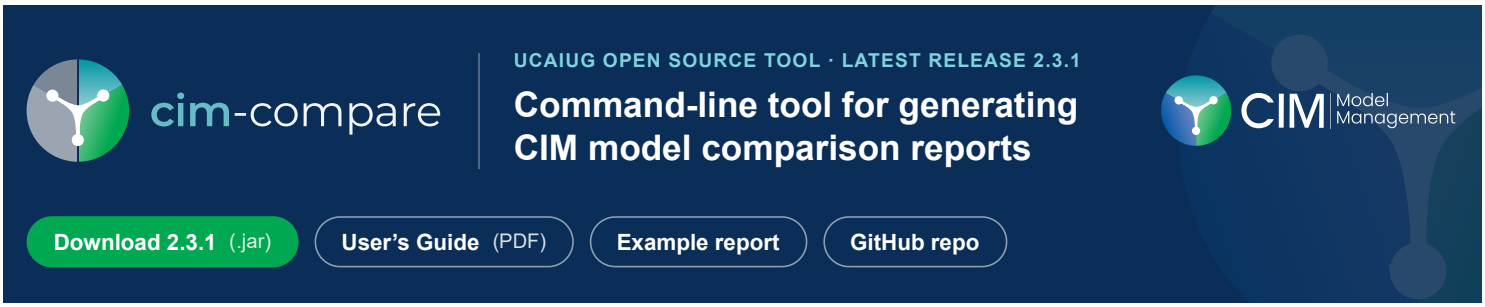


cim-compare is an open-source command-line tool from the UCA International Users Group (UCAIug) that generates readable comparison reports between two versions of a CIM UML model.

- **The need:** Enterprise Architect can compare two models, but only exports the result as XML that is hard to read. As the CIM evolves release by release, model managers, standards editors and implementers need a clear record of what changed.
- **What it does:** given a "baseline" and a "destination" model (EA `.eap` / `.qea` project files, XMI 1.1 exports, or an EA model comparison log), it writes a single, self-contained HTML report of every added, deleted, moved and changed package, class and diagram. The report has search, filters, word-level redlines and highlighted diagram changes, and needs no server or internet connection.
- **Who uses it:** the UCA TF13 model managers use it to publish the changes in each CIM release, and anyone tracking the CIM can use it to see what a new release means for their profiles, interfaces or products.

See an example report, or download this guide as a PDF.

Table of Contents

- Home
- Latest Release
- Overview: CIM Model Comparison Reports
- Reading a Comparison Report
  - Report Layout
  - Status Colors
  - Class Details
  - Diagrams
  - Finding Things
  - Keyboard Shortcuts
  - Printing and PDF
  - Supported Browsers
- Setup & Configuration
  - Java Technical Requirements
    - Java 24 and Later
  - Azul OpenJDK
    - Download 32-bit Java 17
    - Download 64-bit Java 17
  - Important Setup Considerations
  - Warning: Potential Configuration Pitfalls
- Command Line Usage
  - Option #1: Enterprise Architect .EAP or .QEA Project Files as Input

    - Option #2: XMI Baseline and Destination Models as Input
    - Option #3: EA Model Comparison Logs as Input
    - Output Files
- Enterprise Architect XMI Export Procedures
- License
    - Third-Party Software

Latest Release

2.3.1

- For release notes describing the latest features and/or fixes visit latest release.
- To directly download the latest release of the cim-compare command-line application click cim-compare-2.3.1.jar.

▼ What's New in 2.3.1

- **One set of rules for notes, ⓘ pop-ups and the hover / inline setting** (#199). Notes are headed as in Enterprise Architect ("Class notes", "Attribute notes", "Package notes"), and every ⓘ pop-up lists the notes, then the metadata, then the tagged values. In **hover** mode the page shows the changes and the ⓘ holds the notes, metadata and tagged values; in **inline** mode they are on the page, and each attribute's details open with an arrow (▸) or **show details**. A printout reads as inline mode. Added and deleted classes are laid out like changed ones, and the side that does not exist is hatched everywhere. With `--full`, every ⓘ lists all of the item's properties. See Class Details.
- **The report uses the full width of the window** (#219): on a wide monitor, the package tree, tables and diagram images widen with the window instead of stopping at 1,880 px.
- **Both diagram images side by side in the enlarged view** (#225): clicking either image of a changed diagram opens the baseline and destination together; zooming and moving act on both, and the diagram's highlight key is shown across the top.
- **With `--full`, unchanged diagrams show their image** (#204): one image, centred in a full-width panel, since the baseline and destination are the same picture. With EA project files, EA draws it from the destination model, so a `--full` run with `--include-diagrams` takes longer than before. See Diagrams.
- **Fix:** notes showed characters such as `>` and `&` as codes, for example `(>= 0)` (#198). They now show as the characters; formatting tags such as `<i>` are still shown as written.
- **Fix:** notes lost their line breaks, so separate lines and lists ran together (#200). Line breaks are now kept in the comparison and shown in the report.
- **Fix:** the dot that marks a change shown only in an ⓘ pop-up sat outside the ⓘ (#196). It now overlaps the ⓘ's ring.
- **Fix:** with an EA compare log as input, the report left out the compared package's own classes, diagrams and changes, and its header named a sub-package (#188). The report now starts at the compared package.

▼ What's New in 2.3.0

- **Tagged values are compared** (#169): user-defined tags (such as `baseuri` and `baseprefix`) on packages, classes, attributes and associations, which earlier releases never compared. They are recorded in the comparison XML as EA's Compare Utility records them, and shown in the report; an element whose only change is a tag is reported as changed. See Class Details.
- **The summary counts stay in view** (#175) while you scroll; **Off · Counts · Counts + legend** at the right of the counts sets what stays, and `p` cycles it. See Report Layout.
- **Fix:** diagram highlight outlines stopped partway down elements that EA draws taller than their stored size, for example a class whose added stereotypes push its name down (#165). Each outline now covers the element as EA

drew it, in height as well as width.

- **Fix:** a link end's heading, and the heading of an attribute's ⓘ metadata, showed the name in capitals, e.g. "SOURCE END — BASEVOLTAGE" (#172). The name now keeps its own case.
- **Fix:** metadata tables redlined a changed name or alias, though the old value is shown beside it (#179). Tables now redline only free text (notes); a new name is shown as plain text, as for a renamed attribute.

▼ What's New in 2.2.0

- **SVG diagram images** (#139). `--image-type=SVG` gives diagrams that stay sharp at any zoom and in print, with their text kept as text. They need EA 16.1 build 1629 or later on the machine that generates the report (whoever opens the report needs only a browser); with an older EA the report uses JPG and its header says so. See Diagrams.
- **Only the diagrams the report shows are drawn** (#141). With EA project files and `--include-diagrams`, the models are exported without images and EA then draws only the changed diagrams, which shortens long runs and avoids EA running out of memory. A diagram is changed when its picture differs, including when a class or association drawn on it changed; an association whose role names, multiplicities or stereotype changed is listed as **labels changed**.
- **Zoom and pan in the enlarged view** (#153): mouse wheel, pinch, drag, a toolbar and the `+` `−` `0` keys.
- **Search by GUID** (#133), in any of the forms EA, XMI files and scripts use. See Finding Things.
- **A Print / PDF button** (#131) with the settings for a good PDF, and printing that keeps the colors and the header (#128). See Printing and PDF.
- **Empty housekeeping values are left out** (#126): a value against an empty value in Complexity, Phase, Version, GenType and similar fields is not reported unless `--show-empty-values` is given; a line under the legend says how many were left out.
- **Hidden attribute changes are pointed out** (#123): *metadata changed - hover ⓘ* on a row whose change is only in its ⓘ metadata.
- **Fixes:** every stereotype is read and compared, not only the first (#118); Const is read from EA's `changeable` (#122); the legend's ⓘ is drawn at the right size (#137); the comparison XML is written in UTF-8 on every Java version (#143); a moved package says where it came from (#147); a filtered package is not shown as empty (#148); the enlarged view's outlines line up in windows with a scrollbar (#151).

▼ What's New in 2.1.1

- **A value that was cleared or filled in is reported as changed** (#107). When a property of an element that exists in both models has a value on one side only, the row is now blue (changed) instead of red (removed) or green (added), and the empty side shows *(empty)* in light grey. EA leaves an empty value out of its export, so such a property appeared on one side only. Red and green property rows now appear only for elements that were themselves deleted or added.
- **A multiplicity written another way is tagged "notation only"** (#109). An association-end multiplicity whose notation changed but whose meaning did not, such as `1..1` and `1`, or `0..*` and `*`, is still reported as a change, but is tagged **notation only** in the end's table and in the association's signature line.
- **Fix:** in signature lines, a changed multiplicity or role ran the struck old value into the inserted new one, so `0..1` → `1` read as `[0..11]` (#110). A small gap now separates them.

▼ What's New in 2.1.0

- **Every root node of an EA project is compared** (#88). Without `--package`, earlier releases compared only a project's first root node and left the others out of the report. Now each root node is exported and compared, so a project that keeps extension packages in root nodes of their own beside the CIM (for example `Model`, `UtilityExtensions` and `VendorExtensions`) is reported in full. Each root node appears as a top-level package in the report and the package

outline, and a reference from one root node to another, such as an extension class that specializes a CIM class, is reported like any other. `--package=Model` compares a single root node as before. See Option #1.

- **`--package` takes several packages** (#94), separated by commas, for example `--package=Grid,UtilityExtensions` or, when a name contains a space, `--package="Grid,European Standards"`. The report shows them in the order given, and references between them are reported like any others. A package found in only one of the two models, such as a new extension root, is now reported as added or deleted, also for a single `--package`; before, **cim-compare** stopped with an error. A name found in neither model still stops it. See Option #1.
- **Mixed `.eap` / `.qea` comparisons report only real changes** (#86). EA leaves an attribute's AllowDuplicates and Scale blank in an `.eap` project and stores `false` and `0` in a `.qea`, so comparing one of each reported every such attribute, and its class, as changed. A blank value now counts as the default.
- **The report header links to the website and the release** (#87): the logo links to cim-compare.ucaiug.io, and the "cim-compare 2.1.0" line to the GitHub release that produced the report.
- **Fix:** with `--include-diagrams`, the baseline's diagram images could end up in `Images-destination`, and the report then showed the destination image on both sides of a changed diagram (#91). This happened when Windows could not rename the folder EA exported the images to while EA still had the project open. The images are now moved file by file when the folder cannot be renamed, and **cim-compare** stops with an error if they cannot be moved.

▼ What's New in 2.0.1

- **Runs on Java 24 and later without extra settings** (#65). Java 24 lowered its XML entity size limits, and a large comparison stopped with `JAXP00010003` before the report was written; **cim-compare** now removes those limits itself. Its jar also lets EA's Java API load its native library without Java's "restricted method" warning. See Java 24 and Later.
- **An attribute's ⓘ metadata shows its new name as it is** (#67): the Name row of a renamed attribute shows the new name as plain text in the Destination column, like the old name in the Baseline column, instead of a redline.
- **The summary counts filter the report** (#78). Clicking a count, such as **Classes › 9 added**, now shows just those items with their packages opened; before, the matches stayed inside collapsed packages, the counts disappeared, and Classes and Diagrams counts affected each other.
- **The cim-compare logo in the report header.** The top left of the header shows the cim-compare logo, linked to this repository, and in the light theme the header is now black. The logo is part of the report file, so the report still needs no network access.

▼ What's New in 2.0.0

- **A redesigned HTML report** (#38): a package outline, search, status filters, links between classes, GUIDs, word-level redlines of changed descriptions, and keyboard shortcuts. Reports are much smaller: the example report is under 1 MB, down from 5.9 MB. See Reading a Comparison Report.
- **Diagram differences are highlighted** (#37): with `--include-diagrams`, added, removed, changed, moved and restyled elements are outlined on the diagram images, and changed connectors are listed under them. Every diagram that changed is listed, including one where only the layout or a connector changed. See Diagrams.
- **Only changes are reported by default** (#42). The new `--full` option includes identical items as well. `--minimal` is still accepted but no longer needed.
- **`--package` works across renamed packages** (#49): for example `--package=Grid` when comparing CIM17 (where the package is `IEC61970`) with CIM18. If the package can't be found, **cim-compare** now stops with an error instead of comparing the whole model.
- **Fixes:** reports are no longer empty when the model's root package isn't named "Model" (#43); `--cleanup` no longer deletes your own input files (#53).
- **A new output file**, `<name>-enriched.xml`, written beside the report. See Output Files.

Releases before 2.0.0 (1.3.0 and earlier) are described on the GitHub Releases page.

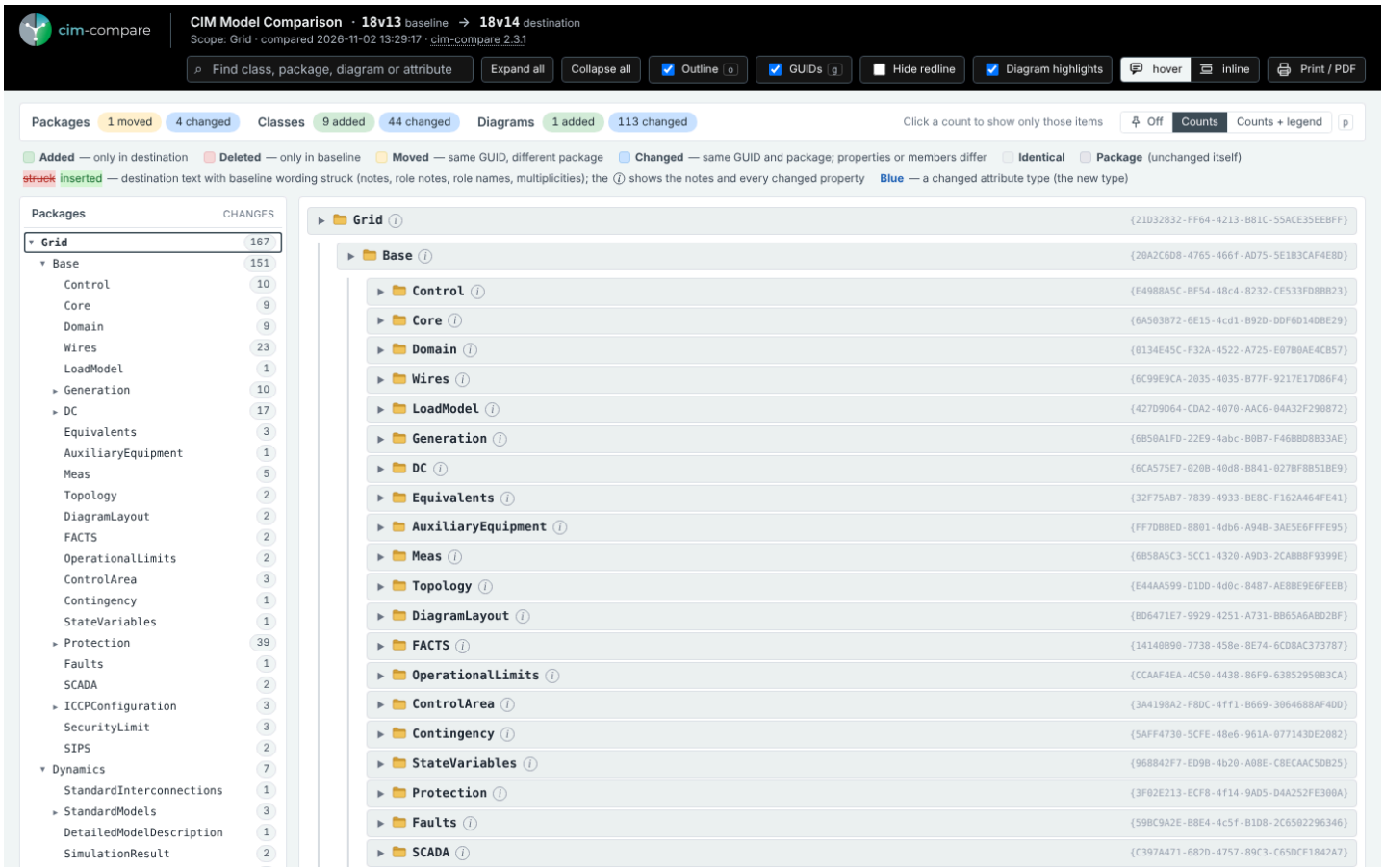
Overview: CIM Model Comparison Reports

cim-compare generates model comparison reports between two points in time.

The term "destination" is used to describe some current version (or latest working copy) of a CIM model that is to be the target of the comparison. The term "baseline" is used to describe the **historical** model against which the "destination" model is to be compared to determine what has changed.

Enterprise Architect (Corporate Edition and above) has the capability to perform these model comparisons; however, the feature is not generally useful as results are only exportable as XML files that are not easily human readable.

The output generated by **cim-compare** is a single, self-contained HTML file covering all changes between a "baseline" and "destination" model. It needs no web server and no internet connection: open it from disk in any current browser (see Supported Browsers). By default the report shows only what changed; `--full` adds the unchanged items.



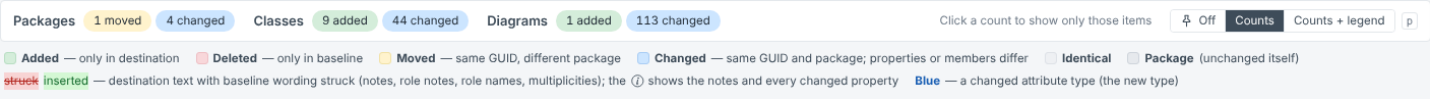
Reading a Comparison Report

Report Layout

- **Header** (always visible at the top): the cim-compare logo (a link to the cim-compare website), the baseline and destination versions, the scope of the comparison (the package given with `--package`, or the model's root nodes and "full model"), when the comparison was made, the cim-compare version (a link to its release), and the controls described below. At the right, **Print / PDF** opens the browser's print dialog (see Printing and PDF).

- **Summary**: counts of added, deleted, moved and changed packages, classes and diagrams. Click a count to show only those items (for example **Classes › 9 added** shows just the 9 added classes, with their packages opened and every other package hidden); click more than one to combine them, and **clear filters** to show everything again. The package outline on the left always lists the whole model. The counts stay under the header while you scroll; **Off · Counts · Counts + legend** at the right of the counts sets what stays in view (Counts by default), and the `p` key cycles it. The choice is remembered in the browser. Printing shows the counts and legend once, at the top.
- **Legend**: what each color means (see Status Colors). When differences between a value and an empty value were left out (see `--show-empty-values`), a line under the legend says how many; point at it for the fields concerned.
- **Package outline** (left): every package with changes, and the number of changed classes and diagrams in it, including its sub-packages. Click a package to jump to it; the package you are reading is highlighted as you scroll. Turn it off with the **Outline** checkbox or the `o` key. It is hidden in narrow windows and when printing.
- **Package tree** (right): packages, and within each package its classes and then its diagrams, grouped by status in the order Deleted, Added, Moved, Changed. Click a row to open or close it; **Expand all** and **Collapse all** open or close everything (in inline mode, Expand all also opens every attribute's details), and Alt-click on an arrow opens a package and everything in it.

Status Colors



Color	Status	Meaning
Green	**Added**	Only in the destination model.
Red	**Deleted**	Only in the baseline model.
Yellow	**Moved**	Same element (same GUID), in a different package. The row says where it came from: for a class or diagram, its baseline package; for a package, its baseline parent package (for example "MOVED from Part303").
Blue	**Changed**	Same element, same package; some of its properties, attributes or links differ.
Grey	**Identical**	Unchanged. Shown only with `--full`.

The status is also written on each row (ADDED, DELETED, MOVED, CHANGED), so the report can be read without relying on color. A renamed element shows its new name, with "renamed from *old name*" in grey beside it (in the tree) or under it (in tables). An element with several stereotypes shows them all, as «ShadowExtension, util»; a change only in their order or case is tagged **order only** or **case only**.

Class Details

Open a class to see what changed in it. In **inline** mode the page shows the sections below in this order; in **hover** mode (the default) the first three are in the class's ⓘ pop-up, and the page shows the attributes and links (see ⓘ hover / ⓘ inline below). Added and deleted classes are laid out the same way as changed ones, with the side that does not exist hatched.

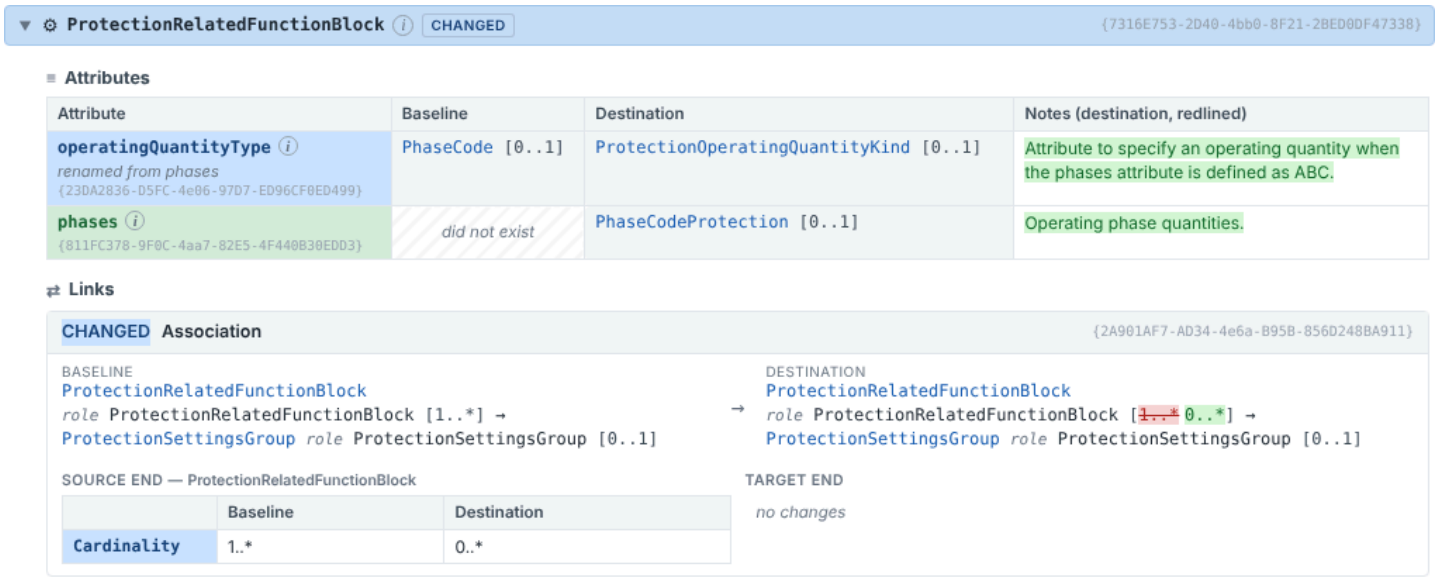
1. **Class notes** – the class's notes, as in Enterprise Architect. When they changed, the heading reads **Class notes (destination, redlined)** and the destination text is *redlined*: wording removed since the baseline is struck through in red and new wording is highlighted in green. When most of a text was rewritten, the whole old text is struck through and the whole new text inserted, which is easier to read than a word-by-word comparison. A deleted class shows **Class notes (baseline)**, and a class without notes shows *No notes defined in the model.*
2. **Metadata** – the class properties that changed (stereotype, abstract, alias and so on), baseline beside destination. A property whose value was cleared or filled in is shown as changed, with *(empty)* on the side that has no value. In the housekeeping fields EA fills in automatically (Complexity, Phase, Version, GenType, and Containment, IsLiteral, Derived,

Const and Static at their defaults), a difference between a value and an empty value is not shown unless the report was generated with `--show-empty-values`. A property that is empty in both models is not listed.

3. **Tagged values** – the class's user-defined tagged values (such as `baseuri` or `baseprefix`) that were added, removed or changed, each with its baseline and destination value. A tag on one side only shows *(no tag)* on the other. Packages show the same section at the top of their contents, and association cards show a **Tagged values** table under Link metadata, and under each end that has tags. An attribute's tags are listed in its ⓘ; a change only in its tags marks the row *metadata changed*. An element whose only change is a tagged value counts as changed.
4. **Attributes** – one row per changed attribute: its type and multiplicity in the baseline and in the destination, and its notes, redlined. In the Destination column a changed type is shown as the new type in blue, and a changed multiplicity is redlined. A renamed attribute shows its new name with "renamed from *old name*" under it. A new attribute shows *did not exist* in the baseline column; a deleted one shows *removed from model* in the destination column. When an attribute changed only in properties its row does not show, such as an enumeration literal given an initial value, in hover mode its ⓘ is blue with a dot and the row reads *metadata changed - hover ⓘ*; in inline mode the row reads *metadata changed*.
5. **Links** – generalizations, associations and aggregations, each as a card showing both ends in the baseline and in the destination, followed by the changed properties of the source end and the target end. Each end of an association reads *class role role name* [*multiplicity*], for example `ACDCTerminal` *role* `ACDCTerminal [0..1]`: the class (a link to it), then its role name and multiplicity. On the destination line changed role names and multiplicities are redlined. A multiplicity written differently with the same meaning, such as `1..1` and `1`, is tagged **notation only**. Long class and role names break between words.

In hover mode, a class whose only changes are in its ⓘ says so under its row, for example *Class 'Switch' has no other changes. The ⓘ shows its notes and every changed property.* A class with nothing to show says *Class 'Name' has no changes to its properties or links.*





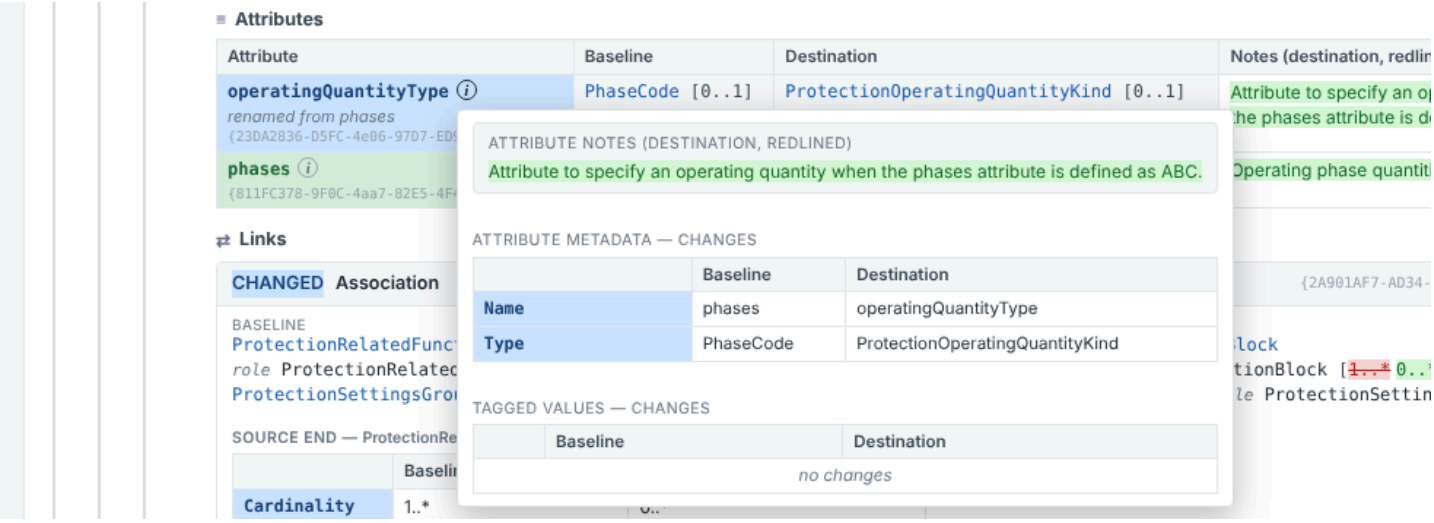
Other controls in the header:

- **ⓘ hover / ⓘ inline** – the ⓘ next to a package, class, attribute or diagram shows its notes and its changed properties, in the same order for every kind of item: the notes, then **‹Kind› metadata — changes**, then **Tagged values — changes** (none for diagrams). A table with no changed rows says *no changes*. In **hover** mode (the default) these appear in a pop-up when you point at the ⓘ; click it to keep the pop-up open. A blue ⓘ with a dot means its pop-up holds a change the page does not show. In **inline** mode they appear in the page instead: a package's, class's or diagram's sections are shown under its row, and each attribute has an arrow (▸) before its name and **show details** under it, which open its details (as does its ⓘ). Attribute details are closed until you open them; **Expand all** opens them all. Inline mode shows everything on the page, so no ⓘ there is blue or has a dot.
- **Hide redline** – shows the destination text without the struck and inserted markings.

Redlining marks changed notes and role notes, role names and multiplicities. A changed attribute type is shown as the new type, in blue, and in tables that show the baseline and destination values side by side, a changed name or other short value is shown as it is. A renamed element also shows "renamed from *old name*" next to its new name.

Notes are shown as they are written in Enterprise Architect, with their line breaks. Characters EA stores as codes, such as `>` and `&`, are shown as the characters (`>` , `&`). Formatting tags (for example `<i>` , `<sub>` , `<li>`) are shown as written; they are not rendered.

- **GUIDs** – each package, class, attribute and link shows its GUID, in grey at the right. Turn them off with the checkbox or the `g` key. The GUID shows when an element was renamed rather than replaced: a renamed element keeps its GUID.

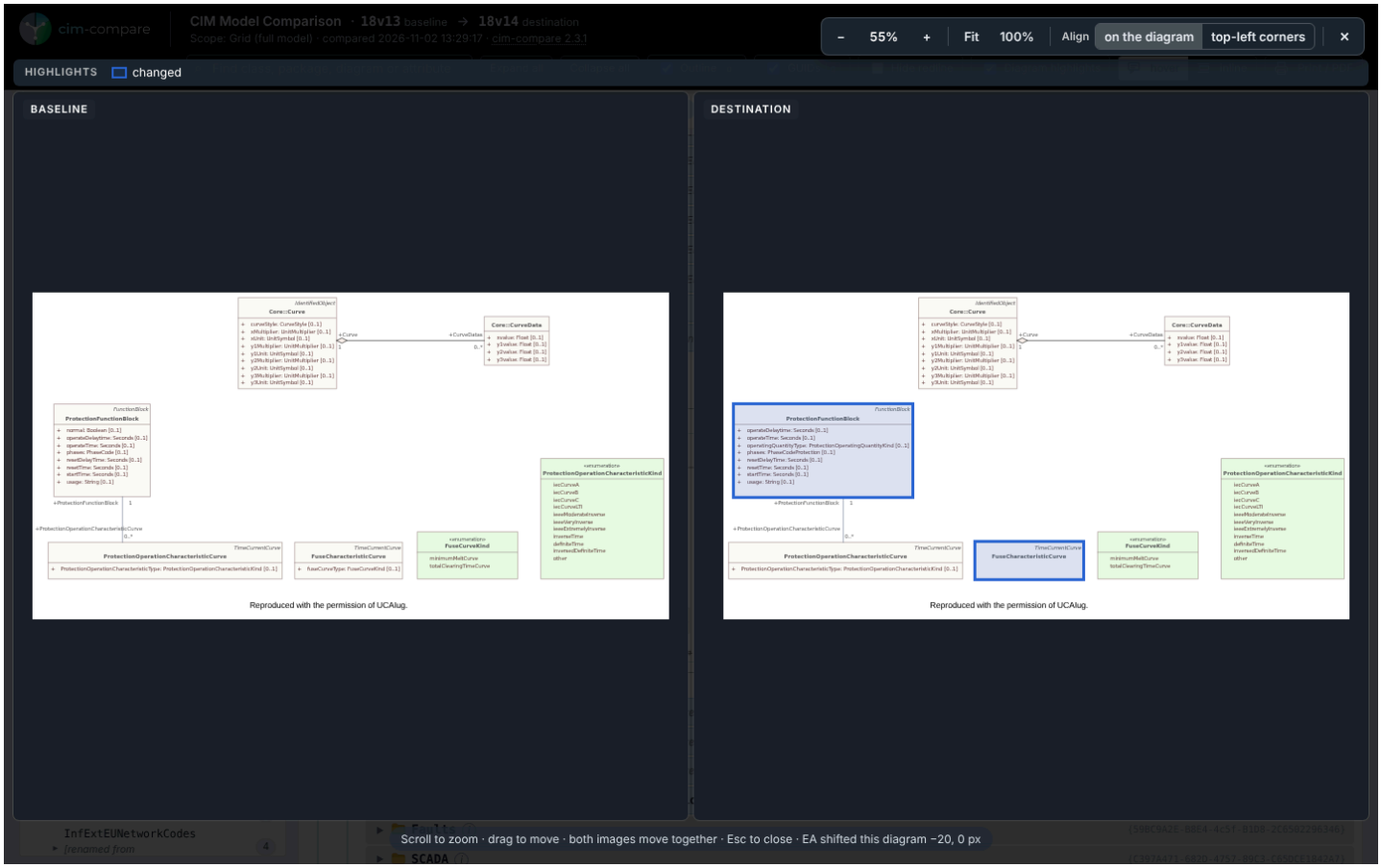


With `--full`, every ⓘ lists all of the item's properties, not only the changed ones, and its headings drop "— changes". A changed item shows the baseline beside the destination, with the changed rows tinted; an unchanged item shows its values in one **Value** column, and in hover mode says, for example, *Class 'Name' is unchanged. The ⓘ shows its notes and metadata.* A table with nothing in it says *none*. Unchanged links show the values of their ends the same way.

Diagrams

With `--include-diagrams`, each changed diagram shows its changed properties and, when something on the diagram changed, the baseline and destination images side by side. A diagram counts as changed when its picture differs: something on it was added, removed, moved, resized or restyled, or a class or association drawn on it changed in a way the diagram shows. Click an image to enlarge it.

In the enlarged view, a diagram with a baseline and a destination image shows the two side by side, at the same scale. The mouse wheel or a trackpad pinch zooms on the point under the cursor, dragging moves the diagrams, and a two-finger pinch zooms on a touch screen; both images zoom and move together, so the same part of the diagram stays in view on each side. When EA shifted the whole diagram between the two versions, the images are lined up so an element on both is at the same place in each pane, and the hint at the bottom says by how much; **Align** switches between that and lining the images up at their top-left corners. A diagram with one image (added, deleted, or unchanged with `--full`) is enlarged on its own. The toolbar has zoom out, zoom in, **Fit** and **100%**; the keys `+`, `-` and `0` do the same, and `Esc` closes the view. A strip beneath the toolbar shows the diagram's highlight key. The outlines follow the diagram at every zoom. SVG images zoom to 800% and stay sharp; JPG, PNG, GIF and BMP images zoom to 200%.



Changes are outlined on the images, and a legend above them lists the kinds present:

- **Green** – an element added to the diagram (destination image).
- **Red** – an element removed from the diagram (baseline image).
- **Blue** – an element whose class changed in something its box shows: its name, stereotype or package, or an attribute added, removed, or changed in its name, type, multiplicity or initial value (destination image). A change only to descriptions is not outlined; it is shown with the class.
- **Amber** – an element moved or resized (both images, each in its own position).
- **Purple** – an element whose style changed, such as its colour or font (destination image).

A shift of the whole diagram, which Enterprise Architect often makes when exporting, is not counted as a move. Turn the outlines off with the **Diagram highlights** checkbox in the header.

▶ **DocEPRIProposedUpdates7** *renamed from EPRIProposedUpdates7* CHANGED {34CB592E-E86B-4fe2-9CEB-0392B6CDD066}

DIAGRAM NOTES

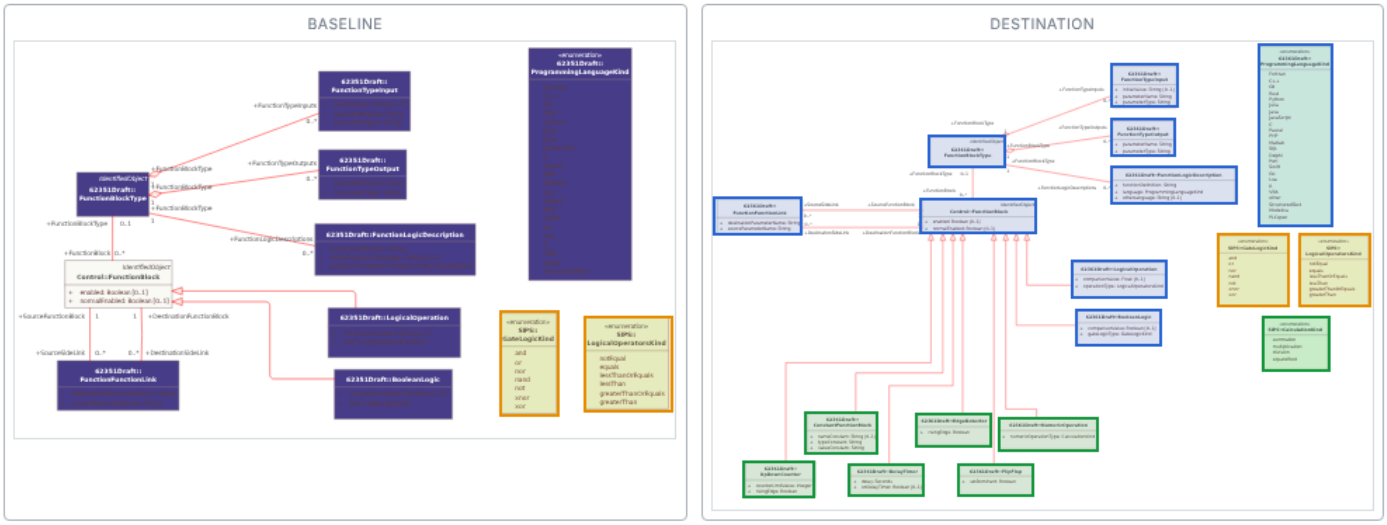
No notes defined in the model.

Metadata

	Baseline	Destination
ModifiedDate	2024-10-18 16:05:08	2024-12-08 16:17:36
Name	EPRIProposedUpdates7	DocEPRIProposedUpdates7

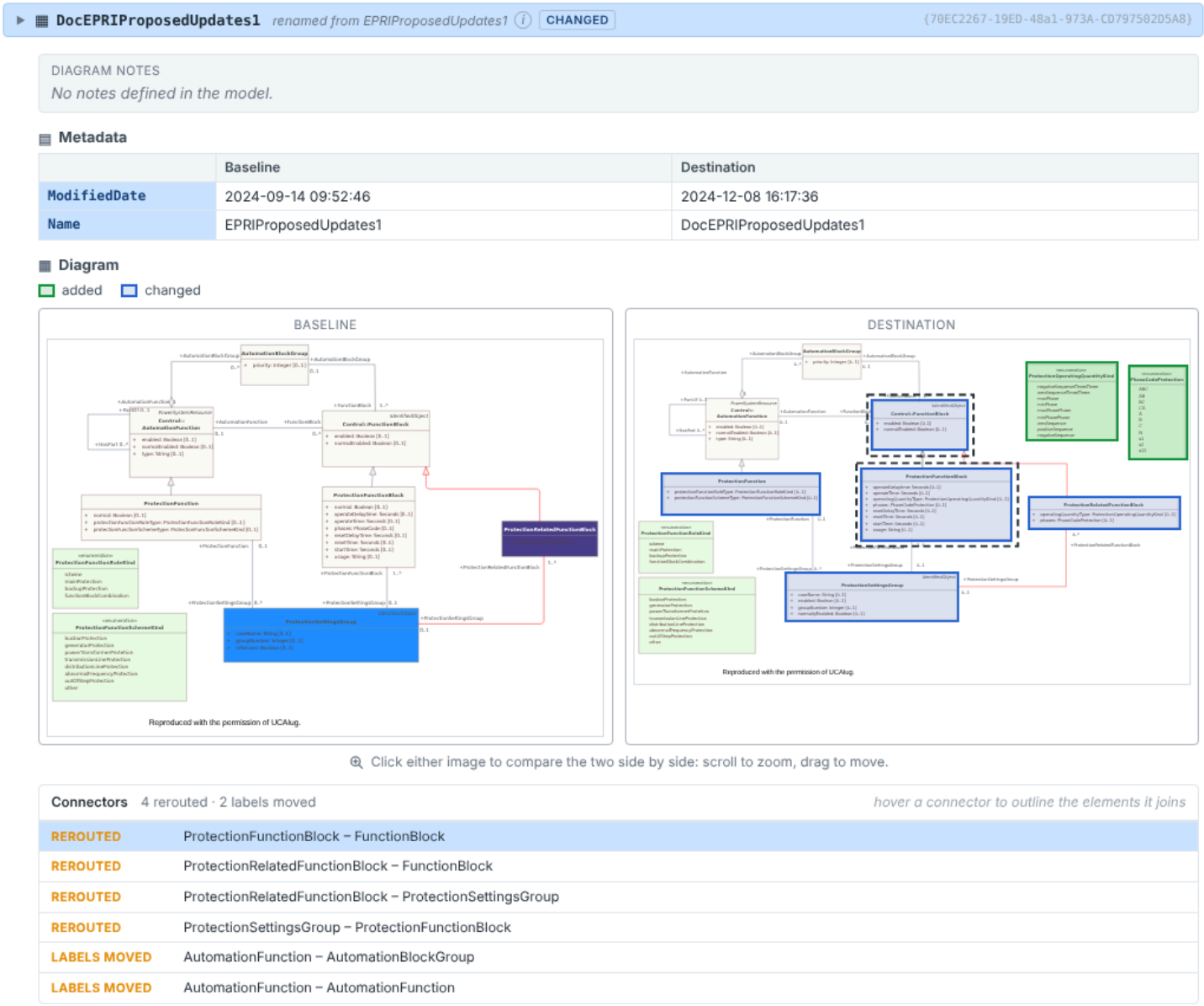
Diagram

added changed moved or resized



Click either image to compare the two side by side: scroll to zoom, drag to move.

Changed connectors (associations, generalizations and other lines) are listed in a **Connectors** table under the images: added, removed, rerouted, labels moved (role names or multiplicities dragged), labels changed (role names, multiplicities or stereotype changed in the model, with what changed, such as “multiplicity 0..* → 1..*”) or restyled, each named by the two elements it joins. Point at a row to outline those two elements with a dashed box on the image it applies to (the baseline for a removed connector); click the row to keep the outline. Connectors are listed rather than drawn because Enterprise Architect does not record the route it draws for them. “(not in this model)” stands for an element the diagram shows but the exported model does not define, such as a class from another model.

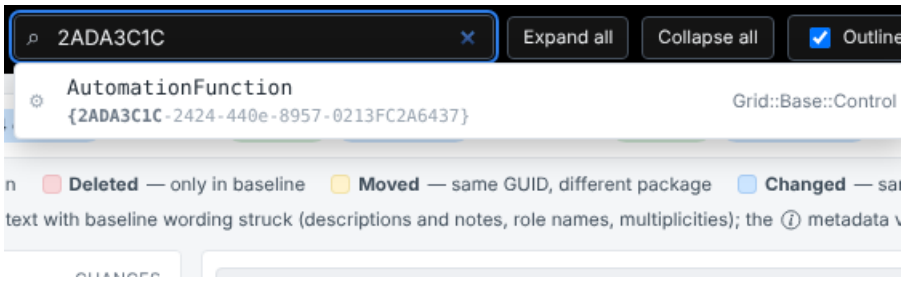


Every diagram that changed is listed. A diagram whose only changes are its name, notes or modified date, with nothing changed on it, shows its metadata alone, without images. An added diagram shows *Diagram did not exist in the baseline model.* on the hatched baseline side, and a removed one *Diagram was removed from the model.* on the destination side.

With `--full`, an unchanged diagram shows its image once, at its own size and centred in a full-width panel labelled **Baseline and destination (unchanged)**, since both models have the same picture. With EA project files as input (Option #1), **cim-compare** has EA draw images only for the diagrams the report shows; with XMI files as input (Option #2), it deletes the images of the other diagrams, so they are not included by `--zip` either way. A diagram EA could not draw says “No image” in place of its image.

Finding Things

- **Search** (or press `/`): type part of the name of a package, class, diagram or attribute, or a GUID. Matches appear as you type, with their package; choose one with the arrow keys and `Enter`, or click it. The report opens at that element and highlights it. A GUID can be typed in any of the forms EA, XMI files and scripts use: `{56C8D23C-C73A-4bea-870E-5C920A3F1066}`, without braces, in lower case, or as `EAID_56C8D23C_C73A_4bea_870E_5C920A3F1066`; eight or more hex digits match by prefix. A GUID match shows the element’s GUID under its name, and associations and generalizations can be found this way too.



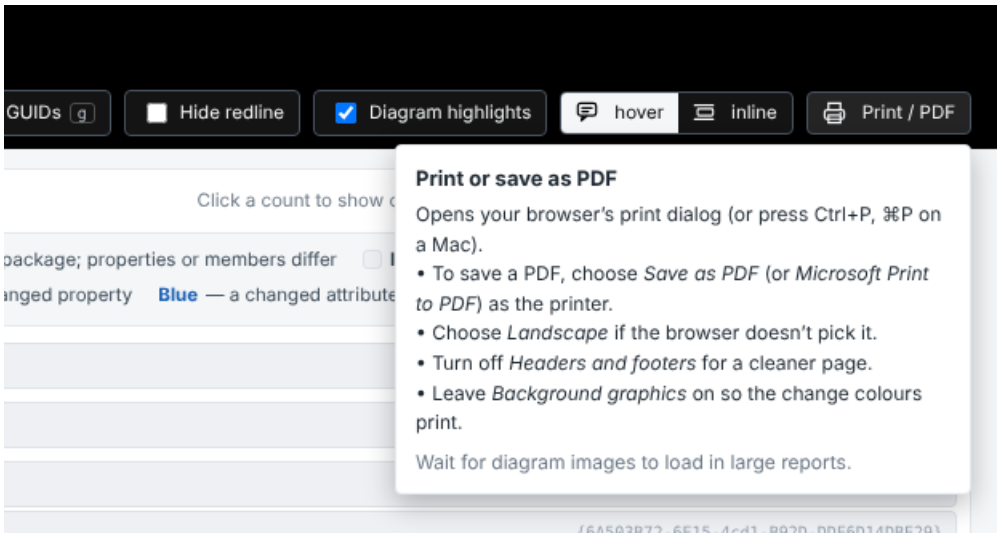
- **Links to other classes**: attribute types and link ends that are classes in the model are links. Following one opens the class; the browser's **Back** button returns you to where you were.
- **Links into a report**: every element has an address of the form `comparison-report.html#EAID_...` (the element's GUID). Copy it from the address bar after opening an element to send someone straight to it.

Keyboard Shortcuts

Key	Action
`/`	Search
`↑` `↓`	Previous / next row (in search results: previous / next match)
`←` `→`	Close / open the current row
`Enter`	Open or close the current row (in search: go to the match)
`Home`	Back to the top (the browser's Back button returns you)
`o`	Show / hide the package outline
`g`	Show / hide GUIDs
`p`	Keep in view while scrolling: off / counts / counts + legend
`+` `-` `0`	In an enlarged diagram: zoom in, zoom out, fit to the window
`Esc`	Clear the search, close a pop-up or an enlarged diagram

Printing and PDF

Print / PDF in the header opens the browser's print dialog (or press Ctrl+P, ⌘P on a Mac); point at it for the settings that give a good PDF. To save a PDF, choose *Save as PDF* (or *Microsoft Print to PDF*) as the printer, with *Landscape*, *Headers and footers* off and *Background graphics* on. The report prints with its colors and its header, the whole package tree expanded, and each diagram's panels kept with their images; the search box, buttons and package outline are left out. A printout reads as inline mode, whichever mode the screen is in: every package's, class's and diagram's notes, metadata and tagged values are printed, no ⓘ is blue or has a dot, and attribute details are printed as they are open on screen (switch to inline mode and use **Expand all** to print them all). In a large report, wait for the diagram images to load before printing.



Supported Browsers

Current versions of Chrome, Edge, Firefox and Safari. Internet Explorer is not supported.

Setup & Configuration

Java Technical Requirements

cim-compare ships as a fully self-contained executable jar file with no external dependencies. To execute it requires that an appropriate release of Java be installed on your system.

Given the large file sizes consumed and produced by the utility, **OutOfMemory** exceptions occur if the default heap (memory) size is used. The following guidelines should be followed:

> For a 32-bit JRE it is recommended to specify a max heap size of at least 1G (i.e. 1024m) resources permitting. With a 1G heap size the execution will typically run slower. Note that the maximum theoretical heap limit for a 32-bit JVM is 4G. However, due to various additional constraints such as available swap, kernel address space usage, memory fragmentation, and VM overhead, in practice the limit can be much lower. On most modern 32-bit Windows systems the maximum heap size will range from 1.4G to 1.6G. A 32-bit Java refuses a larger setting such as `-Xmx4G` at startup with "Invalid maximum heap size". A comparison of the CIM Grid package with diagrams runs on 32-bit Java 17 with `-Xmx1200m`.

```
java -Xmx1G -jar cim-compare-2.3.1.jar iec61970cim16v26a.xmi iec61970cim17v16.xmi D:\reports
```

> For 64 bit JREs it is recommended that the maximum heap size be increased to at least 2G (i.e. 2048m) but more if available. It has been observed that 2G is minimally sufficient to handle larger CIM models one might encounter. In the below example 4G is being allocated.

```
java -Xmx4G -jar cim-compare-2.3.1.jar iec61970cim16v26a.qea iec61970cim17v16.qea D:\reports
```

> Most of that memory is needed to export and compare the models. Producing the report itself takes much less: for a full-model CIM17v40 → CIM18v16 comparison (a 92 MB comparison XML), the report step needed less than 1G, including

> with `--full`.

> For freely available downloads of Java visit Azul Systems' download site at Download Azul JDKs.

Java 24 and Later

Since 2.0.1, **cim-compare** runs on Java 24 and later (including Java 25) with no extra settings. Two changes in Java 24 affect it (#65):

- **XML entity size limits.** Java 24 lowered its default limits on the size of XML entities to 100,000 (JDK-8343006). A comparison of full CIM models easily goes over them, and earlier releases of **cim-compare** then stop before writing the report, with an error such as:

  ```
  JAXP00010003: The length of entity "[xml]" is "100,001" that exceeds the "100,000" limit
  ```

  From 2.0.1, **cim-compare** removes these limits itself at startup, as it already did for Java 11's XPath limits (1.3.0, #21). The files it reads are your own models, so the limits aren't needed.

- **Native access.** Since Java 24, a program that loads a native library prints a "restricted method" warning (JEP 472); a future Java release will block it unless it is allowed. **cim-compare** loads EA's `SSJavaCOM.dll` / `SSJavaCOM64.dll` when `.eap` or `.qea` project files are the input (Option #1). From 2.0.1, the jar allows this itself (`Enable-Native-Access: ALL-UNNAMED` in its manifest), so there is no warning. Earlier Java releases ignore the setting.

With **cim-compare 2.0.0 or earlier** on Java 24 or later, add the settings to the command line yourself, before `-jar`:

```
java -Xmx4G -Djdk.xml.totalEntitySizeLimit=0 -Djdk.xml.maxGeneralEntitySizeLimit=0 --enable-native-access=ALL-UNNAMED
```

or, in a batch file, set them once for every Java program run from it:

```
set "JAVA_TOOL_OPTIONS=-Djdk.xml.totalEntitySizeLimit=0 -Djdk.xml.maxGeneralEntitySizeLimit=0 --enable-native-access=
```

Note that newer Java releases for Windows are 64-bit only (see Azul OpenJDK). Comparing `.eap` project files needs 32-bit Java, so for `.eap` input use Java 17 or earlier; `.qea`, XMI and compare-log input can use any current release.

Azul OpenJDK

We recommend end-users use Azul OpenJDK (Zulu) which is Azul Systems' free, TCK-certified binary distribution of the OpenJDK — the open-source implementation of the Java Platform, Standard Edition (Java SE). Azul provides reliable, production-ready Java runtimes for enterprises and developers who want an open-source alternative to Oracle JDK. OpenJDK is:

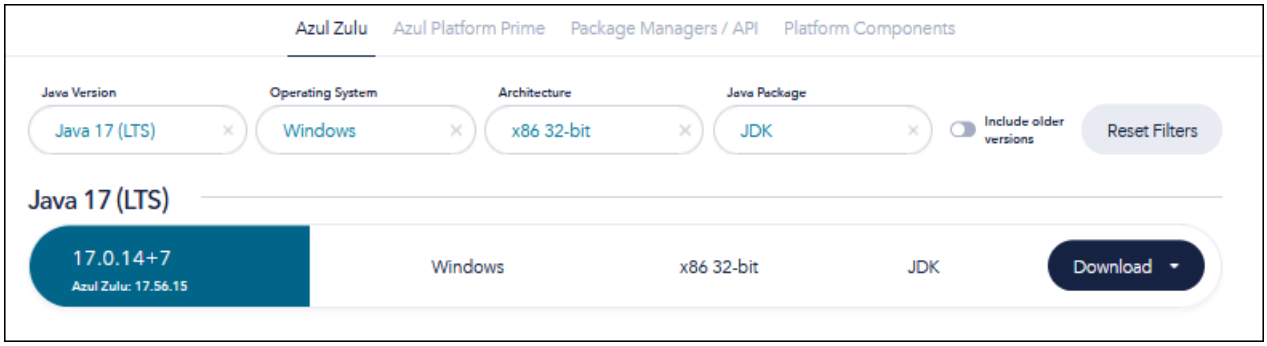
- 100% open-source and fully compliant with Oracle's Java SE standard.
- Offers builds for many Java versions (e.g., 6, 7, 8, 11, 17, 21, and newer).
- Available for multiple platforms: Windows, Linux, macOS, and others.
- Includes free updates, security patches, and optional commercial support.

When downloading there are special considerations related to the 32-bit version. Azul's supported-platforms list offers Windows x86 32-bit builds only of Java 8, 11 and 17; Java 21 and later are 64-bit only on Windows.

The reason for the shift is that it reflects a broader trend. OpenJDK deprecated Windows 32-bit, targeting removal in JDK 23+, making it increasingly rare to see 32-bit builds in newer OpenJDK distributions. Therefore, though **cim-compare** is not tied to Java 17 and can run on newer releases of OpenJDK (from 2.0.1 including Java 24 and later with no extra settings; see Java 24 and Later), the below provided download links simply point to Java 17 where both 32-bit and 64-bit versions are available. For `.qea`, XMI or compare-log input a newer 64-bit release such as Java 21 or 25 works equally well.

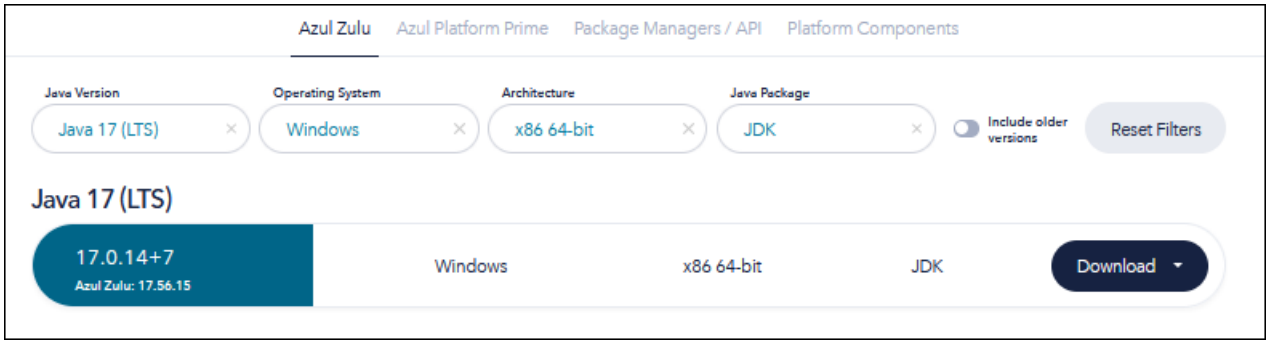
Download 32-bit Java 17

If you are looking to run comparisons of `.eap` files then you'll need 32-bit Java. To directly download a 32-bit Java 17 click the image below:



Download 64-bit Java 17

If you are comparing `.qea` files with a 64-bit EA (EA 16 or later) then you'll need 64-bit Java. To directly download a 64-bit Java 17 click the image below:



Important Setup Considerations

> The **cim-compare** command-line option described in the Option #1: Enterprise Architect .EAP or .QEA Project Files as Input section requires EA's COM automation API. Consequently, if you plan to use it you are required to have a licensed version of Sparx Enterprise Architect (EA) installed for the COM automation API to work. The EA API and the `eaapi.jar` file act as interfaces to the Enterprise Architect application. The following explains why:
>
> - Dependency on the Sparx Enterprise Architect Application:
>   - The EA COM API and `eaapi.jar` rely on the backend functionality of the Enterprise Architect software. They act as a bridge to interact with the models, diagrams, and repositories managed by EA.
>   - Without the Enterprise Architect application installed, there is no core functionality or data source for the API to connect to.
> - Enterprise Architect COM Object:
>   - The `SSJavaCOM.dll` and `SSJavaCOM64.dll` are COM interface wrappers that requires Enterprise Architect's COM objects to function. These COM objects are installed as part of the Enterprise Architect installation process.
> - Licensing Requirements:

    - Sparx Enterprise Architect is a licensed product, and the API requires a valid license to access its features. Without a licensed installation of Enterprise Architect, the API calls will fail or be severely limited.
- Runtime Dependency:
    - When using the COM automation interface, **cim-compare** initializes and communicates with an EA process. The underlying API calls interact with the installed Enterprise Architect software to perform operations like exporting `.xmi` files from an `.eap` or `.qea` file.

The COM interface wrappers ship with each release of Sparx EA in the "Java API" folder of the installation directory, similar to:

`%WINDOWS_PROGRAM_FILES%\Sparx Systems\EAxx\Java API (e.g. "C:\Program Files\Sparx Systems\EA16\Java API")`

The purpose of each is:

- `SSJavaCOM.dll` - the 32-bit COM Module DLL (Dynamic Linked Library) that **cim-compare** will link to when a 32-bit Java JVM/JRE is used to run `cim-compare-2.3.1.jar` from the command line (the usual path for `.eap` project files with a 32-bit EA; `.eap` files use Microsoft's Jet database engine, which is 32-bit only, so only a 32-bit EA can open them, though 64-bit Java with `SSJavaCOM64.dll` can also drive a 32-bit EA, as in the tested mixed comparison under Warning: Potential Configuration Pitfalls).

- `SSJavaCOM64.dll` - the 64-bit COM Module DLL that **cim-compare** will link to when a 64-bit Java JVM/JRE is used to run `cim-compare-2.3.1.jar` from the command line (the usual path for `.qea` project files with a 64-bit EA, EA 16 or later; `.qea` itself is not 64-bit-only, as Sparx states `.qea` files work in both 32-bit and 64-bit EA).

cim-compare communicates to the appropriate COM Module DLL depending on the particular JVM (32-bit or 64-bit) being used on the command line. Which DLL to load is decided at run time: EA's `eaapi.jar` loads `SSJavaCOM.dll` when Java reports its architecture as `x86` (32-bit Java) and `SSJavaCOM64.dll` otherwise.

Another key requirement is that these DLL COM interface wrappers must be where Java can find them at run time. Java looks in the folders listed in `java.library.path`, which on Windows defaults to the folders on the `PATH` (Sparx's own instructions copy `SSJavaCOM.dll` into a Windows system folder). For **cim-compare** we recommend naming the folder explicitly with the `-Djava.library.path` JVM parameter, as in the examples below; on a machine with more than one EA installation it is the way to choose which installation's DLLs are used.

To use **cim-compare** on a system with a dual 32-bit and 64-bit Sparx EA installation (e.g. EA 15.x and EA 16.x) you will need to have a configuration similar to the following:

```
C:\
├── cim-compare
│   ├── cim-compare-2.3.1.jar   (the latest downloaded from https://cim-compare.ucaiug.io)
│   ├── cim-compare-2.3.1.bat   (example batch file for execution)
│   ├── ea15
│   │   ├── SSJavaCOM.dll       (copied from "C:\Program Files (x86)\Sparx Systems\EA15\Java API")
│   │   └── SSJavaCOM64.dll     (copied from "C:\Program Files (x86)\Sparx Systems\EA15\Java API")
│   ├── ea16
│   │   ├── SSJavaCOM.dll       (copied from "C:\Program Files\Sparx Systems\EA16\Java API")
│   │   └── SSJavaCOM64.dll     (copied from "C:\Program Files\Sparx Systems\EA16\Java API")
│   ├── cim17v40.eap
│   ├── cim18v02.eap
│   ├── cim18v12.qea
│   └── cim18v13.qea
├── Program Files (x86)         (32-bit JVM installation, required for .eap files)
│   └── Zulu
│       └── zulu-17
│           └── bin
│               └── java.exe
```

```
├── Program Files              (64-bit JVM installation, for .qea files with the 64-bit EA16)
│   └── Zulu
│       └── zulu-17
│           └── bin
│               └── java.exe
└── ...
```

Of importance is that each EA installation's set of DLL files be located in their own directory. This will allow for the ability to isolate where Java looks for its COM Modules based on the specific release of Sparx EA.

Following is a set of command lines based on the above example configuration and file system structure. The first compares `.eap` files with 32-bit Java and the 32-bit EA15 installation; the second compares `.qea` files with 64-bit Java and the 64-bit EA16 installation. The command lines are split over several lines with `^`, the Windows command prompt's line-continuation character, so they can be pasted into a command prompt as shown.

```
"C:\Program Files (x86)\Zulu\zulu-17\bin\java.exe" -Xmx1200m -Djava.library.path="C:\cim-compare\ea15" ^
  -jar cim-compare-2.3.1.jar cim17v40.eap cim18v02.eap comparison-report.html ^
  --include-diagrams --image-type=JPG
```

The above 32-bit command line example uses:

- a 32-bit Java 17 JRE/JVM (i.e. `"C:\Program Files (x86)\Zulu\zulu-17\bin\java.exe"`)
- a max heap size of 1.2G (i.e. `-Xmx1200m`)
- the 32-bit COM DLL loaded from the `C:\cim-compare\ea15` directory (i.e. via `-Djava.library.path="C:\cim-compare\ea15"`)
- `cim17v40.eap` as the input baseline model (an `.eap` project file, which only 32-bit EA can open)
- `cim18v02.eap` as the input destination model (an `.eap` project file, which only 32-bit EA can open)
- `comparison-report.html` as the name of the generated report
- the inclusion of changed diagrams in the report (i.e. `--include-diagrams`)
- JPG for the type of diagrams (i.e. `--image-type=JPG`)

```
"C:\Program Files\Zulu\zulu-17\bin\java.exe" -Xmx4G -Djava.library.path="C:\cim-compare\ea16" ^
  -jar cim-compare-2.3.1.jar cim17v40.qea cim18v02.qea comparison-report.html ^
  --include-diagrams --image-type=SVG
```

The above 64-bit command line example uses:

- a 64-bit Java 17 JRE/JVM (i.e. `"C:\Program Files\Zulu\zulu-17\bin\java.exe"`)
- a max heap size of 4G (i.e. 4GB specified via `-Xmx4G` or `-Xmx4096m`)
- the 64-bit COM DLL loaded from the `C:\cim-compare\ea16` directory (i.e. via `-Djava.library.path="C:\cim-compare\ea16"`)
- `cim17v40.qea` as the input baseline model (a `.qea` project file, opened here by the 64-bit EA16)
- `cim18v02.qea` as the input destination model (a `.qea` project file, opened here by the 64-bit EA16)
- `comparison-report.html` as the name of the generated report
- the inclusion of changed diagrams in the report (i.e. `--include-diagrams`)
- SVG for the type of diagrams (i.e. `--image-type=SVG`)

Warning: Potential Configuration Pitfalls

A common “gotcha” is a mismatch between the project files, the EA installation and Java. The EA installation decides which files can be opened: `.eap` and `.eapx` files only by a 32-bit EA, because they use Microsoft’s Jet database engine, which is 32-bit only; `.qea` and `.qeax` files by 32-bit and 64-bit EA alike. Java’s bit-width decides which of EA’s COM DLLs is loaded, `SSJavaCOM.dll` for 32-bit Java or `SSJavaCOM64.dll` for 64-bit Java, and that DLL must be in the folder given with `-Djava.library.path`. These combinations have been tested:

Input	EA	Java
two `.eap` files	32-bit	32-bit (`SSJavaCOM.dll`)
two `.qea` files	64-bit (EA 16 or later)	64-bit (`SSJavaCOM64.dll`)
one `.eap` and one `.qea` / `.qeax` (2.1.0 and later)	32-bit (EA 17.2)	64-bit Java 25 (`SSJavaCOM64.dll`)

A mixed pair needs an EA that opens both files, which means a 32-bit EA.

Noteworthy in the previous command line examples is that `java` has been expressly qualified as either `"C:\Program Files (x86)\Zulu\zulu-17\bin\java.exe"` or `"C:\Program Files\Zulu\zulu-17\bin\java.exe"` to ensure that the correct version of Java is used. This is not a requirement but rather one method to eliminate this as a pitfall.

If choosing to simplify the command line via use of the default installation of Java such as:

```
java -Xmx4G -Djava.library.path="C:\cim-compare\ea16" -jar cim-compare-2.3.1.jar ...
```

...then it is suggested that you confirm the bit-wise version of your default Java installation. The following java command will result in output that should indicate if your default installation is 32-bit. This either by explicitly stating as such or by including the installation directory which for 32-bit will be located in `C:\Program Files (x86)\...`. Note that for many of the newest releases of Java, if not explicitly stated as being 32-bit, it is then implied to be 64-bit as illustrated in this example:

```
C:\>java -version
openjdk version "17.0.13" 2024-10-15 LTS
OpenJDK Runtime Environment Zulu17.54+21-CA (build 17.0.13+11-LTS)
OpenJDK Client VM Zulu17.54+21-CA (build 17.0.13+11-LTS, mixed mode, emulated-client)
```

If command line execution results in an error similar to the following then it typically is an indicator that the Java executable being used does not bit-wise match the project files being input:

```
java.lang.Exception: Internal application error.
       at org.sparx.Repository.comGetModels(Native Method)
       at org.sparx.Repository.GetModels(Repository.java:805)
       at org.cimug.compare.app.CIMModelComparisonGenerator.parseFileArguments(CIMModelComparisonGenerator.java:4
       at org.cimug.compare.app.CIMModelComparisonGenerator.main(CIMModelComparisonGenerator.java:107)
ERROR:  Terminating XMI export processing for EA project file [CIMMarket04v16a.eap] due to an unexpected exceptio
```

Finally, keep in mind that when using the default installation you could have a bit-wise compliant version of Java and have no issues creating reports only later to discover that **cim-compare** no longer work. Such a scenario can occur when between uses of cim-compare, software is installed on your system that, as part of its installation process, installs a newer 64-bit version of Java. This would then make this latest installation of Java your default. This typically happens “under the radar” and should be one of the first things checked when troubleshooting potential issues.

Command Line Usage

The **cim-compare** utility has three possible command-line options to produce comparison reports as described next. In the examples, long command lines are split with `^`, the Windows command prompt's line-continuation character, so they can be pasted into a command prompt as they are. Directory and file paths that contain spaces are given in quotes.

Option #1: Enterprise Architect .EAP or .QEA Project Files as Input

The preferred usage is to provide two Enterprise Architect `*.eap` or `*.qea` files for the "baseline" and "destination" input files. Using these file types eliminates the need to manually export XMI and diagrams from EA as described later in the Enterprise Architect XMI Export Procedures section. `.eapx` and `.qeax` files are accepted like `.eap` and `.qea` files. The baseline and destination can be of different types, for example one `.eap` and one `.qeax`, when the EA installation in use opens both (a 32-bit EA does); since 2.1.0 such a comparison reports the same changes as one between two files of the same type (#86). Which EA and Java to use for each kind of file is described under Warning: Potential Configuration Pitfalls. Note that the processing of 64-bit `*.qea` files is more performant.

The command-line usage for this option takes the following form:

```
java [<jvm-parameter-1>]...[<jvm-parameter-n>] -jar cim-compare-2.3.1.jar
     <baseline-model-file> <destination-model-file> [<output-directory-or-html-file>]
     [--package=<package-name>,...] [--full] [--show-empty-values] [--include-diagrams] [--image-type=<image-file-ext
     [--zip] [--cleanup]
```

Parameter Details:

`[<jvm-parameter-x>]` **(Optional)**: A JVM (Java Virtual Machine) parameter that may be needed for execution. JVM parameters are configurations used to control the behavior of the JVM at runtime and take a form such as `-Xmx1G`, `-Xmx4G` or `-Dfile.encoding=UTF-8`. These parameters can influence memory usage, garbage collection, system properties, debugging, and performance tuning. JVM parameters are passed as command-line arguments when starting a Java application and if used should **appear first** in the sequence of command line arguments (before `cim-compare-2.3.1.jar`). This is necessary so that **cim-compare** does not try to process them as part of its command line arguments which should always appear after `cim-compare-2.3.1.jar`. Note that for this Option #1 the `-Djava.library.path=<directory>` JVM parameter is required as will be described later.

`<baseline-model-file>` **(Required)**: An Enterprise Architect baseline `.eap` or `.qea` model file. When not specified as an absolute file path the location is assumed to be the directory **cim-compare** is being executed from.

`<destination-model-file>` **(Required)**: An Enterprise Architect destination `.eap` or `.qea` model file. When not specified as an absolute file path the location is assumed to be the directory **cim-compare** is being executed from.

`[<output-directory-or-html-file>]` **(Optional)**: An output directory or an output HTML file. In the case where a directory is specified but does not exist it will be created. The same is true for the parent directory of a specified HTML file if it does not exist. When an HTML file is provided it may be specified as an absolute file path or as the simple name of an HTML file to be generated. If this command line option is not specified then **cim-compare** will generate an HTML file whose name is derived from the two input files.

`[--package=<package-name>,...]` **(Optional)**: The package to compare, together with everything below it (e.g. `--package=Grid`, `--package=IEC61970`, `--package=IEC61968`, `--package=IEC62325`). Several packages can be given, separated by commas (e.g. `--package=Grid,UtilityExtensions`): the report shows them in that order, and references between them, such as an extension class that specializes a CIM class, are reported like any others. Put the value in quotes when a package name contains a space (e.g. `--package="Grid,European Standards"`). Each package is looked up by name in both models, root nodes included. If it was renamed between the two versions, either name can be given: for example, comparing CIM17 (where the

package is `IEC61970`) with CIM18 (where it is `Grid`), **cim-compare** finds the package by name in one model and by its GUID in the other. A package found in only one of the models (for example a new extension root) is reported as added or deleted. If a name is found in neither model, **cim-compare** stops with an error naming it. If one listed package is inside another (e.g. `--package=TC57CIM,IEC61970`), it is reported as part of the outer package and a warning says so. When `--package` is not given, the whole model is compared: every root node of each EA project. A project with several root nodes, such as a CIM model alongside extension packages, is compared in full, and each root node appears as a top-level package in the report; references from one root node to another (e.g. an extension class that specializes a CIM class) are reported like any others. To compare a single root node, give its name, e.g. `--package=Model`.

`[--full]` **(Optional)**: Include identical packages, classes, attributes, links and diagrams in the report as well as the changes. They are shown collapsed and in grey. With `--include-diagrams`, an identical diagram shows its image (the destination's). Without `--full` the report contains only what changed.

`[--show-empty-values]` **(Optional)**: Report every difference between a value and an empty value. Without it, such a difference is left out in the housekeeping fields EA fills in automatically when it creates an element: Complexity, Phase, Version and GenType (any value); Containment `Not Specified` or `Unspecified`; IsLiteral, Derived, Const and Static `false`. These fields are often empty in a model whose elements were created outside EA, and the differences do not change the model's definitions. The report says how many were left out, under the legend.

`[--minimal]` **(Optional)**: Accepted so that command lines written for 1.x keep working, but has no effect: the report contains only changes by default since 2.0.0.

`[--include-diagrams]` **(Optional)**: Include the changed diagrams in the report, with their images. **cim-compare** exports the two models from EA without images, compares them, and then has EA draw only the diagrams the report shows, into `Images-baseline` and `Images-destination`. With `--full` that includes every identical diagram, drawn from the destination project only, so the run takes longer. A diagram EA fails to draw is tried once more in a new EA session; one it cannot draw at all says "No image" in the report. With EMF images, or with an EA older than 16.1 build 1629, every diagram's image is exported with the XMI instead, and the images the report does not show are then deleted. The image type is set with `--image-type`; JPG is the default.

`[--image-type=<image-file-extension>]` **(Optional)**: The type of diagram images: JPG (the default), SVG, PNG, GIF, BMP or EMF. SVG images stay sharp at any zoom and in print; they need EA 16.1 build 1629 or later on the machine where **cim-compare** runs (viewers of the report need only a browser), and with an older EA **cim-compare** says so and uses JPG, which the report header notes. A diagram EA will not draw as SVG is shown as a PNG, without highlights. Browsers cannot show EMF, so EMF diagrams do not display in the report. Used only with `--include-diagrams`.

`[--zip]` **(Optional)**: When specified **cim-compare** will package up the generated report and any associated diagrams into a single ZIP archive. It is most often utilized for packaging the report when diagram images are included and can help simplify distribution. The ZIP archive contains the report and, with `--include-diagrams`, the `Images-baseline` and `Images-destination` folders; it never contains the XMI or XML files.

`[--cleanup]` **(Optional)**: Used with `--zip`. Deletes the files and folders created during report generation, leaving the ZIP archive: here the report, the exported XMI files, the comparison XML, the enriched XML and the image folders. Your two EA project files are not touched.

Command Line Examples:

```
java -Xmx1G -Djava.library.path="C:\cim-compare\ea15" -jar cim-compare-2.3.1.jar ^
  "C:\exports\15v33.eap" "C:\exports\CIM16v26a.eap" "C:\reports"

java -Xmx4G -Djava.library.path="C:\cim-compare\ea16" -jar cim-compare-2.3.1.jar ^
  "C:\exports\15v33.qea" "C:\exports\CIM16v26a.qea" "C:\reports"

java -Xmx1G -Djava.library.path="C:\cim-compare\ea15" -jar cim-compare-2.3.1.jar ^
```

```
  "C:\exports\15v33.eap" "C:\exports\CIM16v26a.eap" --package=IEC61970

java -Xmx4G -Djava.library.path="C:\cim-compare\ea16" -jar cim-compare-2.3.1.jar ^
  CIM16_Reference.qea CIM16_Extended.qeax --package="Grid,European Standards,UtilityExtensions"

java -Xmx1G -Djava.library.path="C:\cim-compare\ea15" -jar cim-compare-2.3.1.jar ^
  CIM15v33.eap CIM16v26a.eap C:\reports --include-diagrams --zip

java -Xmx4G -Djava.library.path="C:\cim-compare\ea16" -jar cim-compare-2.3.1.jar ^
  CIM15v33.qea CIM16v26a.qea C:\reports\CIM15v33_CIM16v26a_ComparisonReport.html --full

java -Xmx1G -Djava.library.path="C:\cim-compare\ea15" -jar cim-compare-2.3.1.jar ^
  CIM15v33.eap CIM16v26a.eap --package=IEC62325 --include-diagrams --image-type=GIF --zip --cleanup

java -Xmx4G -Djava.library.path="C:\cim-compare\ea16" -jar cim-compare-2.3.1.jar ^
  CIM17v40.qea CIM18v16.qea --package=Grid --include-diagrams --image-type=SVG --zip --cleanup

java -Xmx4G -Djava.library.path="C:\cim-compare\ea16" -jar cim-compare-2.3.1.jar ^
  CIM_Grid18v13.qea CIM_Grid18v14.qea --package=Grid --show-empty-values

java -Xmx4G -Djava.library.path="C:\cim-compare\ea16" -jar cim-compare-2.3.1.jar ^
  CIM16_Reference.qea CIM16_Extended.qeax --package=Model

java -Xmx4G -Djava.library.path="C:\cim-compare\ea16" -jar cim-compare-2.3.1.jar ^
  CIM_Grid18v13.eap CIM_Grid18v14.qeax --package=Grid --include-diagrams
```

NOTE: The above command line examples use both `.eap` and `.qea` EA files as input. An example with two `.eap` files assumes 32-bit Java and a 32-bit EA, and one with two `.qea` files 64-bit Java and a 64-bit EA. The last example compares an `.eap` with a `.qeax` file: it needs a 32-bit EA, which opens both, and was tested with 64-bit Java (see Warning: Potential Configuration Pitfalls).

Option #2: XMI Baseline and Destination Models as Input

The second usage is to directly specify two XMI 1.1 compliant files representing the “baseline” and “destination” models exported as described later in Enterprise Architect XMI Export Procedures. In this scenario the command-line usage takes the following form:

> IMPORTANT: the XMI files must be Enterprise Architect exports of type **“UML 1.3 (XMI 1.1)”**, with **“Unisys/Rose Format”** not checked. **cim-compare** does not read EA’s other XMI formats (XMI 2.x, or XMI 1.1 in Unisys/Rose format) and stops with an error naming the file. See Enterprise Architect XMI Export Procedures.

```
java [<jvm-parameter-1>]...[<jvm-parameter-n>] -jar cim-compare-2.3.1.jar
     <baseline-model-xmi-file> <destination-model-xmi-file> [<output-directory-or-html-file>]
     [--package=<package-name>,...] [--full] [--show-empty-values] [--include-diagrams] [--image-type=<image-file-ext
     [--zip] [--cleanup]
```

Parameter Details:

`[<jvm-parameter-x>]` **(Optional)**: A JVM (Java Virtual Machine) parameter that may be needed for execution. JVM parameters are configurations used to control the behavior of the JVM at runtime and take a form such as `-Xmx1G`, `-Xmx4G` or `-Dfile.encoding=UTF-8`. These parameters can influence memory usage, garbage collection, system properties, debugging, and performance tuning. JVM parameters are passed as command-line arguments when starting a Java application and if used should **appear first** in the sequence of command line arguments (before `cim-compare-2.3.1.jar`). This is necessary so that

cim-compare does not try to process them as part of its command line arguments which should always appear after `cim-compare-2.3.1.jar`.

`<baseline-model-xmi-file>` **(Required)**: A "UML 1.3 (XMI 1.1)" baseline model file exported from EA. When not specified as an absolute file path the location of the file is assumed to be the directory the utility is being executed from.

`<destination-model-xmi-file>` **(Required)**: A "UML 1.3 (XMI 1.1)" destination model file exported from EA. When not specified as an absolute file path the location of the file is assumed to be the directory the utility is being executed from.

`[<output-directory-or-html-file>]` **(Optional)**: An output directory or an output HTML file. In the case where a directory is specified but does not exist it will be created. The same is true for the parent directory of a specified HTML file if it does not exist. When an HTML file is provided it may be specified as an absolute file path or as the simple name of an HTML file to be generated. If this command line option is not specified then **cim-compare** will generate an HTML file whose name is derived from the two input files.

`[--package=<package-name>,...]` **(Optional)**: The package the report starts from, together with everything below it (e.g. `--package=Grid`, `--package=IEC61970`, `--package=IEC61968`, `--package=IEC62325`). Several packages can be given, separated by commas (e.g. `--package=Grid,UtilityExtensions`), and the report shows them in that order; put the value in quotes when a package name contains a space (e.g. `--package="Grid,European Standards"`). A name can be the package's name in either model, so a package renamed between the two versions (e.g. `IEC61970` → `Grid`) can be given by either name. If no package of a given name is in the comparison, **cim-compare** stops with an error naming it. If one listed package is inside another, it is reported as part of the outer package and a warning says so. When `--package` is not given, the report covers the whole model.

`[--full]` **(Optional)**: Include identical packages, classes, attributes, links and diagrams in the report as well as the changes. They are shown collapsed and in grey. With `--include-diagrams`, an identical diagram shows its image (the destination's). Without `--full` the report contains only what changed.

`[--show-empty-values]` **(Optional)**: Report every difference between a value and an empty value. Without it, such a difference is left out in the housekeeping fields EA fills in automatically when it creates an element: Complexity, Phase, Version and GenType (any value); Containment `Not Specified` or `Unspecified`; IsLiteral, Derived, Const and Static `false`. These fields are often empty in a model whose elements were created outside EA, and the differences do not change the model's definitions. The report says how many were left out, under the legend.

`[--minimal]` **(Optional)**: Accepted so that command lines written for 1.x keep working, but has no effect: the report contains only changes by default since 2.0.0.

`[--include-diagrams]` **(Optional)**: When specified **cim-compare** indicates that diagram images should be included in the generated comparison report. Note that if the types of diagrams to be included are not JPG files then the `--image-type` option must also be used in tandem. When this option does not appear on the command line no diagrams will be included in the report.

> IMPORTANT: this command line option does not automatically export diagrams as performed in usage Option #1. Rather the expectation is that they be exported as part of manual exports of XMI baseline and destination models from EA and that images are hosted in folders named `<output directory>\Images-baseline` and `<output directory>\Images-destination` respectively. This must be done prior to running **cim-compare**. Images of diagrams the report does not show (identical diagrams, and diagrams with nothing changed on them) are deleted from those folders. With `--full` the report shows identical diagrams, so their images in `Images-destination` are kept and those in `Images-baseline` deleted. SVG images can be exported with the XMI from EA 16.1 build 1621 or later. Refer to the Enterprise Architect XMI Export Procedures for further details.

`[--image-type=<image-file-extension>]` **(Optional / Conditionally Required)**: Indicates the type of images (i.e. JPG, SVG, GIF, PNG, BMP, or EMF) referenced in the report. Conditionally required when the `--include-diagrams` option is used and the

diagram are not JPG image files. When not specified the default image type of JPG is used. This command line option is only relevant when `--include-diagrams` also appears on the command line.

`[--zip]` **(Optional)**: When specified **cim-compare** will package up the generated report and any associated diagrams into a single ZIP archive. It is most often utilized for packaging the report when diagram images are included and can help simplify distribution. The ZIP archive contains the report and, with `--include-diagrams`, the `Images-baseline` and `Images-destination` folders (everything in them is included, so keep only the diagram images there); it never contains the XMI or XML files.

`[--cleanup]` **(Optional)**: Used with `--zip`. Deletes the files created during report generation, leaving the ZIP archive: here the report, the comparison XML and the enriched XML. Your XMI files and your `Images-baseline` and `Images-destination` folders are not deleted.

Command Line Examples:

```
java -Xmx2G -jar cim-compare-2.3.1.jar "C:\XMI exports\15v33.xmi" "C:\XMI exports\CIM16v26a.xmi" "C:\reports"

java -Xmx2G -jar cim-compare-2.3.1.jar "C:\XMI exports\15v33.xmi" "C:\XMI exports\CIM16v26a.xmi" --package=IEC61970

java -Xmx2G -jar cim-compare-2.3.1.jar CIM16_Reference.xmi CIM16_Extended.xmi --package=Grid,UtilityExtensions

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33.xmi CIM16v26a.xmi C:\reports --full

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33.xmi CIM16v26a.xmi C:\reports ^
  --include-diagrams --image-type=GIF --zip --cleanup

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33.xmi CIM16v26a.xmi C:\reports\CIM15v33_CIM16v26a_ComparisonReport.html

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33.xmi CIM16v26a.xmi CIM15v33_CIM16v26a_ComparisonReport.html ^
  --package=IEC62325 --include-diagrams --image-type=JPG --zip
```

Option #3: EA Model Comparison Logs as Input

The third option is by specifying an **EA model comparison log** file as input on the command line. In order to perform this option it is first required that a comparison XML log file be generated from within Enterprise Architect. Refer to the EA Comparison Log Procedures for detailed information on generating XML Compare Logs from within Enterprise Architect. It is important to note that EA does not support comparisons of UML diagrams and therefore diagrams are not supported as part of reports generated using Option #3.

This particular usage takes the following form:

```
java [<jvm-parameter-1>]...[<jvm-parameter-n>] -jar cim-compare-2.3.1.jar
     <comparison-results-xml-file> [<output-directory-or-html-file>]
     [--package=<package-name>,...] [--full] [--show-empty-values] [--zip] [--cleanup]
```

Parameter Details:

`[<jvm-parameter-x>]` **(Optional)**: A JVM (Java Virtual Machine) parameter that may be needed for execution. JVM parameters are configurations used to control the behavior of the JVM at runtime and take a form such as `-Xmx1G`, `-Xmx4G` or `-Dfile.encoding=UTF-8`. These parameters can influence memory usage, garbage collection, system properties, debugging, and performance tuning. JVM parameters are passed as command-line arguments when starting a Java application and if used should **appear first** in the sequence of command line arguments (before `cim-compare-2.3.1.jar`). This is necessary so that **cim-compare** does not try to process them as part of its command line arguments which should always appear after `cim-compare-2.3.1.jar`.

`<comparison-results-xml-file>` **(Required)**: The model comparison file. When not specified as an absolute file path the location of the file is assumed to be the directory the utility is being executed in.

`[<output-directory-or-html-file>]` **(Optional)**: An output directory or an output HTML file. In the case where a directory is specified but does not exist it will be created. The same is true for the parent directory of a specified HTML file if it does not exist. When an HTML file is provided it may be specified as an absolute file path or as the simple name of an HTML file to be generated. If this command line option is not specified then **cim-compare** will generate an HTML file whose name is derived from the comparison file's name.

`[--package=<package-name>,...]` **(Optional)**: The package the report starts from, together with everything below it (e.g. `--package=Grid`, `--package=IEC61970`, `--package=IEC61968`, `--package=IEC62325`). Several packages can be given, separated by commas (e.g. `--package=Grid,UtilityExtensions`), and the report shows them in that order; put the value in quotes when a package name contains a space (e.g. `--package="Grid,European Standards"`). A name can be the package's name in either model, so a package renamed between the two versions (e.g. `IEC61970` → `Grid`) can be given by either name. If no package of a given name is in the comparison, **cim-compare** stops with an error naming it. If one listed package is inside another, it is reported as part of the outer package and a warning says so. When `--package` is not given, the report covers the whole model.

`[--full]` **(Optional)**: Include identical packages, classes, attributes, links and diagrams in the report as well as the changes. They are shown collapsed and in grey. Without `--full` the report contains only what changed.

`[--show-empty-values]` **(Optional)**: Report every difference between a value and an empty value. Without it, such a difference is left out in the housekeeping fields EA fills in automatically when it creates an element: Complexity, Phase, Version and GenType (any value); Containment `Not Specified` or `Unspecified`; IsLiteral, Derived, Const and Static `false`. These fields are often empty in a model whose elements were created outside EA, and the differences do not change the model's definitions. The report says how many were left out, under the legend.

`[--minimal]` **(Optional)**: Accepted so that command lines written for 1.x keep working, but has no effect: the report contains only changes by default since 2.0.0.

`[--zip]` **(Optional)**: When specified **cim-compare** will package up the generated report into a single ZIP archive. Note that since this option #3 does not support diagram comparisons only the report itself will be included in the archive.

`[--cleanup]` **(Optional)**: Used with `--zip`. Deletes the report and the enriched XML, leaving the ZIP archive. Your comparison XML file is not deleted.

Command Line Examples:

```
java -Xmx2G -jar cim-compare-2.3.1.jar "C:\CIM XMI Exports\CIM15v33_CIM16v26a_EA_Comparison_Report.xml" "C:\Reports"

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33_CIM16v26a_EA_Comparison_Report.xml "C:\Comparison Reports" --full

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33_CIM16v26a_EA_Comparison_Report.xml "C:\Comparison Reports" --package=

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33_CIM16v26a_EA_Comparison_Report.xml ComparisonReport_CIM15v33_CIM16v26

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33_CIM16v26a_EA_Comparison_Report.xml

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33_CIM16v26a_EA_Comparison_Report.xml --package=IEC61970

java -Xmx2G -jar cim-compare-2.3.1.jar CIM15v33_CIM16v26a_EA_Comparison_Report.xml --package=IEC61970 --zip --cleanup
```

Output Files

In the output directory, a run produces:

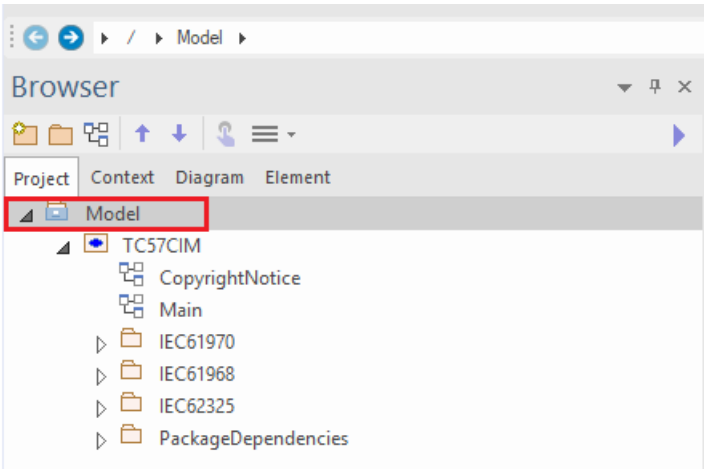
File	Produced by	Contents
`<name>.html`	all options	The comparison report.
`Images-baseline\`, `Images-destination\`	Option #1 with `--include-diagrams`	The images of the diagrams the report shows, drawn by EA from the two projects; with `--full`, also each unchanged diagram's image, in `Images-destination` only. (With Option #2 you provide these folders yourself.)
`<baseline>.xmi`, `<destination>.xmi`	Option #1	The two models exported from EA as XMI 1.1.
`<name>.xml`	Options #1 and #2	The comparison of the two models, in EA's compare log format. Tagged values are recorded as EA's Compare Utility records them: a `Tagged Values` item under the element, with one item per tag.
`<name>-enriched.xml`	all options	The comparison XML as prepared for the report: element kinds and statuses resolved, redlines computed, counts added. Useful for troubleshooting a report or for processing the comparison with other tools.
`<name>.zip`	`--zip`	The report and, with `--include-diagrams`, the two image folders.

With `--zip --cleanup` only the ZIP archive is left of what the run created; files you supplied are never deleted.

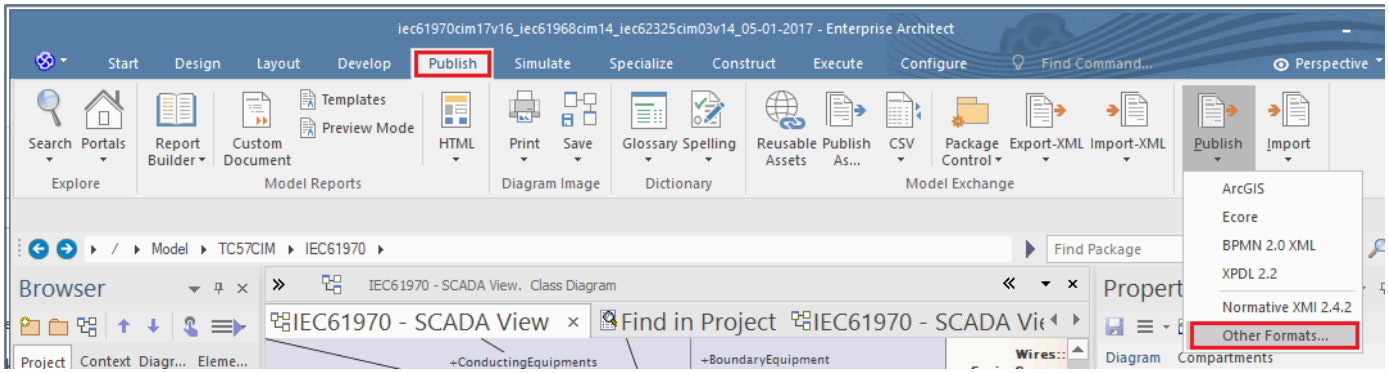
Enterprise Architect XMI Export Procedures

The procedure to perform an export of EA baseline and destination models as **XMI 1.1** compliant files is as follows (screenshots taken from EA v15.1):

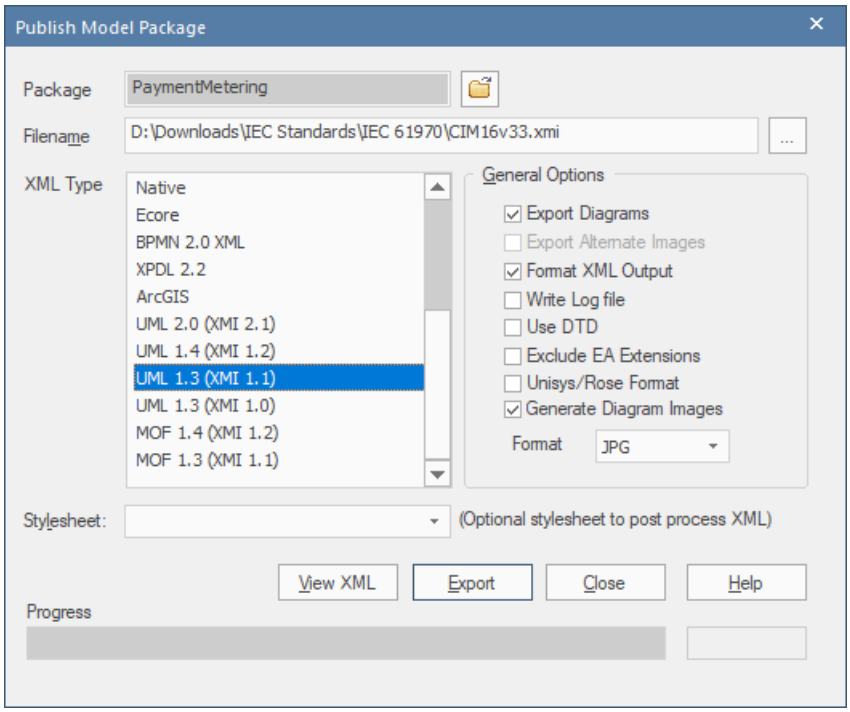
- In Enterprise Architect load the "baseline" (or older) model and select the top-level package of the model as shown.



- Once selected, choose the Publish menu and then the "Other Formats..." menu.



- The dialog box shown next will be presented to allow selection of the specific settings to use for export. Select the **"UML 1.3 (XMI 1.1)"** XML export type. This is the only export format supported for **cim-compare** comparison reports. It is important that the "Unisys/Rose Format" option **not** be checked. Finally, the inclusion of UML diagrams in the comparison report can optionally be selected at this time. This is done by additionally checking the "Export Diagrams" and "Generate Diagram Images" check boxes along with the selection of an image type (e.g. JPG, GIF, PNG, SVG, etc.) to export the diagrams in; SVG is available from EA 16.1 build 1621. Note that when including UML diagrams as part of the export EA will automatically create a directory called "Images" and export all image files into that directory. Note that when exporting an image type other than JPG the `--image-type` command line option described in the Command Line Usage section must be used.



- Finish out by selecting a target directory along with a file name for the "Filename" field.
- Click the **Export** button. Note that at this point it will take a while for the export to complete.
- If UML diagrams were included then upon completion of the export you should navigate to the export directory and rename the diagrams directory from "Images" to "Images-baseline". The naming convention is important and must match what is indicated here including case-sensitivity.
- Now close out of the baseline model load the destination model into EA and perform Steps 1 through 6 again but this time be sure to enter a different name into the "Filename" field for the destination XMI file to be exported. Upon completion the destination model's "Images" directory should be renamed to "Images-destination".

- The result should be a final export directory containing two images directories (i.e. “Images-baseline” and “Images-destination” containing the baseline and destination diagrams respectively) and two distinct XMI files to be used as input on the command line to **cim-compare**.

License

Distributed under the MIT license. See LICENSE for more information.

Third-Party Software

The `cim-compare-2.3.1.jar` includes the following third-party software:

- **Saxon-HE** 12.5 by Saxonica, used to produce the HTML report, distributed under the Mozilla Public License 2.0. Source code is available from github.com/Saxonica/Saxon-HE.
- **XML Resolver** 5.2.2, used by Saxon-HE, distributed under the Apache License 2.0.
- **Sparx Systems Enterprise Architect Java API** (`eaapi.jar`), used to export models from EA project files. See lib/README.md.

This site is open source. Improve this page.