ICES-2020-297

Extending the Capabilities of Thermal Desktop with the OpenTD Application Programming Interface

Hume L. Peabody[1]
NASA-GSFC, Greenbelt, MD 20771

With the release of Thermal Desktop 6.0, users now had the ability to interface with some of the many elements and constructs of a Thermal Desktop model through external applications developed using the TD API (Application Programming Interface). This file allows applications to be developed in the .NET framework and interface to a number of object types within a Thermal Desktop model. The release of 6.1 expands the subset of objects able to be manipulated and now includes the raw geometrical information of surfaces. With the release of 6.1, the API was now referred to as OpenTD. This paper discusses some of the utilities and capabilities developed using the OpenTD API at the NASA Goddard Space Flight Center. These include utilities to help with configuration control of models and case sets, addition of logic to better process heater performance, and a methodology implemented to allow for submodel level processing of radiation couplings to include smaller radks where needed in a cryogenic region without using the same criteria for the warmer portions of the model. This last utility is targeting a reduction in run time without sacrificing accuracy. Lastly, some lessons learned, work-arounds, and wishes for the next release of the OpenTD API are also presented.

Nomenclature

ASCII	= *American Standard Code for Information Interchange*	*MLI*	= *Multi-Layer Insulation*
API	= *Application Programming Interface*	*NASA*	= *National Aeronautics and Space Administration*
GMM	= *Geometric Math Model*	*SINDA*	= *Systems Integrated Numerical Difference Analyzer*
GSFC	= *Goddard Space Flight Center*	*TMM*	= *Thermal Math Model*
GUI	= *Graphical User Interface*	*UV*	= *UltraViolet*
IR	= *InfraRed*	*VB.NET*	= *Visual Basic.NET programming language*

I. Introduction

Thermal Desktop® is an analysis tool commonly used by NASA-GSFC for the thermal modeling of spacecraft and instruments. It utilizes the AutoCAD program as the front end and graphical user interface to allow analysts to construct geometric math models (GMM, which are used to compute radiative exchange factors and radiative heatloads from external sources) as well as generating a network thermal math model (TMM, which is solved to predict temperatures). Thermal Desktop is one tool in a suite of tools from the vendor that allows analysts to tailor the tools they need for particular analysis types. Thermal Desktop provides for the front end building of thermal models and post processing of thermal results; RadCAD extends the capabilities to include Monte Carlo ray tracing computations for radiative exchange and orbital heating; SINDA/FLUINT solves the thermal network model to generate temperatures; FloCAD allows users to construct graphical representation of flow entities in Thermal Desktop to construct coupled fluid-thermal models.

However, being in AutoCAD does come with a limitation; the thermal model data is stored in a proprietary, binary format and is not directly accessible to the end user unlike other tools that use an ASCII format to store model data. This limitation inhibits the ability of end-users to develop additional tools and capabilities that can interface directly with the thermal model data. To address this limitation, the software vendor has recently added an application programming interface (API) that exposes selected data to the user through a dynamic linked library that interfaces with an AutoCAD application instance with the specified drawing file open.

The original introduction of this capability was provided with the 6.0 release of Thermal Desktop (known as the TDAPI) and was limited to a fairly large subset of entity types. These included: Nodes, Lumps, Paths, Ports, Pipes,

[1] Staff Thermal Engineer, Mail Stop 545, Goddard Space Flight Center, Greenbelt MD 20771.

Ties, Optical Properties, ThermoPhysical Properties, CaseSets, Symbols, Heaters, Heat Loads, Layers, and Submodels. Notably missing from this list, however, are surfaces which form the heart of both the GMM and TMM data. The original TDAPI was developed to aid in the conversion of SINAPS models into Thermal Desktop as the software vendor was retiring the SINAPS application around the same time, since most of the SINAPS capabilities could now be duplicated using the aforementioned FloCAD application. Examining the list of supported entities shows a close alignment with the fluid constructs that would be common in a SINAPS model.

The release of Thermal Desktop v6.1 added access to the surface data and the API was renamed as the OpenTD API[1,2]. This paper describes some of the features in the API, how to access them, and some utilities that were developed in Visual Basic .NET at NASA-GSFC using the API. These include utilities to better evaluate heater performance, capture model documentation information, and further filter radiation coupling terms for thermal model run time optimization.

II. Basic API Structure

The API is accessed by including a reference to the OpenTDv61.dll (or alternatively the TDAPIV1.dll if using 6.0) in the project. The OpenTD API was developed in C#.NET and is compatible with any other languages that are compatible with the .NET framework. Including the reference allows for variables to be created based on the OpenTDv61 namespace. To establish a connection with Thermal Desktop, a variable of `ThermalDesktop` type is created (e.g. `Dim TD as New OpenTDv61.ThermalDesktop`). Once created, the drawing file to be accessed is specified by setting the `TD.ConnectConfig.DwgPathname` property. Following this with a call to the `TD.Connect()` method establishes the link with either an existing AutoCAD application that is already open with the specified drawing file or creates a new instance in which the specified drawing file is subsequently opened. At this point, all of the exposed data is now available for further investigation or manipulation.

Further variable types may be created on the program side to hold data retrieved from the model in Thermal Desktop. For example, declaring a variable as a list of CaseSet type [`Dim CaseSets As List(Of OpenTDv61. CaseSet)`] allows the variable to be populated by all the CaseSets in the model via a call to the `GetCaseSets` method [`CaseSets = TD.GetCaseSets()`]. Alternatively, if a CaseSet name is already known, a variable to hold the CaseSet can be declared [`Dim MyCaseSet As OpenTDv61.CaseSet`] and then populated with a call to the `GetCaseSet` method [`MyCaseSet = TD.GetCaseSet("Cold Case")`]. It should be noted that the CaseSet name is expected to be unique and is the identifier by which the API identifies the CaseSet. If two CaseSets are identically named, the results are uncertain as to which one would be associated with the variable. If a CaseSet variable is modified on the program side, any updates must be committed to the Thermal Desktop application by a call to the `Update` method [`MyCaseSet.Update()`]. If a new CaseSet is to be created, the `CreateCaseSet` method should be used with a unique name as the argument [`MyCaseSet = TD.CreateCaseSet("New_CaseSet")`]. Lastly, to delete a CaseSet, the `DeleteCaseSet` method should be used [`TD.DeleteCaseSet("New_CaseSet")`]. Similar Get---s, Get---(), Create---(), and Delete---() methods exist for most of the other object types listed above (e.g. optical property, symbol, etc). The basic process for making changes to a model would be the same regardless of the entity type; (1) retrieve the object from the AutoCAD instance and store in the appropriate variable type, (2) make the desired updates to the variable, and (3) commit the changes to the AutoCAD instance with the `Update` method. If data is only being retrieved from the AutoCAD instance and is then manipulated and organized only on the program side, then the last `Update` step is not necessary.

To access surface properties, a different method is employed since the Get/Set type methods currently do not exist to retrieve all geometric surface type (e.g. rcCone). Instead, a generic approach utilizes the `IDbIterator` object to step through the AutoCAD database and retrieve the objects sought. To accomplish this, a variable of `IDbIterator` type must first be created and set using a call to the `CreateDbIterator` method of a `ThermalDesktop` object. Next, variables and lists of variable types sought should be declared. Then, a WHILE loop should be established that continues until the `.Done` method returns a true value. While in the loop, the object type sought should be set via the `DBObject` call to the `IDbIterator` variable. If this returns back a value that is not null (Nothing), then the current object meets the type sought and it can be extracted and added to a List Of type variable or modified directly. Lastly, a call to the `.Step` method advances to the next object in the database. A snippet of code is shown in Figure 1 that would retrieve all rcRectangle type objects and store them in a List of rcRectangle.

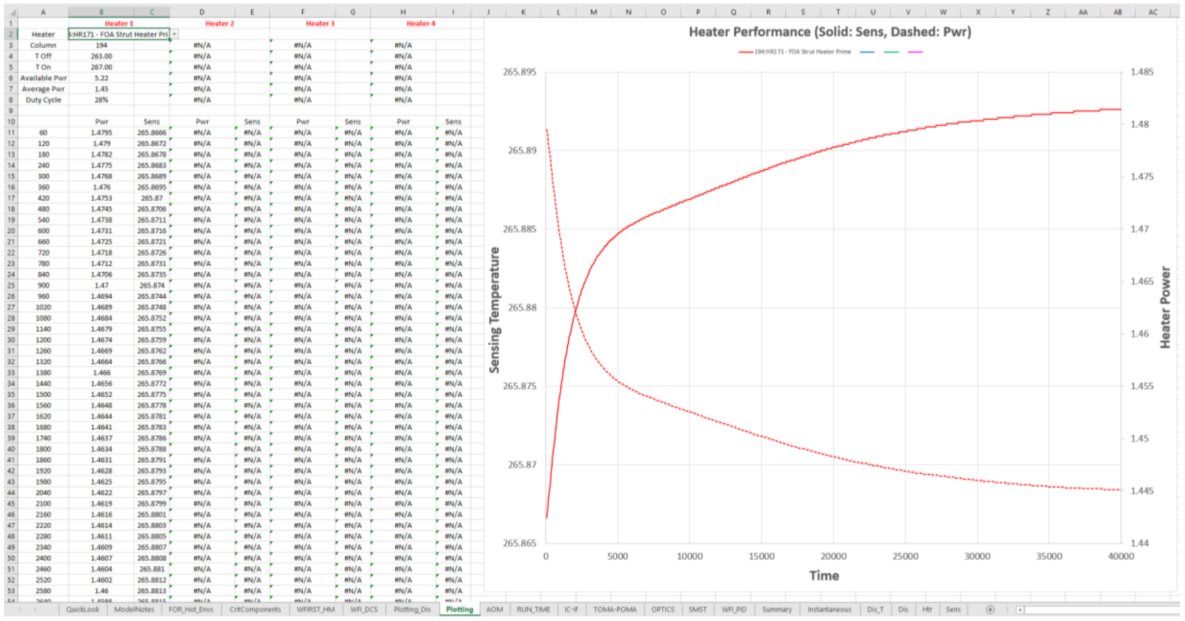
These functions and capabilities have already been utilized to develop utilities to assist with model modification, model documentation, and filtering of radiation couplings and will be used to develop future capabilities as additional needs are identified. Users interested in exploring the OpenTD API are encouraged to begin with the "Getting Started with OpenTD 61.pdf" file[2] included with the Thermal Desktop installation.

```
Dim DB As OpenTDv61.IDbIterator
Dim Rectangles As List(Of OpenTDv61.RadCAD.Rectangle)
Dim Rectangle As OpenTDv61.RadCAD.Rectangle
DB = TD.CreateDbIterator
While Not DB.Done
  Rectangle = DB.DbObject
  If Not IsNothing(Rectangle) Then Rectangles.Add(Rectangle)
  DB.Step()
End While
```
Figure 1. Code snippet to retrieve all rcRectangles: *Accessing geometric entities requires stepping through the AutoCAD database to find objects matching the type sought*

III. Evaluating Heater Performance

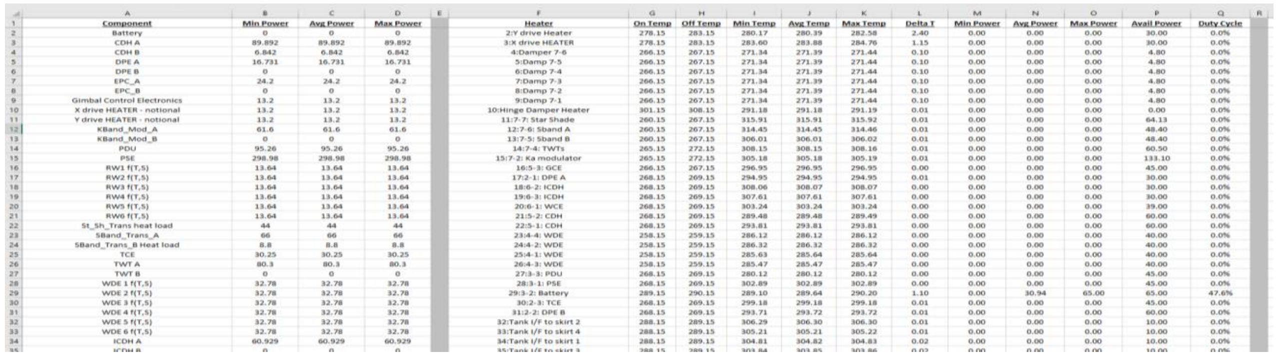
Heaters are a common entity type in thermal analysis. However, capturing the performance of a heater throughout a thermal analysis simulation can be challenging. Typically, predictions from a transient simulation are output periodically, but it is highly unlikely that this output cadence will align with each and every heater in a model. Therefore, it is preferable to output each heater's critical data (sensing temperature, power state) every timestep. Unfortunately, this is not a native capability provided by commercial thermal solvers at this time. To varying degrees, commercial solvers track some metrics and provide a composite value at the end of the run, such as number of cycles, average power, peak temperature, etc. However, this data may be insufficient for evaluation of tight stability requirements, in particular over varying ranges of time (e.g. stability over 180 s, over 2 hours, etc).

A routine was developed to evaluate the CondCap file produced by Thermal Desktop and extract heater information, including: name, on point, off point, available power, sensing temperature, and applied power, with the latter two varying over time. Based on this processing, SINDA logic is generated that outputs the time varying heater parameters (sensing temperature and applied power) at each timestep and tags the output lines with a prefix to allow easy extraction of the relevant lines from the output file. This logic file is then included in the TMM input file prior to model execution. A similar capability was also developed to process the heat load dissipations and the average temperature where the load is applied. A corollary routine was developed to extract these lines based on the aforementioned tags and import the data into Microsoft Excel® for further processing. The data is imported into a new workbook based on a standard template that includes additional worksheets to facilitate plotting (Figure 2), tabular summaries of the heat loads (Figure 3) and the total power usage for the model. One considerable advantage of this approach is that it shows *exactly* what is in the TMM file. For large and complex Thermal Desktop models, it can be difficult to fully understand the configuration in a CaseSet with all the dependencies on symbols and logic.



	Heater 1		Heater 2		Heater 3		Heater 4	
Heater	:HR171 - FOA Strut Heater Pri							
Column	194		#N/A		#N/A		#N/A	
T Off	263.00		#N/A		#N/A		#N/A	
T On	267.00		#N/A		#N/A		#N/A	
Available Pwr	5.22		#N/A		#N/A		#N/A	
Average Pwr	1.45		#N/A		#N/A		#N/A	
Duty Cycle	28%		#N/A		#N/A		#N/A	
	Pwr	Sens	Pwr	Sens	Pwr	Sens	Pwr	Sens
60	1.4795	265.8666	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
120	1.479	265.8672	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
180	1.4782	265.8678	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
240	1.4775	265.8683	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
300	1.4768	265.8689	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
360	1.476	265.8695	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
420	1.4753	265.87	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
480	1.4745	265.8706	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
540	1.4738	265.8711	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
600	1.4731	265.8716	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
660	1.4725	265.8721	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
720	1.4718	265.8726	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
780	1.4712	265.8731	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
840	1.4706	265.8735	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
900	1.47	265.874	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
960	1.4694	265.8744	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1020	1.4689	265.8748	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1080	1.4684	265.8752	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1140	1.4679	265.8755	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1200	1.4674	265.8759	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1260	1.4669	265.8762	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1320	1.4664	265.8766	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1380	1.466	265.8769	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1440	1.4656	265.8772	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1500	1.4652	265.8775	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1560	1.4648	265.8778	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1620	1.4644	265.8781	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1680	1.4641	265.8783	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1740	1.4637	265.8786	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1800	1.4634	265.8788	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1860	1.4631	265.8791	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1920	1.4628	265.8793	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
1980	1.4625	265.8795	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2040	1.4622	265.8797	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2100	1.4619	265.8799	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2160	1.4616	265.8801	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2220	1.4614	265.8803	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2280	1.4611	265.8805	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2340	1.4609	265.8807	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2400	1.4607	265.8808	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2460	1.4604	265.881	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2520	1.4602	265.8812	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
2580	1.46	265.8813	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A

Figure 2. Sample Heater Plotting *Up to 4 heaters may be plotted simultaneously with sensing temperature on the primary axis and heater power on the secondary axis*



Component	Min Power	Avg Power	Max Power	Heater	On Temp	Off Temp	Min Temp	Avg Temp	Max Temp	Delta T	Min Power	Avg Power	Max Power	Avail Power	Duty Cycle
Battery	0	0	0	2:Y drive Heater	278.15	283.15	280.17	280.39	282.58	2.40	0.00	0.00	0.00	30.00	0.0%
CDH A	89.892	89.892	89.892	3:X drive HEATER	278.15	283.15	283.60	283.88	284.76	1.15	0.00	0.00	0.00	30.00	0.0%
CDH B	6.842	6.842	6.842	4:Damper 7-6	266.15	267.15	271.34	271.39	271.44	0.10	0.00	0.00	0.00	4.80	0.0%
DPE A	16.731	16.731	16.731	5:Damp 7-5	266.15	267.15	271.34	271.39	271.44	0.10	0.00	0.00	0.00	4.80	0.0%
DPE B	0	0	0	6:Damp 7-4	266.15	267.15	271.34	271.39	271.44	0.10	0.00	0.00	0.00	4.80	0.0%
EPC_A	24.2	24.2	24.2	7:Damp 7-3	266.15	267.15	271.34	271.39	271.44	0.10	0.00	0.00	0.00	4.80	0.0%
EPC_B	0	0	0	8:Damp 7-2	266.15	267.15	271.34	271.39	271.44	0.10	0.00	0.00	0.00	4.80	0.0%
Gimbal Control Electronics	13.2	13.2	13.2	9:Damp 7-1	266.15	267.15	271.34	271.39	271.44	0.10	0.00	0.00	0.00	4.80	0.0%
X drive HEATER - notional	13.2	13.2	13.2	10:Hinge Damper Heater	301.15	308.15	291.18	291.18	291.19	0.01	0.00	0.00	0.00	0.00	0.0%
Y drive HEATER - notional	13.2	13.2	13.2	11:7-7: Star Shade	260.15	267.15	315.91	315.91	315.92	0.01	0.00	0.00	0.00	64.13	0.0%
KBand_Mod_A	61.6	61.6	61.6	12:7-6: Sband A	260.15	267.15	314.45	314.45	314.46	0.01	0.00	0.00	0.00	48.40	0.0%
KBand_Mod_B	0	0	0	13:7-5: Sband B	260.15	267.15	306.01	306.01	306.02	0.01	0.00	0.00	0.00	48.40	0.0%
PDU	95.26	95.26	95.26	14:7-4: TWTs	265.15	272.15	308.15	308.15	308.16	0.01	0.00	0.00	0.00	60.50	0.0%
PSE	298.98	298.98	298.98	15:7-2: Ka modulator	265.15	272.15	305.18	305.18	305.19	0.01	0.00	0.00	0.00	133.10	0.0%
RW1 f(T,S)	13.64	13.64	13.64	16:5-3: GCE	266.15	267.15	296.95	296.95	296.95	0.00	0.00	0.00	0.00	45.00	0.0%
RW2 f(T,S)	13.64	13.64	13.64	17:2-1: DPE A	268.15	269.15	294.95	294.95	294.95	0.01	0.00	0.00	0.00	30.00	0.0%
RW3 f(T,S)	13.64	13.64	13.64	18:6-2: ICDH	268.15	269.15	308.06	308.07	308.07	0.00	0.00	0.00	0.00	30.00	0.0%
RW4 f(T,S)	13.64	13.64	13.64	19:6-3: ICDH	268.15	269.15	307.61	307.61	307.61	0.00	0.00	0.00	0.00	30.00	0.0%
RW5 f(T,S)	13.64	13.64	13.64	20:6-1: WCE	268.15	269.15	303.24	303.24	303.24	0.00	0.00	0.00	0.00	39.00	0.0%
RW6 f(T,S)	13.64	13.64	13.64	21:5-2: CDH	268.15	269.15	289.48	289.48	289.49	0.00	0.00	0.00	0.00	60.00	0.0%
St_Sh_Trans heat load	44	44	44	22:5-1: CDH	268.15	269.15	293.81	293.81	293.81	0.00	0.00	0.00	0.00	60.00	0.0%
SBand_Trans_A	66	66	66	23:4-4: WDE	258.15	259.15	286.12	286.12	286.12	0.00	0.00	0.00	0.00	40.00	0.0%
SBand_Trans_B Heat load	8.8	8.8	8.8	24:4-2: WDE	258.15	259.15	286.32	286.32	286.32	0.00	0.00	0.00	0.00	40.00	0.0%
TCE	30.25	30.25	30.25	25:4-1: WDE	258.15	259.15	285.63	285.64	285.64	0.00	0.00	0.00	0.00	40.00	0.0%
TWT A	80.3	80.3	80.3	26:4-3: WDE	258.15	259.15	285.47	285.47	285.47	0.00	0.00	0.00	0.00	40.00	0.0%
TWT B	0	0	0	27:3-3: PDU	268.15	269.15	280.12	280.12	280.12	0.00	0.00	0.00	0.00	45.00	0.0%
WDE 1 f(T,S)	32.78	32.78	32.78	28:3-1: PSE	268.15	269.15	302.89	302.89	302.89	0.00	0.00	0.00	0.00	45.00	0.0%
WDE 2 f(T,S)	32.78	32.78	32.78	29:3-2: Battery	289.15	290.15	289.10	289.64	290.20	1.10	0.00	30.94	65.00	65.00	47.6%
WDE 3 f(T,S)	32.78	32.78	32.78	30:2-3: TCE	268.15	269.15	299.18	299.18	299.18	0.01	0.00	0.00	0.00	45.00	0.0%
WDE 4 f(T,S)	32.78	32.78	32.78	31:2-2: DPE B	268.15	269.15	293.71	293.72	293.72	0.01	0.00	0.00	0.00	60.00	0.0%
WDE 5 f(T,S)	32.78	32.78	32.78	32:Tank I/F to skirt 2	288.15	289.15	306.29	306.30	306.30	0.01	0.00	0.00	0.00	10.00	0.0%
WDE 6 f(T,S)	32.78	32.78	32.78	33:Tank I/F to skirt 4	288.15	289.15	305.21	305.21	305.22	0.01	0.00	0.00	0.00	10.00	0.0%
ICDH A	60.929	60.929	60.929	34:Tank I/F to skirt 1	288.15	289.15	304.81	304.82	304.83	0.02	0.00	0.00	0.00	10.00	0.0%
ICDH B	0	0	0	35:Tank I/F to skirt 3	288.15	289.15	303.84	303.85	303.86	0.02	0.00	0.00	0.00	10.00	0.0%

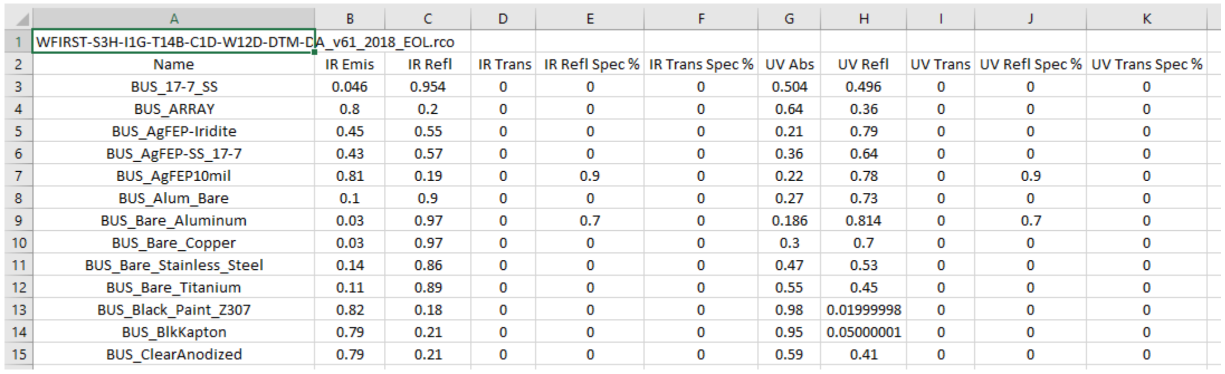
Figure 3. Sample Heater and Dissipation Summary *Heatload and heater information is output to summary table, including min, average, and max power, heater on/off points, duty cycle, etc*

Although this capability does not explicitly utilize the OpenTD API, it was later added as an option to automatically add the heater processing logic when executing a model using the API. This is discussed further in Section IV.

IV. Model Documentation

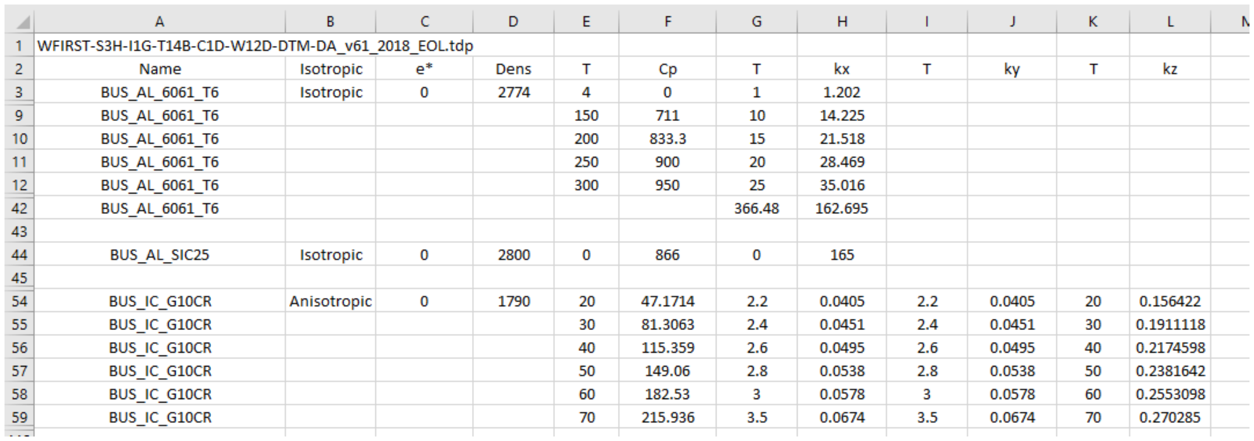
Model documentation is a fairly tedious process. Thermal model data is not neatly compartmentalized into discrete "cards" or "blocks" like finite element models (one card for nodes, one card for materials, one card for elements, etc). Thermal data can include heaters, conductors, heat loads, optical properties, material properties, etc Any of these entities may also have a further dependence on symbols. Likewise, symbols may also have interdependencies on other symbols making it difficult to ascertain a given symbol's value without tracking all the relationships. Furthermore, CaseSets can include over-rides for symbols, optical properties, and thermophysical properties that are only applicable to that specific case. For example, a cold case might include Beginning of Life optical properties as well as lower values for heat dissipations whereas a hot case may include End of Life optical properties and higher values for dissipations. These values may be defined only in the CaseSet or they may be defined in symbols that depend on another symbol to indicate the configuration (e.g. a state flag).

A VB.NET framework was developed including functions to extract optical property values, thermophysical property values, symbol values, and model notes directly from the base drawing file. Additional functions were also developed to extract the same information for a specified CaseSet based on the values in the over-rides. For the optical property extraction, the UV and IR Absorptivity, Reflectivity, and Transmissivity as well as the specular percentages for reflectance and transmittance are extracted for each property with the IR absorptivity serving as the emissivity. It should be noted that with the current version of the API, there is not the ability to identify the current optical property and thermophysical property files used by the drawing file; therefore, it is possible to extract the optical property and thermophysical property values, but the associated filename is not available. However, for CaseSet property over-rides, the specified filenames are available. Sample output of optical property data is shown in Figure 4. For thermophysical properties, the possibility exists for both anisotropic thermal conductivity as well as temperature dependent thermal conductivity and specific heat. Furthermore, the density and MLI effective emissivity values are also extracted Sample output is shown in Figure 5.



WFIRST-S3H-I1G-T14B-C1D-W12D-DTM-DA_v61_2018_EOL.rco										
Name	IR Emis	IR Refl	IR Trans	IR Refl Spec %	IR Trans Spec %	UV Abs	UV Refl	UV Trans	UV Refl Spec %	UV Trans Spec %
BUS_17-7_SS	0.046	0.954	0	0	0	0.504	0.496	0	0	0
BUS_ARRAY	0.8	0.2	0	0	0	0.64	0.36	0	0	0
BUS_AgFEP-Iridite	0.45	0.55	0	0	0	0.21	0.79	0	0	0
BUS_AgFEP-SS_17-7	0.43	0.57	0	0	0	0.36	0.64	0	0	0
BUS_AgFEP10mil	0.81	0.19	0	0.9	0	0.22	0.78	0	0.9	0
BUS_Alum_Bare	0.1	0.9	0	0	0	0.27	0.73	0	0	0
BUS_Bare_Aluminum	0.03	0.97	0	0.7	0	0.186	0.814	0	0.7	0
BUS_Bare_Copper	0.03	0.97	0	0	0	0.3	0.7	0	0	0
BUS_Bare_Stainless_Steel	0.14	0.86	0	0	0	0.47	0.53	0	0	0
BUS_Bare_Titanium	0.11	0.89	0	0	0	0.55	0.45	0	0	0
BUS_Black_Paint_Z307	0.82	0.18	0	0	0	0.98	0.01999998	0	0	0
BUS_BlkKapton	0.79	0.21	0	0	0	0.95	0.05000001	0	0	0
BUS_ClearAnodized	0.79	0.21	0	0	0	0.59	0.41	0	0	0

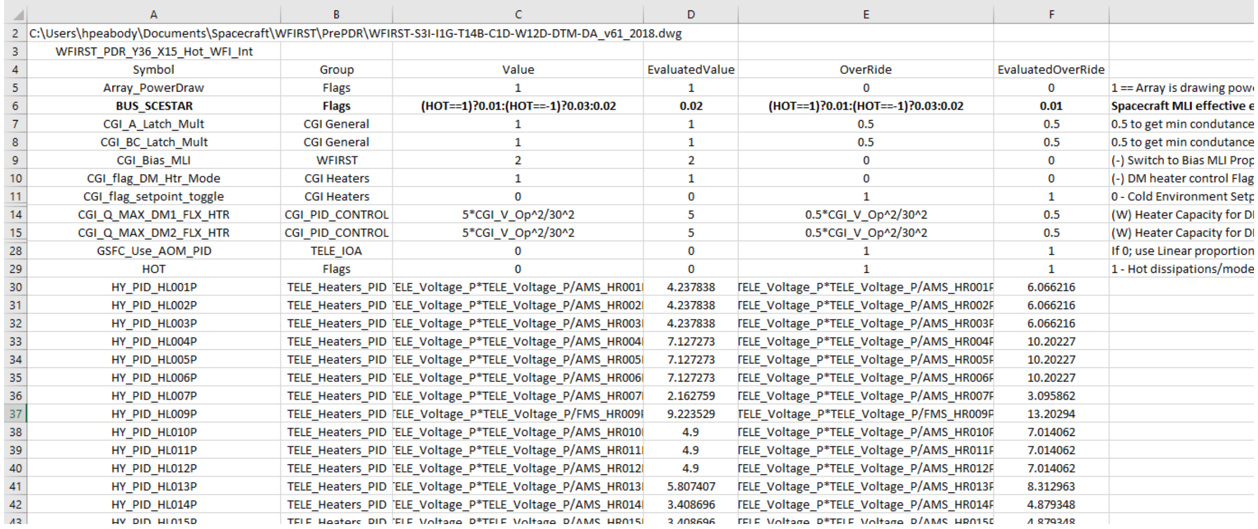
Figure 4. Sample Optical Property Extraction from Thermal Desktop Model *Optical property values are listed along with source file (if available).*



	A	B	C	D	E	F	G	H	I	J	K	L
1	WFIRST-S3H-I1G-T14B-C1D-W12D-DTM-DA_v61_2018_EOL.tdp											
2	Name	Isotropic	e*	Dens	T	Cp	T	kx	T	ky	T	kz
3	BUS_AL_6061_T6	Isotropic	0	2774	4	0	1	1.202				
9	BUS_AL_6061_T6				150	711	10	14.225				
10	BUS_AL_6061_T6				200	833.3	15	21.518				
11	BUS_AL_6061_T6				250	900	20	28.469				
12	BUS_AL_6061_T6				300	950	25	35.016				
42	BUS_AL_6061_T6						366.48	162.695				
43												
44	BUS_AL_SIC25	Isotropic	0	2800	0	866	0	165				
45												
54	BUS_IC_G10CR	Anisotropic	0	1790	20	47.1714	2.2	0.0405	2.2	0.0405	20	0.156422
55	BUS_IC_G10CR				30	81.3063	2.4	0.0451	2.4	0.0451	30	0.1911118
56	BUS_IC_G10CR				40	115.359	2.6	0.0495	2.6	0.0495	40	0.2174598
57	BUS_IC_G10CR				50	149.06	2.8	0.0538	2.8	0.0538	50	0.2381642
58	BUS_IC_G10CR				60	182.53	3	0.0578	3	0.0578	60	0.2553098
59	BUS_IC_G10CR				70	215.936	3.5	0.0674	3.5	0.0674	70	0.270285

Figure 5. Sample Thermophysical Property Extraction from Thermal Desktop Model *Thermophysical property values (including temperature dependence and anisotropy) are listed along with source file (if available).*

For symbols, the challenge exists that the API only provides access to the symbol expression and not its evaluated value. The evaluation of symbols, which can have very complex inter-dependencies including (Expr) ? TrueVal: FalseVal type logic, would require considerable development to evaluate based purely on the data available through the API. However, a work-around was employed that creates a temporary CaseSet and generates the CondCap file, which includes all the symbols as well as their evaluated values in the header comment block. This file is then processed and the connections established between symbol name, group, expression, comment, and evaluated value This approach allows Thermal Desktop to use its built in capabilities that evaluate all the symbol inter-dependencies to evaluate a symbol's value and does not rely on reinventing the well tested code within Thermal Desktop. For the nominal drawing symbol extraction, all the Symbols are included in the output. However, for a CaseSet symbol over-ride extraction, the evaluated symbol values are compared to the base values and if no differences exist, then only the symbols affected by the over-rides are output. This makes for a much smaller subset of values rather than the entire set of symbols but does include any symbols that are over-ridden as well as symbols whose evaluated values also depend on the over-rides. Figure 6 shows sample output for the symbol extraction for a specified CaseSet. A similar work-around could also potentially be used to extract the thermophysical and optical property filenames from the CondCap file, alleviating the aforementioned limitation that the files referenced by the drawing file were not available through the API.



	A	B	C	D	E	F	
2	C:\Users\hpeabody\Documents\Spacecraft\WFIRST\PrePDR\WFIRST-S3I-I1G-T14B-C1D-W12D-DTM-DA_v61_2018.dwg						
3	WFIRST_PDR_Y36_X15_Hot_WFI_Int						
4	Symbol	Group	Value	EvaluatedValue	OverRide	EvaluatedOverRide	
5	Array_PowerDraw	Flags	1	1	0	0	1 == Array is drawing pow
6	**BUS_SCESTAR**	**Flags**	**(HOT==1)?0.01:(HOT==-1)?0.03:0.02**	**0.02**	**(HOT==1)?0.01:(HOT==-1)?0.03:0.02**	**0.01**	**Spacecraft MLI effective e**
7	CGI_A_Latch_Mult	CGI General	1	1	0.5	0.5	0.5 to get min condutance
8	CGI_BC_Latch_Mult	CGI General	1	1	0.5	0.5	0.5 to get min condutance
9	CGI_Bias_MLI	WFIRST	2	2	0	0	(-) Switch to Bias MLI Prop
10	CGI_flag_DM_Htr_Mode	CGI Heaters	1	1	0	0	(-) DM heater control Flag
11	CGI_flag_setpoint_toggle	CGI Heaters	0	0	1	1	0 - Cold Environment Setp
14	CGI_Q_MAX_DM1_FLX_HTR	CGI_PID_CONTROL	5*CGI_V_Op^2/30^2	5	0.5*CGI_V_Op^2/30^2	0.5	(W) Heater Capacity for D
15	CGI_Q_MAX_DM2_FLX_HTR	CGI_PID_CONTROL	5*CGI_V_Op^2/30^2	5	0.5*CGI_V_Op^2/30^2	0.5	(W) Heater Capacity for D
28	GSFC_Use_AOM_PID	TELE_IOA	0	0	1	1	If 0; use Linear proportion
29	HOT	Flags	0	0	1	1	1 - Hot dissipations/mode
30	HY_PID_HL001P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR001	4.237838	TELE_Voltage_P*TELE_Voltage_P/AMS_HR001	6.066216	
31	HY_PID_HL002P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR002	4.237838	TELE_Voltage_P*TELE_Voltage_P/AMS_HR002	6.066216	
32	HY_PID_HL003P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR003	4.237838	TELE_Voltage_P*TELE_Voltage_P/AMS_HR003	6.066216	
33	HY_PID_HL004P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR004	7.127273	TELE_Voltage_P*TELE_Voltage_P/AMS_HR004	10.20227	
34	HY_PID_HL005P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR005	7.127273	TELE_Voltage_P*TELE_Voltage_P/AMS_HR005	10.20227	
35	HY_PID_HL006P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR006	7.127273	TELE_Voltage_P*TELE_Voltage_P/AMS_HR006	10.20227	
36	HY_PID_HL007P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR007	2.162759	TELE_Voltage_P*TELE_Voltage_P/AMS_HR007	3.095862	
37	HY_PID_HL009P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/FMS_HR009	9.223529	TELE_Voltage_P*TELE_Voltage_P/FMS_HR009	13.20294	
38	HY_PID_HL010P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR010	4.9	TELE_Voltage_P*TELE_Voltage_P/AMS_HR010	7.014062	
39	HY_PID_HL011P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR011	4.9	TELE_Voltage_P*TELE_Voltage_P/AMS_HR011	7.014062	
40	HY_PID_HL012P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR012	4.9	TELE_Voltage_P*TELE_Voltage_P/AMS_HR012	7.014062	
41	HY_PID_HL013P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR013	5.807407	TELE_Voltage_P*TELE_Voltage_P/AMS_HR013	8.312963	
42	HY_PID_HL014P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR014	3.408696	TELE_Voltage_P*TELE_Voltage_P/AMS_HR014	4.879348	
43	HY_PID_HL015P	TELE_Heaters_PID	ELE_Voltage_P*TELE_Voltage_P/AMS_HR015	3.408696	TELE_Voltage_P*TELE_Voltage_P/AMS_HR015	4.879348	

Figure 6. Sample Symbol Extraction from Thermal Desktop Model (CaseSet) *Note that while BUS_SCESTAR is not explicitly over-ridden, its evaluated value depends on HOT, which is explicitly over-ridden The CaseSet and drawing file are also identified.*

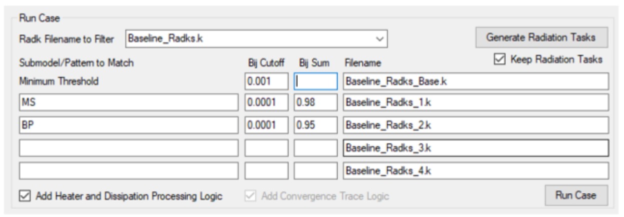
Lastly, the development of a generic routine to extract tagged data (e.g. lines beginning with a specified string) and import them into Excel (similar to the heater and dissipation lines) further allows for user specified logic to be included to output data of particular interest at each timestep and extracted to Microsoft Excel.

V. Radiation Coupling Filtering

Early in the Wide Field InfraRed Survey Telescope program, as concerns grew regarding model run time, a hybrid radiation coupling filtering approach was proposed[3]. This hybrid approach would output two (or more) sets of radiation couplings with different filter criteria (B_{ij} Cutoffs, B_{ij} Sums) but from the same computation run. Then, couplings would be extracted connected to a subset of nodes from one set and all but the subset of nodes from the second set. This approach resulted in sufficient detail for a cryogenic region (where smaller radiation couplings may be more significant) while not needing the small radiation couplings throughout the model and increasing the associated run time. Although the project did not implement this as a matter of course due to the frequent updating of the model and the effort required to configure the model for this, it was recently revisited with the availability of the OpenTD framework due to the ability to automate this process using the API.

Aside from being able to modify and manipulate objects with the OpenTD API, a user also has the ability to execute CaseSets using the `.Run` method. Therefore, code was developed to execute a user specified CaseSet with some modifications to the execution process. First, the function allowed for the execution of the model in its own directory based on the CaseSet name. Determining the directory requires some care as the model may be written to the same location as the drawing file, or if defined by the `.UserDirectory` properties may be in an alternate directory. The usage of "`..\`" and "`.\`" relative path type syntax also complicates determination of the intended directory for model execution as these locations need to be resolved relative to the specified drawing file. A function was developed to determine this directory for a specified CaseSet. Once determined, this CaseSet subdirectory is created and after the input file is created, a copy of the input file is placed into the newly created directory. Furthermore, a paths.txt file is also created in the directory pointing back to the location of all files included via INSERT directives by SINDA. An option was also provided to allow the user to add the heater processing logic automatically as described in Section III.

Having developed the capabilities to manipulate parts of the model execution process, a second routine built on this ability and added radiation coupling filtering as an option. This filtering process is illustrated in Figure 7. The top portion shows four available levels of filtering, where each filter includes: Pattern, B_{ij} Cutoff, B_{ij} Sum, and filtered Radk Filename. The user also specifies the CaseSet and base Radk Filename, which identifies which RadiationTask in the CaseSet should be used to spawn the additional RadiationTasks with the new B_{ij} Cutoff, B_{ij} Sum, and output filenames as specified by the user. The updated CaseSet is then executed and the additional radk output files are created based on the new B_{ij} Cutoff and Sum values. These output files are then further processed and all couplings connected to nodes not included in the pattern are removed. For the example in Figure 7, the pattern for filter A is MS, and only couplings connected to a submodel with MS are kept. The pattern for filter B is BP, and only couplings connected to BP are kept, unless they are also connecting MS and BP nodes, since these were already included in processing filter A. Figure 7 shows the couplings removed in strikethrough and couplings that are kept in bold red. This process repeats for filters C and D if applicable. Lastly, any couplings connected to nodes not meeting any of the pattern matches are output from the base RadiationTask output and its filter criteria, which should be less stringent than all the other RadiationTasks that were spawned. The end result is multiple radk files, but with fewer total radiation couplings. Smaller couplings are kept only for portions of the model where the benefits are justified.



Run Case

Radk Filename to Filter: Baseline_Radks.k

Generate Radiation Tasks

☑ Keep Radiation Tasks

Submodel/Pattern to Match	Bij Cutoff	Bij Sum	Filename
Minimum Threshold	0.001		Baseline_Radks_Base.k
MS	0.0001	0.98	Baseline_Radks_1.k
BP	0.0001	0.95	Baseline_Radks_2.k
			Baseline_Radks_3.k
			Baseline_Radks_4.k

☑ Add Heater and Dissipation Processing Logic ☑ Add Convergence Trace Logic

Run Case

98% Sum Radks Pattern: MS	95% Sum Radks Pattern: BP	Baseline Radks
~~1, BP.1, BP.2, X~~	**1, BP.1, BP.2, X**	~~1, BP.1, BP.2, X~~
~~2, BP.1, BP.3, X~~	**2, BP.1, BP.3, X**	~~2, BP.1, BP.3, X~~
~~3, BP.2, BP.3, X~~	**3, BP.2, BP.3, X**	~~3, BP.2, BP.3, X~~
~~4, BP.2, BP.4, X~~	**4, BP.2, BP.4, X**	
~~5, BP.3, BP.4, X~~		
6, BP.1, MS.1, X	~~5, BP.1, MS.1, X~~	~~4, BP.1, MS.1, X~~
7, BP.2, MS.1, X	~~6, BP.2, MS.1, X~~	~~5, BP.2, MS.1, X~~
8, MS.1, MS.2, X	~~7, MS.1, MS.2, X~~	~~6, MS.1, MS.2, X~~
9, MS.2, MS.3, X		
10, MS.2, MS.4, X		
~~11, BP.1, LDR.1, X~~	**8, BP.1, LDR.1, X**	~~7, BP.1, LDR.1, X~~
~~12, BP.2, LDR.1, X~~	**9, BP.2, LDR.1, X**	~~8, BP.2, LDR.1, X~~
~~13, BP.1, LDR.2, X~~	**10, BP.1, LDR.2, X**	~~9, BP.1, LDR.2, X~~
~~14, BP.2, LDR.2, X~~		
~~15, LDR.1, LDR.2, X~~	~~11, LDR.1, LDR.2, X~~	**10, LDR.1, LDR.2, X**
~~16, LDR.1, LDR.3, X~~	~~12, LDR.1, LDR.3, X~~	

Figure 7. Sample Process for Radk Filtering: *The first file removes all couplings not connected to the pattern. Subsequent files must also account for couplings already included from previous filter steps. In this example, MS couplings are kept from the 98% file, and BP couplings are kept from 95% file as long as they were not already kept from MS processing. All remaining nodes not matching any patterns are captured last.*

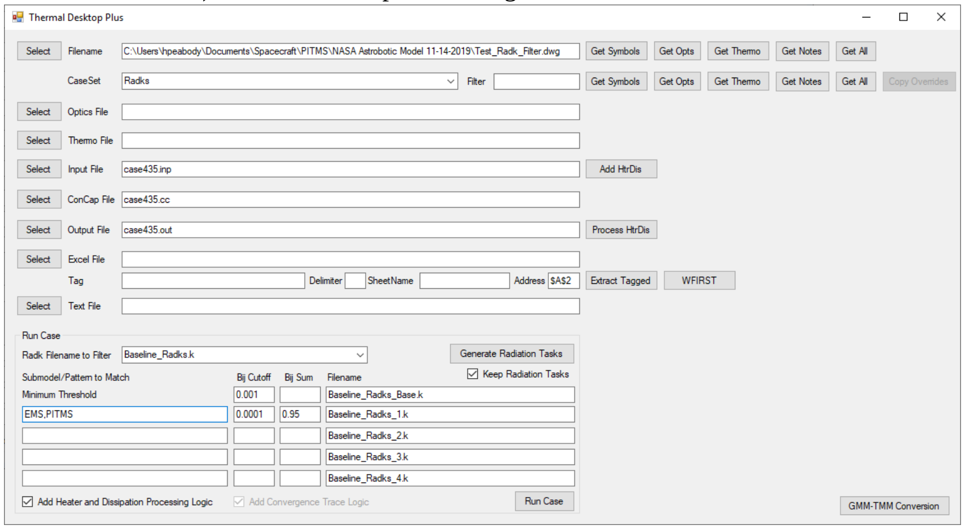
VI. Framework and Interface

The various capabilities described in this paper form the framework of functions listed in Table 1. The goal was to minimize, as much as possible, the need for a graphical interface to access the capabilities of the framework, allowing the user to develop their own interface to the framework capabilities. Entries in italics utilize the OpenTD API, while the others are utility functions that do not require a connection to the API.

Framework Function	**Purpose**
GenerateHeaterDissipationLogic	Process SINDA .inp file and add logic to .htr file to output Heater and Dissipation output logic for every timestep
ProcessHeaterDissipationResults	Process SINDA .out file and retrieve output generated from GenerateHeaterDissipationLogic and import into Excel template workbook
ExtractTaggedLinesAndImportIntoExcel	Extract lines beginning with TagID from OutFile and import into specified Excel location, parsing on specified delimiter
ImportFileIntoExcel	Import specified text file into specified Excel location, parsing on delimiter
EvaluateSymbolsInDWGFile	*Generates temporary case set, outputs CC file and processes this for evaluated symbol values*
EvaluateSymbolsForSpecifiedCaseSet	*Generates temporary case set spawned from user specified case, outputs CC file and processes this for evaluated symbol values*
ExtractSymbolEvaluatedValuesFromCCFile	Process CC file header and retrieves symbol names and evaluated values
Write2DArrayToCSVFile	Convert 2D array to comma-separated value output file (can include Header row and Output Mask)
GetTDOptProps	*Read TD object and extract optical properties and store in GMM_OpticalProperties collection*
GetTDThermoProps	*Read TD object and extract thermophysical properties and store in GMM_ThermophiysicalProperties collection*
GetTDNotes	*Read TD object and extract Notes data and Splice together all Tabs into single output text file*
GetTDRunDirectory	*Return path to either DWG file if No CaseSet specified with UserDirectory, or to UserDirectory*
RunSpecifiedCaseSet	*Execute case set in its own directory with options to add heater dissipation and convergence trace logic*
CreateSubmodelSpecificRadiationTasks	*Create multiple Radiation Tasks with various Bij Sum/Cutoff and then filters output files for associated specified submodels/ patterns*

Table 1 – List of Function in GSFC Developed Framework

A simple Graphical User Interface (GUI) was developed at GSFC to facilitate easier usage of the functions in the framework with efforts made to minimize the amount of "action" code associated with the GUI controls (e.g. a button click simply passed a textbox value to a framework function rather than having the framework capability defined in the button click event). This GUI is depicted in Figure 8.



Filename C:\Users\hpeabody\Documents\Spacecraft\PITMS\NASA Astrobotic Model 11-14-2019\Test_Radk_Filter.dwg

CaseSet Radks Filter

Optics File

Thermo File

Input File case435.inp

ConCap File case435.cc

Output File case435.out

Excel File

Tag Delimiter SheetName Address A2

Text File

Run Case

Radk Filename to Filter Baseline_Radks.k

☑ Keep Radiation Tasks

Submodel/Pattern to Match	Bij Cutoff	Bij Sum	Filename
Minimum Threshold	0.001		Baseline_Radks_Base.k
EMS,PITMS	0.0001	0.95	Baseline_Radks_1.k
			Baseline_Radks_2.k
			Baseline_Radks_3.k
			Baseline_Radks_4.k

☑ Add Heater and Dissipation Processing Logic ☑ Add Convergence Trace Logic

Figure 8. Simplistic Graphical User Interface to access framework functions: *A simplistic interface was developed to more easily specify files and parameters to access the framework capabilities*

VII. Limitations and Workarounds

Aside from some of the workarounds described earlier, such as extracting evaluated symbol values and baseline property file names from the generated .cc file, some additional workarounds were found to further access data not yet exposed by the API. One of the major limitations of the API for version 6.1 is the lack of access to existing Finite Element data, prevalent in many Thermal Desktop models. Elements may be created, but existing elements cannot currently be polled to inspect or extract their data. However, a methodology was developed that can determine: (1) the nodes used to form an element and their locations, (2) the thickness, (3) material, (4) material orienter assigned, (5) the optical properties assigned to the top and bottom sides, (6) the assignment of active sides for each radiation analysis group, and (7) the assignment of MLI on top and bottom. At the heart of this method is the use of the entity handles (the unique identifiers for each object in the AutoCAD database) as well as the ability to send commands and retrieve the command history through the API. While this process is neither efficient nor quick, it is at this time the only way known to access the Finite Element data.

To begin, the DBIterator should be used to step through the entire database, one object at a time. Each object includes both a *Type* property and a *Handle* property. Knowing the object type, allows an application to select how to access data associated with the object. Knowing the *Handle* allows the object to be selected using API methods or AutoLISP code to interface with the AutoCAD instance. Use of the *SendCommand* method of the API in combination with the *handent* AutoLISP function and the *LIST* AutoCAD command will execute the command and send the output to the AutoCAD text window [e.g. `TD.SendCommand("list (handent ""A1A1A"") ")`]. The *GetCommandHistory* method of the API retrieves a string of the result of the sent commands. This text can then be parsed to extract the node numbers, material, thickness, and material orienters. Knowing the node numbers, the API can then access the XYZ locations in space. Unfortunately, the LIST output for a Finite Element does not include any further information about the radiative properties, so an alternate method must be used to access that data.

To extract the radiative activity, optical property assignments, and application of insulation information, a model check from the Thermal Desktop interface (Output Node Optical Property Summary) can be executed that indirectly provides this information in the form of an Excel spreadsheet. Fortunately, the API does include the ability to specify the active Radiation Analysis Group and *SendCommand* can be used with *rcOutputNodeOpticsSummary* and r*cOutputAnalGroupSummary* to generate the Excel files. Data included in the *OpticsSummary* spreadsheet is for the current active Radiation Analysis Group and includes: (1) node number, (2) optical property, and, most importantly, (3) object type, handle, and side (e.g. Quad Elem-TOP::A1A1A). If a node number is *associated* with an element, but is not used to *define* the element, it can be inferred that this node must be an insulation node. For example, node 100001 is found for element A1A1A, but the element is made up of nodes 1, 2, 3, and 4. Since finite elements do not allow partial insulation to be assigned, the other nodes associated with element A1A1A can be used to determine the insulation offset. Lastly, the radiative activity can be inferred from this file based on which node numbers are found in the file, as it only includes surfaces assigned to the active Radiation Analysis Group. Varying the active group and evaluating each of the spreadsheets generated would allow the full radiative activity to be defined. The *AnalysisGroup* spreadsheet output includes the active sides for all groups associated with the node numbers.

The above method has been preliminarily tested, but has not yet been robustly developed at GSFC. Once development is complete, this method will be integrated into a previously developed framework for the conversion of geometrical models[4]. With the geometric data stored in an application, metrics could then be used to establish if two surfaces in different models are indeed identical (e.g. same shape, location, size, nodes, etc) even with differing handles. This could be used to ensure consistency between delivered and integrated models (independent of their assigned handles) by comparing all surfaces, conductors, contactors, heatloads, etc. Furthermore, this ability could also allow for the import of Domain Tag Sets from a source model to the destination model. By identifying the properties of surfaces in a source Domain Tag Set and comparing them to surfaces in a destination model, the Domain Tag Set could then be redefined in the destination model.

One last capability that is not inherently available in the API is the ability to turn on or off the local display of a specific entity. Furthermore, there is currently no dedicated API method to determine the global visibility for a given entity type. However, again the use of the *SendCommand* method can be employed with the *rcToggleSurfaceVis* command (or any of the other rcToggle commands). By sending this command twice, it results in no change to the global visibility, but the resultant text can be extracted using *GetCommandHistory* and parsed to determine the current visibility state. Furthermore, the use of the *handent* AutoLISP routine in combination with the *SendCommand* method and the *rcVisOn* and *rcVisOff* commands would allow for the local visibility of an object to be specified based on its handle. This could allow for the save and retrieval of dedicated views of specific geometry or other thermal objects and could be further extended to include the view direction and magnification for saved views, which may be useful in the generation of consistent images for reports or presentations.

VIII. Conclusions and Path Forward

The introduction of an API to access the internal parameters of a Thermal Desktop model provides users with an immense amount of power to further the capabilities of the software. Already this API has been used in the development of the Veritrek tool[5] and by the European Space Agency to improve model exchange via the STEP-TAS protocol. Early usage of the API has also improved productivity at GSFC, allowing external modification of Thermal Desktop generated files prior to model execution to improve the output of heater and dissipation performance as well as to allow the extraction and capture of thermal model data such as properties, symbols, and notes. Furthermore, the ability to filter radiation couplings based on submodel or pattern specific cutoff and sum values can result in smaller matrices for the thermal solver without sacrificing accuracy in areas where radiation heat exchange is critical, such as a cryogenic region, thus improving model run time. The utilities developed in the framework as well as other external tools have been used extensively to capture results and configuration details of a large scale observatory as the design evolved[6] to track the impact of updates on the thermal performance.

Future efforts are already envisioned to further develop productivity improvement utilities including CaseSet comparisons across more than two CaseSets as well as allowing symbol over-rides to be copied from one CaseSet to other CaseSets. The model conversion efforts[4], began earlier, may also now be updated to support import and export of Thermal Desktop models. Some additional desired features that could be added to the API by the vendor include the ability to access post-processing data mappers through the API and the ability to pass a selection set of objects from the AutoCAD instance to the program through the API. With potential future additions to the API by the vendor, even more utilities can be conceived, such as standard image generation for reports and generation of comparison contour plots for thermal distortion mapping efforts. While it is not covered in this paper, potential users should also be aware of a second aspect of the OpenTD API which addresses the processing of results files and is accessed through a second reference to the appropriate namespace. The long term plan for the development of these tools at GSFC is to provide the source code as open source in the hopes that other organizations may follow suit to improve the thermal analysis capabilities of the community at large. With the development and sharing of productivity enhancing utilities, the entire thermal community stands to benefit from the groundwork laid by the development of the OpenTD API with Thermal Desktop.

Acknowledgments

The primary author would like to thank Matt Garrett at Cullimore and Ring Technologies for helping to come up to speed on the API and provide the way forward when obstacles were encountered during development.

References

[1] OpenTD API Product Brochure: https://www.crtech.com/sites/default/files/files/Brochures/api.pdf

[2] Documentation with Thermal Desktop install: Getting Started with OpenTD 61.pdf

[3] Peabody, H., et al, "Addressing Thermal Model Run Time Concerns of the Wide Field Infrared Survey Telescope using Astrophysics Focused Telescope Assets (WFIRST-AFTA)" ICES-2016-188, *46th International Conference on Environmental Systems*, Vienna Austria, 2016.

[4] Yang, K., Peabody, H., "Preliminary Development of a TSS and SINDA/FLUINT to ESARAD/ESATAN Thermal Model Converter" TFAWS-2014-PT-03, *2014 Thermal And Fluids Analysis Workshop*, Cleveland OH, 2014.

[5] Hengeveld, D., Moulton, J., "Automatic creation of reduced-order models using Thermal Desktop®" ICES-2018-80, *48th International Conference on Environmental Systems*, Albuquerque NM, 2018.

[6] Peabody, H., "Tracking Critical Thermal Metrics throughout the Life Cycle of a Large Observatory Thermal Model" ICES-2020-298