
clean <- deidentify(x, safeHarborPolicy())
# A pass means no configured residual identifier was found; it is not a
# HIPAA, GDPR, or Safe Harbor determination.
auditDeidentification(clean)$status
```

---

`auditTrail`    *Extract the compliance audit trail*

---

### Description

Extract the compliance audit trail

### Usage

```
auditTrail(x)
```

**Arguments**

`x` A `PhysioExperiment::PhysioExperiment` object.

**Value**

An `audit_trail` data frame with a `details` list-column. An uninitialized object returns a zero-row trail.

**Examples**

```
x <- PhysioExperiment::PhysioExperiment(
  assays = list(raw = matrix(1:6, nrow = 3)),
  samplingRate = 100
)
x <- initializeAuditTrail(x, actor = "operator-01")
auditTrail(x)
```

---

|                               |                                                              |
|-------------------------------|--------------------------------------------------------------|
| <code>conformanceCheck</code> | <i>Run a read-only ecosystem engineering-readiness audit</i> |
|-------------------------------|--------------------------------------------------------------|

---

**Description**

This function statically parses package metadata and source. It does not load namespaces, execute package code, run tests, or launch external tools.

**Usage**

```
conformanceCheck(
  root,
  packages = NULL,
  criteria = conformanceCriteria(),
  coverage = NULL,
  external_results = NULL,
  applicability = NULL,
  checked_at = Sys.time()
)
```

**Arguments**

|                               |                                                                |
|-------------------------------|----------------------------------------------------------------|
| <code>root</code>             | Repository or package root.                                    |
| <code>packages</code>         | Optional exact package-name filter.                            |
| <code>criteria</code>         | Project criteria from <code>conformanceCriteria()</code> .     |
| <code>coverage</code>         | Caller-supplied coverage evidence or <code>NULL</code> .       |
| <code>external_results</code> | Caller-supplied external check evidence or <code>NULL</code> . |
| <code>applicability</code>    | Reviewed applicability decisions or <code>NULL</code> .        |
| <code>checked_at</code>       | Fixed check timestamp.                                         |

**Value**

A deterministic conformance\_report.

---

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| conformanceCriteria | <i>Define PhysioExperiment project conformance criteria</i> |
|---------------------|-------------------------------------------------------------|

---

**Description**

Thresholds are project policies for an engineering-readiness check. They are not regulatory thresholds or clauses of a standard.

**Usage**

```
conformanceCriteria(example_floor = 0.5, coverage_floor = 0.8)
```

**Arguments**

- example\_floor    Minimum fraction of documented exports with examples.
- coverage\_floor    Minimum caller-supplied line coverage.

**Value**

A conformance\_criteria data frame.

**Examples**

```
criteria <- conformanceCriteria()
criteria[, c("criterion_id", "default_applicability", "threshold")]
```

---

|                  |                                             |
|------------------|---------------------------------------------|
| dataSubjectErase | <i>Plan or apply a data-subject erasure</i> |
|------------------|---------------------------------------------|

---

**Description**

Builds a no-content plan before authorization. Applying the plan changes only the returned in-memory collection; it does not delete files, databases, backups, replicas, caches, or remote systems. Identity verification, applicable legal bases, Article 17 exceptions, retention obligations, and operational deletion remain controller responsibilities.

**Usage**

```
dataSubjectErase(  
  records,  
  subject_id,  
  locate,  
  authorize,  
  retain = NULL,  
  reason,  
  mode = c("plan", "apply")  
)
```

**Arguments**

|            |                                                                                  |
|------------|----------------------------------------------------------------------------------|
| records    | A named plain list of records.                                                   |
| subject_id | One requested subject identifier.                                                |
| locate     | A callback called exactly once per record.                                       |
| authorize  | A callback invoked exactly once in apply mode with the plan.                     |
| retain     | Optional callback returning a non-empty retention reason or NULL/NA for erasure. |
| reason     | Non-empty operational reason. It must not contain the subject identifier.        |
| mode       | Build a no-op plan or apply an authorized plan.                                  |

**Value**

A data\_subject\_erasure plan or applied result.

**Examples**

```
records <- list(  
  visit_01 = list(subject = "S1"),  
  visit_02 = list(subject = "S2")  
)  
# Plan mode builds a no-content plan and performs no deletion; apply mode  
# changes only the returned in-memory collection, never files or backups.  
plan <- dataSubjectErase(  
  records, "S1",  
  locate = function(record) record$subject,  
  authorize = function(plan) TRUE,  
  reason = "verified erasure request",  
  mode = "plan"  
)  
plan$plan
```

---

|                   |                                             |
|-------------------|---------------------------------------------|
| dataSubjectExport | <i>Prepare a data-subject access export</i> |
|-------------------|---------------------------------------------|

---

### Description

Selects records through a caller-owned identity matcher and returns deep copies with a content manifest. Identity verification, the scope and format of an Article 15 response, third-party rights, and secure delivery remain controller responsibilities.

### Usage

```
dataSubjectExport(records, subject_id, locate)
```

### Arguments

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| records    | A named plain list of records.                                                      |
| subject_id | One requested subject identifier.                                                   |
| locate     | A callback called exactly once per record. It returns one subject identifier or NA. |

### Value

A `data_subject_export` object.

### Examples

```
records <- list(
  visit_01 = list(subject = "S1", note = "baseline"),
  visit_02 = list(subject = "S2", note = "baseline")
)
# `locate` is a caller-owned identity matcher called once per record.
dataSubjectExport(records, "S1", locate = function(record) record$subject)
```

---

|           |                                                |
|-----------|------------------------------------------------|
| dateShift | <i>Shift dates consistently within subject</i> |
|-----------|------------------------------------------------|

---

### Description

Derives one deterministic integer-day offset per subject using keyed HMAC-SHA-256. Exact elapsed intervals are preserved. Full shifted dates are pseudonymized personal data, not a HIPAA Safe Harbor output.

**Usage**

```
dateShift(
  dates,
  subject_id,
  key,
  range_days = c(-3650L, 3650L),
  inverse = FALSE
)
```

**Arguments**

|            |                                                           |
|------------|-----------------------------------------------------------|
| dates      | A Date or POSIXct vector.                                 |
| subject_id | One subject ID or a vector aligned to dates.              |
| key        | Raw vector of at least 32 cryptographically random bytes. |
| range_days | Inclusive integer offset range.                           |
| inverse    | Apply the exact negative keyed offset.                    |

**Value**

A vector with the same length and date class.

**Examples**

```
key <- openssl::rand_bytes(32)
visits <- as.Date(c("2026-01-05", "2026-02-02", "2026-03-09"))
# One deterministic per-subject offset; exact intervals are preserved.
shifted <- dateShift(visits, subject_id = "subject-a", key = key)
all(diff(shifted) == diff(visits))
```

---

deidentify

*De-identify PhysioCore records*


---

**Description**

Applies an explicit field policy while preserving the input S4 class. Assays are never inspected for identifying signal content; retained assays are therefore listed for manual review. Reports contain paths and counts, never removed values, keys, or re-identification material.

**Usage**

```
deidentify(
  x,
  policy = safeHarborPolicy(),
  subject_id = NULL,
  pseudonymization = NULL,
  date_key = NULL,
```

```

    audit_actor = NULL,
    audit_reason = "record de-identified"
  )

```

### Arguments

|                               |                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                | A <code>PhysioExperiment</code> , <code>MultiPhysioExperiment</code> , <code>PhysioLongitudinal</code> , or <code>PhysioCohort</code> (a multi-subject container whose subject-level <code>colData</code> and every subject timeline are de-identified; unlike <code>MultiRate/Longitudinal</code> it has a metadata slot, so its merged report is stored there). |
| <code>policy</code>           | A policy from <code>safeHarborPolicy()</code> or <code>pseudonymizedPolicy()</code> .                                                                                                                                                                                                                                                                             |
| <code>subject_id</code>       | A scalar or field-aligned subject identifier used only for keyed date shifting.                                                                                                                                                                                                                                                                                   |
| <code>pseudonymization</code> | A protected result from <code>pseudonymize()</code> used to replace direct identifiers.                                                                                                                                                                                                                                                                           |
| <code>date_key</code>         | A caller-owned raw key of at least 32 bytes.                                                                                                                                                                                                                                                                                                                      |
| <code>audit_actor</code>      | Responsible actor when an input audit trail is initialized.                                                                                                                                                                                                                                                                                                       |
| <code>audit_reason</code>     | Non-empty reason stored in linked audit events.                                                                                                                                                                                                                                                                                                                   |

### Details

`MultiPhysioExperiment` and `PhysioLongitudinal` do not provide a metadata slot. Their aggregate report is stored as a serializable deidentification attribute, while every child `PhysioExperiment` keeps its report in `metadata()`. Existing audit trails are linked per child.

### Value

A modified object of the same S4 class.

### Examples

```

x <- PhysioExperiment::PhysioExperiment(
  assays = list(raw = matrix(1:6, nrow = 3)),
  colData = S4Vectors::DataFrame(patient_name = c("A", "B")),
  samplingRate = 100
)
# Changes only the returned in-memory object; stored evidence is not repaired.
clean <- deidentify(x, safeHarborPolicy())
auditDeidentification(clean)$status

```

---

eSign

---

*Add an externally authenticated electronic signature*


---

### Description

The callback is responsible for authenticating and authorizing the signer. It is called exactly once with canonical challenge bytes and a sanitized public credential descriptor. The package never accepts passwords, private keys, API tokens, or reusable secrets.

**Usage**

```
eSign(x, signer, meaning, credential, sign, timestamp = Sys.time())
```

**Arguments**

|            |                                                                                                               |
|------------|---------------------------------------------------------------------------------------------------------------|
| x          | An initialized, currently valid <a href="#">PhysioExperiment::PhysioExperiment</a> .                          |
| signer     | Non-empty signer identity.                                                                                    |
| meaning    | Non-empty signature meaning.                                                                                  |
| credential | Named list containing id, algorithm, fingerprint, and optional non-secret public_data.                        |
| sign       | Function called once as <code>sign(challenge_raw, credential)</code> . It must return a non-empty raw vector. |
| timestamp  | One finite POSIXct value that does not precede the audit head.                                                |

**Details**

The callback's algorithm and credential service determine cryptographic strength, identity assurance, and non-repudiation properties. This function is a technical control and does not itself establish regulatory compliance.

**Value**

A modified copy of x with one signature and one cross-linked audit event.

**Examples**

```
x <- PhysioExperiment::PhysioExperiment(
  assays = list(raw = matrix(1:6, nrow = 3)),
  samplingRate = 100
)
x <- initializeAuditTrail(x, actor = "operator-01")
# Signing is delegated to a caller-supplied callback. This illustrative one
# hashes the challenge; a real deployment authenticates the signer and uses
# its own credential service. The package stores no secrets.
credential <- list(id = "credential-01", algorithm = "demo-sha256",
  fingerprint = "01:23:45:67")
sign <- function(challenge, credential) {
  digest::digest(c(challenge, serialize(credential, NULL, version = 3)),
    algo = "sha256", serialize = FALSE, raw = TRUE)
}
x <- eSign(x, signer = "reviewer-02", meaning = "reviewed",
  credential = credential, sign = sign)
verifyAuditTrail(x)$n_signatures
```

---

|             |                                                       |
|-------------|-------------------------------------------------------|
| headerScrub | <i>Scrub a parsed physiological or imaging header</i> |
|-------------|-------------------------------------------------------|

---

### Description

Scrubs a named structured header without parsing or rewriting a binary file. The DICOM-adjacent mode implements a documented conservative subset, not a DICOM PS3.15 conformance statement. DICOM private attributes, pixels, overlays, and burned-in annotations always remain manual-review items.

### Usage

```
headerScrub(
  header,
  format = c("auto", "edf", "brainvision", "snirf", "dicom"),
  subject_token = NULL,
  date_key = NULL,
  subject_id = NULL,
  free_text = c("error", "drop")
)
```

### Arguments

|               |                                                                                                  |
|---------------|--------------------------------------------------------------------------------------------------|
| header        | A named plain list, data frame, or <code>DataFrame</code> .                                      |
| format        | Explicit format, or "auto" when the object carries exactly one explicit supported format marker. |
| subject_token | Optional safe replacement token.                                                                 |
| date_key      | Caller-owned raw date-shifting key.                                                              |
| subject_id    | Subject identifier used only to derive a date offset.                                            |
| free_text     | Refuse or drop detected free-text fields.                                                        |

### Value

A `header_scrub` object containing the scrubbed header and reports.

### Examples

```
header <- list(
  patient_id = "source-patient",
  recording_id = "source-recording",
  start_date = as.Date("2024-02-29"),
  comments = "free text note",
  signal_headers = list(label = c("ECG", "EMG"), unit = c("mV", "mV"))
)
# Scrubs a parsed header only; it does not read or rewrite the binary file.
result <- headerScrub(header, "edf", subject_token = "psn_safe",
  free_text = "drop")
```

```
result$header$patient_id  
result
```

---

|                      |                                                |
|----------------------|------------------------------------------------|
| initializeAuditTrail | <i>Initialize a tamper-evident audit trail</i> |
|----------------------|------------------------------------------------|

---

## Description

Initializes a versioned SHA-256 audit chain over the complete `PhysioExperiment` record. The genesis event snapshots the existing `PhysioCore` provenance log. Existing compliance metadata is never replaced.

## Usage

```
initializeAuditTrail(  
  x,  
  actor,  
  reason = "audit trail initialized",  
  timestamp = Sys.time()  
)
```

## Arguments

|                        |                                                           |
|------------------------|-----------------------------------------------------------|
| <code>x</code>         | A <code>PhysioExperiment::PhysioExperiment</code> object. |
| <code>actor</code>     | Non-empty identifier for the responsible actor.           |
| <code>reason</code>    | Non-empty reason for initialization.                      |
| <code>timestamp</code> | One finite <code>POSIXct</code> value.                    |

## Value

A modified copy of `x` with a genesis audit event.

## Examples

```
x <- PhysioExperiment::PhysioExperiment(  
  assays = list(raw = matrix(1:6, nrow = 3)),  
  samplingRate = 100  
)  
x <- initializeAuditTrail(x, actor = "operator-01")  
verifyAuditTrail(x)
```

---

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| lifecycleTemplate | <i>Render a project-owned software lifecycle template set</i> |
|-------------------|---------------------------------------------------------------|

---

## Description

Render a project-owned software lifecycle template set

## Usage

```
lifecycleTemplate(
  out_dir,
  project,
  intended_use,
  software_safety_class = c("unclassified", "A", "B", "C"),
  owner,
  effective_date = Sys.Date(),
  source_editions = standardsSources(),
  overwrite = FALSE
)
```

## Arguments

|                       |                                                           |
|-----------------------|-----------------------------------------------------------|
| out_dir               | Destination child directory.                              |
| project               | Project name. Path separators are not allowed.            |
| intended_use          | Intended-use statement supplied by the project.           |
| software_safety_class | Project-supplied class or "unclassified".                 |
| owner                 | Recorded document owner.                                  |
| effective_date        | Effective date to render.                                 |
| source_editions       | Metadata returned by <a href="#">standardsSources()</a> . |
| overwrite             | Whether to replace an existing destination atomically.    |

## Value

A lifecycle\_template manifest.

## Examples

```
parent <- file.path(tempdir(), "pc-lifecycle-demo")
dir.create(parent, showWarnings = FALSE)
tpl <- lifecycleTemplate(
  out_dir = file.path(parent, "lifecycle"),
  project = "Example Project",
  intended_use = "Research data processing",
  owner = "project-owner",
```

```

    effective_date = as.Date("2026-07-28")
  )
  tmp1
  unlink(parent, recursive = TRUE)

```

pseudonymize

*Pseudonymize identifiers with a caller-owned key*

## Description

Creates deterministic HMAC-derived or random 128-bit tokens and stores the one-to-one re-identification map in an authenticated encrypted envelope. The encrypted map remains sensitive additional information and should be stored separately from pseudonymized records with deployment-level access controls.

## Usage

```

pseudonymize(
  identifiers,
  key,
  namespace,
  method = c("hmac_sha256", "random"),
  encrypted_map = NULL
)

```

## Arguments

|                            |                                                                                              |
|----------------------------|----------------------------------------------------------------------------------------------|
| <code>identifiers</code>   | Character vector. NA values remain NA.                                                       |
| <code>key</code>           | Raw vector of at least 32 cryptographically random bytes.                                    |
| <code>namespace</code>     | Non-empty deployment-specific namespace.                                                     |
| <code>method</code>        | Token method.                                                                                |
| <code>encrypted_map</code> | Optional prior pseudonymization object or protected map envelope used to continue a mapping. |

## Details

This is a pseudonymization primitive, not a declaration of anonymization, HIPAA Safe Harbor status, or GDPR compliance.

## Value

A pseudonymization object containing tokens and an authenticated encrypted map, but no plain-text identifier map or key.

**Examples**

```
key <- openssl::rand_bytes(32)
p <- pseudonymize(c("subject-a", "subject-b", "subject-a"),
                  key, namespace = "study-example")
p
identical(reidentify(p, key), c("subject-a", "subject-b", "subject-a"))
```

---

|            |                                                                  |
|------------|------------------------------------------------------------------|
| reidentify | <i>Recover identifiers from a protected pseudonymization map</i> |
|------------|------------------------------------------------------------------|

---

**Description**

Recover identifiers from a protected pseudonymization map

**Usage**

```
reidentify(x, key)
```

**Arguments**

|     |                                                  |
|-----|--------------------------------------------------|
| x   | A pseudonymization object.                       |
| key | The caller-owned raw key used to create the map. |

**Value**

Character vector aligned to x\$values.

**Examples**

```
key <- openssl::rand_bytes(32)
p <- pseudonymize(c("subject-a", "subject-b"), key,
                  namespace = "study-example")
reidentify(p, key)
```

---

|                        |                                                            |
|------------------------|------------------------------------------------------------|
| riskManagementTemplate | <i>Render a project-owned risk-management template set</i> |
|------------------------|------------------------------------------------------------|

---

**Description**

Render a project-owned risk-management template set

## Usage

```
riskManagementTemplate(  
  out_dir,  
  project,  
  intended_use,  
  owner,  
  effective_date = Sys.Date(),  
  risk_matrix = NULL,  
  source_editions = standardsSources(),  
  overwrite = FALSE  
)
```

## Arguments

|                 |                                                                       |
|-----------------|-----------------------------------------------------------------------|
| out_dir         | Destination child directory.                                          |
| project         | Project name. Path separators are not allowed.                        |
| intended_use    | Intended-use statement supplied by the project.                       |
| owner           | Recorded document owner.                                              |
| effective_date  | Effective date to render.                                             |
| risk_matrix     | Optional project-owned <a href="#">riskMatrix()</a> object to record. |
| source_editions | Metadata returned by <a href="#">standardsSources()</a> .             |
| overwrite       | Whether to replace an existing destination atomically.                |

## Value

A lifecycle\_template manifest.

## Examples

```
parent <- file.path(tempdir(), "pc-risk-demo")  
dir.create(parent, showWarnings = FALSE)  
tmpl <- riskManagementTemplate(  
  out_dir = file.path(parent, "risk"),  
  project = "Example Project",  
  intended_use = "Research data processing",  
  owner = "project-owner",  
  effective_date = as.Date("2026-07-28")  
)  
tmpl  
unlink(parent, recursive = TRUE)
```

---

riskMatrix

*Define a project-owned risk decision matrix*


---

## Description

The constructor records caller-supplied ordinal levels and one explicit decision for every severity/probability pair. It does not multiply ranks or infer a risk-acceptability policy.

## Usage

```
riskMatrix(severity, probability, decisions, matrix_id, version, rationale)
```

## Arguments

|                               |                                                             |
|-------------------------------|-------------------------------------------------------------|
| severity                      | Severity levels with level_id, rank, label, and definition. |
| probability                   | Probability levels with the same columns.                   |
| decisions                     | One explicit decision for every Cartesian level pair.       |
| matrix_id, version, rationale | Non-empty project identifiers and policy rationale.         |

## Value

A deterministic risk\_matrix object.

## Examples

```
severity <- data.frame(
  level_id = c("S1", "S2"), rank = 1:2,
  label = c("Minor", "Serious"),
  definition = c("Limited impact", "Significant impact"),
  stringsAsFactors = FALSE
)
probability <- data.frame(
  level_id = c("P1", "P2"), rank = 1:2,
  label = c("Unlikely", "Likely"),
  definition = c("Rarely occurs", "Often occurs"),
  stringsAsFactors = FALSE
)
decisions <- expand.grid(
  severity_id = severity$level_id, probability_id = probability$level_id,
  stringsAsFactors = FALSE
)
# One explicit project decision for every pair; ranks are never multiplied
# and no universal acceptability threshold is supplied.
decisions$decision <- c("acceptable", "review_required",
  "review_required", "unacceptable")
decisions$rationale <- paste0("POLICY-", seq_len(nrow(decisions)))
riskMatrix(severity, probability, decisions,
```

```
matrix_id = "RM01", version = "1.0",
rationale = "Project-owned decision policy")
```

---

safeHarborPolicy

*Conservative de-identification policies*


---

## Description

Constructs a versioned field policy. These policies are technical controls, not legal determinations. A `safe_harbor_candidate` still requires the no-actual-knowledge and organizational review required by the applicable workflow. Pseudonymized data remain personal data when separately held information permits re-identification.

## Usage

```
safeHarborPolicy(
  additional_fields = list(),
  date_action = c("year", "remove"),
  free_text = c("error", "drop"),
  biometric_data = c("error", "drop")
)

pseudonymizedPolicy(
  additional_fields = list(),
  free_text = c("error", "drop", "retain")
)
```

## Arguments

|                                |                                                                                   |
|--------------------------------|-----------------------------------------------------------------------------------|
| <code>additional_fields</code> | A named plain list mapping canonical category IDs to deployment-specific aliases. |
| <code>date_action</code>       | Replace person-related dates with year-only values or remove them.                |
| <code>free_text</code>         | Refuse, drop, or (for pseudonymized data only) retain detected free text.         |
| <code>biometric_data</code>    | Refuse or drop detected biometric/image content.                                  |

## Value

A `deidentification_policy` data frame.

## Examples

```
# A conservative field policy. The `safe_harbor_candidate` label is a
# configured-field result, not a legal de-identification determination.
policy <- safeHarborPolicy(free_text = "drop", biometric_data = "drop")
head(policy[, c("category", "action")])

# The pseudonymized variant shifts dates instead of reducing them to a year.
head(pseudonymizedPolicy(free_text = "drop")[, c("category", "action")])
```

---

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| standardsSources | <i>Read the bundled standards and guidance source metadata</i> |
|------------------|----------------------------------------------------------------|

---

**Description**

Reads edition metadata recorded when this package was built. It performs no network request and does not determine whether a source remains applicable to a project.

**Usage**

```
standardsSources(max_age_days = 365L, as_of = Sys.Date())
```

**Arguments**

|              |                                                                            |
|--------------|----------------------------------------------------------------------------|
| max_age_days | Non-negative integer age after which source metadata is marked for review. |
| as_of        | Date used to calculate metadata age.                                       |

**Value**

A standards\_sources data frame.

**Examples**

```
# Reads edition metadata recorded at build time; performs no network request.
sources <- standardsSources(as_of = as.Date("2026-07-28"))
sources[, c("source_id", "edition", "review_status")]
```

---

|                    |                                                                                |
|--------------------|--------------------------------------------------------------------------------|
| traceabilityMatrix | <i>Construct a normalized requirement-risk-control-test traceability graph</i> |
|--------------------|--------------------------------------------------------------------------------|

---

**Description**

Construct a normalized requirement-risk-control-test traceability graph

**Usage**

```
traceabilityMatrix(
  requirements,
  risks,
  controls,
  tests,
  links,
  evidence_root = NULL,
  risk_matrix = NULL
)
```

**Arguments**

requirements, risks, controls, tests, links  
Plain data frames using the schemas documented in vignette("PhysioCompliance") and the package reference.

evidence\_root Optional evidence directory. Evidence URIs remain relative to this root.

risk\_matrix Optional project-owned `riskMatrix()`.

**Value**

A deterministic traceability\_matrix object.

---

validateTraceability    *Verify traceability structure, completeness, and objective evidence*

---

**Description**

Verify traceability structure, completeness, and objective evidence

**Usage**

```
validateTraceability(x, verify_evidence = TRUE)
```

**Arguments**

x                    A traceability\_matrix.

verify\_evidence    Whether to hash evidence files under the recorded evidence root.

**Value**

A traceability\_verification object.

---

verifyAuditTrail    *Verify a compliance audit trail*

---

**Description**

Verification reports tampering and malformed storage without modifying, truncating, repairing, or rehashing the supplied object.

**Usage**

```
verifyAuditTrail(x)
```

**Arguments**

`x` A `PhysioExperiment::PhysioExperiment` object.

**Details**

A self-contained object cannot distinguish an internally consistent earlier copy from the record as it existed at that time. Detect whole-object rollback or removal of the current tail event by retaining each verified head\_hash in a validated external append-only store and comparing it on retrieval.

**Value**

A `compliance_verification` object containing validity, deterministic issues, the verified head hash, current record hash, and event counts.

**Examples**

```
x <- PhysioExperiment::PhysioExperiment(
  assays = list(raw = matrix(1:6, nrow = 3)),
  samplingRate = 100
)
x <- initializeAuditTrail(x, actor = "operator-01")
verifyAuditTrail(x)
```

---

|                   |                                            |
|-------------------|--------------------------------------------|
| verifyESignatures | <i>Verify stored electronic signatures</i> |
|-------------------|--------------------------------------------|

---

**Description**

Reconstructs each canonical challenge and delegates cryptographic and credential verification to a caller-supplied callback. Provider errors are reported without storing or exposing provider diagnostics.

**Usage**

```
verifyESignatures(x, verify)
```

**Arguments**

`x` A `PhysioExperiment::PhysioExperiment` object.

`verify` Function called once for each structurally valid signature as `verify(challenge_raw, signature_raw, credential)`. It must return one non-missing logical value.

**Value**

A `compliance_verification` object containing audit and signature issues.

**Examples**

```

x <- PhysioExperiment::PhysioExperiment(
  assays = list(raw = matrix(1:6, nrow = 3)),
  samplingRate = 100
)
x <- initializeAuditTrail(x, actor = "operator-01")
credential <- list(id = "credential-01", algorithm = "demo-sha256",
  fingerprint = "01:23:45:67")
bytes <- function(challenge, credential) {
  digest::digest(c(challenge, serialize(credential, NULL, version = 3)),
    algo = "sha256", serialize = FALSE, raw = TRUE)
}
x <- eSign(x, signer = "reviewer-02", meaning = "reviewed",
  credential = credential, sign = bytes)
# The verifier recomputes the same bytes; a real one checks a cryptographic
# signature against the signer's credential.
verifyESignatures(x, verify = function(challenge, signature, credential) {
  identical(signature, bytes(challenge, credential))
})$valid

```

---

writeConformanceReport

*Write a deterministic conformance report*


---

**Description**

Write a deterministic conformance report

**Usage**

```

writeConformanceReport(
  x,
  path,
  format = c("json", "csv", "markdown"),
  overwrite = FALSE
)

```

**Arguments**

|           |                                                |
|-----------|------------------------------------------------|
| x         | A conformance_report.                          |
| path      | Destination path.                              |
| format    | JSON, CSV, or Markdown output.                 |
| overwrite | Whether to replace existing output atomically. |

**Value**

Normalized output path or paths, invisibly.

# Index

appendAuditEvent, [3](#)  
auditDeidentification, [4](#)  
auditTrail, [4](#)  
  
conformanceCheck, [5](#)  
conformanceCriteria, [6](#)  
conformanceCriteria(), [5](#)  
  
dataSubjectErase, [6](#)  
dataSubjectExport, [8](#)  
dateShift, [8](#)  
deidentify, [9](#)  
  
eSign, [10](#)  
  
headerScrub, [12](#)  
  
initializeAuditTrail, [13](#)  
  
lifecycleTemplate, [14](#)  
  
PhysioExperiment::PhysioExperiment, [3](#),  
[5](#), [11](#), [13](#), [22](#)  
pseudonymize, [15](#)  
pseudonymize(), [10](#)  
pseudonymizedPolicy (safeHarborPolicy),  
[19](#)  
pseudonymizedPolicy(), [10](#)  
  
reidentify, [16](#)  
riskManagementTemplate, [16](#)  
riskMatrix, [18](#)  
riskMatrix(), [17](#), [21](#)  
  
safeHarborPolicy, [19](#)  
safeHarborPolicy(), [10](#)  
standardsSources, [20](#)  
standardsSources(), [14](#), [17](#)  
  
traceabilityMatrix, [20](#)  
  
validateTraceability, [21](#)  
  
verifyAuditTrail, [21](#)  
verifyESignatures, [22](#)  
  
writeConformanceReport, [23](#)