CIM Assessment Toolkit (CAT) v2.10.0

Machine Data Insights Inc.

Measure and validate CIM (Common Information Model) compliance across Splunk® data models at the data model + dataset level. Provides CIM compliance level, field mapping quality, prescribed value validation, acceleration health monitoring, data model testing, automated report generation, and scheduled email distribution.

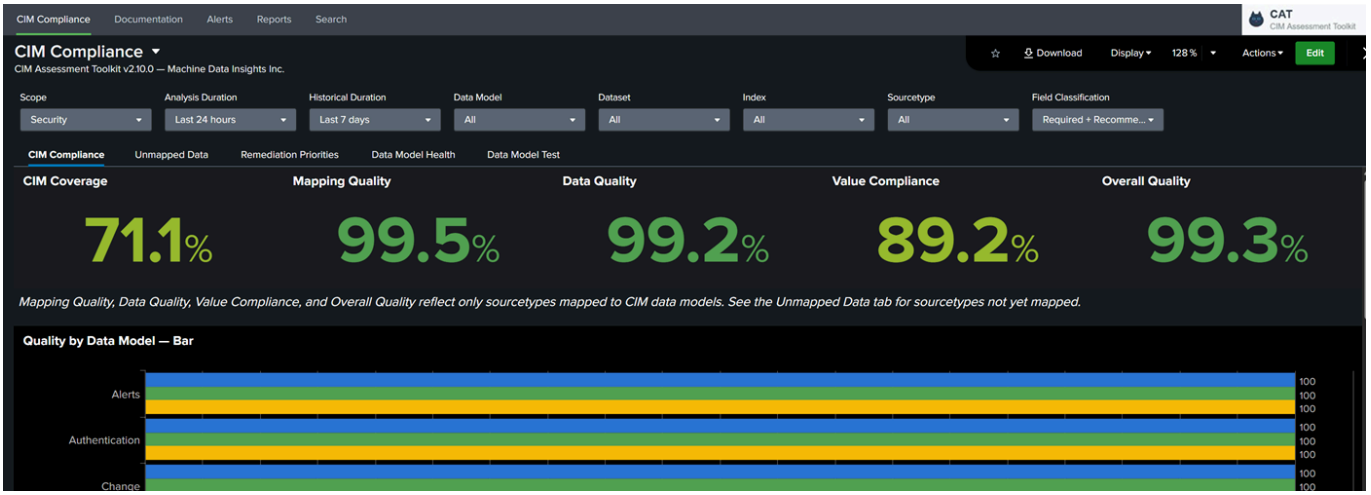


Table of Contents

CIM Assessment Toolkit Installation & Configuration

Prerequisites:

1. The latest Splunk Common Information Model app (https://splunkbase.splunk.com/app/1621) should be installed on the Splunk Enterprise Security, IT Service Intelligence, or other Splunk server upon which data model acceleration will be configured and data models used.

2. The applicable indexes and sourcetypes for data sources to be included in each data model should be configured in the CIM macros: Settings → Advanced search → select 'Splunk Common Information Model (Splunk_SA_CIM)' in the App dropdown → enter `cim_*_indexes` in the search filter and click the hourglass icon. Then click and edit the macro contents for each applicable data model and 'Save'.

3. Confirm or enable data model acceleration: Settings → Data Models → enable acceleration for CIM models relevant to your environment and ensure a recent acceleration search has completed for each applicable data model.

4. Create a new Splunk summary (events) index (recommended: cat_compliance) or identify an existing index to collect CIM validation data in. A decision factor is the data retention timeframe - you should keep CAT data for as long as you want to track historical CIM compliance levels (recommend 1 year minimum).

Installation and Configuration

Note: If your environment utilizes a Splunk Enterprise Security (ES) or IT Service Intelligence (ITSI) server it is highly recommended to install CAT on that server (or the server from which data model acceleration is run) to allow accurate data model acceleration health measurements.

1. Install the CIM Assessment Toolkit app: In Splunk Web, navigate to Apps → Manage Apps → Install App from File. Click "Browse" and select the `CIM_Assessment_Toolkit_xxxx.spl` file. Check "Upgrade app" if applicable, then click Upload. You can also navigate to Apps → Manage Apps → Browse More Apps → Type CIM in the search field → find "CIM Assessment Toolkit" → Install to install from Splunkbase. Alternatively, copy an unpacked `CIM_Assessment_Toolkit` folder to `$SPLUNK_HOME/etc/apps/` and restart Splunk.

   The first time CAT is started you will be presented with a "CIM Assessment Toolkit Setup" view. If your Splunk server has email configured you can enter the SMTP password (which gets encrypted in Splunk's credential vault) to allow CAT to send periodic CIM assessment reports via email - or you can leave this blank for now and just click 'Save and continue'.

2. Open the CAT dashboard: Apps → CIM Assessment Toolkit → CIM Compliance. The CIM metrics and views will not populate until an initial CIM compliance data collection sequence has run.

3. In another Splunk Web tab, navigate to Settings → Advanced search → Search macros and select 'CIM Assessment Toolkit' from the App drop-down. Configure the following CAT macros:

   - cim_validator_index - Set this to reflect the Splunk index that the CIM validation collection data will be sent to
   - cim_validator_data_model_list - Edit the settings to remove any data models you don't want to collect data for and monitor
   - cim_validator_base_search - DO NOT CHANGE THIS

4. Initiate an initial CIM validation collection search: In the CAT app, select the `Reports` tab. For the `cim_validator_collection` report click 'Open in search'. If you get a "We've identified a potential security risk" warning click "Run Query Anyway" and allow the search to complete. Depending on the size of your environment and the number of data models and sourcetypes to be collected, this search may take anywhere from 15 to 75 minutes. This search populates the summary index with field-level CIM compliance metrics.

5. By default, the `cim_validator_collection` report is scheduled to run each morning at 1:17 AM to allow the collection search to run when the number of ad-hoc and other scheduled searches are expected to be minimal. You can change this as desired to suit your environment by selecting Edit → Edit schedule.

6. When the `cim_validator_collection` search has completed, execute the `cim_validator_coverage_collection` report in like fashion. This collects metrics reflecting the overall level of CIM compliance.

7. By default the `cim_validator_coverage_collection` report is scheduled to run each morning at 4:17 AM. You can change this as desired to suit your environment.

8. When the `cim_validator_coverage_collection` search has completed the CIM Assessment Toolkit dashboards and panels should all populate properly. See Using the CIM Assessment Toolkit and Dashboard Reference sections of this document for guidance.

Top

Using the CIM Assessment Toolkit

What CAT Measures

CAT evaluates three dimensions of CIM compliance for every required and recommended field in each data model:

Mapping Quality — What percentage of required and recommended fields have at least some data? A field with even one event containing that field counts as mapped. 100% means all expected fields exist in your data.

Data Quality — Of the fields that are mapped, how consistently are they populated? A field that appears in 80% of events gets an 80% data quality score. Low data quality flags fields that exist but are only sporadically populated.

Value Compliance — For fields with CIM-prescribed values (like Authentication.action expecting "success", "failure", "error", "pending"), what percentage of events use the correct values? This catches fields that are mapped but contain non-standard values that ES correlation searches won't match.

Overall Quality — Combined score of Mapping Quality and Data Quality, providing a single metric for executive reporting.

Quick-Reference Workflow

Daily monitoring:

- Open the CIM Compliance dashboard -- the five KPI scorecards give you an instant CIM compliance level and health check
- Investigate any scores below 90% by drilling into the Compliance Details and Field-Level Detail sections
- Check the Data Model Health tab if scores are unexpectedly low (acceleration issues cause missing data)

After adding new data sources or TAs:

1. Re-run `cim_validator_collection` from Settings → Searches, reports, and alerts
2. Check the Unmapped Data tab for any new sourcetypes not yet mapped to CIM
3. Review Field-Level Gaps in the report for specific fields that need extraction configuration

Weekly review:

- Use the Compliance Trends graphs to track improvement over time
- Review the Remediation Priorities tab to focus effort where it has the most impact (low quality + high volume = highest priority)
- Generate and distribute the CIM Assessment Report (see Report Generation below)

Interpreting Dashboard Scores

Score	Rating	Action
90–100%	Excellent	Maintain. Monitor for regression with weekly reports.
70–89%	Good	Review missing fields. Prioritize high-volume sourcetypes.
50–69%	Fair	Significant gaps. Focus on required and recommended fields first, then optional.
Below 50%	Needs Improvement	Major CIM compliance issues. May impact ES detection coverage.

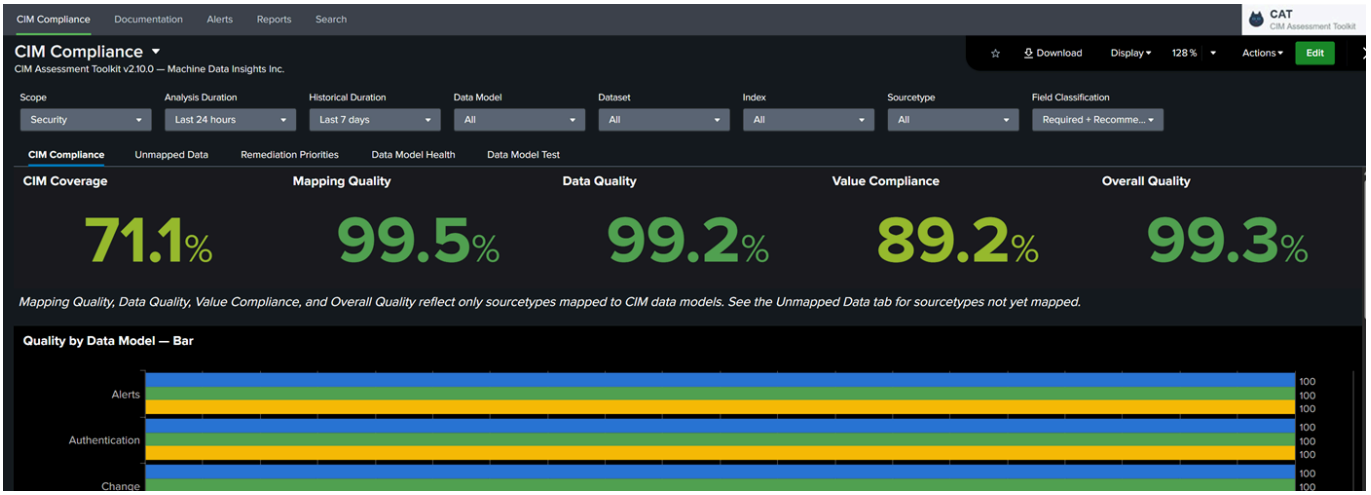
Tips for Improving CIM Compliance

- **Start with required and recommended fields** — these are used by Splunk ES correlation searches. Missing required and recommended fields directly impact detection capability.
- **Focus on high-volume sourcetypes first** — the Remediation Priorities tab ranks by impact (low quality × high volume = highest priority).
- **Check the Required Tags column** — this tells you exactly which tags need to be applied in your TA's `tags.conf` for the data to appear in the data model.
- **Use the Field-Level Gaps section** — it shows exactly which fields are unmapped (count = 0) or have non-compliant values, making TA remediation specific and actionable.
- **Don't ignore Data Quality** — a field that appears in only 10% of events may indicate a field extraction that only works for some event formats within a sourcetype.
- **Re-run collection after TA changes** — after updating field extractions or tags, run `cim_validator_collection` immediately to verify the improvement.

Top

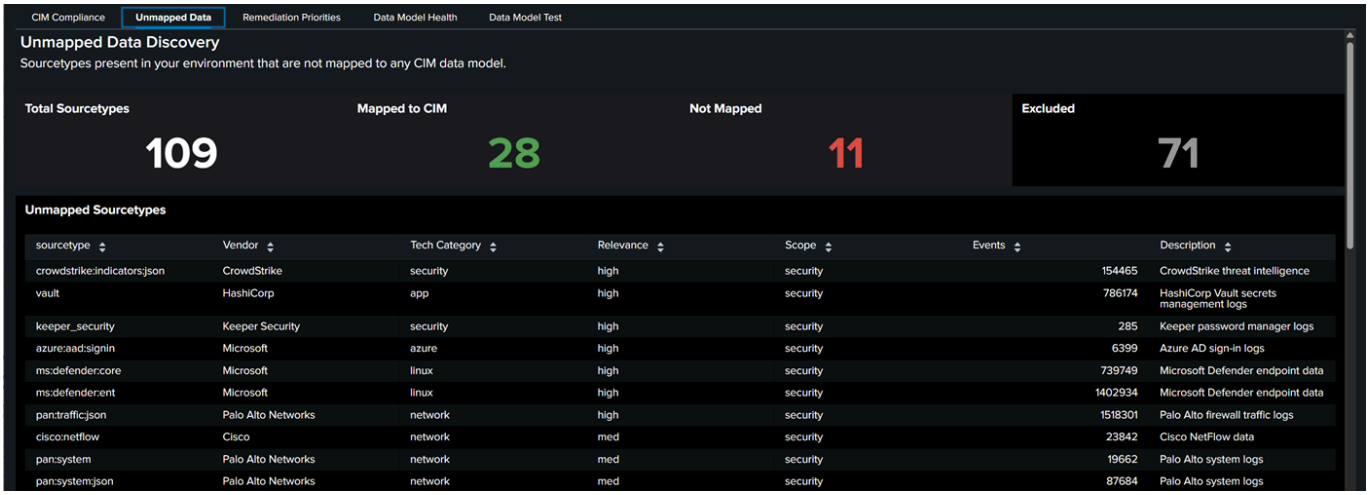
Dashboard Reference

CIM Compliance Tab



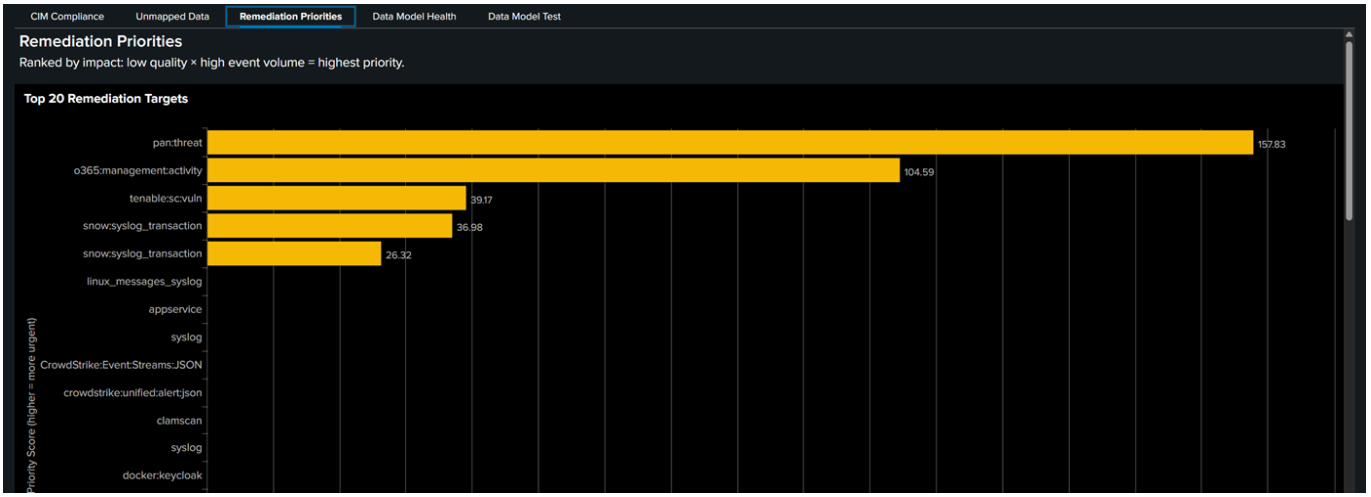
- **KPI Scorecards** — Five color-coded percentage scores: CIM Coverage (live tstats -- what fraction of sourcetypes are mapped), Mapping Quality, Data Quality, Value Compliance, Overall Quality. CIM Coverage uses live index data; the other four use summary index data from the collection search.
- **Quality by Data Model (bar chart)** — Visual comparison across all data models
- **Compliance Trends** — Three line charts showing Mapping, Data Quality, and Overall trends over time
- **Compliance Details** — Summary table with per-model/dataset quality scores
- **Compliance by Data Source** — Per index/sourcetype breakdown showing exactly which data sources have gaps
- **Field-Level Detail** — Expandable table of every field with count, distinct count, coverage %, and value compliance. Use the Field Classification dropdown to filter by required, recommended, or optional.

Unmapped Data Tab



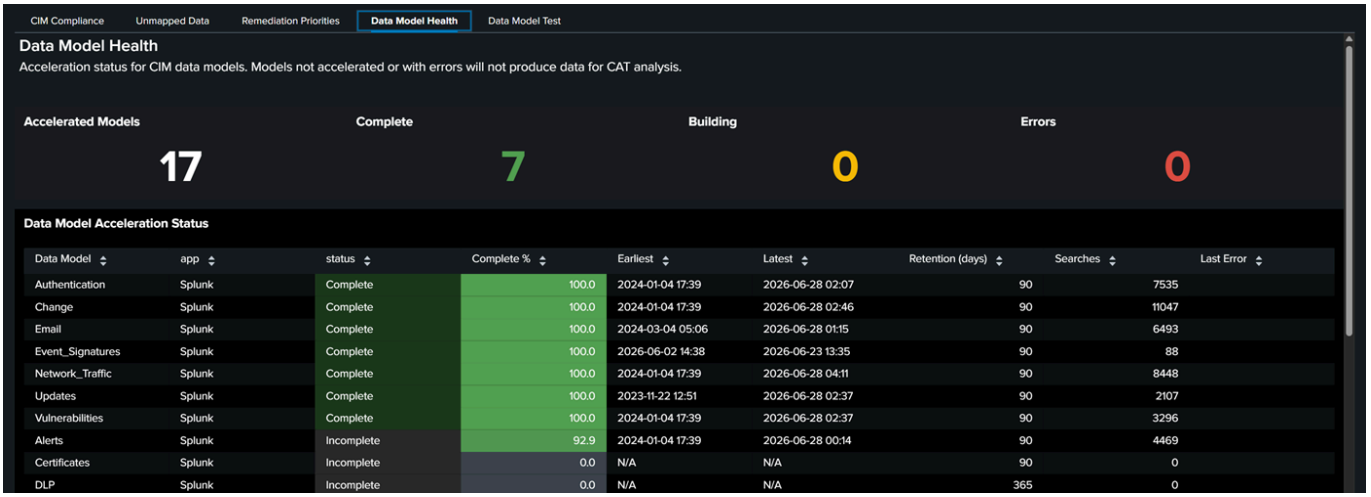
- **Sourcetype counts** — Total, mapped, and unmapped sourcetypes in the environment
- **Unmapped Sourcetypes** — Sourcetypes present in your indexes but not associated with any CIM data model. These represent blind spots in your security monitoring. Enriched from the sourcetype inventory with Vendor, Tech Category, Relevance, and Scope columns.
- **Mapped Sourcetypes** — Which models each sourcetype feeds, with event counts

Remediation Priorities Tab



- **Priority Matrix** — Scatter plot of quality vs. event volume
- **Priority Ranking** — Table sorted by impact score. Low quality on high-volume sources ranks highest. Includes the required tags and missing field count for actionable remediation.

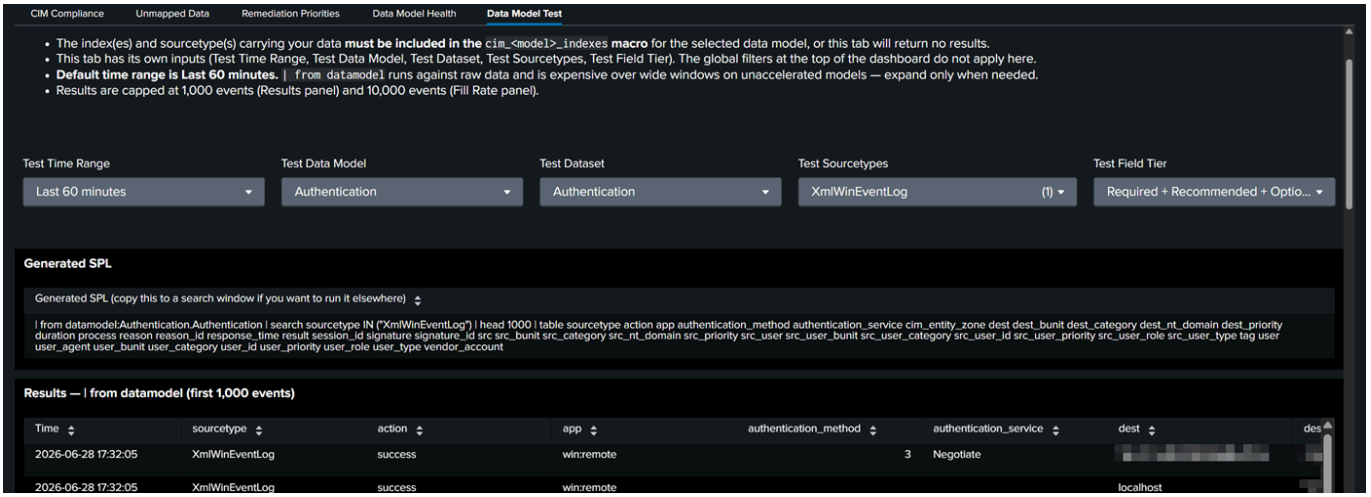
Data Model Health Tab



- **Acceleration scorecards** — Count of Complete, Building, Incomplete, and Error models
- **Acceleration Status table** — Per-model details including completion %, time range, retention, and search count. Models not fully accelerated will produce incomplete CIM compliance data.

  Note: The CIM Assessment Toolkit should be installed on the Splunk server from which data model accelerations are run to ensure accurate health measurements.

Data Model Test Tab



For validating CIM normalization at the end of a TA deployment effort. Select a data model, dataset, one or more sourcetypes, and a field tier (Required / Recommended / Optional / combinations); the tab generates and runs the equivalent `| from datamodel:Model.Dataset | search sourcetype IN (...) | table sourcetype <tier-filtered fields>` search so you do not have to remember the `| from datamodel` syntax or look up which fields belong to each dataset.

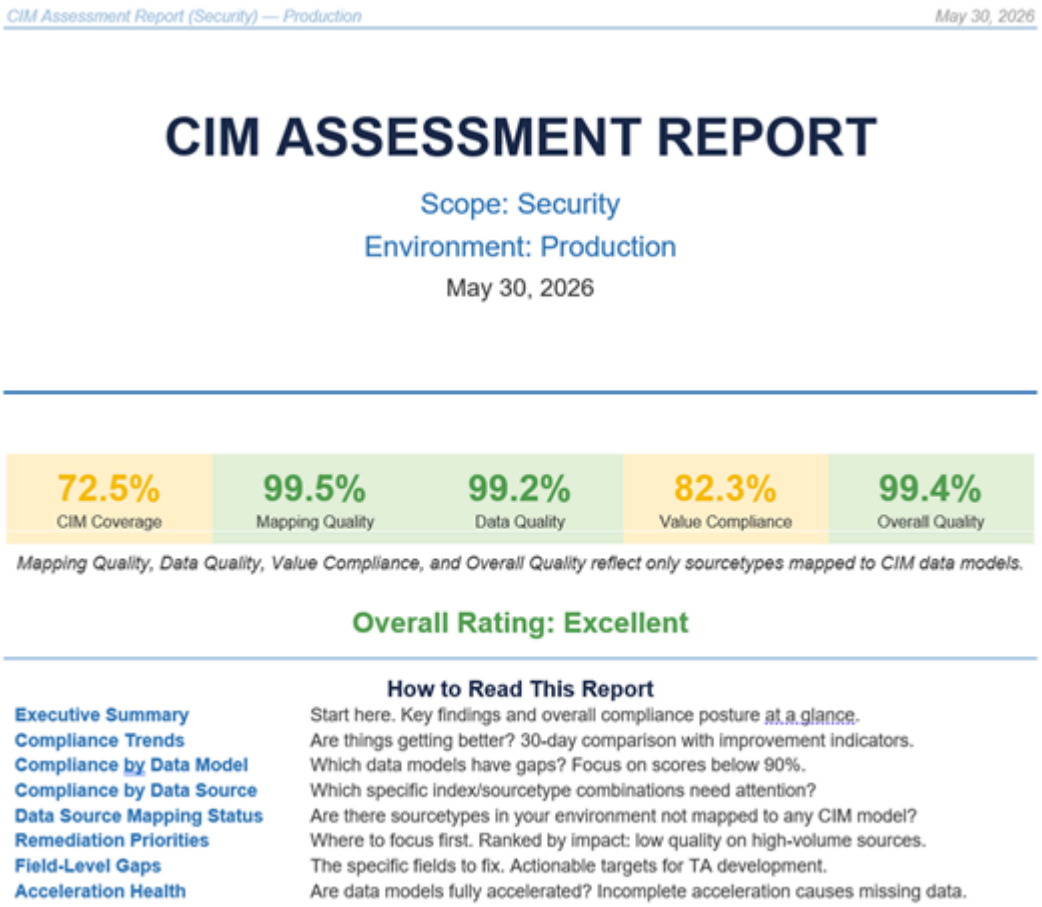
- **Generated SPL** — Copy-pasteable SPL string for the equivalent search, useful for running outside the dashboard.
- **Results** — Up to 1,000 rows from `| from datamodel:Model.Dataset | search sourcetype IN (...)` so you can see the actual normalized events.
- **Field Fill Rate** — Sampled over up to 10,000 events: percent non-null per field per sourcetype, tier-colored, sorted by tier and fill rate. This is the diagnostic view -- it tells you whether the *required and recommended* fields are actually populated, not just whether any data flowed.
- **CIM Field Reference** — Field, tier, description, and prescribed values for every field in the selected dataset.

Requirements: The relevant index(es) and sourcetype(s) must be included in the `cim_<model>_indexes` macro for the chosen data model or the tab returns no results. Default time range is **Last 60 minutes** -- `| from datamodel` runs against raw data and is expensive over wide windows on unaccelerated models. This tab has its own dedicated inputs; the global filters at the top of the dashboard do not apply here. The sourcetype picker is populated from `| tstats values(sourcetype) where cim_<model>_indexes`, not the inventory, so freshly-deployed TAs whose sourcetypes are not yet in the inventory still appear.

Top

Report Generation

The CIM Assessment Report is a professional Word document (.docx) containing KPI scorecards, compliance tables, field-level gap analysis, remediation priorities, and acceleration health. Generated entirely in Python using only the standard library -- no external dependencies. Designed for C-Suite, PM, and manager-level consumption.



CIM Assessment Report (Security) — Production *May 30, 2026*

CIM ASSESSMENT REPORT

Scope: Security

Environment: Production

May 30, 2026

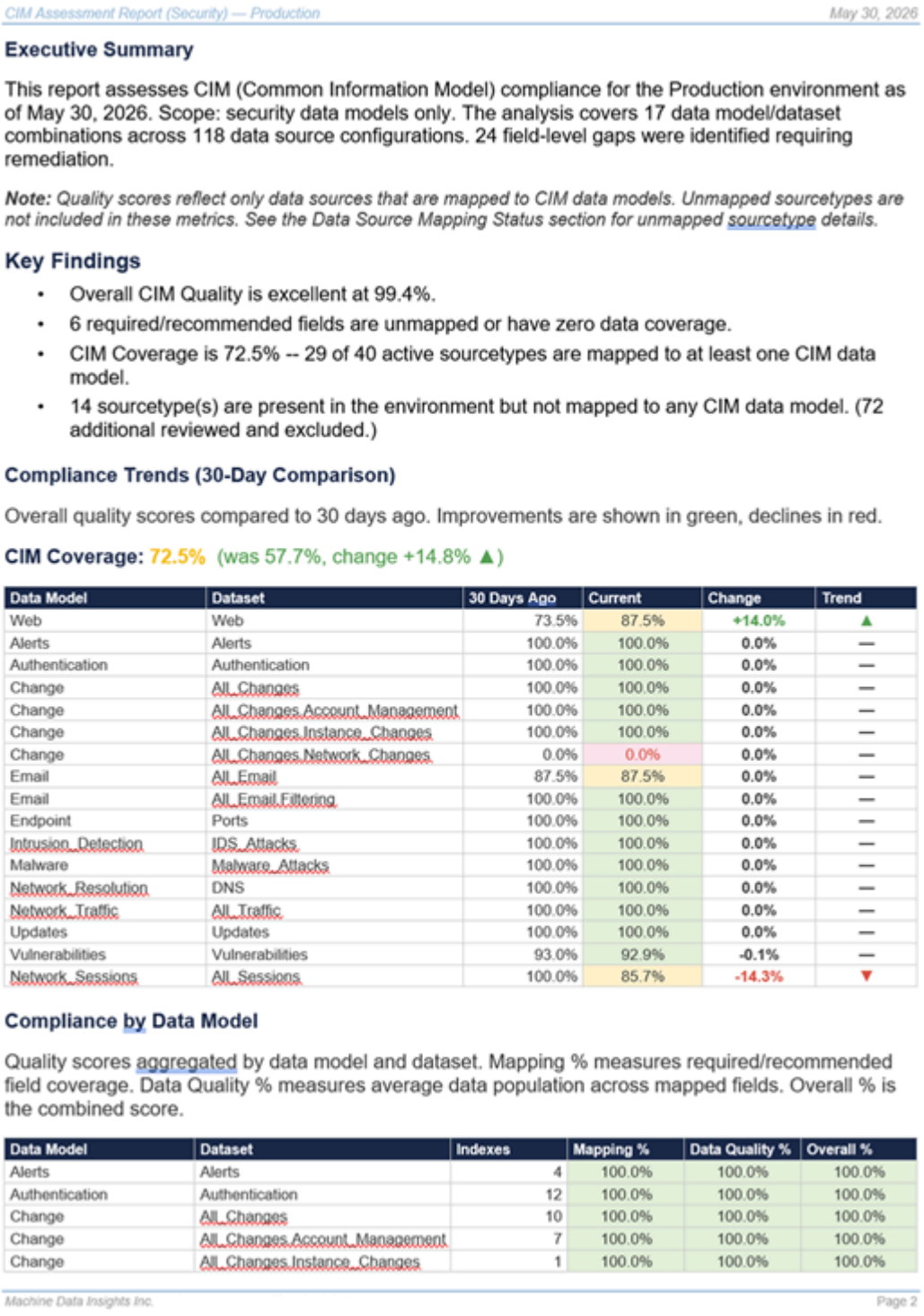
72.5%	99.5%	99.2%	82.3%	99.4%
CIM Coverage	Mapping Quality	Data Quality	Value Compliance	Overall Quality

Mapping Quality, Data Quality, Value Compliance, and Overall Quality reflect only sourcetypes mapped to CIM data models.

Overall Rating: Excellent

How to Read This Report

Executive Summary	Start here. Key findings and overall compliance posture at a glance.
Compliance Trends	Are things getting better? 30-day comparison with improvement indicators.
Compliance by Data Model	Which data models have gaps? Focus on scores below 90%.
Compliance by Data Source	Which specific index/sourcetype combinations need attention?
Data Source Mapping Status	Are there sourcetypes in your environment not mapped to any CIM model?
Remediation Priorities	Where to focus first. Ranked by impact: low quality on high-volume sources.
Field-Level Gaps	The specific fields to fix. Actionable targets for TA development.
Acceleration Health	Are data models fully accelerated? Incomplete acceleration causes missing data.



CIM Assessment Report (Security) — Production *May 30, 2026*

Executive Summary

This report assesses CIM (Common Information Model) compliance for the Production environment as of May 30, 2026. Scope: security data models only. The analysis covers 17 data model/dataset combinations across 118 data source configurations. 24 field-level gaps were identified requiring remediation.

Note: *Quality scores reflect only data sources that are mapped to CIM data models. Unmapped sourcetypes are not included in these metrics. See the Data Source Mapping Status section for unmapped sourcetype details.*

Key Findings

- Overall CIM Quality is excellent at 99.4%.
- 6 required/recommended fields are unmapped or have zero data coverage.
- CIM Coverage is 72.5% -- 29 of 40 active sourcetypes are mapped to at least one CIM data model.
- 14 sourcetype(s) are present in the environment but not mapped to any CIM data model. (72 additional reviewed and excluded.)

Compliance Trends (30-Day Comparison)

Overall quality scores compared to 30 days ago. Improvements are shown in green, declines in red.

CIM Coverage: 72.5% (was 57.7%, change +14.8% ▲)

Data Model	Dataset	30 Days Ago	Current	Change	Trend
Web	Web	73.5%	87.5%	+14.0%	▲
Alerts	Alerts	100.0%	100.0%	0.0%	—
Authentication	Authentication	100.0%	100.0%	0.0%	—
Change	All_Changes	100.0%	100.0%	0.0%	—
Change	All_Changes.Account_Management	100.0%	100.0%	0.0%	—
Change	All_Changes.Instance_Changes	100.0%	100.0%	0.0%	—
Change	All_Changes.Network_Changes	0.0%	0.0%	0.0%	—
Email	All_Email	87.5%	87.5%	0.0%	—
Email	All_Email.Filtering	100.0%	100.0%	0.0%	—
Endpoint	Ports	100.0%	100.0%	0.0%	—
Intrusion_Detection	IDS_Attacks	100.0%	100.0%	0.0%	—
Malware	Malware_Attacks	100.0%	100.0%	0.0%	—
Network_Resolution	DNS	100.0%	100.0%	0.0%	—
Network_Traffic	All_Traffic	100.0%	100.0%	0.0%	—
Updates	Updates	100.0%	100.0%	0.0%	—
Vulnerabilities	Vulnerabilities	93.0%	92.9%	-0.1%	—
Network_Sessions	All_Sessions	100.0%	85.7%	-14.3%	▼

Compliance by Data Model

Quality scores aggregated by data model and dataset. Mapping % measures required/recommended field coverage. Data Quality % measures average data population across mapped fields. Overall % is the combined score.

Data Model	Dataset	Indexes	Mapping %	Data Quality %	Overall %
Alerts	Alerts	4	100.0%	100.0%	100.0%
Authentication	Authentication	12	100.0%	100.0%	100.0%
Change	All_Changes	10	100.0%	100.0%	100.0%
Change	All_Changes.Account_Management	7	100.0%	100.0%	100.0%
Change	All_Changes.Instance_Changes	1	100.0%	100.0%	100.0%

Machine Data Insights Inc. *Page 2*

Prerequisites

Python 3.9+ (included with Splunk):

The report generator (`generate_report.py`) uses only the Python standard library -- no external packages required. Splunk ships with Python 3, so no additional installation is needed on the Splunk server.

For manual report generation outside of Splunk, any Python 3.9+ installation will work.

Top

Automated Weekly/Monthly Report & Email

The recommended approach — fully Splunk-native, cross-platform, no cron or Task Scheduler needed.

> **This setup is optional.** It is only needed for scheduled email delivery of the report -- the dashboard and every CIM assessment feature work without it. The first-launch setup page covered under Installation & Configuration is where the SMTP password is stored; it also remains available any time via Settings → Apps → Manage Apps → CIM Assessment Toolkit → Set up. Configure email below only if you want scheduled report delivery.

Setup (one-time, all through Splunk UI)

Step 1: Configure Splunk's email (SMTP) settings

CAT sends report email through the same SMTP server Splunk uses for its own alerting, so you configure it once in the Splunk Web UI and every app shares it. Navigate to Settings → Server settings → Email settings and fill in:

- **Mail host:** your SMTP server and port (e.g., `smtp.office365.com:587`)
- **Email security:** enable TLS (or SSL) if your server requires it - port 587 uses TLS
- **Send emails as:** the From address (e.g., `splunk@yourcompany.com`)
- **Username** and **Password:** the SMTP account credentials, if your server requires authentication

Click **Save**. That is all that is needed here - Splunk stores these settings globally and CAT picks them up automatically.

For Microsoft 365 SMTP: The sending account needs an Exchange Online license (Plan 1, $4/month minimum), Authenticated SMTP enabled (admin.microsoft.com → Users → Mail → Manage email apps), and Security Defaults disabled or the account excluded from MFA (Microsoft blocks SMTP AUTH by default under Security Defaults).

Step 2: Store the SMTP password for CAT

The password you entered in Step 1 is stored encrypted, and Splunk's REST API only ever returns it in encrypted form -- which `email_report.py` cannot decrypt. CAT therefore needs its own copy of the SMTP password in Splunk's credential vault. Store it using one of these methods (listed in priority order):

Preferred: Setup Page

Settings → Apps → Manage Apps → CIM Assessment Toolkit → Set up:

1. Enter and confirm the SMTP password
2. Click **Save Credential**

This stores the password encrypted at rest in Splunk's credential vault (realm=CIM_Assessment_Toolkit, username=smtp_password). The report script retrieves it securely via REST API. Re-run setup at any time to update the password.

Alternative: Splunk REST API

If you prefer to store the credential via CLI instead of the setup page:

```
curl -k -u admin:changeme \

https://localhost:8089/servicesNS/nobody/CIM_Assessment_Toolkit/storage/passwords
\
  -d realm=CIM_Assessment_Toolkit -d name=smtp_password -d password=YOUR_PASSWORD
```

CLI override: Pass `--smtp-password` on the command line for one-time use.

Step 3: Set report recipients

Settings → Advanced search → Search macros → find `cim_report_recipients` → Edit:

```
"security-team@company.com, manager@company.com"
```

Multiple recipients are comma-separated. The quotes are required.

Step 4: Set environment name

Settings → Advanced search → Search macros → find `cim_validator_environment` → Edit:

```
"Production"
```

This appears on the report cover page, headers, and executive summary. Use it to distinguish reports across environments (e.g., "Production", "Staging", "QA-West", "Dev").

Step 5: Set sender address (optional)

Settings → Advanced search → Search macros → find `cim_report_sender` → Edit:

```
"cim-reports@company.com"
```

If left empty, the report uses Splunk's configured sender from the email settings.

Step 6: Verify Python is available (Splunk ships with Python 3 -- no additional installation needed)

Step 7: Enable the schedule

From the CAT dashboard, click **Reports** → find `cim_validator_scheduled_report` → **Edit** → **Edit schedule**:

- Check **Schedule report**
- **Cron expression:** `0 8 * * 1` (every Monday at 8:00 AM, after the overnight collection and coverage-collection searches complete). For a monthly report, use `0 8 1 * *` (the 1st of each month at 8:00 AM).
- **Time range:** Last 24 hours

Then click the **Trigger actions** tab. The "CIM Assessment Report" action should already be listed. If not, click **+ Add Actions** → select **CIM Assessment Report** → **Save**.

Note: Leave the "New rendering method" toggle OFF on the trigger action configuration screen. CAT's alert action UI uses a classic HTML template (`email_report.html`), not a React component. With the toggle on, Splunk's newer React-based renderer has nothing to display, so the configuration guidance text disappears. This is cosmetic only -- the alert action still executes normally regardless of the toggle setting.

The report runs every Monday at 8:00 AM by default. Edit the cron schedule to change the frequency (for example, `0 8 1 * *` for a monthly report on the 1st).

How It Works

1. `cim_validator_collection` runs at 1:17 AM daily, populating the summary index
2. `cim_validator_scheduled_report` triggers at 8:00 AM Monday
3. Splunk fires `email_report.py` as a **custom alert action**, passing a JSON payload with the session token via stdin — no long command-line paths, no MAX_PATH issues on Windows
4. The script authenticates using the session token (no stored passwords needed for Splunk REST)
5. It reads SMTP settings from Splunk's mail configuration and the SMTP password from the credential store
6. Exports all report data via the local REST API
7. Imports `generate_report.py` and generates the .docx using Python's standard library
8. Emails the report directly via Python's `smtplib` (bypasses Splunk's built-in `sendemail.py`)

Top

Manual Report Generation

Four options for generating the report on demand.

Option 1: Python Script (Recommended — Cross-Platform)

Runs on the Splunk server, authenticates interactively:

```
cd $SPLUNK_HOME/etc/apps/CIM_Assessment_Toolkit/bin
python email_report.py --env "Production" --no-email
```

On Windows, use the same command from `C:\Program Files\Splunk\etc\apps\CIM_Assessment_Toolkit\bin`.

The `--no-email` flag generates the report without sending. The report is saved in the current working directory. You can override recipients with `--recipients "user@co.com"` or specify a different output location with `--output-dir "C:\Reports"`.

To generate a security-focused report:

```
python email_report.py --env "Production" --scope security --no-email
```

The `--scope` flag filters to data models matching that category in the `cim_model_categories` lookup. The report title, headers, and filename automatically reflect the scope (e.g., `CIM_Assessment_Report_Production_Security_20260313.docx`). Set the default scope via the `cim_report_scope` macro in Splunk.

Option 2: PowerShell Script (Windows)

Connects to Splunk REST API, exports data, and generates the report:

```
cd "C:\Program Files\Splunk\etc\apps\CIM_Assessment_Toolkit\bin"
.\Export-CIMReport.ps1 -Env "Production" -SplunkUri "https://localhost:8089"
```

Prompts for Splunk username and password. Optionally specify an output directory:

```
.\Export-CIMReport.ps1 -Env "Production" -Scope security -SplunkUri
"https://localhost:8089" -OutputDir "C:\Reports"
```

CSVs are retained in `bin\report_data\` for manual review.

Option 3: Bash Script (Linux / macOS)

Thin wrapper around `email_report.py` -- all arguments are passed through directly.

```
cd $SPLUNK_HOME/etc/apps/CIM_Assessment_Toolkit/bin
./export_report.sh --env "Production" --no-email
```

With email delivery and custom output directory:

```
./export_report.sh --env "Production" --scope security --output-dir /tmp/reports
```

Option 4: Manual CSV Export

If REST API access is unavailable:

1. Run each of the 11 SPL queries from the SPL Query Reference section below in the Splunk search bar (time range: Last 24 hours)
2. Export each result as CSV with the exact filename specified
3. Place all CSVs in a single directory
4. Run the report generator directly:

```
python generate_report.py --env "Production" --data-dir /path/to/csvs --output
CIM_Assessment_Report.docx
```

Report Sections

#	Section	Content
1	Cover Page	Title, environment name, date, color-coded KPI scorecards, overall rating
2	Executive Summary	Auto-generated narrative with key findings
3	Compliance by Data Model	Quality scores per data model and dataset
4	Compliance by Data Source	Per index/sourcetype compliance with event counts
5	Data Source Mapping Status	Unmapped and mapped sourcetypes
6	Remediation Priorities	Impact-ranked table with required tags and missing field counts
7	Field-Level Gaps	Specific unmapped and non-compliant fields
8	Acceleration Health	Data model acceleration status, completion %, retention

Top

Configuration Reference

Macros (Settings → Advanced search → Search macros)

Macro	Default	Description
`cim_validator_environment`	`"Production"`	Environment name displayed on reports
`cim_report_recipients`	`""`	Comma-separated email addresses for the weekly report
`cim_report_sender`	`""`	From address (empty = use Splunk's configured sender)
`cim_report_trend_days`	`7`	Days back for Compliance Trends comparison (7 = weekly, 30 = monthly)
`cim_report_scope`	`all`	Report scope filter: "all", "security", "operational", or any custom category from `cim_model_categories` lookup
`cim_validator_base_search`	(see below)	Base search for all dashboard and report queries. Reads from the summary index and renames index/sourcetype fields.

Macro	Default	Description
`cim_validator_index`	`cim_validator_index`	Summary index name
`cim_validator_data_model_list`	(all CIM models)	Comma-separated data models to analyze

`cim_validator_base_search` default definition:

```
index=`cim_validator_index` ReportType="cim_validator_v2"
| rename orig_index AS index, orig_sourcetype AS sourcetype
```

This macro is referenced by all 9 report queries and most dashboard panels. It pulls field-level compliance data from the summary index populated by `cim_validator_collection`.

Saved Searches (Settings → Searches, reports, and alerts)

Search	Schedule	Description
`cim_validator_collection`	Daily 1:17 AM	Main collection search — populates the summary index
`cim_validator_coverage_collection`	Daily 4:17 AM	Collects overall CIM coverage snapshots (mapped vs. total sourcetypes) per scope
`cim_validator_scheduled_report`	Monday 8:00 AM (disabled by default)	Generates and emails the report
`cim_validator_collection_search`	Manual	View raw collection data
`splunk_data_model_objects_fields_search`	Manual	View the CIM field reference lookup
`cim_validator_acceleration_health`	Manual	Check data model acceleration status
`Data Model Field Mapping Quality Alert`	Monday 8:00 AM (disabled)	Alert when mapping coverage drops below 80%

Lookups (Settings → Lookups → Lookup table files)

Lookup	Description
`splunk_data_model_objects_fields`	CIM field reference -- all data models, datasets, fields, prescribed values, and descriptions. Ships as `splunk_data_model_objects_fields.csv` (currently sourced from Splunk CIM v8.5.0). To refresh between app releases, replace the CSV in

Lookup	Description
	`$SPLUNK_HOME/etc/apps/CIM_Assessment_Toolkit/lookups/` with an updated copy of the same filename -- no conf changes required.
`cim_model_categories`	Data model classification for report scoping. Edit to customize categories for your environment.
`cim_sourcetype_inventory`	Master reference (enrichment only) for all known sourcetypes -- provides technology category, vendor, scope, security classification, security relevance, candidate data models, and descriptions. Doubles as a general-purpose CIM sourcetype catalog. Ships with a broad set of common industry sourcetypes.
`cim_sourcetype_exclusions`	Short list of sourcetypes to omit from CIM-compliance calculations and the unmapped/coverage views (`sourcetype_exclusions.csv`). Seeded with Splunk/Cribl internal plumbing sourcetypes. Edit to add your own.

Customizing Model Categories:

Edit `cim_model_categories.csv` via Settings → Lookups → Lookup table files, or directly in the app's `lookups/` directory. Each row maps a model name to one or more comma-separated categories:

```
model,category
Authentication,security
Network_Traffic,"security, operational"
Performance,operational
Custom_Model,"security, pci"
```

Any category string works — the report filters using `like(category, "%value%")`, so custom tags like `pci`, `hipaa`, `sox`, or `cloud` work immediately without code changes.

Managing the Sourcetype Inventory:

As of v2.10.0 the inventory is enrichment-only -- it no longer controls exclusions (those moved to the separate `cim_sourcetype_exclusions` lookup, described below). Each row of `cim_sourcetype_inventory.csv` describes a sourcetype with classification and enrichment context:

```
sourcetype,tech_category,scope,security_classification,security_relevance,data_mod
els,vendor,description,expanded_desc
splunkd,linux,operational,,none,,Splunk,Splunk daemon logs,Internal Splunk daemon
logs covering indexer/search-head/forwarder operations.
pan:traffic,network,security,NGFW,high,Network_Traffic.All_Traffic,Palo Alto
Networks,Palo Alto firewall traffic logs,Palo Alto Networks NGFW session traffic
logs with source/destination addressing and application identification.
```

Key columns: `tech_category` is a single canonical technology family per sourcetype from a controlled vocabulary (`app`, `aws`, `azure`, `gcp`, `linux`, `network`, `security`, `windows`); `scope` and `security_relevance` enrich the dashboard and report; `data_models` lists the candidate CIM data models a sourcetype maps to; `vendor`, `security_classification`, and `description` provide reference context. `expanded_desc` is a longer curator-facing note and is intentionally not surfaced in the dashboard or report.

The lookup ships with a broad set of common industry sourcetypes. Sourcetypes not in the inventory still appear in the unmapped list -- they simply have no enrichment data.

Updating or replacing the inventory via Splunk Web:

To load a supplied custom inventory (for example, one produced by the Data Refinery Scope Discovery tab for a specific client) or an updated copy:

1. Settings -> Lookups -> Lookup table files. Set the app context to **CIM Assessment Toolkit**, locate `cim_sourcetype_inventory.csv`, and open it.
2. Upload the replacement file, keeping the **same filename** (`cim_sourcetype_inventory.csv`) so the existing lookup definition resolves with no further changes. The file must keep the column headers shown above; extra columns are ignored, and a missing `exclude`, `scope`, or `security_relevance` column only reduces filtering and enrichment.
3. Confirm scope under Settings -> Lookups -> Lookup definitions -> `cim_sourcetype_inventory`. The definition just points at the filename, so no edit is needed when the filename is unchanged.
4. Re-run the `cim_validator_collection` saved search (or wait for its schedule) so the dashboard and the next report reflect the new inventory.

Treat a supplied inventory as a superset: keep the existing rows and append the new sourcetypes Scope Discovery surfaced, so previously classified sourcetypes retain their enrichment.

Excluding Sourcetypes from CIM Compliance

Some sourcetypes should never count toward CIM compliance -- Splunk's own internal logs, data-pipeline plumbing, and any noisy sources you have decided are out of scope. Listing them in `cim_sourcetype_exclusions.csv` removes them from the CIM Coverage score, the Unmapped Data tab, and the report, so your numbers reflect only the data you actually care about.

CAT ships this lookup pre-seeded with the common Splunk/Cribl internal sourcetypes (`splunkd`, `splunkd_access`, `kvstore`, `mongod`, `scheduler`, `stash`, `cribl:internal`, and similar), so you usually do not need to touch it. Add to it when you want to exclude something specific to your environment.

The simplest way to exclude a sourcetype: add a row with just the sourcetype name. A row with only a `sourcetype` value (everything else blank) is treated as excluded, so a bare list is all you need:

```
sourcetype,exclude,exclude_reason,reviewed_by,reviewed_date
my_noisy_sourcetype
another_internal_sourcetype
```

The fully documented form (recommended for shared environments, so the next person knows why):

```
sourcetype,exclude,exclude_reason,reviewed_by,reviewed_date
my_noisy_sourcetype,yes,Heartbeat data -- no security value,Jane Smith,2026-06-11
```

Step by step (Splunk Web):

1. Go to **Settings -> Lookups -> Lookup table files**, set the app context to **CIM Assessment Toolkit**, and open `sourcetype_exclusions.csv` (the Splunk App for Lookup File Editing gives you a spreadsheet-style editor; otherwise download, edit, and re-upload).
2. Add one row per sourcetype you want to exclude. The sourcetype name must match exactly what Splunk reports (case-sensitive, including any colons, e.g. `pan:traffic`).
3. Save.
4. Re-run the **`cim_validator_coverage_collection`** saved search (Settings -> Searches, reports, and alerts -> Run) so the collected CIM Coverage trend reflects the change. The live dashboard scorecards and Unmapped Data tab update on their own the next time they run -- no collection needed for those.

What goes in the `exclude` column: it is optional and case-insensitive.

Value	Meaning
(blank), `y`, `yes`, `t`, `true`, `1`	Exclude this sourcetype
`n`, `no`, `f`, `false`, `0`	Keep this sourcetype (do not exclude)

The `n`/`no`/... values let you **temporarily turn an exclusion off without deleting the row** -- handy for keeping the `exclude_reason` note while re-including a sourcetype. `exclude_reason`, `reviewed_by`, and `reviewed_date` are free-text notes for your own audit trail; CAT does not parse them.

If the file is missing or broken, CAT keeps working. Every consumer (dashboard, coverage collection, and report) reads this lookup through a fault-tolerant subsearch. If the file is deleted, unreadable, or pointed at a path that does not exist, nothing is excluded and the rest of CAT runs normally -- you simply see every sourcetype in the unmapped/coverage views. CAT never breaks because of an exclusions problem.

Making your exclusions survive an app upgrade. Editing the shipped `sourcetype_exclusions.csv` in place works, but a CAT upgrade replaces that file with the latest seed and your additions are lost. To keep a custom list permanently:

1. Put your sourcetypes in a CSV with a **new filename**, for example `sourcetype_exclusions_local.csv` (same column headers).
2. Upload it under **Settings -> Lookups -> Lookup table files** (app context: CIM Assessment Toolkit).
3. Go to **Settings -> Lookups -> Lookup definitions**, open `cim_sourcetype_exclusions`, and change its **filename** to your new file.

Because your file has a name CAT does not ship, the upgrade leaves it alone; and the definition change is saved to `local/transforms.conf`, which upgrades never touch. Both survive -- only the original seed gets refreshed. (Splunk has no `local/lookups` layering for lookup table files, so using a differently-named file is what makes this upgrade-safe.)

Inventory changelog (sidecar): `cim_sourcetype_inventory.csv.version.csv` sits next to the lookup as a small, human-maintained changelog (columns

`last_updated,updated_by,note,base_catalog_last_updated`). It is a glance-level "how current is this copy?" note, not a programmatically verified manifest. `last_updated` is when this inventory copy was last changed; `base_catalog_last_updated` is the `last_updated` of the standard shipped catalog this copy was derived from -- so when a client receives a custom inventory carrying unique or additional sourcetypes, both the custom copy's version and the base catalog version it built on are tracked. When you replace the inventory, update this file too; the report prints `last_updated` (with `updated_by` and `note`) and the `base_catalog_last_updated` as an "as of" line in the Data Source Mapping Status section -- wherever the inventory version is surfaced, both dates are shown. The file is paired with its own lookup definition (`cim_sourcetype_inventory_version`) so it is a recognized lookup -- query it directly with `| inputlookup cim_sourcetype_inventory_version`.

Script Reference

Script	Platform	Usage
`email_report.py`	All (Python)	Splunk alert action + manual report generation and email
`generate_report.py`	All (Python)	Core report generator -- reads CSVs, produces .docx (standard library only)
`Export-CIMReport.ps1`	Windows	PowerShell wrapper for manual report generation
`export_report.sh`	Linux/macOS	Bash wrapper for manual report generation

PowerShell Parameters (Export-CIMReport.ps1)

Parameter	Required	Description
`-Env`	Yes	Environment name for the report
`-SplunkUri`	No	Splunk REST API URI (e.g., `https://localhost:8089`)
`-SplunkUser`	No	Splunk username (prompted if `-SplunkUri` is set)
`-SplunkPass`	No	Splunk password as SecureString (prompted if `-SplunkUri` is set)
`-DataDir`	No	Directory with pre-exported CSV files (manual mode)
`-OutputDir`	No	Output directory for the report (default: current directory)
`-TrendDays`	No	Days back for trends comparison (default: 7)
`-Scope`	No	Report scope: all, security, operational, or custom category (default: all)

Python Parameters (email_report.py)

Flag	Default	Description
`--env`	From macro	Environment name
`--recipients`	From macro	Comma-separated email addresses
`--sender`	From macro/Splunk	From email address

Flag	Default	Description
`--username`	(prompted)	Splunk username for manual auth
`--password`	(prompted)	Splunk password for manual auth
`--token`	(auto)	Splunk session token (used by alert action)
`--output-dir`	current directory	Output directory for the report
`--scope`	From macro	Report scope: all, security, operational, or custom category
`--smtp-password`	From credential store	SMTP password override (highest priority, bypasses credential store)
`--no-email`	false	Generate report only, don't email

Top

SPL Query Reference

These 11 queries populate the report. The automated scripts run these internally. They are documented here for manual CSV export and troubleshooting.

All queries reference `cim_validator_base_search`, which expands to `index=`cim_validator_index` ReportType="cim_validator_v2" | rename orig_index AS index, orig_sourcetype AS sourcetype` (see Configuration Reference above). The inner `cim_validator_index` is a macro that resolves to your configured summary index name (default: `cim_validator_index`).

Set the time range to **Last 24 hours** for all queries.

1. kpi.csv — KPI Summary Scores

```
`cim_validator_base_search`
| bin _time span=1d
| eval is_rec = if(field_class IN ("required", "recommended"), 1, 0)
| eval is_rec_mapped = if(is_rec=1 AND field_count > 0, 1, 0)
| eval has_pv = if(isnotnull(value_compliance_pct), 1, 0)
| eval pv_compliant = if(has_pv=1 AND value_compliance_pct >= 80, 1, 0)
| eventstats sum(is_rec) as total_rec_fields sum(is_rec_mapped) as
mapped_rec_fields
    avg(eval(if(is_rec=1, percent_coverage, null()))) as percent_data_coverage
    sum(has_pv) as total_pv_fields sum(pv_compliant) as compliant_pv_fields
    by modelName dataset index sourcetype
| eval rec_field_coverage_pct = round(mapped_rec_fields / total_rec_fields * 100,
2)
| eval percent_data_coverage = round(percent_data_coverage, 2)
| eval value_quality_pct = if(total_pv_fields > 0, round(compliant_pv_fields /
total_pv_fields * 100, 2), null())
| eval overall_quality_pct = round((rec_field_coverage_pct +
percent_data_coverage) / 2, 2)
| dedup modelName dataset index sourcetype
```

```
| stats avg(rec_field_coverage_pct) as mapping_quality
    avg(percent_data_coverage) as data_quality
    avg(eval(if(isnotnull(value_quality_pct), value_quality_pct, null()))) as
value_compliance
    avg(overall_quality_pct) as overall_quality
| eval mapping_quality = round(mapping_quality, 1)
| eval data_quality = round(data_quality, 1)
| eval value_compliance = round(value_compliance, 1)
| eval overall_quality = round(overall_quality, 1)
```

2. compliance_detail.csv — Compliance by Data Model

```
`cim_validator_base_search`
| bin _time span=1d
| eval is_rec = if(field_class IN ("required", "recommended"), 1, 0)
| eval is_rec_mapped = if(is_rec=1 AND field_count > 0, 1, 0)
| eventstats sum(is_rec) as total_rec_fields sum(is_rec_mapped) as
mapped_rec_fields
    avg(eval(if(is_rec=1, percent_coverage, null()))) as percent_data_coverage
    by modelName dataset index sourcetype
| eval rec_field_coverage_pct = round(mapped_rec_fields / total_rec_fields * 100,
2)
| eval percent_data_coverage = round(percent_data_coverage, 2)
| eval overall_quality_pct = round((rec_field_coverage_pct +
percent_data_coverage) / 2, 2)
| dedup modelName dataset index sourcetype
| stats avg(rec_field_coverage_pct) as "Mapping %"
    avg(percent_data_coverage) as "Data Quality %"
    avg(overall_quality_pct) as "Overall %"
    dc(index) as indexes dc(sourcetype) as sourcetypes
    by modelName dataset
| eval "Mapping %" = round('Mapping %', 1)
| eval "Data Quality %" = round('Data Quality %', 1)
| eval "Overall %" = round('Overall %', 1)
| rename modelName as "Data Model" dataset as "Dataset"
| sort "Data Model" Dataset
```

3. compliance_summary.csv — Compliance by Data Source

```
`cim_validator_base_search`
| bin _time span=1d
| eval is_rec = if(field_class IN ("required", "recommended"), 1, 0)
| eval is_rec_mapped = if(is_rec=1 AND field_count > 0, 1, 0)
| eval has_pv = if(isnotnull(value_compliance_pct), 1, 0)
| eval pv_compliant = if(has_pv=1 AND value_compliance_pct >= 80, 1, 0)
| eventstats sum(is_rec) as total_rec_fields sum(is_rec_mapped) as
mapped_rec_fields
    avg(eval(if(is_rec=1, percent_coverage, null()))) as percent_data_coverage
    max(total_count) as event_count
```

```
    sum(has_pv) as total_pv_fields sum(pv_compliant) as compliant_pv_fields
    values(eval(if(is_rec=1 AND is_rec_mapped=0, field, null()))) as
missing_rec_fields
    by modelName dataset index sourcetype
| eval rec_field_coverage_pct = round(mapped_rec_fields / total_rec_fields * 100,
2)
| eval percent_data_coverage = round(percent_data_coverage, 2)
| eval value_quality_pct = if(total_pv_fields > 0, round(compliant_pv_fields /
total_pv_fields * 100, 2), null())
| eval overall_quality_pct = round((rec_field_coverage_pct +
percent_data_coverage) / 2, 2)
| dedup modelName dataset index sourcetype
| rename modelName as "Data Model" dataset as "Dataset"
    rec_field_coverage_pct as "Mapping %"
    percent_data_coverage as "Data Quality %"
    value_quality_pct as "Value Compliance"
    overall_quality_pct as "Overall %"
    event_count as "Events"
    missing_rec_fields as "Missing Fields"
| table "Data Model" Dataset index sourcetype "Mapping %" "Data Quality %" "Value
Compliance" "Overall %" Events "Missing Fields"
| sort "Data Model" Dataset sourcetype
```

4. field_gaps.csv — Unmapped & Non-Compliant Fields

```
`cim_validator_base_search`
| search field_class IN ("required", "recommended")
| stats max(total_count) as total_count sum(field_count) as field_count
    max(distinct_count) as distinct_count
    avg(percent_coverage) as percent_coverage
    latest(field_class) as field_class
    sum(compliant_count) as compliant_count
    avg(value_compliance_pct) as value_compliance_pct
    by modelName dataset sourcetype field
| eval pv_rounded = round(value_compliance_pct, 1)
| where field_count=0 OR (isnotnull(value_compliance_pct) AND pv_rounded < 100)
| eval pv_display = if(isnotnull(value_compliance_pct), tostring(pv_rounded)."%",
"---")
| fields - pv_rounded
| rename modelName as "Data Model" dataset as "Dataset" field as "Field"
    field_class as "Class" field_count as "Count"
    percent_coverage as "Coverage %" pv_display as "Value Compliance"
| table "Data Model" Dataset sourcetype Field Class Count "Coverage %" "Value
Compliance"
| sort "Data Model" Dataset sourcetype -Class Field
```

5. unmapped.csv — Unmapped Sourcetypes

```
| tstats count WHERE index=* NOT index=_* NOT index=`cim_validator_index` BY
index, sourcetype
| stats sum(count) as event_count values(index) as indexes by sourcetype
| join type=left sourcetype
    [| search `cim_validator_base_search`
    | stats dc(modelName) as model_count values(modelName) as mapped_models by
sourcetype]
| eval is_mapped = if(isnotnull(model_count) AND model_count > 0, "Yes", "No")
| search is_mapped="No"
| lookup cim_sourcetype_inventory sourcetype OUTPUT vendor, tech_category,
security_relevance, scope
| rename event_count as "Events" vendor as "Vendor" tech_category as "Tech
Category" security_relevance as "Relevance" scope as "Scope"
| table sourcetype Events Vendor "Tech Category" Relevance Scope
| sort Relevance Vendor sourcetype
```

6. mapped.csv — Mapped Sourcetypes

```
| tstats count WHERE index=* NOT index=_* NOT index=`cim_validator_index` BY
index, sourcetype
| stats sum(count) as event_count values(index) as indexes by sourcetype
| join type=left sourcetype
    [| search `cim_validator_base_search`
    | stats dc(modelName) as model_count values(modelName) as mapped_models by
sourcetype]
| eval is_mapped = if(isnotnull(model_count) AND model_count > 0, "Yes", "No")
| search is_mapped="Yes"
| eval mapped_list = mvjoin(mapped_models, ", ")
| lookup cim_sourcetype_inventory sourcetype OUTPUT vendor, security_relevance,
scope
| rename event_count as "Events" vendor as "Vendor" security_relevance as
"Relevance" scope as "Scope" mapped_list as "Mapped To"
| table sourcetype Vendor Relevance Scope Events "Mapped To"
| sort Relevance Vendor sourcetype
```

7. remediation.csv — Remediation Priority Ranking

```
`cim_validator_base_search`
| search field_class IN ("required", "recommended")
| bin _time span=1d
| eval is_mapped = if(field_count > 0, 1, 0)
| stats sum(is_mapped) as mapped_rec_fields dc(field) as total_rec_fields
    avg(percent_coverage) as percent_data_coverage
    max(total_count) as event_count
    values(eval(if(is_mapped=0, field, null()))) as missing_rec_fields
    avg(eval(if(isnotnull(value_compliance_pct), value_compliance_pct, null())))
as avg_value_compliance
    by modelName dataset index sourcetype
```

```
| eval rec_field_coverage_pct = round(mapped_rec_fields / total_rec_fields * 100,
2)
| eval percent_data_coverage = round(percent_data_coverage, 2)
| eval overall_quality_pct = round((rec_field_coverage_pct +
percent_data_coverage) / 2, 2)
| eval priority_score = round((100 - overall_quality_pct) * log(event_count + 1),
2)
| eval missing_rec_count = mvcount(missing_rec_fields)
| fillnull value=0 missing_rec_count
| lookup splunk_data_model_objects_fields model as modelName dataset OUTPUT
constraints
| eval required_tags = mvdedup(constraints)
| eval required_tags = mvindex(required_tags, 0)
| rex field=required_tags "tag=(?<required_tags>.+)"
| rename modelName as "Data Model" dataset as "Dataset"
    overall_quality_pct as "Overall %" rec_field_coverage_pct as "Mapping %"
    event_count as "Events" priority_score as "Priority"
    missing_rec_count as "Missing #" required_tags as "Required Tags"
| where Priority > 0
| table "Data Model" Dataset sourcetype "Overall %" "Mapping %" Events Priority
"Missing #" "Required Tags"
| sort "Data Model" Dataset sourcetype
```

8. acceleration.csv — Data Model Acceleration Status

```
| rest /services/admin/summarization by_tstats=t splunk_server=local
| regex title="tstats:DM_"
| rex field=title "tstats:DM_(?<app>[^_]+)_(?<modelName>.+)"
| eval modelName = replace(modelName, "^SA_CIM_", "")
| eval complete_pct = round(coalesce('summary.complete', complete, 0) * 100, 1)
| eval is_building = coalesce('summary.is_inprogress', is_inprogress, 0)
| eval error_msg = coalesce('summary.last_error', last_error, "")
| eval status = case(error_msg!="" AND error_msg!="None", "Error", complete_pct >=
99.9, "Complete", is_building=1, "Building", 1==1, "Incomplete")
| eval e_time = coalesce('summary.earliest_time', earliest_time)
| eval l_time = coalesce('summary.latest_time', latest_time)
| eval earliest_time = if(isnotnull(e_time) AND e_time > 0, strftime(e_time, "%Y-
%m-%d %H:%M"), "N/A")
| eval latest_time = if(isnotnull(l_time) AND l_time > 0, strftime(l_time, "%Y-%m-
%d %H:%M"), "N/A")
| eval ret = coalesce('summary.retention', retention)
| eval retention_days = if(isnotnull(ret) AND ret > 0, round(ret / 86400, 1),
"N/A")
| eval access_count = coalesce('summary.access_count', access_count, 0)
| rename modelName as "Data Model" complete_pct as "Complete %"
    earliest_time as "Earliest" latest_time as "Latest"
    retention_days as "Retention (days)" access_count as "Searches"
    error_msg as "Last Error"
| table "Data Model" app status "Complete %" Earliest Latest "Retention (days)"
```

```
Searches "Last Error"
| sort "Data Model"
```

9. trends.csv -- Compliance Trends (Prior Snapshot)

Set the time range to the trend comparison window (default: 3-day window centered on 7 days ago).

```
`cim_validator_base_search`
| eval percent_coverage = if(percent_coverage > 100, 100, percent_coverage)
| bin _time span=1d
| eval is_rec = if(field_class IN ("required", "recommended"), 1, 0)
| eval is_rec_mapped = if(is_rec=1 AND field_count > 0, 1, 0)
| eventstats sum(is_rec) as total_rec_fields sum(is_rec_mapped) as
mapped_rec_fields
    avg(eval(if(is_rec=1, percent_coverage, null()))) as percent_data_coverage
    by modelName dataset index sourcetype
| eval rec_field_coverage_pct = round(mapped_rec_fields / total_rec_fields * 100,
2)
| eval percent_data_coverage = round(percent_data_coverage, 2)
| eval overall_quality_pct = round((rec_field_coverage_pct +
percent_data_coverage) / 2, 2)
| dedup modelName dataset index sourcetype
| stats latest(overall_quality_pct) as "Prior Overall %" by modelName dataset
| eval "Prior Overall %" = round('Prior Overall %', 1)
| rename modelName as "Data Model" dataset as "Dataset"
```

10. cim_coverage.csv -- CIM Coverage (Current)

Uses live tstats to calculate the percentage of active sourcetypes mapped to at least one CIM data model. When scope is set, filters models by `cim_model_categories` category and sourcetypes by `cim_sourcetype_inventory` scope. Sourcetypes listed in the `cim_sourcetype_exclusions` lookup are removed from the denominator (read through a fault-tolerant subsearch, so a missing file excludes nothing).

```
| tstats count WHERE index=* NOT index=_* NOT index=`cim_validator_index` BY
sourcetype
| join type=left sourcetype
    [| search `cim_validator_base_search`
    | stats dc(modelName) as model_count by sourcetype]
| eval is_mapped = if(isnotnull(model_count) AND model_count > 0, 1, 0)
| join type=left sourcetype
    [| inputlookup cim_sourcetype_exclusions
    | eval exclude = if(match(lower(trim(exclude)), "^(n|no|f|false|0)$"), "N",
"Y")
    | fields sourcetype exclude]
| eval exclude = if(isnull(exclude), "N", exclude)
| where NOT (exclude="Y")
| stats sum(is_mapped) as mapped count as total
| eval cim_coverage = round(mapped / total * 100, 1)
```

11. cim_coverage_prior.csv -- CIM Coverage (Prior Snapshot)

Reads the prior CIM Coverage value from the summary index, collected daily by `cim_validator_coverage_collection`. The collection writes one row per scope (security, operational, etc.) plus an "all" row, so the prior query filters by the current report scope. Set the time range to the trend comparison window (same as query 9).

```
index=`cim_validator_index` ReportType="cim_coverage" scope="all"
| stats latest(cim_coverage) as prior_cim_coverage latest(mapped) as prior_mapped
latest(total) as prior_total
```

Top

Troubleshooting

Dashboard Issues

Symptom	Cause	Fix
All panels show "No results"	Collection hasn't run	Run `cim_validator_collection` from Settings → Searches, reports, and alerts
KPIs show 0% or N/A	Summary index empty or wrong time range	Set Analysis Duration to Last 24 hours; verify `cim_validator_index` has events
Acceleration shows Incomplete	Data models not accelerated	Settings → Data Models → enable acceleration
Missing data models	Model not in `cim_validator_data_model_list`	Add the model name to the macro

Symptom	Cause	Fix
	macro	

Report Generation Issues

Symptom	Cause	Fix
`python` not recognized	Python not in PATH	Ensure Python 3.9+ is installed and in your system PATH
Report shows N/A for all KPIs	Collection hasn't run recently	Run `cim_validator_collection`, then regenerate the report
Empty CSV files from REST export	Time range mismatch	Ensure collection data exists within the last 24 hours
PowerShell execution policy	Script execution disabled	Run `Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope Process`
Acceleration CSV empty	REST API listing doesn't work	Report scripts query each model individually (already handled)
`unmapped.csv` shows 0 bytes	All sourcetypes are mapped to CIM	This is normal — the report correctly shows "All sourcetypes are mapped"
Report generated in unexpected location	Default is current working directory	Use `--output-dir` to specify, or `cd` to desired folder before running

Email Delivery Issues

Important: `email_report.py` sends email directly using Python's `smtplib`, bypassing Splunk's built-in `sendemail.py`. It reads the SMTP server settings from Splunk's email configuration (Settings → Server settings → Email settings), so those must be configured correctly.

Diagnostic searches:

```
-- Check effective email configuration
| rest /services/configs/conf-alert_actions/email
| table mailserver from auth_username use_tls

-- Check for email errors in Splunk logs
index=_internal (sendemail OR SendMail OR smtp) earliest=-1h
| sort - _time
| table _time log_level message

-- Check if a scheduled alert fired
index=_internal sourcetype=scheduler status=* earliest=-1h
| sort - _time
| table _time savedsearch_name status
```

Symptom	Cause	Fix
`[WinError 10061] No connection could be made`	Splunk email settings not configured, or wrong mail host/port	Set the mail host, security, and From address under Settings → Server settings → Email settings, then Save. Confirm the mail host and port are correct (e.g., `smtp.office365.com:587`).
`mailserver` shows `localhost` in REST query	Email settings still at default (never configured)	Configure them under Settings → Server settings → Email settings and Save; Splunk stores them globally so all apps, including CAT, pick them up.
`SMTP AUTH extension not supported by server`	TLS not starting before auth	Update `email_report.py` to v2.7.3+ which adds `ehlo()` before `starttls()`. Ensure TLS is enabled under Settings → Server settings → Email settings.
`SendAsDenied` / `not allowed to send as`	Splunk using hostname as sender	Ensure the **Send emails as** address (Settings → Server settings → Email settings) is set and matches the authenticated account.
`Got encrypted password` warning	Splunk REST API won't return clear password	Use the setup page (Apps → CIM Assessment Toolkit → Set up) to store the SMTP password in the credential store, or use `--smtp-password` on the command line.
`Authentication unsuccessful, user is locked by security defaults`	Microsoft 365 Security Defaults blocks SMTP AUTH	Disable Security Defaults in Azure portal (Microsoft Entra ID → Properties → Manage Security defaults), then re-enable MFA per-user for human accounts.
Email works from `| sendemail` SPL but not from alerts	Splunk's alert email action has a known `from` field bug	CAT's `email_report.py` bypasses this by sending directly via Python — use the scheduled alert action with `email_report.py`, not Splunk's built-in email action.

Microsoft 365 SMTP checklist:

1. Exchange Online license assigned to the sending account (Plan 1 minimum, $4/month)
2. Authenticated SMTP enabled: admin.microsoft.com → Users → select user → Mail → Manage email apps → enable Authenticated SMTP
3. Security Defaults disabled (or the account excluded via Conditional Access policy)
4. Splunk email settings (Settings → Server settings → Email settings) have mail host `smtp.office365.com:587`, TLS enabled, and the **Send emails as** address matching the authenticated account

Collection Search Issues

Symptom	Cause	Fix
Collection returns no results	No accelerated data models	Enable acceleration and wait for completion

Symptom	Cause	Fix
Collection takes >10 minutes	One tstats per field is slow	Expected for large environments; runs at 1:17 AM so doesn't impact users
`field_count` is 0 for all fields	Acceleration not complete	Check Data Model Health tab; wait for 100% completion
`compliant_count` is blank	No prescribed values for that field, or values don't match CIM spec	Verify field values match prescribed values in the CIM reference

Top

File Inventory

```
CIM_Assessment_Toolkit/
├── app.manifest                # Splunkbase app manifest (version, dependencies)
├── bin/
│   ├── email_report.py         # Splunk alert action: generate + email report
(Python)
│   ├── generate_report.py      # Core report generator (Python, standard library
only)
│   ├── Export-CIMReport.ps1    # Windows PowerShell wrapper
│   └── export_report.sh        # Linux/macOS bash wrapper
├── default/
│   ├── alert_actions.conf      # Custom alert action definition (CIM Assessment
Report)
│   ├── app.conf                # App metadata and build number
│   ├── macros.conf             # Environment, recipients, index, model list
│   ├── savedsearches.conf      # Collection, report export, weekly email, alerts
│   ├── transforms.conf         # Lookup definitions
│   └── data/ui/
│       ├── alerts/
│       │   └── email_report.html    # Alert action UI form
│       ├── nav/
│       │   └── default.xml          # App navigation
│       └── views/
│           ├── cim_validator.xml        # Main dashboard (Splunk Dashboard
Studio)
│           └── cat_documentation.xml    # In-app documentation
├── lookups/
│   ├── cim_model_categories.csv                 # Data model category
classifications
│   ├── cim_sourcetype_inventory.csv             # Master sourcetype reference
(enrichment only)
│   ├── cim_sourcetype_inventory.csv.version.csv  # Inventory changelog sidecar
(as-of dates)
│   ├── sourcetype_exclusions.csv                # Sourcetypes omitted from CIM-
compliance views
│   └── splunk_data_model_objects_fields.csv      # CIM field reference (Splunk
CIM v8.5.0)
```

```
├── appserver/static/
│   └── appIcon.png              # Alert action icon
├── metadata/
│   └── default.meta             # Permissions
├── static/                      # App icons and logos
├── LICENSE                      # Apache License 2.0
└── NOTICE                       # Third-party attribution notices
```

Top

License

Top

Support

Documentation: CAT ships a complete in-app **Documentation** tab; the same material is available as the CIM Assessment Toolkit User Guide (PDF).

Bug reports, feature requests, and feedback are welcome at support@machinedatainsights.com. CAT is built for experienced Splunk administrators; installation and configuration are covered in the User Guide and the in-app Documentation tab.

Commercial support, custom inventory engagements, and consulting on CIM normalization gap remediation are available from **Machine Data Insights Inc.**

Splunk platform compatibility: Splunk Enterprise 9.0 and later, and Splunk Cloud Platform. CAT passed Splunk Cloud vetting; its code paths are Cloud-safe (report data is written under `$SPLUNK_HOME/var/run/splunk/`).

Top

Acknowledgments

The **CIM Assessment Toolkit** was created by **James H. Baxter**, founder of **Machine Data Insights Inc.**

CAT evolved from the excellent Data Model Wrangler app by **Nick von Korff**, which introduced a structured approach to assessing CIM data model coverage in Splunk environments.

That app was itself inspired by the pioneering SA-cim_vladiator, developed by **Vladimir** and **Splunk Works** (GitHub repository), which introduced many of the original CIM validation concepts and scoring logic that remain foundational to this work.

We are grateful for the contributions of these authors — all the tools we rely on today are made possible by the work of those who came before us.

Top

Version History

v2.10.0 (June 28, 2026)

- **Sourcetype exclusions split into their own lookup** -- CIM-compliance exclusions moved out of `cim_sourcetype_inventory` into a dedicated, user-editable `cim_sourcetype_exclusions` lookup (`sourcetype_exclusions.csv`), seeded with the Splunk/Cribl internal plumbing sourcetypes. This keeps the inventory a clean, enrichment-only master catalog that doubles as a general-purpose CIM sourcetype reference, while exclusions become a short list the user owns.
- **Flexible, forgiving exclude semantics** -- the `exclude` column is case-insensitive and accepts `y`/`yes`/`t`/`true`/`1` (exclude) or `n`/`no`/`f`/`false`/`0` (keep); a blank `exclude` -- or a bare `sourcetype` row with no other columns -- is treated as excluded, so a plain list of sourcetypes is enough to exclude them.
- **Fault-tolerant by design** -- every consumer (dashboard, coverage collection, `email_report.py`, and `Export-CIMReport.ps1`) reads exclusions through a graceful subsearch / guarded file read. If the exclusions file is missing, unreadable, or redirected to a deleted path, nothing is excluded and the rest of CAT keeps working -- no broken panels.
- **Upgrade-safe custom exclusions** -- editing the shipped `sourcetype_exclusions.csv` in place works but is overwritten on upgrade; uploading your list under a new filename and repointing the `cim_sourcetype_exclusions` lookup definition keeps both the file and the override (in `local/transforms.conf`) across upgrades.
- **Inventory slimmed to enrichment-only columns** -- `sourcetype, tech_category, scope, security_classification, security_relevance, data_models, vendor, description, expanded_desc`. Dropped `exclude`, `exclude_reason`, `reviewed_by`, `reviewed_date`, and `provenance`.
- **New `tech_category` column** -- a single canonical technology family per sourcetype from a controlled vocabulary (`app`, `aws`, `azure`, `gcp`, `linux`, `network`, `security`, `windows`), identifying the index technology type a sourcetype's data belongs to. Surfaced as a Tech Category column on the dashboard Unmapped Data tab and the report's Unmapped Sourcetypes table, and lays groundwork for index-type-based CIM macro administration where new workloads of a given technology type are picked up automatically.

- **Provenance removed** -- the uniformly-`human_curated` `provenance` field and its Provenance column were removed from the inventory, the Unmapped Data tab, and the report's Unmapped Sourcetypes table.
- **Removed the `cim_validator_custom_model_generator` utility** -- a non-functional placeholder saved search that generated lookup CSV rows for custom data models. Deleted from the Reports tab and the in-app documentation.
- **Schedule adjustments** -- moved the scheduled report to Monday 8:00 AM (from 4:00 AM) and the `cim_validator_coverage_collection` search to 4:17 AM (from 1:47 AM), so the overnight collection searches have ample time to finish on large environments before coverage and reporting run.

v2.9.0 (May 29, 2026)

- New **Data Model Test** tab on the main dashboard for validating CIM normalization at the end of a TA deployment effort. Select a data model, dataset, one or more sourcetypes, and a field tier (Required / Recommended / Optional / combinations); the tab generates and runs the equivalent `| from datamodel:Model.Dataset | search sourcetype IN (...) | table sourcetype <tier-filtered fields>` search so you do not have to remember the `| from datamodel` syntax or look up which fields belong to each dataset.
- Three result panels: a copy-pasteable Generated SPL preview, the actual `| from datamodel` results (capped at 1,000 events), and a Field Fill Rate panel (sampled to 10,000 events) showing percent non-null per field per sourcetype with tier coloring. A CIM Field Reference panel underneath shows the tier, description, and prescribed values for every field in the selected dataset.
- Default time range is **Last 60 minutes** with an in-tab banner reminding users that the relevant index(es) and sourcetype(s) must be included in the `cim_<model>_indexes` macro for results to appear, and that wide time windows on unaccelerated models are expensive.
- Sourcetype picker is populated from `| tstats values(sourcetype) where cim_<model>_indexes` (not the inventory), so a freshly-deployed TA whose sourcetypes are not yet in the inventory still appears.
- Field reference comes from the existing `splunk_data_model_objects_fields.csv` lookup -- no new lookups, refresh scripts, or REST dependencies. The tab has its own dedicated inputs that are independent of the global dashboard filters.
- **Setup page UX** -- replaced the credential-required setup with a "Save and continue" affordance. The SMTP password field is now explicitly optional; clicking Save (with or without a password) stores the credential if provided and sets `is_configured = 1`, sending the user straight to the dashboard. Resolves the AppInspect `check_that_setup_has_not_been_performed` failure without changing the no-setup-needed UX.
- **App icons** -- re-encoded as standards-compliant PNGs at Splunkbase dimensions (36x36 / 72x72 for `appIcon`, 160x40 / 320x80 for `appLogo`) so the CAT icon displays correctly in Splunk Web's Apps menu, on the launcher tile, and in Windows Explorer / iOS thumbnails. Previous icons were JPEG masquerading as PNG and were silently rejected by newer Splunk versions.
- **Data Model Test column widths** -- tuned per-field widths on the Results table (Time 170, sourcetype 260, CIM fields 200) so long headers like `http_user_agent_length` render without truncation while short values like `allowed` or `280` don't waste viewport.
- **Splunk Cloud-safe code paths** -- redirected `email_report.py` data writes from `bin/report_data/` to `$SPLUNK_HOME/var/run/splunk/CIM_Assessment_Toolkit/` (Splunk Cloud mounts `bin/` read-only). Pinned `python.required = 3.9` in `alert_actions.conf` for broad version compatibility.

Foundation for Splunk Cloud certification; pending Cloud trial validation before the manifest declares Cloud support.

v2.8.4 (May 18, 2026)

- Sourcetype inventory schema update -- dropped the unused `events` column and added a `provenance` column (`ta_derived`, `human_curated`, `client_engagement`) recording how each inventory row was sourced. `client_engagement` rows are remapped to `human_curated` in the shipped lookup.
- Provenance surfaced -- new Provenance column on the dashboard Unmapped Data tab and in the report's Unmapped Sourcetypes table, with readable labels (TA-derived, Human-curated, Client engagement).
- Inventory changelog sidecar -- ships `cim_sourcetype_inventory.csv.version.csv` next to the lookup (columns `last_updated,updated_by,note,base_catalog_last_updated`); the report prints `last_updated`, `updated_by`, `note`, and `base_catalog_last_updated` as an "as of" line in the Data Source Mapping Status section. `base_catalog_last_updated` records the standard catalog version a custom inventory was derived from, so a client's customized copy and the base catalog it built on are tracked together.
- New `cim_sourcetype_inventory_version` lookup definition -- pairs the changelog sidecar with a `transforms.conf` definition so it is a recognized lookup (AppInspect) and queryable via `| inputlookup cim_sourcetype_inventory_version`.
- Case-insensitive `exclude` -- the inventory `exclude` column now accepts `y`/`yes`/`n`/`no` in any case (previously only an exact uppercase `Y` excluded a row), applied consistently across the dashboard and the report.
- Documented how to load a supplied custom or updated inventory via Splunk Web (Lookup table files and Lookup definitions).

v2.8.3 (May 11, 2026)

- Refreshed CIM field reference to Splunk CIM v8.5.0 -- `splunk_data_model_objects_fields.csv` regenerated from CIM v8.5.0 data model definitions (1,474 field rows, up from 1,392).
- Versionless lookup filename -- renamed `splunk_data_model_objects_fields_604.csv` to `splunk_data_model_objects_fields.csv`. The lookup stanza in `transforms.conf` no longer needs editing when refreshing the CIM reference; admins can drop an updated CSV of the same name into `$SPLUNK_HOME/etc/apps/CIM_Assessment_Toolkit/lookups/` and reload the lookups without redeploying the app. Future CIM-only refreshes can be shipped as a standalone CSV from the project `docs/` folder rather than requiring a full app release.

v2.8.2 (April 29, 2026)

- Fixed CIM Coverage scorecard showing the mapped sourcetype count instead of the coverage percentage. The `ds_sv_cim_coverage` query produced three columns (`mapped`, `total`, `CIM Coverage`); Dashboard Studio's singlevalue viz defaults to the first numeric column, so it displayed `mapped`. Appended `| fields "CIM Coverage"` so only the intended column remains.

v2.8.1 (April 22, 2026)

- Fixed Excluded scorecard error on the Unmapped Data tab -- `ds_unmapped_excluded_count` had `"type": "chain"` which Dashboard Studio does not recognize ("Data source ..., of type chain, cannot

be a chain data source"). Corrected to `"type": "ds.chain"` to match its sibling count datasources.
- Fixed multi-recipient email parsing -- `email_report.py` now strips quotes before splitting the `cim_report_recipients` macro value, so both `"a@x.com,b@y.com"` and `"a@x.com","b@y.com"` formats are accepted. Previously the quoted-per-address form left embedded quotes in each parsed recipient and only the first address received mail.

v2.8.0 (April 14, 2026)

- CIM Coverage KPI -- new 5th scorecard on the CIM Compliance tab and in the CIM Assessment Report, showing what percentage of active sourcetypes are mapped to CIM data models. Scope-aware: when a scope is selected, both the mapped models (via `cim_model_categories`) and the sourcetype denominator (via `cim_sourcetype_inventory` scope) are filtered accordingly. Uses live tstats against actual indexes (not summary index). Excluded sourcetypes (exclude=Y) do not count against the score. Daily CIM Coverage snapshots are collected per scope into the summary index by `cim_validator_coverage_collection` (one row per scope plus an "all" row) and displayed as a trend chart alongside the existing quality trend charts. Both the dashboard trend chart and the report's Compliance Trends section filter by the selected scope when reading historical coverage.
- Removed `cim_smtp_password` macro -- the clear-text SMTP password macro has been removed entirely. SMTP passwords must now be stored in Splunk's encrypted credential store via the setup page (Apps -> CIM Assessment Toolkit -> Set up). The `--smtp-password` CLI argument remains available as an override.
- Setup page no longer gates first launch -- `is_configured = 1` ships as default, so users go straight to the dashboard. SMTP credential setup is available any time via Settings -> Apps -> Manage Apps -> CIM Assessment Toolkit -> Set up.
- Removed the quality scores disclaimer note beneath the KPI scorecards -- CIM Coverage makes the gap self-evident.

v2.7.6 (April 13, 2026)

- Secure SMTP password storage -- `email_report.py` now reads SMTP credentials from Splunk's built-in credential store (`/services/storage/passwords`, realm=CIM_Assessment_Toolkit, username=smtp_password). Password priority chain: `--smtp-password` CLI argument, then credential store.
- Setup page -- new first-run setup page (Apps → CIM Assessment Toolkit → Set up) for storing the SMTP credential. Handles both initial creation and password updates. Sets `is_configured` automatically on save.

v2.7.5 (April 3, 2026)

- Splunk AppInspect validated -- all checks pass. Added `[package]`, `[id]` stanzas to app.conf, `python.version` to alert_actions.conf, set `is_configured = 0`.
- Report formatting -- scope moved from title suffix to its own line on the cover page. Reduced cover page top whitespace. Added note in Executive Summary clarifying quality scores reflect mapped sources only.
- Report table sort orders -- all tables sort by Data Model/Dataset/sourcetype. Remediation filtered to sources needing attention (`Priority > 0`). Field gaps deduplicated across indexes.
- Mapped sourcetypes table -- restructured to match unmapped format: added Vendor, Relevance, Scope columns; removed Models count.

- Remediation table -- added sourcetype column after Dataset for granular identification.
- Dashboard -- KPI scorecards use analysis base (fixes timezone-dependent N/A/0% issue). Unmapped/mapped tabs respond to index/sourcetype filters. Added quality scores context note.
- Removed Node.js dependency -- deleted `generate_report.js`. `Export-CIMReport.ps1` and `export_report.sh` now call `generate_report.py` directly.
- Renamed `cim_validator_weekly_report` to `cim_validator_scheduled_report` (default 4:00 AM Monday).
- Fixed `cim_validator_custom_model_generator` saved search.
- Sourcetype inventory expanded to 750 entries.
- Dynamic scope dropdown -- dashboard scope filter now auto-populates from categories in `cim_model_categories.csv`. Custom scopes (application, IoT, pci, etc.) appear automatically.
- Collection search time range widened from `-4h@h/-3h@h` to `-24h@h/-1h@h` to prevent sourcetypes with infrequent events from appearing unmapped.
- README rewritten to reflect current project state.

v2.7.4 (March 30, 2026)

- Python report generator -- replaced Node.js `generate_report.js` with Python `generate_report.py` using only the standard library (zipfile + XML). Eliminates Node.js dependency entirely. `email_report.py` now imports the generator directly instead of calling Node via subprocess. No external packages required -- zero-install deployment.

v2.7.3 (March 26, 2026)

- Custom alert action — replaced generic "Run a script" with a proper Splunk custom alert action (`email_report`). Eliminates the Windows MAX_PATH issue that prevented script execution. Splunk passes session token via JSON on stdin instead of long command-line arguments. Removed `bin/scripts/` wrapper.
- `parse_known_args` — main script now silently ignores unexpected positional arguments from Splunk.

v2.7.2 (March 26, 2026)

- SMTP fix -- added `ehlo()` before `starttls()`, auto-detect TLS on port 587, diagnostic output for SMTP settings. Encrypted password detection warns and falls back. New `--smtp-password` argument for manual override.

v2.7.1 (March 25, 2026)

- Inventory file resolution — `email_report.py` now reads the actual filename from `transforms.conf` (checking `local/` first, then `default/`), so renaming the lookup via Splunk's UI works correctly for both the dashboard and the report.

v2.7.0 (March 25, 2026)

- Scope-aware unmapped filtering — when scope is "security", the unmapped section in both the dashboard and report shows only sourcetypes classified as security-relevant in the inventory. Sorted by relevance (high → med → low) then by event count.
- Dashboard unmapped tab scope integration — unmapped table and scorecard counts filter by inventory scope when a scope is selected.

v2.6.4 (March 25, 2026)

- Remediation chart redesign — single-series Priority Score bar chart (yellow), taller layout (700px) for readable sourcetype labels. Cap on `overall_quality_pct` added to base query to fix historical >100% values.

v2.6.3 (March 25, 2026)

- Remediation bar chart polish — blue for quality %, yellow for priority score. Capped display at 100%. Labels show sourcetype names.

v2.6.2 (March 25, 2026)

- Remediation Priorities visualization — replaced non-functional `splunk.scatter` with horizontal bar chart showing Top 20 Remediation Targets sorted by priority score.

v2.6.1 (March 25, 2026)

- Quality Matrix scatter chart fix — restructured datasource column order for `splunk.scatter` (superseded by v2.7.3).

v2.6.0 (March 25, 2026)

- Sourcetype Inventory — new `cim_sourcetype_inventory` lookup replaces `cim_unmapped_exclusions` as the master reference for all known sourcetypes. Ships with 311 common industry sourcetypes classified by vendor, security relevance, scope, and potential CIM data models.
- Enriched Unmapped Data — dashboard and report unmapped tables now show Vendor, Relevance, and Scope columns from the inventory lookup.
- Exclusion via inventory — sourcetypes with `exclude=Y` in the inventory are filtered from unmapped counts and tables. 15 Splunk-internal sourcetypes pre-excluded.

v2.5.0 (March 19, 2026)

- Unmapped Sourcetype Exclusions — new `cim_unmapped_exclusions` lookup for sourcetypes reviewed and determined to not need CIM mapping. Ships with 15 common Splunk-internal sourcetypes pre-excluded.
- Dashboard Excluded scorecard — Unmapped Data tab now shows Total, Mapped, Unmapped (active), and Excluded counts. Unmapped table filters out excluded sourcetypes.
- Report exclusion context — unmapped section and Key Findings show excluded count for audit transparency.
- Apache License 2.0 — LICENSE file, copyright headers on all source files, and Acknowledgments section in README.

v2.4.3 (March 2026)

- Unmapped/mapped export rewrite — replaced failing metadata/search index=* chain with async job approach. Uses `| tstats` via job create/poll/fetch which works for all search types including generating commands. Single query produces both unmapped and mapped CSVs. Falls back to summary index if async fails.

v2.4.2 (March 2026)

- Unmapped sourcetypes report fix — added `search index=*` fallback when `| metadata` fails on the REST export endpoint. Three-tier fallback: metadata → search index=* → summary index.

v2.4.1 (March 2026)

- Coverage cap fix — `percent_coverage` capped at 100% in collection search, dashboard, and report queries. Multi-valued fields in accelerated data models (e.g., DNS) could inflate `tstats` counts beyond event totals, producing >100% Data Quality scores.
- Bar chart height increased to accommodate 15+ data models without clipping Y-axis labels

v2.4.0 (March 2026)

- Data Model Category Scoping — new `cim_model_categories` lookup classifies models as security, operational, or both
- Scoped reports — `cim_report_scope` macro filters all report sections by category (security, operational, all, or custom)
- Report title, header, filename, and executive summary reflect selected scope
- Dashboard Mapped Sourcetypes table shows Category column
- Custom categories supported — add any tag to the lookup for client-specific scoping (pci, hipaa, etc.)

v2.3.0 (March 2026)

- Unmapped/Mapped sourcetype consolidation — one row per sourcetype with comma-separated indexes on the dashboard
- Unmapped detection fix — join on sourcetype instead of index+sourcetype, matching how CIM normalization works
- Compliance Trends time window fix — widened search window to reliably capture data regardless of collection timing
- Configurable trend period — `cim_report_trend_days` macro (default 7) controls comparison window

v2.2.1 (March 2026)

- Acceleration Retention bug fix — `tonumber()` wrapping for Data Model Health tab
- Dashboard tab renamed from "CIM Validator" to "CIM Compliance"

v2.1.1 (March 2026)

- Compliance Trends — configurable comparison table in the report with improvement indicators (▲/▼/—). Default 7 days, adjustable via `cim_report_trend_days` macro.
- Report CSV persistence — export data retained in `bin/report_data/` for manual investigation
- Acceleration Retention fix — sourced from data model configuration instead of missing summarization field
- Air-gapped deployment — docx package bundled; Node.js local install support in all scripts
- Report column layout — fixed header wrapping in Compliance by Data Source and Remediation Priorities
- How to Read This Report — cover page guide with section descriptions

v2.1.0 (March 2026)

- Prescribed Value Validation — validates field values against CIM-defined expected values
- Data Model Health tab — acceleration status, completion %, time range, and errors
- CIM Field Reference — expected values shown in field detail table
- Custom Data Model Support — utility search to auto-generate lookup entries
- Automated Report Generation — Word document with KPI scorecards, compliance tables, field gaps
- Scheduled Email Distribution — Splunk-native weekly delivery using configured SMTP settings
- Cross-Platform Report Scripts — Python, PowerShell, and Bash wrappers

v2.0.0 (July 2025)

- Dataset-level CIM assessment with improved accuracy over data-model-only approach
- Optimized search logic — single `inputlookup` → `map` → `tstats` pipeline
- Unmapped Data Discovery — identifies sourcetypes not mapped to CIM
- Remediation Priority Matrix — ranks by impact (low quality × high volume)
- Modern Dashboard Studio UI

v1.0.0 (Original)

- Initial adaptation from Data Model Wrangler with dataset-level validation concept

Top

CIM Assessment Toolkit v2.10.0 — Machine Data Insights Inc. "There's Gold In That Data!"®