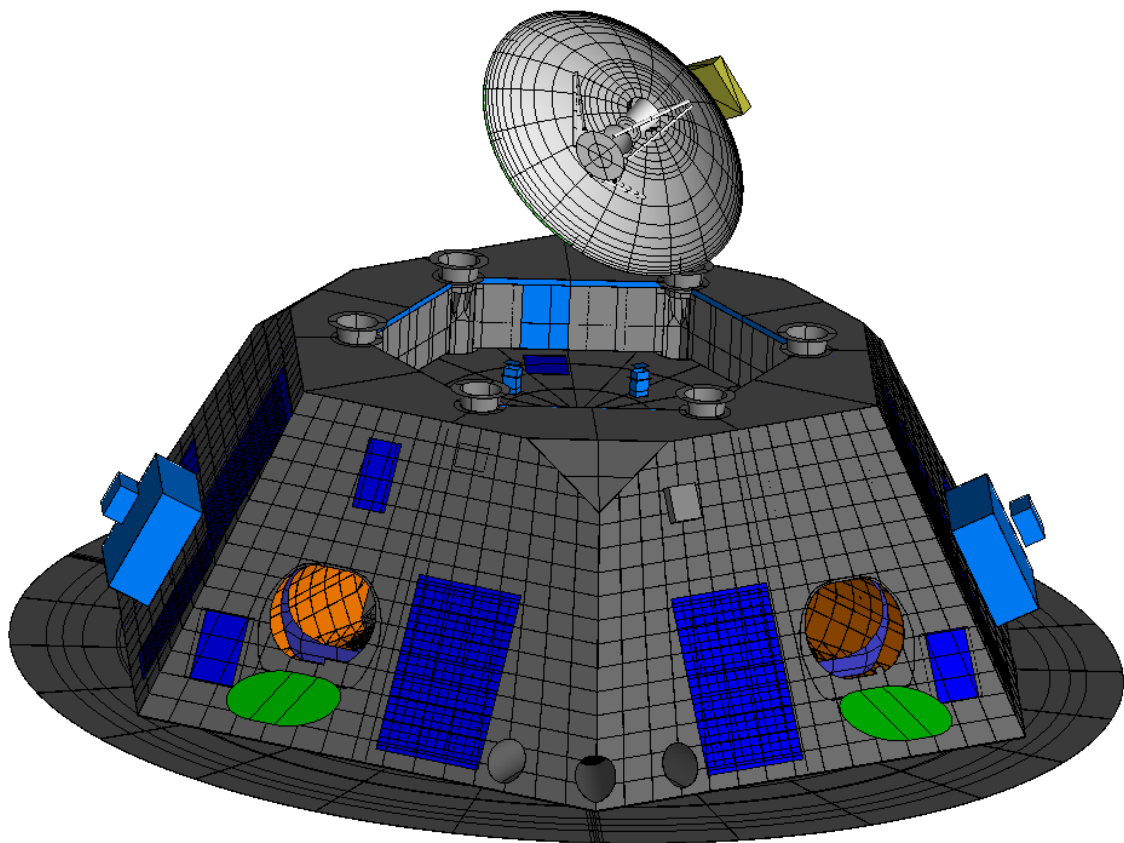


ESATAN-TMS

Release Notes



Front page picture courtesy of Thales Alenia Space Italy

Table of Contents

1	Introduction.....	4
2	Developments.....	5
	ESATAN-TMS 2025 sp1	5
	Analysis Case Optimised Thermal Solution Improvements.....	5
	Arrow to Indicate Secondary Body Position	6
	ESATAN-TMS 2025	7
	Multibody orbital modelling.....	7
	Python API	8
	Visualise point labels on point variables.....	13
	Enhancements to STEP-TAS export.....	13
	Frequency Solver Improvements	14
	Post Processing – Schematic Chart Layout	15
3	Problems fixed.....	16
	ESATAN-TMS 2025 sp1	16
	Workbench	16
	ESATAN-TMS 2025.....	17
	Workbench	17
	Thermal.....	24
4	Points to note	25
5	Migrate from previous version.....	27
6	Contact information	28

1 Introduction

ESATAN-TMS 2025 is another significant evolution of the product, providing new functionality that significantly enhances the thermal modelling capability of Workbench. The major new features are the completion of the Multibody orbital modelling extensions, a significant update to the Python API functionality and improvements to the Analysis Case optimised thermal solution.

The ESATAN-TMS 2025 development also delivers a number of feature enhancements that result from discussions with and requests from users.

The following sections describe the main changes introduced within the ESATAN-TMS 2025 sp1 release.

- **Analysis Case Optimised Thermal Solution Improvements** [<more detail>](#)
- **Arrow to indicate Secondary Body Position** [<more detail>](#)

The following sections describe the main changes introduced within the ESATAN-TMS 2025 release.

- **Multibody orbital modelling** [<more detail>](#)
- **Python API** [<more detail>](#)
- **Visualise point labels on point variables** [<more detail>](#)
- **Enhancements to STEP-TAS export** [<more detail>](#)
- **Frequency Solver Improvements** [<more detail>](#)
- **Post Processing – Schematic Chart Layout** [<more detail>](#)

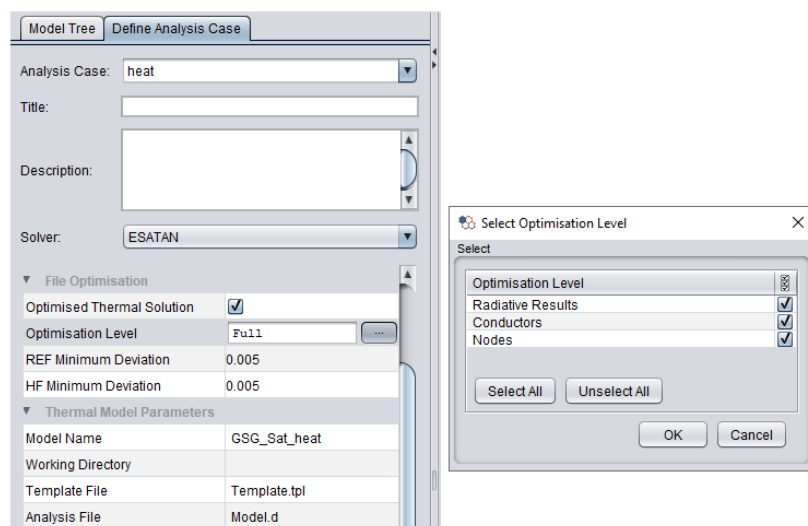
2 Developments

ESATAN-TMS 2025 sp1

Analysis Case Optimised Thermal Solution Improvements

The analysis case optimised thermal solution has been enhanced. Additional Thermal Model components can now be exported to the binary Analysis Case Definition (ACD) file, controlled by using a new optimisation level option.

This allows for greater reductions in the size of analysis files whilst helping to make the remaining content more readable.



Within the Analysis Case dialog's "File Optimisation" category there is now an additional "Optimisation Level" field. This field provides access to a dialog to select the optimisation level. There are the following three optimisation levels:

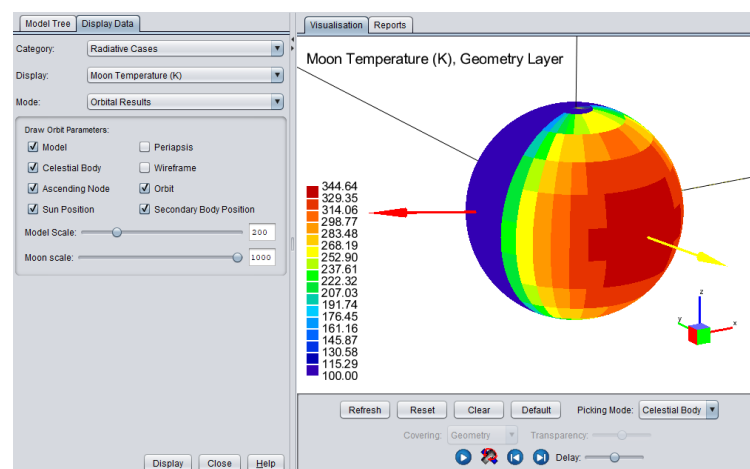
Optimisation Level	Description
Radiative Results	<p>The default optimisation level for analysis cases that are associated with one or more radiative cases when an optimised thermal solution is requested.</p> <p>The option places the results from the radiative case(s) GRs (obtained from REF calculations) and Heat Fluxes into the ACD file instead of the analysis file.</p>

Optimisation Level	Description
	This level of optimisation covers most of what was available in previous versions of ESATAN-TMS but with one important difference. Static GRs are now output to the ACD file instead of the analysis file.
Conductors	Outputs conductor definitions from sources other than radiative case results to the ACD file. This includes conductors generated from user defined conductor symbols, contact zones, conductive interfaces, insulation, intra-primitive and through thickness geometry conductors. Note: excludes conductors added directly to the template file.
Nodes	Outputs all fluid and thermal node definitions to the ACD file. Only the environment and inactive nodes definitions remain in the analysis file. Note: excludes nodes added directly to the template file.

Each optimisation level can be selected separately or in combination to achieve greater reductions in the overall size of the analysis file than possible in previous versions of ESATAN-TMS.

Arrow to Indicate Secondary Body Position

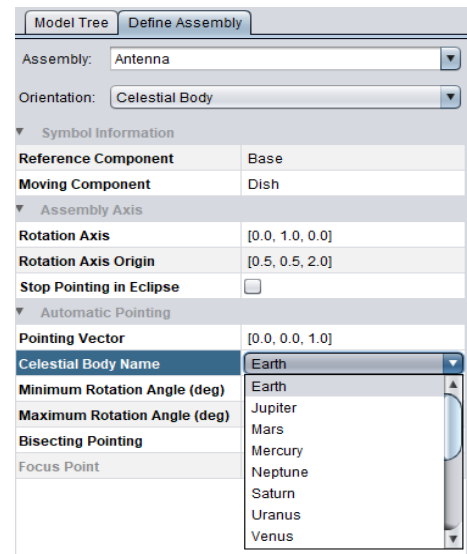
When visualising a moon-centred radiative case there is a new “Secondary Body Position” option that will draw a red arrow to indicate the direction of the secondary body (planet) with respect to the primary body. The existing “Sun Position” option, indicated by a yellow arrow, is now orbit position dependent. Both arrows will change when stepping through orbital results to show the current position as the orbit progresses.



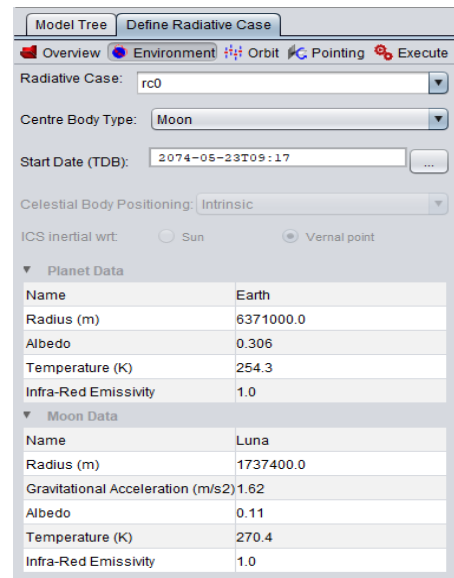
ESATAN-TMS 2025

Multibody orbital modelling

A planet or moon can now be specified as the target direction for both assembly pointing and spacecraft attitude control (main-body pointing). For example, a spacecraft orbiting Jupiter's moon Europa may have an antenna that is required to point continually toward Earth.



When modelling missions where the centre body is a moon, the radiative analysis now includes the blackbody and albedo heat fluxes from the parent planet (as well as from the moon), taking into account occultation of the planet by the moon.



The Application Programming Interface (API) has been extended to include the definition of a radiative case in which intrinsic celestial body positioning is employed, e.g. for moon-centred missions.

Python API

The embedded interpreter has been updated from Python 3.9 to 3.12, thus supports newer Python functionality such as structural pattern matching and type parameter syntax.

User support material has been improved. Along with the installation, there is now an API Guide (including two new example plugins), browsable HTML documentation of the API, and .pyi (Python Interface) stub files that can be used with an IDE (integrated development environment) such as VSCode to provide static type-checking and auto-completion.

```
x: API[MeshType] = get_attribute(
    name=1.0,
    attribute=ShellPropertiesAttribute.MESH_TYPE_1,
    row = 2)
```

etms.lang.get_radiative_times

```
etms.lang.get_radiative_times(radiative_case: str, spin: int /
    SpinPositions = SpinPositions.ALL, position: int / OrbitPositions =
    OrbitPositions.ALL) → etms.api_types.API[list[float]]
```

Retrieves a times series for a radiative case, aligned with the following (for the same parameters):

- get_radiative_face_hf()
- get_radiative_node_hf()
- get_radiative_face_factors()
- get_radiative_node_factors()

The returned series is orbit-major and spin-minor, i.e. [O1S1, O1S2, O1..., O2S1, ...]. Can only be called whilst a model is open.

- Parameters:**
- **radiative_case** (str) – Name of the radiative case.
 - **spin** (Union[int, SpinPositions]) – The spin position(s) at which to retrieve results.
 - **position** (Union[int, OrbitPositions]) – The orbit position(s) at which to retrieve results.

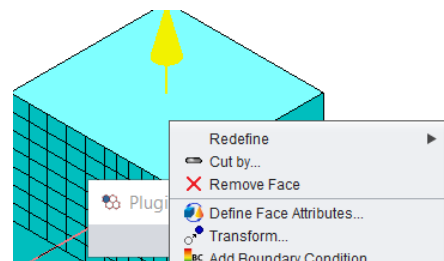
Returns: API result (The time series (orbit-major, spin-minor).)

Return type: etms.api_types.API[list[float]]

The value accessor of result objects returned from API calls now throws if not successful (as it is meaningless otherwise), this can be used as a built-in alternative to wrapping API function calls to throw exceptions. The prefix unary + operator `__pos__` has been defined as a shorthand for this, likewise `__bool__` (conversion to boolean) is now an alias of `success()`.

Before	After
<pre>xr = get_attribute(...) if not xr.success(): raise Exception(...) xv = xr.value yr = get_attribute(...) yv = yr.value if yr.success() else 1</pre>	<pre>xv = +get_attribute(...) yr = get_attribute(...) yv = +yr if yr else 1</pre>

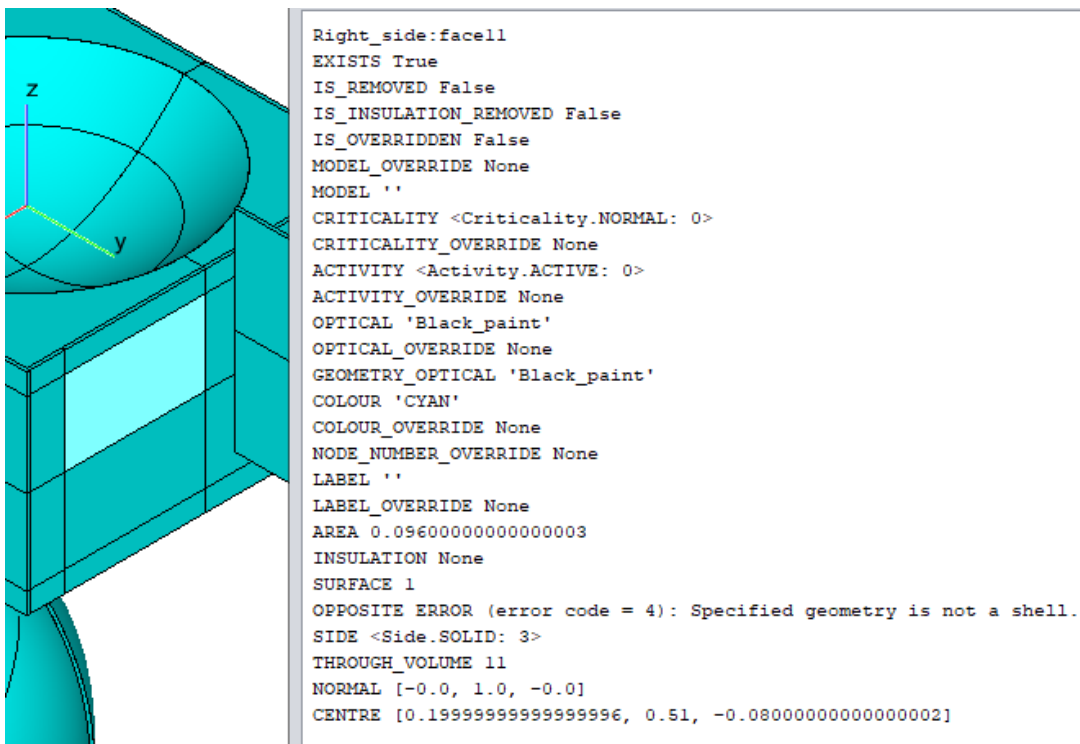
A batch mode has been introduced. During batch mode, multiple API calls are faster than before, by maintaining an exclusive lock. Outside of batch mode, Workbench is now more interactive than before – the visualisation is kept refreshed, and API dialogs no longer block the rest of the GUI. Plugins will be in batch mode by default, and can exit or re-enter it via `begin_batch` and `end_batch`.



A new function `evaluate_expression` provides support for evaluating real, integer and point expressions to literals (may be extended in future). This can be used on custom expressions, or for example on the result of many `get_attribute` calls.

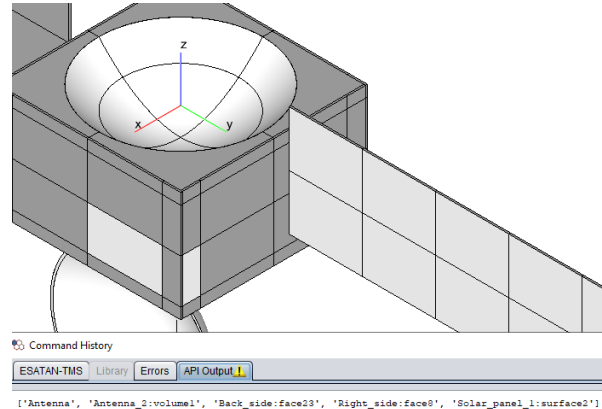
```
# x, y, z: float
[x,y,z] = +evaluate_expression(
    EvaluationType.POINT,
    +get_attribute('Shell', ShellByPointsAttribute.POINT4),
    default = [0., -1., 2.] # if unset
)
```

Face, surface and volume attributes are now supported by `get_attribute`. These include properties and overrides, along with geometric and topologic information such as the normal of a face or its corresponding surface number. In addition to this, `get_attribute` now has built-in support for expanded names and geometric references (when appropriate) as an alternative to parameters such as row or index.

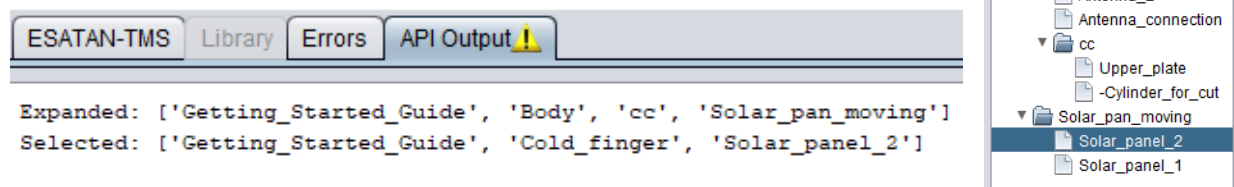


A new `set_attribute` function – currently supported only for description, and the points and parameters of geometry primitives – supports modifying individual attributes, without needing to re-define symbols wholesale. For primitives, this in most cases preserves face and volume overrides, surface properties, removed faces, removed insulation and hierarchy information.

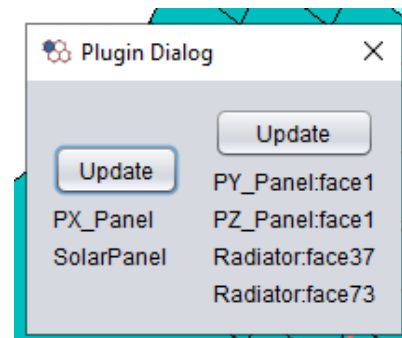
The most recent visualisation selection can now be retrieved as geometric reference strings using a function in the new module `etms.vis`. Together with the out-of-batch-mode dialog change, it is now possible for example to have the user supply input multiple times via visualisation selection whilst a plugin runs.



A tree component has been added to the GUI module. This can present arbitrary hierarchical data to the user – such as the model tree, part of a file system, or primitive mesh elements or properties. The selection and expansion states can be set and retrieved before and after the dialog is dismissed.

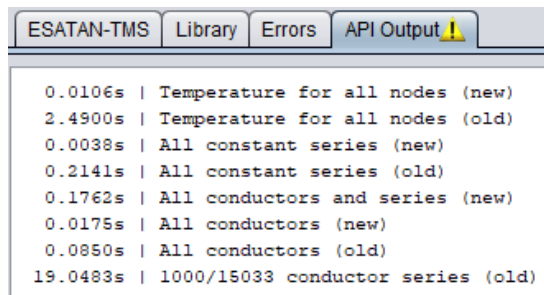


An abstract component class `CustomComponent` has been added to the GUI module. Subclasses of this run custom user code to generate a component to display, and update the component after the dialog is dismissed. Some smaller additional changes have been made to synergise with this: `Button` has a new `getRecentlyPressed` method, subclasses of `Button` should no longer be sliced when returned from `showDialog`, and `Dialog` will accept any component, not just `Panel`.



Use site	Custom component logic
<pre># Enable visualisation interactivity end_batch() p: Component = Picker(items = set()) q: Component = Picker(items = set()) # Custom components act like built-in components dia = Dialog(Panel([[p, q]])) # Re-display dialog until X is pressed while showDialog(dia) is not None: pass # Print user input print(p.items) print(q.items)</pre>	<pre>class Picker(CustomComponent): def __init__(self, items: set[str]) -> None: super().__init__() # essential self.items = items self.button = Button('Update') # Produce the displayed component def get_displayed(self): return Panel([[self.button], *([Label(i)] for i in self.items)]) # Update after display def post_display(self): if self.button.getRecentlyPressed(): self.items = +current_selection()</pre>

There have been several improvements to thermal results retrieval from TMD (thermal model data) files,



```
ESATAN-TMS  Library  Errors  API Output ⚠
0.0106s | Temperature for all nodes (new)
2.4900s | Temperature for all nodes (old)
0.0038s | All constant series (new)
0.2141s | All constant series (old)
0.1762s | All conductors and series (new)
0.0175s | All conductors (new)
0.0850s | All conductors (old)
19.0483s | 1000/15033 conductor series (old)
```

particularly performance and convenience. The new functions `get_thermal_`: `attribute_multi_series`, `constant_multi_series`, and `conductor_data` can retrieve many series much more quickly than making many calls to existing functions, and often more conveniently.

The node type parameter for `get_thermal_attribute_series` can now be deduced from the node passed in and has thus been made optional. `get_thermal_constant_series` has gained an additional overload taking a type parameter for enforcing the expected scalar type, improving the return type annotation.

```
# Specify type to validate constant type and aid type checker
timem: Sequence[float] = +get_thermal_constant_series(
    tmd_path = tmd, constant = 'TIMEM', type = ThermalConstantScalarType.REAL)

# The 'Temperature' attribute series for all nodes in the TMD (both types)
# Optional filters: nodes (superset of returned keys), type (node type - thermal or fluid)
temperatures: Mapping[NodeRef, Sequence[float]]
temperatures = +get_thermal_attribute_multi_series(
    tmd_path = tmd,
    attribute = 'Temperature',
    attribute_type = ThermalAttributeType.REAL)

# All conductors in the TMD, including their value and status series
```

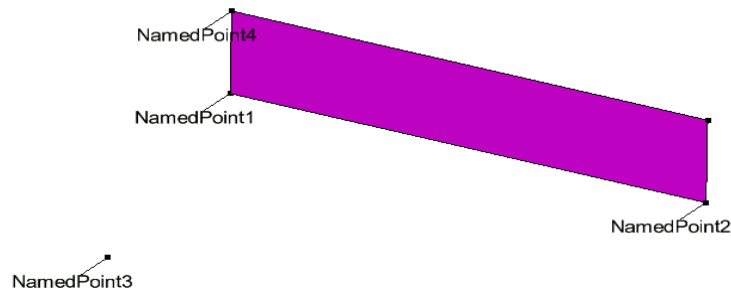
```
# Optional filters: types (conductor types), sources, destinations (conductor end nodes)
# Optional output controls: include_status, include_value (False if series aren't needed)
conductors: Mapping[ThermalConductorType, Sequence[ThermalConductorFullData]]
conductors = +get_thermal_conductor_data(
    tmd_path = tmd)
```

In addition the following minor improvements have been made:

- NodeRef can now be constructed from a single positional number argument e.g. NodeRef(5000)
- rc_execute now uses the settings saved on a radiative case for unsupplied parameters, thus can now be called with just the case as its argument e.g. rc_execute('Polar')
- ComboBox and RadioButtonGroup now accept an iterable of arbitrary Python objects as options, their string presentation can be customised by an optional parameter as_str (defaults to str). getSelectedItem will return one of the Python objects passed in, simplifying several use cases that previously would require getSelectedIndex or a dictionary lookup.
- Updates to the 3rd party library used to interact with Python mean that set parameters (get_radiative_...) now accept frozenset, and tmd_path parameters (get_thermal_...) now accept os.Pathlike arguments, such as pathlib.Path.

Visualise point labels on point variables

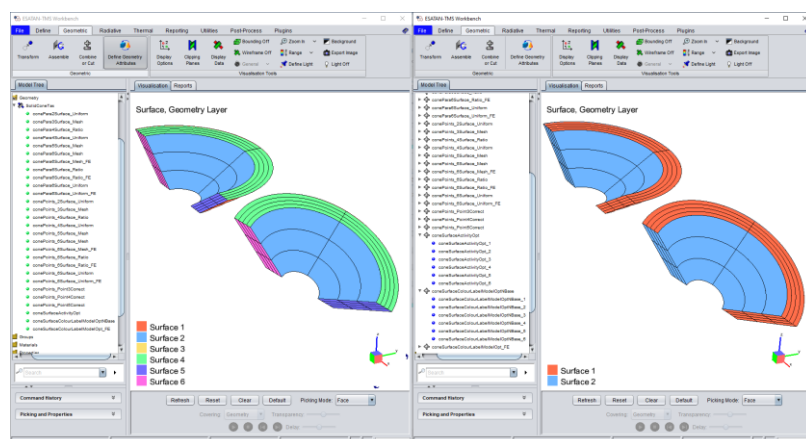
It is now possible to display the labels of named variable points in the visualisation, for example “NamedPoint1” to help identify point variables, including those used to define geometry. Point labels can be controlled through the model tree and the visualisation. Text size can also be controlled through the Display Options.



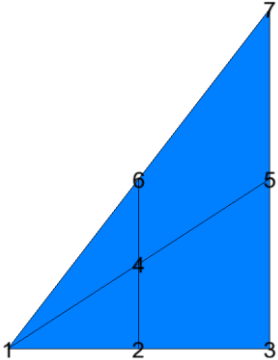
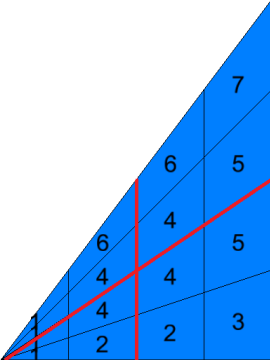
Enhancements to STEP-TAS export

The functionality for exporting data to STEP-TAS has been improved by introducing support for exporting both solid geometry and geometry with a finite element thermal mesh to STEP-TAS. Previous versions of the STEP-TAS converter omitted elements that fell into either of these categories.

There are no direct equivalents to the ESATAN-TMS solids in STEP-TAS. Instead, the surfaces of each solid are converted based on the type of geometry they represent to the equivalent STEP-TAS `mgm_meshed_primitive_bounded_surface`. Each converted solid can have between 1 and 6 surfaces. The converted surfaces are combined using a `mgm_compound_meshed_geometric_item`. This forms an equivalent representation of the original solid geometry with its surface attributes, meshing and node numbering.



There is currently no direct support for defining finite element meshes within STEP-TAS. To export a finite element mesh from ESATAN-TMS to STEP-TAS the exported geometry is modified by splitting each face in two exactly halfway along its two mesh directions. Each sub-face can then be assigned a single node number. This approximates the original finite mesh by converting it into an alternative lumped parameter form.

 <p>Example ESATAN-TMS finite element mesh on Triangle meshed 2 x 2. Node numbers are shown on the corner vertices of each face.</p>	 <p>Triangle converted to STEP-TAS. Each original face is split into 4 sub-faces. Each face has a single thermal node number. The red lines show the original mesh lines.</p>
---	--

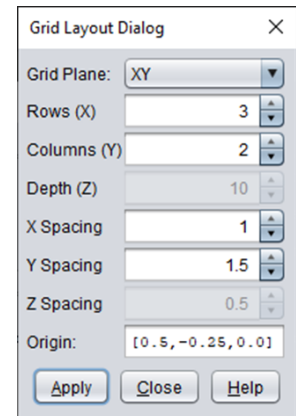
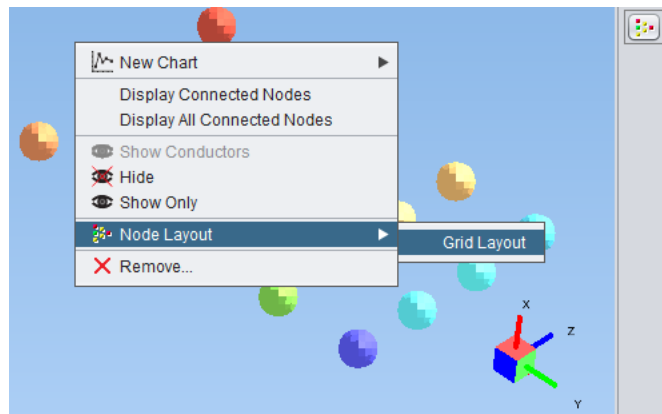
Frequency Solver Improvements

The following improvements have been implemented to ESATAN Thermal's frequency solver (SLFRTF):

- Make use of Fortran 90 Linear Algebra Package (LAPACK) to improve performance of the frequency response calculations.
- Add new library routines (SAVEFRQRESP & LOADFREQRESP) to save/load calculated frequency response data.

Post Processing – Schematic Chart Layout

The ability to reposition nodes and container elements on a Schematic chart has been improved with the provision of a new layout option. The user will be able to reposition either a selection of or all the elements on the chart using a 2D grid layout by selecting a grid plane. There are settings to control the starting coordinate, the spacing and ordering of the elements in the grid.



3 Problems fixed

ESATAN-TMS 2025 sp1

Workbench

-
- 1 Omit SpecialFace.(INACTIVE | DEEP_SPACE) from the results of flux-based HF retrieval in the Python API, as they lack well-defined area and return non-finite floating-point series.

 - 2 Python API function get_radiative_(face|node) factors gave 0 series for inputs not present in the radiative model instead of omitting them.

 - 3 Factors returned by Python API function get_radiative_(face|node) factors are now sparse (many zero series and sub-dictionaries will be omitted) to avoid very excessive memory usage and time taken for large numbers of faces or nodes.

 - 4 Python API function get_radiative_node_hf mixes up 99998 (inactive node) and 99999 (environment node).

 - 5 ESATAN-TMS exits unexpectedly generating conductors for a model containing insulation if the insulation submodel is not uppercase.

 - 6 ESATAN-TMS exits unexpectedly when deleting a real matrix which is referenced by a currently displayed radiative case.

 - 7 Execute radiative case progress bar appears blank when no calculations have been selected.

 - 8 Renaming all the geometry under a parent symbol is very slow with increasing memory usage.

 - 9 Moon faces cannot be picked when displaying the albedo or temperature overlays.

 - 10 Python API function get_attribute shouldn't extract name from a face/surface/volume reference when face/surface/volume is already supplied.

 - 11 Calculated moon temperature maps are not recalculated when the radiative case is attribute edited.

 - 12 Picking mode may be reset to distance after using transform dialog.

-
- 13** Transform dialog's distance field should be empty when first opened to allow for it being populated when a translation vector is picked.
-
- 14** The FLEXIm utilities provided with ESATAN-TMS on Linux do not support running on versions of Linux that are not compliant with Linux Standard Base.
-
- 15** Orbit positions are sometimes incorrectly flagged as being in eclipse for daytime only surface radiative cases.
-
- 16** Missing values for Albedo and Absorbed Planet flux at some orbit positions. Causing average flux values to report as not a number (nan).
-
- 17** Surface radiative cases using daytime only produces smaller than expected final angle.
-
- 18** The primary and secondary pointing vectors of surface radiative cases are not output to the log file, when defined without using language, or exported when using export model.
-
- 19** Some finite element triangles do not display correctly in the visualisation.
-
- 20** ESATAN-TMS exits unexpectedly when reporting REFs or VFs for a model with no radiative surfaces.
-
- 21** ESATAN-TMS exits unexpectedly when referring to a half space shell in a geometric reference.
-

ESATAN-TMS 2025

Workbench

-
- 22** Convective conductor does not allow thermal non-geometric nodes as source reference for wall-to-fluid option.
-
- 23** Defining a paraboloid by points can give spurious coincident point messages.
-
- 24** Node reference to FE vertex treated differently by group overlay.
-
- 25** Reported solar constant changes after radiative case is run.
-

-
- 26** Multiple calls to the DEFINE_MISSION language statement may reset parameters to default values.
-
- 27** Chart element reference error message is too general.
-
- 28** Analysis case output goes to wrong monitor window when pre-processing fails.
-
- 29** Naming suffixes applied to bulk properties exported to ESATAN can be repeated when generating analysis files from STEP-TAS models with anonymous material properties.
-
- 30** Typing multiple characters after moving the cursor in the Radiative Case dialogs title or description fields causes cursor to jump to end of text.
-
- 31** Deleting all elements from the list in the cavity dialog causes an exception when apply is clicked.
-
- 32** Point variable does not display when shown if it was hidden multiple times.
-
- 33** Albedo map cannot be displayed for moon centred cases.
-
- 34** Displaying a TMD file from an analysis case with a non-existent working directory generates an exception.
-
- 35** Radiative Case Albedo and Temperature sub dialog layouts inconsistent.
-
- 36** Radiative Case dialog Albedo and Temperature fields behave incorrectly when changing planet or moon.
-
- 37** Expand as faces includes cutting tools.
-
- 38** Radiative case language failed to warn user when the rotate map parameter was provided without a start date.
-
- 39** Setting the REF multiplier using a real defined in the sub dialog will not be applied on the analysis case.
-
- 40** Hide un-editable radiative case fields.
-
- 41** Memory leak in material orientation overlay.
-
- 42** Displaying GFF orbital results overlays showing all positions in a custom time interval selection causes ESATAN-TMS to exit unexpectedly.
-

-
- 43** Analysis cases chaining interbody radiative cases always use the same celestial body image.
-
- 44** Eclipse Entry Exit Points flag cannot be set on radiative cases with Moon centre type.
-
- 45** Ephemeris files fail validation check on Linux if Celestial Body Positioning is Intrinsic.
-
- 46** Memory leak calculating VF/REF using statistical accuracy control.
-
- 47** Define Geometry Attributes language validation of submodel too permissive.
-
- 48** The Python API description attribute does not support view symbols.
-
- 49** Restoring insulation on geometry then adding an insulation reference to it on a schematic chart generates an exception.
-
- 50** Make default Python API plugin more helpful.
-
- 51** Orbital arcs not displayed for certain combinations of initial and final anomalies.
-
- 52** Analysis Case monitor sometimes outputs either incomplete text or nothing at all.
-
- 53** Radiative Case does not range check last row and column of the albedo or temperature matrices.
-
- 54** Changing moon case fields in the Radiative Case dialog does not update other fields.
-
- 55** Picking mode is reset to Geometry if the Define Geometry Attributes dialog is closed without changing selection type.
-
- 56** LOCS orientation enabled for intrinsic cases.
-
- 57** Radiative Case Planet or Moon Temperature and Albedo sub-dialogs are not validated when opened.
-
- 58** Geometry dialog does not update a Solid that has surface attribute changes.
-
- 59** When a non-default optical property environment is set, selecting a face with optical that references that property environment causes ESATAN-TMS to exit unexpectedly.
-
- 60** An exception occurs selecting a point in the model tree that has been declared but has not been assigned a value.
-

-
- 61** Defining an intrinsic radiative case without the planet_name field can cause ESATAN-TMS to exit unexpectedly.
-
- 62** Radiative Case start date, right ascension and declination fields are not set correctly when planet changed.
-
- 63** Define Geometry dialog does not warn the user that properties changed at the surface level may be lost before applying the changes.
-
- 64** Entering an invalid start date for a radiative case via the language causes ESATAN-TMS to exit unexpectedly.
-
- 65** Python API function get_attribute fails for node delta attributes.
-
- 66** Geometry dialog allows literal opticals with negative properties when the properties sum to 1.0.
-
- 67** Literal bulk expressions do not validate property symbol dependencies.
-
- 68** Surface Radiative Case sunset/sunrise not taken into account.
-
- 69** Pipe constituent assignment is allowed in cases that should be considered invalid causing ESATAN-TMS to exit unexpectedly.
-
- 70** The Python API CavityAttribute.TYPE attribute is ignored.
-
- 71** ICS coordinate system "inertial with respect to" option available for Sun centred radiative cases.
-
- 72** Memory leak defining primitive geometry.
-
- 73** Shell disc parameters angmin and angmax are not being validated. Entering invalid values could cause ESATAN-TMS to exit unexpectedly.
-
- 74** Calling the Python API get_attribute function and requesting the attribute NonGeometricNodeAttribute.MODEL_NAME for a shell causes ESATAN-TMS to exit unexpectedly.
-
- 75** Python API functions get_radiative_face_factors and get_radiative_node_factors sometimes return empty lists instead of an all zero series.
-
- 76** Defining a Solid Quad Rotate with a small non-zero angular extent causes ESATAN-TMS to exit unexpectedly.
-

-
- 77** Modifying a property of a pipe or pipe bend can cause ESATAN-TMS to exit unexpectedly.
-
- 78** Python API function `get_geometry_faces` ignores conductive faces.
-
- 79** Python API function `remove_insulation` incorrectly succeeds for the face of some uninsulated surfaces.
-
- 80** Attribute editing the height of a shell rectangle was not updating the surfaces of the shell.
-
- 81** Python API function `generate_analysis_file` doesn't always write `GENERATE_ANALYSIS_FILE` language to the log file.
-
- 82** An exception occurs when clicking on the visualisation window with no model loaded and picking mode set to Point.
-
- 83** Radiative Case end date is not being validated.
-
- 84** Python API functions `get_geometry_faces` and `get_geometry_nodes` do not support FE vertex references or solid volume nodes.
-
- 85** Temperature and Albedo maps for Moon centred orbits do not get reported.
-
- 86** Python API functions `point_vector` and `point_matrix` report errors on success.
-
- 87** When 'HF Output Format' is set on an analysis case to 'S+A+P' summed radiative results, HF values are applied to QS nodal heat source values in the analysis file. If 'HF Multiplying Factors' are also applied on the analysis case, the factors are incorrectly applied to QR nodal heat source values.
-
- 88** Editing an analysis case through the dialog can cause its 'HF Output Format' to be reset to the default value.
-
- 89** Post processing "Sum Total Internal Heat Source" attribute missing from the container attributes of node-based charts.
-
- 90** Python API function `rc_execute` does not update radiative case execution settings.
-
- 91** Visualisation transformations are incorrectly applied in certain cases to points or geometry outside of the model.
-
- 92** Convective conductor allows selecting calculated area for a group containing a non-geometric node.
-

-
- 93** Non-fatal exception generated whilst closing model with a Schematic chart displayed.
-
- 94** Point corrections applied to Solid Quad Rotate can lead to incorrectly formed geometry.
-
- 95** DEFINE_GEOMETRY_ATTRIBUTES language statement and equivalent Python API function permit invalid surface numbers.
-
- 96** Point matrix and vector elements are not listed in the Point dialog's symbol name list.
-
- 97** Changing conductor text attribute in Schematic Chart causes an exception to occur.
-
- 98** Sun centred radiative case treated as Planet centred when defined through language.
-
- 99** Defining an analysis case with a parametric variable that refers to itself causes ESATAN-TMS to exit unexpectedly.
-
- 100** Defining insulation through dialog does not validate negative values in a literal optical.
-
- 101** Applying a 'HF Multiplying Factor' to a face can cause ESATAN-TMS to exit unexpectedly if the face activity has previously been switched from 'Conductive' to 'Inactive'.
-
- 102** Model tree collapsing when symbols are deleted or renamed.
-
- 103** When an Optimised Thermal Solution (ACD) model contains time-dependent internal heat source applied as QR, the application of the heat source in \$VARIABLES2 is overwritten at the start of \$VARIABLES1 when the current radiative results are loaded from the ACD file.
-
- 104** Python API float input for some real parameters of conductors limited to 2 decimal places.
-
- 105** Python API calling get_attribute (with PrimitiveAttribute.Solidity) on a non-geometric node or a half space incorrectly returned Shell – it now returns None.
-
- 106** Assembly validation incorrectly prevents redefinition of model containing just two components.
-
- 107** Python API function get_geometry_faces fails when called with a non-geometric node, and sometimes wrongly lists a "non-geometric thermal node's face" when its number is referenced.
-
- 108** ESATAN-TMS to SINDA translator option fails translating model, due to model name incorrectly being treated as ".".
-
- 109** Python API get_thermal_attribute_series succeeds with spurious result for mismatched attribute, node and supplied type.
-

-
- 110** Editing attribute HMIN of a solid paraboloid is not properly validated.
-
- 111** Shell cylinder defined by circle does not ensure thickness is set when a bulk is set in the dialog.
-
- 112** Log file language generated from using Include Model through the dialog does not set the parameter use_model_transformations to TRUE when it has been applied.
-
- 113** Thickness parameters of shell geometry should not accept property symbols.
-
- 114** Entries in right click menus that exceed the screen height cannot be selected. Provide smaller scrollable menus when there are a variable number of elements on a menu.
-
- 115** Model tree collapses after clicking apply in Radiative Case dialog when an associated Analysis Case was open.
-
- 116** Radiative results considered invalid and are deleted when model is reopened.
-
- 117** Export model is slow for large models.
-
- 118** Delays updating dependencies of an open analysis case.
-
- 119** Analysis case dialog validation is very slow after clicking Apply for an analysis case with several thousand boundary conditions.
-
- 120** Avoid saving literal values as expressions to speed up model saving.
-
- 121** Avoid repeated group/cavity validation to speed up certain operations including model load.
-
- 122** Validation errors are not shown if a HF multiplier or thermostat reference is set to inactive geometry.
-
- 123** A valid Solid Torus defined by parameters maybe incorrectly rejected.
-
- 124** Redefining a pipe as an assembly then attempting to interact with the orphaned symbols can cause ESATAN-TMS to exit unexpectedly.
-

Thermal

-
- 1 Error attempting to call Thermal library routine DMPFR (to dump frequency response) two or more times in succession.
-
- 2 Error setting external subroutines (in global data file) with names greater than 6 characters.

4 Points to note

-
- 1 The developments undertaken for ESATAN-TMS 2025 have required an update to the model store format. Models will therefore be updated from the ESATAN-TMS 2023 and 2024 formats when first opened in ESATAN-TMS 2025.

Due to changes in ESATAN-TMS 2025 it will be necessary to rerun some Radiative Cases after upgrading the model to regenerate the results.

-
- 2 The embedded Python interpreter used by the API has been updated from Python 3.9 to Python 3.12. Hence an installation of Python 3.12 is required to use the API with ESATAN-TMS 2025, and 3.9 is no longer required – consult the installation notes. Plugins that make use of deprecated Python functionality may need updating.

Installation errors can occur when attempting to install Python 3.12 on a Windows machine that already has Python 3.9 installed. To avoid these errors, it is advisable to uninstall Python 3.9 before installing Python 3.12.

-
- 3 Passing most Python API parameters positionally has been deprecated, except for “obvious” or memorable cases, e.g. the name of a symbol being re-defined being the first parameter. Plugins passing these parameters positionally should pass them by keyword instead before this is enforced. Consult the documentation to see which positional parameters are still supported.

There have been several other minor deprecations. The parts deprecated are not part of the new documentation – consider setting up a Python IDE (integrated development environment) such as described in the new API Guide to detect discrepancies between plugins and the documentation due to deprecation.

-
- `delete_file` (will be removed): use appropriate standard library functionality, such as `os.remove`
 - `generate_analysis_file`'s `boundary_conditions_qr` parameter (will be removed): this should be specified per boundary condition
 - `setCIConnectionType`'s `connect_type` parameter (will no longer support `str`): instead supply the appropriate `ConnectType` constant.
 - `ShellByDirectionAndCircleAttribute`'s `DIRECTION4` constant (will be removed): this should not be used; no geometry has a 4th direction parameter.
 - Uses of the record types `ThermalParameter` and `ThermalConductorData` beyond reading their attributes has been deprecated and will be removed. This includes construction,

comparison and modifying. Consider reading the attributes into a tuple or dataclass if needed.

- The optional pipe(_bend) parameter constituent_prefix has been moved, this is a breaking change for calls to pipe(_bend) functions relying on positional arguments other than the name of the pipe(_bend) to define. Calls to pipe(_bend) functions should be updated to pass most arguments by keyword instead.

-
- 4 The Fluid Non-Geometric Node's heat-transfer area parameter has been removed from ESATAN-TMS Workbench as it was not being used.
-

- 5 The use of Properties in the definition of Bulk materials has been changed to disallow cases that could lead to unrealistic models. The density attribute can no longer be defined using properties. The conductivity and specific heat attributes now only allow temperature dependent properties. The use of literal values or expressions to define Bulk material attributes remains unchanged.
-

- 6 ESATAN-TMS Thermal no longer supports so-called Virtual submodels, hence the \$MODEL statement:

```
$MODEL name [,REAL | VIRTUAL] [,GLOBALFILE =myglobalfile]
           [,STANDARD | VCHP] [,TEMP = tdesig] [,ACDFILE]
```

no longer accepts the 'VIRTUAL' keyword. The 'REAL' keyword (default) is still accepted but is essentially redundant.

- 7 The version of the FLEXlm licence manager used by ESATAN-TMS 2025 sp1 has been updated. Users will need to ensure they are running a licence server using the latest version of the FLEXlm utilities supplied with the ESATAN-TMS software. Attempting to use an older version of the utilities to licence the software will fail with an error message due to the age of the vendor daemon. The new licence utilities will work with all versions of ESATAN-TMS.

5 Migrate from previous version

ESATAN-TMS Workbench provides a smooth upgrade path from ESATAN-TMS 2023 and ESATAN-TMS 2024.

On launch of Workbench, ESATAN-TMS checks the version of the associated user-library file (if present) and prompts to update the library. On opening the model, Workbench checks the model version number and, if a model from one of the versions listed above is detected, the option is given to automatically update the model to the new version.

Note that the upgrade of the model and the user-library is a permanent update and therefore after the update the files will not open in the previous version. It is therefore recommended that a copy of the model and the user-library file be made before proceeding with the update.

Please also note that due to changes in ESATAN-TMS 2025 it will be necessary to rerun some Radiative Cases after upgrading the model to regenerate the results.

6 Contact information

If you have any questions regarding the new version or require further information, please contact our customer support at:

ESATAN-TMS User Support



+44 (0)116 284 5748



support@esatan-tms.com



www.esatan-tms.com

ESATAN-TMS is a trademark of ITP Engines UK Ltd.