Package: dicomtk.ai (via r-universe)

November 14, 2026

Title AI Agents, MCP Tooling, and Clinical NLP for 'DICOM' Imaging

Version 0.2.0

Description Satellite Artificial Intelligence ('AI') and clinical natural language processing toolkit for 'dicomtk'. Provides 'ellmer'-based clinical imaging agents with unbypassable monotonic de-identification guardrails, a Model Context Protocol ('MCP') server via 'mcptools', a pluggable REST and offline backend registry for segmentation models ('TotalSegmentator' by Wasserthal et al. (2023) <doi:10.1148/ryai.230024>, 'nnU-Net', 'VISTA3D', and 'MedSAM2' by Ma et al. (2024) <doi:10.1038/s41467-024-44824-z>), Integrating the Healthcare Enterprise AI Results ('IHE AIR') standardized 'DICOM SEG' and 'TID 1500' outputs, radiology report de-identification and weak-label extraction, 'RadLex' and 'LOINC-RSNA' protocol harmonisation, and 'DuckDB' vector similarity search.

License GPL (>= 2)

URL https://lauyeehow1986-hub.github.io/dicomtk.ai/, https://github.com/lauyeehow1986-hub/dicomtk.ai

BugReports https://github.com/lauyeehow1986-hub/dicomtk.ai/issues

Depends R (>= 4.1.0)

Imports dicomtk (>= 0.4.0), ellmer (>= 0.4.0), mcptools (>= 0.2.0), duckdb (>= 1.0.0), DBI (>= 1.2.0), httr2 (>= 1.0.0), curl, jsonlite (>= 1.8.0), tibble (>= 3.0.0), cli (>= 3.6.0), rlang (>= 1.1.0), stats, utils

Suggests testthat (>= 3.0.0), tesseract, withr, knitr, rmarkdown

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs make libicu-dev libssl-dev zlib1g-dev

Repository https://lauyeehow1986-hub.r-universe.dev

Date/Publication 2026-11-14 15:26:17 UTC

RemoteUrl https://github.com/lauyeehow1986-hub/dicomtk.ai

RemoteRef main

RemoteSha bee54d8893066e26c656c28584e008778b96d055

Contents

ai_agent_approve . . . 3
ai_agent_dry_run . . . 3
ai_agent_execute . . . 4
ai_agent_plan . . . 4
ai_agent_verify . . . 5
ai_backend_health . . . 6
ai_chat . . . 6
ai_combat_harmonize . . . 7
ai_deid_report . . . 8
ai_embed . . . 8
ai_embedding_index . . . 9
ai_export_results . . . 9
ai_extract_labels . . . 10
ai_extract_radiomics . . . 11
ai_generate_maisi . . . 12
ai_harmonize_protocol . . . 13
ai_index_add . . . 13
ai_index_close . . . 14
ai_search_similar . . . 14
ai_segment . . . 15
as_ellmer_chat . . . 16
combat_apply . . . 16
combat_fit . . . 17
dicomtk_mcp_tools . . . 17
enforce_non_weakening_profile . . . 18
get_ai_backend . . . 19
ibsi_digital_phantom . . . 19
list_ai_backends . . . 20
radlex_playbook . . . 20
register_ai_backend . . . 21
start_mcp_server . . . 22

Index **23**

ai_agent_approve *Approve a Dry-Run Clinical Imaging Agent Plan*

Description

Stage 3 of the agent loop. Requires explicit approval before any mutation or de-identification output is written.

Usage

```
ai_agent_approve(dry_run, auto_approve = FALSE, approver = NULL)
```

Arguments

dry_run	A "dicom_agent_plan" object at stage = "dry_run".
auto_approve	Logical. Set TRUE for non-interactive batch pipelines.
approver	Optional character name/identifier of the approving user.

Value

Updated "dicom_agent_plan" object at stage = "approved".

ai_agent_dry_run *Dry-Run a Planned Clinical Imaging Agent Workflow*

Description

Stage 2 of the agent loop. Evaluates effective tag actions and optional image QC without writing or mutating any files on disk.

Usage

```
ai_agent_dry_run(plan)
```

Arguments

plan	A "dicom_agent_plan" object from ai_agent_plan().

Value

Updated "dicom_agent_plan" object at stage = "dry_run".

ai_agent_execute *Execute an Approved Clinical Imaging Agent Plan*

Description

Stage 4 of the agent loop. Re-verifies the non-weakening guardrail and executes de-identification and/or QC into `out_dir`.

Usage

```
ai_agent_execute(approved_plan, out_dir)
```

Arguments

`approved_plan`	An approved `"dicom_agent_plan"` object.
`out_dir`	Output directory for de-identified files and provenance logs.

Value

Updated `"dicom_agent_plan"` object at `stage = "executed"`.

ai_agent_plan *Plan a Guardrailed Clinical Imaging Agent Workflow*

Description

Stage 1 of the 5-stage clinical agent loop (`plan -> dry-run -> approve -> execute -> verify`). Enforces the non-weakening de-identification guardrail up front.

Usage

```
ai_agent_plan(
  task = c("deid_and_qc", "deid_only", "qc_only"),
  path,
  baseline_profile = dicomtk::dicom_deid_profile(),
  override_tags = list(),
  options = character(0),
  chat = ai_chat("offline")
)
```

Arguments

task	Workflow task: "deid_and_qc", "deid_only", or "qc_only".
path	Input DICOM file or directory path.
baseline_profile	Baseline dicomtk::dicom_deid_profile() object.
override_tags	Optional named list of tag-action overrides (must not weaken baseline_profile).
options	Optional character vector of PS3.15 options.
chat	A "dicom_ai_chat" object from ai_chat().

Value

An S3 "dicom_agent_plan" object at stage = "planned".

Examples

```
synth <- dicomtk::dicom_synthesize(modality = "CT", dims = c(8L, 8L, 2L))
on.exit(unlink(synth$out_dir, recursive = TRUE), add = TRUE)
plan <- ai_agent_plan("deid_only", path = synth$out_dir)
dry <- ai_agent_dry_run(plan)
```

---

ai_agent_verify *Verify Executed Clinical Imaging Agent Outputs*

Description

Stage 5 of the agent loop. Runs dicomtk::dicom_deid_verify() on the executed output directory to confirm zero residual PHI leakage.

Usage

```
ai_agent_verify(exec_result)
```

Arguments

exec_result	A "dicom_agent_plan" object at stage = "executed".

Value

Updated "dicom_agent_plan" object at stage = "verified" with $passed logical status.

ai_backend_health *Check Health of a Registered AI Backend*

Description

Probes a registered backend's readiness ("offline_ready" for offline backends, or HTTP GET /health / base URL for REST backends).

Usage

```
ai_backend_health(name, timeout_sec = 3)
```

Arguments

name	Backend name.
timeout_sec	HTTP timeout in seconds for REST backends (default 3).

Value

A list with name, mode, status, and endpoint.

ai_chat *Configure an* ellmer*-Backed Chat Session for* dicomtk.ai

Description

Creates a "dicom_ai_chat" object wrapping an ellmer chat client (or a deterministic "offline" rule-based planner for CI and air-gapped testing).

Usage

```
ai_chat(
 provider = c("offline", "ollama", "openai", "anthropic", "gemini", "openai_compatible"),
  model = NULL,
  base_url = NULL,
  api_key = NULL,
  system_prompt = NULL,
  ...
)
```

Arguments

provider	Chat provider: "offline", "ollama", "openai", "anthropic", "gemini", or "openai_compatible".
model	Optional model identifier.
base_url	Optional API base URL (used for "ollama" and "openai_compatible").
api_key	Optional API key (defaults to standard provider environment variables).
system_prompt	Optional system prompt string.
...	Additional arguments passed to the underlying ellmer chat constructor.

Value

An object of class "dicom_ai_chat".

Examples

```
chat <- ai_chat("offline")
```

---

ai_combat_harmonize *Fit and Apply Empirical Bayes ComBat Harmonisation*

Description

Fit and Apply Empirical Bayes ComBat Harmonisation

Usage

```
ai_combat_harmonize(features, batch, mod = NULL, eb = TRUE, ref_batch = NULL)
```

Arguments

features	Numeric matrix (n_features x n_samples).
batch	Factor or character vector of scanner/site batch labels of length ncol(features).
mod	Optional biological covariate design matrix (n_samples x q, e.g. from model.matrix(~ age + sex)).
eb	Logical; if TRUE (default), applies Empirical Bayes shrinkage across features.
ref_batch	Optional reference batch name.

Value

A list with harmonized (harmonized matrix) and fit ("dicom_combat_fit").

Examples

```
set.seed(1)
mat <- matrix(rnorm(40), nrow = 4, ncol = 10)
b <- rep(c("A", "B"), each = 5)
res <- ai_combat_harmonize(mat, b)
dim(res$harmonized)
```

ai_deid_report *De-Identify Free-Text Radiology Reports or Structured Reports*

Description

Redacts or pseudonymizes protected health information (PHI) in free-text radiology reports or `dicom_sr` objects using a registered NLP backend (`"presidio"` REST or built-in deterministic HIPAA/GDPR/SG pattern NER). Enforces a monotonic union invariant: optional LLM audit passes can only add missed spans, never remove deterministic detections.

Usage

```
ai_deid_report(
  x,
  backend = "presidio",
  chat = NULL,
  mode = c("redact", "pseudonymize")
)
```

Arguments

`x`	Character report text or a `dicom_sr` object from `dicomtk::read_dicom_sr()`.
`backend`	Registered NLP de-id backend name (default `"presidio"`).
`chat`	Optional `dicom_ai_chat` object for contextual review.
`mode`	`"redact"` (default) or `"pseudonymize"`.

Value

A list with `text` (de-identified string), `spans` (tibble of detected PHI spans), and `mode`.

Examples

```
ai_deid_report("Patient John Tan (MRN: MRN-12345) scanned on 2026-03-14.")
```

ai_embed *Extract a Normalized Study Embedding Vector*

Description

Computes an L2-normalized feature embedding from a `dicom_volume` using a registered embedding backend (`"offline"` deterministic intensity/spatial/modality signature, or a REST foundation-model endpoint such as RAD-DINO / BiomedCLIP).

Usage

```
ai_embed(vol, backend = "embeddings", dim_out = 32L)
```

Arguments

vol	A dicom_volume object or DICOM path.
backend	Registered embedding backend name (default "embeddings").
dim_out	Target embedding dimension for the offline extractor (default 32L).

Value

A numeric L2-normalized embedding vector.

---

ai_embedding_index *Open or Initialize a DuckDB Study Embedding Index*

Description

Open or Initialize a DuckDB Study Embedding Index

Usage

```
ai_embedding_index(db_path = ":memory:")
```

Arguments

db_path	Path to a DuckDB database file (or ":memory:").

Value

A "dicom_ai_index" object.

---

ai_export_results *Export AI Segmentation Results in IHE AIR Standardized DICOM Format*

Description

Writes segmentation masks as a DICOM SEG object (write_dicom_seg()) and quantitative summary metrics as a DICOM TID 1500 Measurement Report (dicom_sr_tid1500()), aligned with IHE AI Results (AIR) conventions.

Usage

```
ai_export_results(
  seg_res,
  vol,
  out_dir,
  title = "IHE AIR AI Segmentation Report"
)
```

Arguments

seg_res	A "dicom_ai_segmentation" object from ai_segment(), or a dicom_seg object.
vol	Reference dicom_volume object.
out_dir	Output directory for standardized DICOM files.
title	Report title for the TID 1500 Measurement Report.

Value

Invisibly returns a list with seg_path, sr_path, out_dir, and metrics.

---

ai_extract_labels *Extract Weak Clinical Labels From Radiology Report Text*

Description

Extracts structured pathology findings with assertion polarity ("present", "absent", or "uncertain") using deterministic ConText negation/uncertainty rules (when "offline") or an ellmer structured-output chat session.

Usage

```
ai_extract_labels(text, findings = NULL, chat = ai_chat("offline"))
```

Arguments

text	Character vector of report text.
findings	Optional character vector of extra finding terms to track.
chat	A dicom_ai_chat object from ai_chat().

Value

A tibble::tibble with columns finding, anatomy, polarity, confidence, and sentence.

Examples

```
ai_extract_labels("No evidence of pneumothorax. Moderate pleural effusion.")
```

`ai_extract_radiomics` *Extract IBSI-Standardized 3D Radiomics Features*

Description

Computes morphology, first-order intensity, 3D GLCM, 3D GLRLM, and 3D GLSZM features compliant with the Image Biomarker Standardisation Initiative (IBSI) Phase-1 specification.

Usage

```
ai_extract_radiomics(
  volume,
  mask = NULL,
  discretization = c("none", "fbn", "fbw"),
  n_bins = 16L,
  bin_width = 25,
  backend = c("native", "pyradiomics")
)
```

Arguments

`volume`	A 3D numeric array or a `dicom_volume` / list with `$voxels` and `$spacing`.
`mask`	Optional 3D binary/integer mask array (same dimensions as `volume`). If `NULL`, all non-NA voxels are included.
`discretization`	Intensity discretization: `"none"` (for integer phantoms), `"fbn"` (fixed bin number), or `"fbw"` (fixed bin width).
`n_bins`	Number of bins when `discretization = "fbn"` (default `16L`).
`bin_width`	Bin width when `discretization = "fbw"` (default `25`).
`backend`	Either `"native"` (pure R IBSI engine) or `"pyradiomics"`.

Value

A `tibble::tibble` with columns `family`, `feature`, and `value`.

Examples

```
ph <- ibsi_digital_phantom()
rad <- ai_extract_radiomics(ph$volume, ph$mask, discretization = "none")
head(rad)
```

`ai_generate_maisi` *Generate MONAI MAISI 3D Synthetic CT and Multi-Organ Label Masks*

Description

Generates paired 3D synthetic CT DICOM slices (`ImageType = "DERIVED\\SECONDARY\\SYNTHETIC"`) and multi-organ integer segmentation masks along with a provenance manifest (`maisi_manifest.json`). Supports local deterministic `"phantom"` synthesis or an external MONAI MAISI `"http"` inference endpoint.

Usage

```
ai_generate_maisi(
  out_dir = tempfile("maisi_"),
  anatomy_list = c("liver", "spleen", "left_kidney", "right_kidney"),
  dims = c(32L, 32L, 16L),
  spacing = c(1.5, 1.5, 2),
  mode = c("phantom", "http"),
  endpoint = NULL
)
```

Arguments

`out_dir`	Output directory for synthetic DICOM series and manifest.
`anatomy_list`	Character vector of target organ names (e.g. `c("liver", "spleen", "left_kidney", "right_kidney")`).
`dims`	Integer vector `c(rows, cols, slices)` (default `c(32L, 32L, 16L)`).
`spacing`	Numeric vector `c(dx, dy, dz)` in mm.
`mode`	Synthesis mode: `"phantom"` (pure-R offline) or `"http"` (MAISI REST service).
`endpoint`	Optional MAISI HTTP endpoint URL when `mode = "http"`.

Value

An S3 `"dicom_maisi_result"` object with `dicom_dir`, `mask`, `anatomy_map`, `manifest_path`, and `synthetic = TRUE`.

Examples

```
out <- tempfile("maisi_ex_")
res <- ai_generate_maisi(out, dims = c(16L, 16L, 4L))
dim(res$mask)
unlink(out, recursive = TRUE)
```

ai_harmonize_protocol *Harmonise Messy DICOM Series Descriptions to RadLex and LOINC-RSNA Playbook*

Description

Maps heterogeneous `SeriesDescription`, `Modality`, and `BodyPartExamined` headers onto standardized LOINC-RSNA Radiology Playbook IDs (`rpid`) and `RadLex` anatomical identifiers (`radlex_id`).

Usage

```
ai_harmonize_protocol(headers, chat = NULL)
```

Arguments

`headers`	A data frame / tibble with `SeriesDescription` (and optional `Modality`, `BodyPartExamined` columns).
`chat`	Optional `dicom_ai_chat` object for semantic tie-breaking on low-confidence headers.

Value

A tibble::tibble with columns `SeriesDescription`, `Modality`, `rpid`, `long_common_name`, `short_name`, `body_region`, `radlex_id`, `contrast`, and `confidence`.

Examples

```
df <- data.frame(SeriesDescription = "CHEST WO CON 3.0mm", Modality = "CT")
ai_harmonize_protocol(df)
```

ai_index_add *Add a Study Volume to a DuckDB Embedding Index*

Description

Add a Study Volume to a DuckDB Embedding Index

Usage

```
ai_index_add(index, study_id, vol, metadata = list(), backend = "embeddings")
```

Arguments

index	A "dicom_ai_index" from ai_embedding_index().
study_id	Unique study identifier string.
vol	A dicom_volume object.
metadata	Optional named list of study metadata.
backend	Registered embedding backend name (default "embeddings").

Value

Invisibly returns study_id.

---

ai_index_close *Close a DuckDB Study Embedding Index*

Description

Close a DuckDB Study Embedding Index

Usage

```
ai_index_close(index)
```

Arguments

index	A "dicom_ai_index" object.

Value

Invisibly returns TRUE.

---

ai_search_similar *Retrieve Top-K Similar Studies From a DuckDB Embedding Index*

Description

Uses DuckDB vss cosine similarity when installed, falling back seamlessly to vectorized cosine scoring for air-gapped environments.

Usage

```
ai_search_similar(index, query_vol, k = 5L, backend = "embeddings")
```

Arguments

index	A "dicom_ai_index" object.
query_vol	Query dicom_volume object.
k	Integer number of nearest neighbors to return (default 5L).
backend	Registered embedding backend name (default "embeddings").

Value

A tibble::tibble with study_id, modality, similarity, and metadata_json, sorted by descending cosine similarity.

---

ai_segment *Run Pluggable AI Segmentation With Uncertainty Estimation*

Description

Runs segmentation via a registered backend ("totalsegmentator", "nnunet", "vista3d", "medsam2", or custom registered backends), aligns the mask to the reference dicom_volume, and optionally computes voxel-wise Shannon entropy uncertainty and Dice agreement.

Usage

```
ai_segment(
  vol,
  backend = "totalsegmentator",
  prompts = NULL,
  estimate_uncertainty = TRUE,
  reference_seg = NULL
)
```

Arguments

vol	A dicom_volume object or path to a DICOM file/directory.
backend	Registered segmentation backend name.
prompts	Optional prompt list for interactive models such as "medsam2" (e.g., list(points = matrix(c(r, c, s), ncol = 3))).
estimate_uncertainty	Logical. Compute voxel-wise entropy map (TRUE by default).
reference_seg	Optional reference dicom_seg for Dice evaluation.

Value

A list of class "dicom_ai_segmentation" with seg (dicom_seg), uncertainty (dicom_volume or NULL), probs, metrics, and backend.

as_ellmer_chat *Convert a Legacy* dicomtk::dicom_llm *Object to* dicom_ai_chat

Description

Convert a Legacy dicomtk::dicom_llm Object to dicom_ai_chat

Usage

```
as_ellmer_chat(x)
```

Arguments

x	A dicom_llm, dicom_ai_chat, or ellmer Chat object.

Value

A "dicom_ai_chat" object.

combat_apply *Apply Fitted ComBat Harmonisation Parameters Without Test-Set Leakage*

Description

Apply Fitted ComBat Harmonisation Parameters Without Test-Set Leakage

Usage

```
combat_apply(fit, features, batch, mod = NULL)
```

Arguments

fit	A "dicom_combat_fit" from combat_fit().
features	Numeric matrix (n_features x n_samples).
batch	Factor or character vector of batch labels matching fit$batch_levels.
mod	Optional biological covariate matrix (n_samples x q).

Value

Harmonized numeric matrix of dimensions dim(features).

combat_fit — *Fit Train/Test-Safe Empirical Bayes ComBat Harmonisation Parameters*

Description

Estimates batch location and scale parameters on a training cohort while preserving biological covariates (mod). Apply the resulting fit to training or held-out test cohorts with combat_apply() without test-set leakage.

Usage

```
combat_fit(features, batch, mod = NULL, eb = TRUE, ref_batch = NULL)
```

Arguments

features	Numeric matrix (n_features x n_samples).
batch	Factor or character vector of scanner/site batch labels of length ncol(features).
mod	Optional biological covariate design matrix (n_samples x q, e.g. from model.matrix(~ age + sex)).
eb	Logical; if TRUE (default), applies Empirical Bayes shrinkage across features.
ref_batch	Optional reference batch name.

Value

An S3 "dicom_combat_fit" object.

dicomtk_mcp_tools — *Build* ellmer *Tool Definitions for the* dicomtk.ai *MCP Server*

Description

Constructs a named list of ellmer::tool() definitions exposing header inspection, automated QC, guardrailed de-identification dry-runs, residual-PHI verification, protocol harmonisation, and AI segmentation.

Usage

```
dicomtk_mcp_tools()
```

Value

A named list of ellmer ToolDef objects.

Examples

```
tools <- dicomtk_mcp_tools()
names(tools)
```

`enforce_non_weakening_profile` *Enforce Non-Weakening Monotonic De-Identification Guardrail*

Description

Validates or clamps any agent-proposed overrides, options, or policy flags against a baseline `dicomtk::dicom_deid_profile` along the strict PS3.15 join-semilattice `K < C < U < D < Z < X`.

Usage

```
enforce_non_weakening_profile(
  baseline,
  override_tags = list(),
  options = character(0),
  jurisdictions = NULL,
  deidentify = NULL,
  patient_deid = NULL,
  mode = c("reject", "clamp")
)
```

Arguments

`baseline`	A baseline `dicom_deid_profile` object.
`override_tags`	Named list of tag-action overrides.
`options`	Character vector of additional PS3.15 options.
`jurisdictions`	Optional character vector of additional jurisdictions.
`deidentify`	Optional logical flag override.
`patient_deid`	Optional logical flag override.
`mode`	`"reject"` (default; throws an error on any weakening attempt) or `"clamp"` (clamps every tag and flag to at least baseline strength).

Value

A validated (or monotonically clamped) `dicom_deid_profile` object.

Examples

```
base <- dicomtk::dicom_deid_profile("ps3.15")
stronger <- enforce_non_weakening_profile(base, override_tags = list("(0008,1030)" = "X"))
```

get_ai_backend *Retrieve a Registered AI Backend*

Description

Retrieve a Registered AI Backend

Usage

```
get_ai_backend(name)
```

Arguments

name	Backend name (case-insensitive).

Value

A dicom_ai_backend object.

ibsi_digital_phantom *Construct the Canonical IBSI Phase-1 Digital Phantom*

Description

Returns the 4 x 5 x 4 (74-voxel ROI, 2 x 2 x 2 mm voxel spacing) IBSI Phase-1 digital phantom for standardized radiomics verification.

Usage

```
ibsi_digital_phantom()
```

Value

A list with voxels (4x5x4 array), mask (4x5x4 integer array), spacing (c(2, 2, 2)), and volume (list(voxels = ..., spacing = ...)).

Examples

```
ph <- ibsi_digital_phantom()
sum(ph$mask)
```

list_ai_backends *List Registered AI Backends*

Description

List Registered AI Backends

Usage

```
list_ai_backends(task = NULL)
```

Arguments

task	Optional task filter ("segmentation", "nlp_deid", "ocr", or "embedding").

Value

A tibble::tibble summarizing registered backends.

Examples

```
list_ai_backends()
```

radlex_playbook *Curated LOINC-RSNA Radiology Playbook and RadLex Protocol Table*

Description

A compact reference dataset of common diagnostic imaging protocols across CT, MR, PT, NM, DX, MG, US, and XA modalities mapped to LOINC-RSNA Radiology Playbook IDs (rpid) and RadLex anatomical region identifiers (radlex_id).

Usage

```
radlex_playbook
```

Format

A tibble with 46 rows and 8 variables:

rpid LOINC-RSNA Radiology Playbook identifier.

long_common_name Standardized long protocol name.

short_name Abbreviated display name.

modality DICOM Modality code.

body_region Normalized anatomical region.

radlex_id RadLex anatomical identifier (`RID...`).

contrast Logical indicating IV/radiotracer contrast usage.

keywords Space-delimited matching keywords.

Source

Curated subset of LOINC-RSNA Radiology Playbook and RadLex ontology.

`register_ai_backend` *Register a Pluggable AI Backend*

Description

Registers or updates an AI model backend (segmentation, NLP de-identification, OCR, or embedding extraction) in the session registry.

Usage

```
register_ai_backend(
  name,
  task = c("segmentation", "nlp_deid", "ocr", "embedding"),
  mode = c("offline", "rest"),
  endpoint = NULL,
  predict_fn = NULL,
  headers = character(0)
)
```

Arguments

`name`	Unique backend identifier (e.g., `"totalsegmentator"`, `"medsam2"`, `"presidio"`).
`task`	Task category: `"segmentation"`, `"nlp_deid"`, `"ocr"`, or `"embedding"`.
`mode`	Execution mode: `"offline"` (deterministic local engine) or `"rest"` (HTTP endpoint via `httr2`).
`endpoint`	Base URL or local URI for the backend.
`predict_fn`	Optional custom `function(input, backend, ...)` override.
`headers`	Optional named character vector of HTTP headers.

Value

Invisibly returns the registered `dicom_ai_backend` object.

start_mcp_server *Start the* dicomtk.ai *Model Context Protocol (MCP) Server*

Description

Launches an MCP server via mcptools::mcp_server() exposing dicomtk_mcp_tools() over "stdio" or "http".

Usage

```
start_mcp_server(
  type = c("stdio", "http"),
  host = "127.0.0.1",
  port = 8080L,
  tools = dicomtk_mcp_tools()
)
```

Arguments

type	Transport type: "stdio" or "http".
host	Host interface for HTTP transport (default "127.0.0.1").
port	Port for HTTP transport (default 8080L).
tools	List of ellmer::ToolDef objects (default dicomtk_mcp_tools()).

Value

Called for its side effect (runs the MCP server loop).

Index

∗ **datasets**
    radlex_playbook, 20

ai_agent_approve, 3
ai_agent_dry_run, 3
ai_agent_execute, 4
ai_agent_plan, 4
ai_agent_plan(), *3*
ai_agent_verify, 5
ai_backend_health, 6
ai_chat, 6
ai_chat(), *5*, *10*
ai_combat_harmonize, 7
ai_deid_report, 8
ai_embed, 8
ai_embedding_index, 9
ai_embedding_index(), *14*
ai_export_results, 9
ai_extract_labels, 10
ai_extract_radiomics, 11
ai_generate_maisi, 12
ai_harmonize_protocol, 13
ai_index_add, 13
ai_index_close, 14
ai_search_similar, 14
ai_segment, 15
ai_segment(), *10*
as_ellmer_chat, 16

combat_apply, 16
combat_apply(), *17*
combat_fit, 17
combat_fit(), *16*

dicomtk::dicom_deid_profile(), *5*, *18*
dicomtk::dicom_deid_verify(), *5*
dicomtk_mcp_tools, 17
dicomtk_mcp_tools(), *22*

enforce_non_weakening_profile, 18

get_ai_backend, 19

ibsi_digital_phantom, 19

list_ai_backends, 20

mcptools::mcp_server(), *22*

radlex_playbook, 20
register_ai_backend, 21

start_mcp_server, 22

tibble::tibble, *10*, *13*, *15*, *20*