FINAL REPORT

ENERGY BENCHMARKING TOOL DEVELOPMENT

TAF NEW CONCEPT DEVELOPMENT PROGRAM

RWDI #1701187

September 12, 2017

SUBMITTED TO

Ian Klesmer
Grants Manager and Policy Advisor
iklesmer@taf.ca

Ekaterina Tzekova
Building Research Manager
etzekova@taf.ca

The Atmospheric Fund
75 Elizabeth Street
Toronto, Ontario, M5G 1P4
T: 416.392.0271
F: 416.338.0616

SUBMITTED BY

Ontario Association of Architects
111 Moatfield Drive
Toronto, Ontario, M3B 3L6

PREPARED BY

Chris Frankowski
Sustainability Consultant
chris.frankowski@rwdi.com

Mike Williams
Principal, Sustainability
mike.williams@rwdi.com

RWDI
400-901 King Street West
Toronto, Ontario, M5V 3H5
T: 519.823.1311 x2062
F: 519.823.1316

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The following report and appendices demonstrate the development of an automated data harvesting tool that extracts over 400 building characteristics from an eQUEST or IES-VE energy model output file, stores the data in a centralized database, and returns completed submission forms for five building programs and standards. Development is ongoing to expand this tool to include EnergyPlus energy model output files, and a sixth building program.

Two specific outcomes of this project are of immediate benefit to the building industry:

1. A robust matrix that identifies the over 400 key building characteristics needed to complete the required submission forms for six existing programs and standards (LEED, HPNC, TGS, OBC SB-10, SBD, and 2030 Challenge), and correlates these characteristics to the location of each data point within the output files of three prominent energy modelling software (eQUEST, EnergyPlus, and IES-VE). While the information in this matrix can be used to inform the manual extraction of data from an energy model file, the vast quantity of characteristics proves the need for an automated data extraction process.

2. This concept development project is a valuable part of a larger project, which includes the development of an end-user interface, online portal, and data storage architecture, to form an Energy Benchmarking Tool that is freely and easily accessed by the entire design community in Ontario. To accomplish this, additional funding has been secured through the Independent Electricity System Operator (IESO). The project work that has been completed using funding from TAF was instrumental as a proof-of-concept, and allowed for greater clarity in the scope and intention of the overall project, resulting in a successful application to the IESO Conservation Fund.

Despite these successes, there have been some challenges that limited the final delivery of this project.

1. The Toronto Green Standard (TGS) energy model files were not available for the completion of Task Areas 3 and 4, testing and analysis of the tool on external energy model files. Negotiations are ongoing to coordinate the completion of these tasks, but this will fall within the broader project, and will not be completed using funding from TAF.

2. The number of data points extracted from each building energy model expanded from 210 in the original proposal to over 400 in the full matrix. The inclusion of these characteristics has doubled the number of data points that are extracted from each energy model, which doubled the effort required for the project. RWDI was therefore not able to complete the full scope of script development within the funding time frame. The remaining scripting will be completed as part of the larger project, under the IESO funding described above.

1 INTRODUCTION

This report provides a final update of the New Construction Energy Benchmarking Tool Development project, in accordance with the terms and conditions outlined in the agreement "*RE: GRANT #G-DEC 2016-1 – New Construction Energy Benchmarking Tool Development.*" It is submitted to The Atmospheric Fund by the Ontario Association of Architects. For the technical development of the tool, the OAA has contracted RWDI engineers, one of the leading research engineering firms in Canada. This report has been prepared by RWDI.

1.1 Description of Project

In the current marketplace, regulations, voluntary standards and regulated incentive programs have been put in place to motivate newly constructed buildings to reduce their energy use and carbon footprint. Largely as a result of these programs, energy modelling has become a mainstream design exercise that is now completed on hundreds of projects annually in Ontario. To date, however, data contained within energy simulation files has not been leveraged either for broad analysis of performance trends or to benchmark like-buildings to improve energy performance.

Through the development of an Energy Benchmarking Tool, this project aims to automate the harvesting of information from completed energy models, and process the data in two ways to inform the building design industry in Ontario:

1. Completed program submission forms for six building standards and programs provided to the user; and
2. A data visualization platform available to the user to enable benchmarking of their building against the aggregated database of all harvested information.

Table 1 summarizes the six standards and programs that are included in the Energy Benchmarking Tool.

The target market for the project is new construction design decision makers in Ontario including architects, engineers, builders, and municipal building authorities. The standardized program submission forms provided by the Tool will result in streamlined and consistent reporting, which will be of benefit to both the design community and the reporting agencies. The aggregated data set generated through this reporting will be a valuable benchmarking tool to inform the design processes through comparison of energy and emission characteristics to other projects and best practice targets. Further, it may be and a way for program administrators to track and analyze performance trends across all new buildings.

Three energy modelling applications have been identified as priorities for this project, based on their prevalence in the market: eQUEST, EnergyPlus, and Integrated Environmental Solutions – Virtual Environment (IES-VE).

Table 1: Summary of Standards and Programs

Standard / Program	Compatible Version(s)	Submission Form(s)

Leadership in Energy and Environmental Design (LEED)	LEED v2009;
	LEED v4;
		LEED Letter Template (v2009);
		LEED v4 online submission form;

Toronto Green Standard (TGS)	New Mid to High-Rise Residential and All Non-Residential Development, Version 2.1 April 2017;	Appendix A Better Buildings Partnership – New Construction Energy Modelling Report Summary;

Save On Energy - High Performance New Construction (HPNC)	HPNC Version 6.0, May 2016;	Energy and Demand Summary – May 18, 2016;
		Program Custom Project Worksheet – May 18, 2016;

Ontario Building Code (OBC)	Supplementary Bulletin 10-2012;
	*Submission forms for SB10-2017 have not been released however it will be included once available;
		SB-10 Form A;
		SB-10 Form 11;

Savings by Design (SBD)	N/A	RWDI customized template (no official modelling submission form);
2030 Challenge	The 2030 Challenge 2015;	N/A

2 PROJECT ACTIVITIES

2.1 Program Requirement Analysis

The first task of this project was to prepare a concise and comprehensive summary of the reporting requirements for LEED, TGS, HPNC, OBC, SBD and 2030 Challenge into a single matrix. Through this process, 300 building and energy characteristics were identified for extraction from each energy model, to enable completion of the submission forms for each standard/program. An additional 100 characteristics were selected for inclusion as useful metrics for the benchmarking process. These characteristics were selected based on RWDI's industry experience and the recommendation of the TAF staff. The Program Requirements Matrix was submitted for review by the OAA in March, 2017.

Having identified the program requirements and building characteristics for extraction, the next task involved locating each characteristic in the output files of all three selected energy modelling applications. The full matrix of over 400 building characteristics with identified energy modelling output locations can be found in Appendix A.

2.2 Tool Development

A Use Case Diagram and System Flowchart were developed to outline the structure of the tool. These can be found in Appendix B. From this outline, and using the information detailed in the Program Requirements Matrix described in Section 2.1, the Energy Benchmarking Tool was developed using the Python programming language.

The tool consists of a library of modules that are capable of extracting information from the output files of three different energy modelling applications and using that information to populate the submission forms for six different building programs. There is an extraction module for each of the three energy modelling application output files covered in the scope of this project. The extraction modules go through the modelling output files, extract the required information, and return it to the main module. There is also a populating module for each of the program submission documents that can be generated by the tool. The populating modules take the information passed to them by the main module, and create the required output files by inserting the data into template documents. The main module scans the project folder for the modelling output files, reads additional information out of the setup workbook, and calls the extraction and output modules.

The extraction modules were created by reviewing multiple output files from each energy modelling application and reverse engineering the output file format. By understanding the output file format fully, a reader can be created that is capable of interpreting any output from the energy modelling application.

The populating modules were developed using the submission documents that are provided by the programs as templates that can be populated by the modules.

The main module contains the logic for scanning the project folder for modelling output files, reading the project setup workbook, and calling the extraction and populating modules.

The tool was developed and tested on Python 3.5.1. The project utilizes several packages beyond those included in the Python standard library. beautifulsoup4 is used for html file scraping, openpyxl is used for reading and writing excel files, and fdfgen is used for creating Acrobat Forms Data Format files. The tool also requires pdftk to be installed on the system for creating PDF files.

A copy of the current program source code is found in Appendix C.

2.3 Testing

The tool was developed and tested using multiple example modelling output files to which RWDI had access, for each modelling application. The modules were written in an iterative fashion by adding the logic for extracting the next data point, running the module against the test output files, and confirming that the data was extracted correctly for all test output files. The scripts were developed by reverse engineering the output files, so having a more diverse set of output files to test against improve the robustness of the extraction scripts.

Unfortunately, we were unable to access the City of Toronto modelling files (see Section 3.2.1 for more details), and as a result have not been able to test the tool against external files. As the development of the tool continues, the modules will continue to be tested against a larger and more diverse set of modelling output files.

3 PROJECT STATUS

Through the project activities described in Section 2, the project has progressed as follows:

1. Over 400 building characteristics have been identified as required for six programs and standards, and the location of these characteristics has been identified in three energy modelling software.
2. A library of modules has been developed, using Python programming language, to extract information from the output files of eQUEST and IES-VE energy modelling applications.
    a. Note: extraction of all 400 building characteristics from IES-VE output files was not possible, and is discussed further in Section 3.3.3.
3. Additional Python modules have been developed to populate the submission forms for four programs and standards.
4. The modules were tested using sample eQUEST and IES-VE energy modelling files in an iterative fashion to confirm that data extraction and submission form generation were completed as designed. The items highlighted in yellow in Appendix A are still under development.

Appendix D contains example program submission forms that were created by the tool from an example energy modelling output file. The outputs cover 4 of the 5 program submission forms included in the proposal (LEED, TGS, OBC, and SBD).

There is no program submission form for the 2030 Challenge. At this time, the relevant outputs are extracted into the database of building characteristics, but are not presented in an output file. With the full development of the tool, through the IESO funded project, a user input that identifies the use-type of the building will enable a comparison with the specific 2030 Challenge target for that building type. At that time, a reporting strategy will be developed to present the 2030 Challenge results.

The development of the EnergyPlus extraction module, and the HPNC populating module are still in progress. Improvements to all of the modules will continue as the tool is further developed, as is discussed in the following sections, which identify the challenges and opportunities encountered by the project and the program modifications that have been undertaken. These ongoing improvements will be completed as part of the IESO Conservation Funding project. A full list of the next steps that will be completed using the IESO funding is found in Section 4.

3.1 Expanded Project Scope

3.1.1 Opportunity: Successful IESO Conservation Fund Application

Parallel to the work that has been done for this Energy Benchmarking Tool Development project, RWDI has partnered with the OAA and the Toronto 2030 District to apply to the Independent Electricity Systems Operator (IESO) Conservation Fund. The IESO application proposed an expansion of this current project to include a robust online user interface, beta-testing with industry partners from the design and regulatory sectors, and 2-years of user engagement to support the market launch of the Benchmarking Tool.

The application was successful, and the project has been awarded its full funding request. This is an excellent opportunity to build upon the groundwork that has been completed in this current TAF project, and move the Energy Benchmarking Tool from the concept development stage into full market deployment. It is believed that the TAF project strongly contributed to the success of the IESO application by demonstrating the demand for a benchmarking tool in the current market.

3.1.2 Challenge: Forward-looking Tool Development

The successful receipt of IESO Conservation Fund funding means that the larger initiative, of which the current project is the first step, is moving forward. This has altered the approach taken on the tasks and deliverables for the TAF funding, in reflection of the expanded overall scope. Specifically, the scope and function of the data extraction scripts has been expanded to accommodate the future project development, in order to avoid rewriting code in the future.

As a result, the scripts are more robust than initially planned, however this added detail is reflected in the increased effort spent in their development beyond the proposed TAF funding budget. Further, it has proven to be a challenge to complete the additional work within the timeframe of the TAF project.

3.2 Toronto Green Standard Harvesting and Analysis

3.2.1 Challenge: City of Toronto Data Unavailable

The TAF Funding Agreement included as a deliverable the results from a test run of the Energy Benchmarking Tool using energy model files submitted to the City of Toronto for application to the Toronto Green Standard. To facilitate this, RWDI met with representatives from the City to discuss the use of their modelling files. Unfortunately, due to confidentiality concerns, the City officials were unwilling to provide files, preventing RWDI from producing this deliverable.

In order to address the confidentiality concerns, an alternative was proposed where RWDI would produce a standalone version of the tool that could be installed and tested on-site at the City of Toronto Energy Efficiency Office, by City employees. Negotiations are underway to coordinate the completion of this collaborative testing. As such, this deliverable cannot currently be produced, however it is anticipated that this test run will be successfully completed once the arrangements are finalized and the stand alone version is produced by RWDI.

3.2.2 Opportunity: Municipal Beta Testing Partnership

Automated reporting to the most prominent new construction energy conservation programs in Ontario is a key function of our tool. Beyond providing a service to design professionals, this has been identified as an excellent opportunity to provide additional value to energy conservation program regulators. While all of the targeted standards and programs share a common objective of energy conservation, their varied reporting and compliance requirements have created confusion and detracted from their underlying intent. Developing a single reporting platform for all of these programs will simplify the process for applicants and the program's governing bodies alike.

Our belief is that by aligning the Energy Benchmarking Tool with all of these programs, the use of our tool will be seen as a means to simplify and streamline the required program reporting and therefore use of the tool will be adopted rapidly and broadly by the design and program regulation communities. Despite not connecting for the TGS data harvesting and analysis, we are hopeful that the City of Toronto will participate in the development of the full Energy Benchmarking Tool as beta testers, and that the use of this tool will be beneficial to their offices. Working with the City of Toronto as beta testers will also allow the early resolution of their confidentiality concerns within the full tool.

3.3 Unanticipated Hurdles

3.3.1 Challenging Opportunity: Increased Number of Characteristics

The TAF funding proposal estimated that 210 data points would be extracted from each energy model to complete the Energy Benchmarking Tool objectives, based on a working beta version of the tool. During the Program Requirements Assessment task of this project, two expansions were made to the project scope:

1. The 2030 Challenge was added as a sixth program/standard for consideration; and
2. It was decided that tool would be compatible with both LEED 2009 and LEED v4 submission requirements and forms.

This meant that the original proposal greatly underestimated the number of characteristics that would be required to complete the submission forms for all six programs and standards. An additional 190 data points were identified for inclusion in the extraction process, for a total of over 400 building characteristics per energy model – double what was anticipated during the TAF funding application process.

The additional data points greatly increased the effort required for this project, both in the program requirement analysis tasks and in the tool development: for each additional characteristic, a feature needs to be added to each of the three data extraction scripts – one for each modelling software. Additional script development is also needed to process the data into the new program submission forms.

Despite this increased effort, the expanded number of building characteristics included in the extraction process enables the Energy Benchmarking Tool to be a more thorough and overall useful tool.

3.3.2 Challenge: Post-Processed Results

The process of energy modelling, and the limitations of current energy modelling software, frequently require the energy modeller to use software features in unintended ways in order to achieve a desired outcome. This process creates the necessity for manual post-processing by the energy modeller in order to convert the model results into what they are meant to represent.

The Energy Benchmarking Tool is incapable of identifying when this situation arises, and instead interprets the model files literally. Therefore, the results presented through this tool will, at times, not agree with an energy modelling report, or other post-processed interpretation of the results. This is an unavoidable challenge of the automated data extraction process, yet the intention of this project has never been to replace the experience of the design team. Rather, the Energy Benchmarking Tool is designed to provide an opportunity for the team to verify and review their design choices in the context of other similar buildings, and to relieve the time required for submission to building programs and standards. It is the responsibility of the energy modeller to review the model outputs that are extracted by the tool and make any necessary modifications.

3.3.3 Challenge: IES-VE Output File Formats

IES-VE is a proprietary energy simulation software. A summary report is produced when each model is run, but the detailed simulation results are only accessible through the IES-VE user interface. While many energy performance characteristics are provided in the summary report, several building characteristics, such as the seasonal efficiency of mechanical equipment, are not included and can only be determined through manual review of the model results. This has presented a challenge for the automated extraction of data from the model outputs.

RWDI has reached out to IES in an attempt to solve this data extraction issue but has received limited response. We hope that as this project expands under the IESO Conservation Funding, our partnerships with municipal authorities, modelling associations, and industry leaders will help demonstrate to IES the demand for the collaborative creation of an output format for IES-VE that aligns with the Energy Benchmarking Tool. In the meantime, the Energy Benchmarking Tool will extract as much information as possible from the current IES-VE summary report – the information that can be extracted from current summary report is sufficient for the completion of the program submission forms, but not all 400 data points can be extracted to the database. Appendix A identifies which characteristics have been successfully captured by the current scripting.

4 NEXT STEPS

Building upon the foundational work that was accomplished during this concept development project, the next steps for the Energy Benchmarking Tool Development project, which will be completed using the IESO Conservation Funding, can be summarized as follows:

1. Complete the development of the tool, including finalizing the automated data extraction scripts and preliminary output reports, designing a user interface, and developing data security and stewardship protocols;
    a. This will include development and testing of the scripting modules that were not completed during this concept development stage, e.g.: extraction from EnergyPlus, output submission form for HPNC, and development of a reporting strategy for the 2030 Challenge based on user inputs.
2. Initial industry engagement through beta testing and tool technical refinement;
3. 24-month full rollout to industry, including province-wide education and engagement sessions; and
4. Ongoing technical refinement and support for any updates to the standards/programs or software.
    a. In order to maintain the relevance of the Energy Benchmarking Tool for users, the standards and programs that are included will be reviewed regularly, and scripting will be developed or modified as required to accommodate changes or updates.

5 FINANCIAL REPORT

5.1 External Funding Secured

The funding provided by TAF has proven beneficial in leveraging external funding from the IESO Conservation Fund, as described in Section 3.1.1. The additional funding that has been secured will enable the expansion of this project into a larger project, which includes the development of an end-user interface, online portal, data storage architecture, and end-user agreements, to make the Energy Benchmarking Tool accessible to the building design and construction industry, and the building regulatory community across Ontario.

5.2 Discussion of Any Budget Variances

As discussed in detail in Section 3 of this report, a number of challenges and opportunities have impacted the outcome of this project.

1. The successful receipt of funding from the IESO Conservation Fund, and the corresponding expansion of the overall project has altered the approach taken for the scripting of the Energy Benchmarking Tool. The scripts are being written with this expansion in mind, to avoid future adjustments and re-programming. This added flexibility has taken greater time than was anticipated by the original TAF funding application.

2. The Toronto Green Standard (TGS) energy model files were not available for the completion of Task Areas 3 and 4. Negotiations are ongoing to coordinate the completion of these tasks, but this will fall within the broader project, and will not be completed using funding from TAF.
3. Beyond the 210 building characteristics that were extracted in the beta version of the tool, as described in the grant proposal, an additional 190 characteristics have been included in the Energy Benchmarking Tool as required for program submission forms, or as useful metrics for the benchmarking process. These characteristics were selected based on RWDI's industry experience and the recommendation of the TAF staff, however their inclusion has doubled the number of data points that are extracted from each energy model, which therefore greatly increased the effort required for the project.

6 CARBON EMISSION REDUCTIONS

The purpose of this project is to develop a benchmarking process for new construction energy models. In contrast to the application of a specific energy conservation measure it is difficult to quantifiably estimate the benefits of a benchmarking program. However, we know that when benchmarked metrics are introduced and managed it often leads to improvement – "you can't manage what you don't measure". To provide an estimated potential benefit of our project we will look to the well-established benefits of benchmarking in the existing building sector.

In Toronto, Civic Action's Race to Reduce achieved a 12% energy reduction over four years of benchmarking. While this was only one building type (office) and participants were self-selected, which likely skews performance higher, the results are nonetheless impressive. To be more conservative, we look to New York City and San Francisco who have had mandatory energy reporting of existing building energy use through EnergyStar Portfolio Manager in place for 3 and 4 years, respectively. Over this time period the cities have reported near 6% and 8% energy reductions in their existing building stock as a result of the implementation of their benchmarking programs. We believe that this Energy Benchmarking Tool could see a similar benefit, i.e. a 7% reduction in energy consumption, in the performance of new construction buildings.

To estimate the absolute potential of the project requires a forecast of anticipated new construction activity in Ontario. Obtaining the data necessary to prepare this estimate proved challenging; however, the following provides our best effort estimate.

We were able to obtain TOCore data, which estimates that there are currently 76,687 residential units in the development pipeline, all of which could potentially be impacted by our tool. Assuming an average unit size of 70 m^2 and a gross-up ratio of 1.2 the total residential area in the pipeline equates to a gross floor area of 6,441,708 m^2. To estimate our Project's potential impact on the commercial building sector we used historic TOCore data, which estimates commercial development at 950,000 m^2 since 2011 and have assumed that a similar gross floor area of commercial development is also in the pipeline. To annualize these estimates, we divided the total GFA estimates by 5 years, which equates to estimates of 1,200,000 and 190,000 m^2/yr of residential and commercial development in Toronto's core, respectively.

To scale these estimates up to represent total City of Toronto and Province of Ontario estimates we have assumed that development in Toronto's core represents 40% of development in the City and that development in Toronto represents 20% of all development in the Province. This scales the total residential and commercial development in Ontario to 16,000,000 and 2,400,000 m^2/yr, respectively.

Recent work completed for the City of Toronto in preparation for TGS v3 estimated the energy use and GHG intensity of residential and commercial buildings, which meet the minimum performance requirements of the OBC to be 225 and 233 kWh/m^2/yr, and 0.035 tonnes/m2 and 0.037 tonnes/m^2, respectively. For the purposes of this analysis we will assume an average for both residential and commercial buildings of 230 kWh/m^2/yr and 0.036 tonnes CO_2e/m^2.

Based on the above assumptions we estimate our Project's province-wide annual potential as follows:

Energy Savings: 296,000,000 kWh/yr
= [7% reduction in energy consumption] x [16,000,000 m^2/yr + 2,400,000 m^2/yr] x [230 kWh/m^2/yr]

GHG Emissions: 46,368 tonnes of CO_2e/yr
= [7% reduction in energy consumption] x [16,000,000 m^2/yr + 2,400,000 m^2/yr] x [0.036 tonnes/m^2]

We believe these estimates to be conservative given that our project will target not only the residential and commercial sectors, but any building required to prepare an energy model, which would add savings realized by institutional and industrial developments to the potential benefit of this Project.

7 FINAL REPORT QUESTIONAIRE NARRATIVE

Schedule C – Final Report Questionnaire

1. *Did you carry out the project as planned? If not, what changed and why? Did anything happen that you did not anticipate? If you had to do it all over again, what would you do differently? Describe the single most positive and negative aspects of the project.*

The project was not carried out precisely as planned due to a number of challenges and opportunities that arose during the project, as detailed in Section 3. In hindsight, the time and effort required to complete each of the deliverables was underestimated by the original funding proposal. Anticipating this and either reducing the overall project scope or expanding the project budget (in terms of time and funding) would allow for a more smoothly delivered project.

The most positive aspect of this project has been the enthusiastic response it has generated from the wider building industry and community. At RWDI, we have been striving towards the expanded use of aggregated data to inform new construction design through benchmarking of performance trends, and have found that our clients are very interested in learning from this analysis. In hearing that we are developing a tool to facilitate this process, several clients and collaborators have signed up to be volunteer beta testers.

A negative aspect of this project has been the challenge to extract data from a proprietary software, IES-VE, as described in Section 3.3.3. We have reached out to IES for support and in an attempt to develop a collaborative relationship, but at this time no such relationship has been established. As a result, we have had to reduce the scope of our data extraction when IES-VE is the energy modelling software used. We are hopeful that as the Energy Benchmarking Tool gains traction in the industry, the developers of IES-VE will be encouraged to collaborate with us.

2. *What results have you accomplished? How do you measure success and how have your contributions led to specific measurable results? How do the results of your work compare with the objectives identified in the proposal? What other progress have you made in achieving your goals?*

At this time, 400 building characteristics have been identified from six programs and standards relevant in the Ontario building context, and each has been correlated with its specific data output location in three leading energy modelling software. Functioning Python modules have been written for two of these three energy modelling applications (eQUEST and IES-VE), and the development of the module for EnergyPlus is in progress.

This does not fully achieve the results and objectives that were outlined in the proposal, which include full tool development for all three applications, and testing of 25 energy models provided by the City of Toronto. Full discussion of the missed objectives can be found in Section 3 of this report.

Despite these shortcomings, the project has successfully achieved its goal of receiving funding from the IESO through its Conservation Fund. The additional budget will be used to expand the Energy Benchmarking Tool to its full intended scope, facilitate a beta testing phase with industry volunteers, and introduce the final tool to the building industry using educational programming and ongoing technical support.

3. *What have you learned from your project? What were the critical elements of your success? How will you apply your lessons? Lessons learned may relate to collaboration strategies, communications, policy, scientific matters or other interesting insights gained from your work.*

The funding provided by TAF allowed RWDI to research the outputs generated by the energy modelling applications, the similarities and differences between these outputs, and how they can be utilized beyond the modelling process itself. The lessons learned during this initial phase of the project will be carried forward into the further development of the tool as part of the IESO funding that has been received.

4. *How has your project contributed to the greater city-wide/provincial/national campaigns or strategies for climate change and/or air pollution in Canada? What will happen as a result of the project in the next five years or beyond?*

Buildings are responsible for a significant portion of greenhouse gas emissions at the federal, provincial, and municipal levels – approximately 40%. This statistic has brought energy efficient and environmentally responsible building design forward as a focus of climate change action plans and strategies at all levels, as authorities recognize the potential for greenhouse gas emission reductions in the building industry. The Energy

Benchmarking Tool holds the potential to positively impact the building industry by reducing energy consumption in new construction buildings through benchmarking and energy conservation measures.

5. *How did your collaboration efforts contribute to the project? Describe your collaboration activities with a comment on how you measured impact and what results can be traced back specifically to your collaboration efforts? What challenges did you face with your collaborations?*

This positive attention that the TAF funding has brought to this Energy Benchmarking Tool project has been instrumental in strengthening the collaborative opportunities at the core of the project. Through this project, relationships have been fostered between RWDI and the Ontario Association of Architects, the Toronto 2030 District, the IESO, the Canadian chapter of the International Building Performance Simulation Association, and 10 volunteer beta testing firms and organizations.

A major collaboration effort in this project was the attempted collaboration with the City of Toronto, which had the goal of using Toronto Green Standard energy model submissions to test the reliability and robustness of the Energy Benchmarking Tool. This collaboration faced a major challenge in the confidentiality concerns of the City, which led to their unwillingness to participate. As a result, two of the deliverables of this project, which were tied to this collaboration, were unable to be achieved. Nevertheless, there are ongoing negotiations underway with the City of Toronto to determine how a future partnership could be established. By accommodating the privacy concerns of the City, the Energy Benchmarking Tool will be stronger, overall, and more easily adopted by similar municipal or regulatory bodies. Please refer to Section 3.2 of this report for more discussion.

6. *How did you find your experience working with the The Atmospheric Fund? Please provide feedback on positive aspects and areas where the relationship could be improved.*

The experience of working with TAF has been overall a positive one. Each deliverable has been well-received by the TAF team, and constructive feedback has been offered to improve the overall project. The enthusiasm with which TAF adopted this concept development project has driven our ability to promote the full Energy Benchmarking Tool project, and was instrumental in the success of the IESO Conservation Fund application.

7. *We encourage you to provide interesting high resolution photos and images (no more than three), or internet links related to your project. For those that are publishable, please include permission for us to use them in our publications and include credit details*.

N/A

APPENDICES

Appendix A – Full Matrix w/ status within tool

Appendix B – Use Case Diagram and System Flowchart

Appendix C – Source Code

Appendix D – Sample Submission Reports from Test Runs

APPENDIX A:

FULL MATRIX

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0000	Project Name	Y			
Output-0001	Anonymous Name	Y			
Output-0002	Energy Model Status	Y			
Output-0003	Design Phase	Y			
Output-0004	Objective	Y			
Output-0005	Completion Year	Y			
Output-0006	Tender Year	Y			
Output-0007	SPA-Number	Y			
Output-0008	Energy Modeller Name	Y			
Output-0009	Architect Name	Y			
Output-0010	Code compliance path	Y			
Output-0011	Name of Applicant	Y			
Output-0012	Company Name	Y			
Output-0013	Primary Use Type	Y			
Output-0014	Secondary Use Type	Y			
Output-0015	Building Type	Y			
Output-0016	Gross Floor Area	Y			
Output-0017	Modelled Floor Area		LV-B	BPRM: 1.1 General Information	ABUPS: Building Area
Output-0018	Total Floor Area		LV-B	BPRM: 1.1 General Information	ABUPS: Building Area
Output-0019	Energy Modelled Year (for Hourly data)	Y		Reports\PRM\UseDemand,DetailedTables	
Output-0020	Floor Area Primary Use Type (%)	Y		Reports\PRM\UseDemand	
Output-0021	Floor Area Secondary Use Type (%)	Y		Reports\PRM\UseDemand	
Output-0022	Country	Y			
Output-0023	Province	Y			
Output-0024	City	Y			
Output-0025	Street	Y			
Output-0026	Address	Y			
Output-0027	Latitude	Y		APSimResults/Weather File	
Output-0028	Longitude	Y		APSimResults/Weather File	
Output-0029	Heating Degree Days			Weather file	Climate Data Summary: V
Output-0030	Cooling Degree Days			Weather file	Climate Data Summary: V
Output-0031	2030 Region	Y			
Output-0032	2030 Reference EUI (kWh/m2)	Y			
Output-0033	2030 Weighted Reference EUI (kWh/m2)	Y			
Output-0034	2030 Target Reduction (%)	Y			
Output-0035	2030 Target EUI (kWh/m2)	Y			
Output-0036	2030 Reduction	Y			

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0037	Proposed - Plug load energy (MJ)		BEPS - MISC Equip	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0038	Proposed - receptacle equipment energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0039	Proposed - receptacle equipment consumption		BEPS - MISC Equip	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0040	Proposed - receptacle equipment demand		PS-F	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0041	Proposed - Lights energy (MJ, kWh)		BEPS - AREA LIGHTS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0042	Proposed - Lights electrical annual consumption (kWh)		BEPS - AREA LIGHTS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0043	Proposed - Lights natural gas annual consumption (kWh)		BEPS - AREA LIGHTS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0044	Proposed - Lights energy use intensity (kWh/m2-yr)		Calculated	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0045	Proposed - interior lighting energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0046	Proposed - interior lighting consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0047	Proposed - interior lighting demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0048	Proposed - exterior lighting energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0049	Proposed - exterior lighting consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0050	Proposed - exterior lighting demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0051	Proposed - interior lighting - process energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0052	Proposed - interior lighting - process consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0053	Proposed - interior lighting - process demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0054	Proposed - Misc. Equipment electrical annual consumption (kWh)		BEPS - MISC EQUIPMT	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0055	Proposed - Misc. Equipment natural gas annual consumption (kWh)		BEPS - MISC EQUPMT	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0056	Proposed - Misc. Equipment energy use intensity (kWh/m2-yr)		Calculated	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0057	Proposed - Space heating energy (MJ, kWh)		BEPS - SPACE HEAT	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0058	Proposed - Space Heating electrical annual consumption (kWh)		BEPS - SPACE HEAT	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0059	Proposed - Space Heating natural gas annual consumption (kWh)		BEPS - SPACE HEAT	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0060	Proposed - Space Heating energy use intensity (kWh/m2-yr)		Calculated	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0061	Proposed - space heating energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0062	Proposed - space heating consumption		BEPS - SPACE HEAT	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0063	Proposed - space heating demand		PS-F	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0064	Proposed - Space cooling energy (MJ, kWh)		BEPS - SPACE COOL	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0065	Proposed - Space Cooling electrical annual consumption (kWh)		BEPS - SPACE COOL	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0066	Proposed - Space Cooling natural gas annual consumption (kWh)		BEPS - SPACE COOL	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0067	Proposed - Space Cooling energy use intensity (kWh/m2-yr)		Calculated	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0068	Proposed - space cooling energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0069	Proposed - space cooling consumption		BEPS - SPACE COOL	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0070	Proposed - space cooling demand		PS-F	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0071	Proposed - heat rejection energy type		PS-F	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0072	Proposed - heat rejection consumption		BEPS - HEAT REJECT	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0073	Proposed - heat rejection demand		PS-F	Reports\PRM\UseDemand	DEUCS: End Uses by Subd

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0074	Proposed - Pumps energy (MJ)		BEPS - PUMPS & MISC	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0075	Proposed - Pumps electrical annual consumption (kWh)		BEPS - PUMPS & MISC	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0076	Proposed - Pumps natural gas annual consumption (kWh)		BEPS - PUMPS & MISC	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0077	Proposed - Pumps energy use intensity (kWh/m2-yr)		Calculated	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0078	Proposed - pumps energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0079	Proposed - pumps consumption		BEPS - PUMPS & MISC	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0080	Proposed - pumps demand		PS-F	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0081	Proposed - Fans energy (MJ)		BEPS - VENT FANS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0082	Proposed - Fans electrical annual consumption (kWh)		BEPS - VENT FANS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0083	Proposed - Fans natural gas annual consumption (kWh)		BEPS - VENT FANS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0084	Proposed - Fans energy use intensity (kWh/m2-yr)		Calculated	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0085	Proposed - fans - interior ventilation energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0086	Proposed - fans - interior ventilation consumption		BEPS - VENT FANS	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0087	Proposed - fans - interior ventilation demand		PS-F	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0088	Proposed - fans - parking garage energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0089	Proposed - fans - parking garage consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0090	Proposed - fans - parking garage demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0091	Proposed - Service Hot Water energy (MJ, kWh)		BEPS - DOMHOT WATER	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0092	Proposed - Service Hot Water electrical annual consumption (kWh)		BEPS - DOMHOT WATER	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0093	Proposed - Service Hot Water natural gas annual consumption (kWh)		BEPS - DOMHOT WATER	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0094	Proposed - Service Hot Water energy use intensity (kWh/m2-yr)		Calculated	Calculated from Reports\PRM\UseDemand	Calculated from ABUPS
Output-0095	Proposed - service water heating energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0096	Proposed - service water heating consumption		BEPS - DOMHOT WATER	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0097	Proposed - service water heating demand		PS-F	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0098	Proposed - IT equipment energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0099	Proposed - IT equipment consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0100	Proposed - IT equipment demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0101	Proposed - refrigeration equipment energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0102	Proposed - refrigeration equipment consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0103	Proposed - refrigeration equipment demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0104	Proposed - fans - kitchen ventilation energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0105	Proposed - fans - kitchen ventilation consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0106	Proposed - fans - kitchen ventilation demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0107	Proposed - cooking energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0108	Proposed - cooking consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0109	Proposed - cooking demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0110	Proposed - industrial process energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0111	Proposed - industrial process consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0112	Proposed - industrial process demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0113	Proposed - elevators and escalators energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0114	Proposed - elevators and escalators consumption		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0115	Proposed - elevators and escalators demand		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0116	Proposed - heat pump supplementary energy type		BEPS or N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0117	Proposed - heat pump supplementary consumption		BEPS or N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0118	Proposed - heat pump supplementary demand		BEPS or N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0119	Proposed - Total peak demand summer (kW)			Reports\PRM\VistaResults,Detailed Tables	Calculated from ECPDM /
Output-0120	Proposed - Total peak demand winter (kW)			Reports\PRM\VistaResults,Detailed Tables	Calculated from ECPDM /
Output-0121	Proposed - Total peak (kW)			Reports\PRM\UseDemand	Calculated from DEUCS
Output-0122	Proposed - Renewables - Electric (kWh)			Reports\PRM\UseDemand	ABUPS: Electric Loads Sat
Output-0123	Proposed - Total - Electric (kWh), (MJ)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses
Output-0124	Proposed - Renewables - Natural Gas (kWh)		N/A	Reports\PRM\UseDemand	ABUPS: On-Site Thermal
Output-0125	Proposed - Total - Natural Gas (kWh),(m3),(MJ)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses
Output-0126	Proposed - Total - Oil / Other fuels (MJ)		N/A	Reports\PRM\UseDemand	ABUPS: End Uses
Output-0127	Proposed - Hourly electricity consumption data (kWh)				
Output-0128	Proposed - HVAC Auxilary		BEPS / NA	Primary VS Other HVAC system	Calculated from ABUPS: E
Output-0129	Proposed - Misc Electrical		BEPS / NA	Reports\PRM\UseDemand	Calculated from ABUPS: E
Output-0130	Proposed - Other Energy Use 1				
Output-0131	Proposed - Other Energy Use 2				
Output-0132	Proposed - Total Annual Energy		BEPS	Reports\PRM\UseDemand	ABUPS: Site and Source E
Output-0133	Proposed - Total Annual Energy Cost		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0134	Proposed - annual energy cost - electricity		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0135	Proposed - annual energy cost - natural gas		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0136	Proposed - annual energy cost - district cooling		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0137	Proposed - Total Annual co2e emissions		BEPS & Calculation	Reports\PRM\UseDemand	Calculated from ABUPS: E
Output-0138	Proposed - Percentage less co2 emmissions		BEPS & Calculation	Reports\PRM\UseDemand	Calculated
Output-0139	Reference - Plug loads energy (MJ)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0140	Reference - receptacle equipment energy type		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0141	Reference - receptacle equipment consumption - model output x4 orientati		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subd
Output-0142	Reference - receptacle equipment demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subd
Output-0143	Reference - Lights total annual energy usage (kWh, MJ)		BEPS - AREA LIGHTS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0144	Reference - Lights electrical annual consumption (kWh)		BEPS - AREA LIGHTS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0145	Reference - Lights natural gas annual consumption (kWh)		BEPS - AREA LIGHTS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0146	Reference - Lights energy use intensity (kWh/m2-yr)		Calcualted	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0147	Reference - interior lighting energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subd

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0148	Reference - interior lighting consumption - model output x4 orientation		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0149	Reference - interior lighting demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0150	Reference - interior lighting - process energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0151	Reference - interior lighting - process consumption - model output x4 orient		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0152	Reference - interior lighting - process demand - model output x4 orientation		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0153	Reference - exterior lighting energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0154	Reference - exterior lighting consumption - model output x4 orientation		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0155	Reference - exterior lighting demand - model output x4 orientation		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0156	Reference - Misc. Equipment electrical annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0157	Reference - Misc. Equipment natural gas annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0158	Reference - Misc. Equipment energy use intensity (kWh/m2-yr)		BEPS	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0159	Reference - Space Heating total annual energy usage (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0160	Reference - Space Heating electrical annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0161	Reference - Space Heating natural gas annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0162	Reference - Space Heating energy use intensity (kWh/m2-yr)		BEPS & Calculation	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0163	Reference - space heating energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0164	Reference - space heating consumption - model output x4 orientation		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0165	Reference - space heating demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0166	Reference - Space Cooling total annual energy usage (kWh, MJ)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0167	Reference - Space Cooling electrical annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0168	Reference - Space Cooling natural gas annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0169	Reference - Space Cooling energy use intensity (kWh/m2-yr)		BEPS & Calculation	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0170	Reference - space cooling energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0171	Reference - space cooling consumption - model output x4 orientation		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0172	Reference - space cooling demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0173	Reference - heat rejection energy type		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0174	Reference - heat rejection consumption - model output x4 orientation		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0175	Reference - heat rejection demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0176	Reference - Pumps energy (MJ)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0177	Reference - Pumps electrical annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0178	Reference - Pumps natural gas annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0179	Reference - Pumps energy use intensity (kWh/m2-yr)		BEPS & Calculation	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0180	Reference - pumps energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0181	Reference - pumps consumption - model output x4 orientation		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0182	Reference - pumps demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0183	Reference - Fans energy (MJ)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0184	Reference - Fans electrical annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0185	Reference - Fans natural gas annual consumption (kWh)		BEPS & Calculation	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0186	Reference - Fans energy use intensity (kWh/m2-yr)		BEPS & Calculation	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0187	Reference - fans - interior ventilation energy type		BEPS & Calculation	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0188	Reference - fans - interior ventilation consumption - model output x4 orient		BEPS & Calculation	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0189	Reference - fans - interior ventilation demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0190	Reference - fans - parking garage energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0191	Reference - fans - parking garage consumption - model output x4 orientatio		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0192	Reference - fans - parking garage demand - model output x4 orientation		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0193	Reference - fans - kitchen ventilation energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0194	Reference - fans - kitchen ventilation consumption - model output x4 orientation			Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0195	Reference - fans - kitchen ventilation demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0196	Reference - Service Hot Water total annual energy usage (kWh, MJ)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0197	Reference - Service Hot Water electrical annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0198	Reference - Service Hot Water natural gas annual consumption (kWh)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0199	Reference - Service Hot Water energy use intensity (kWh/m2-yr)		BEPS & Calculation	Reports\PRM\UseDemand	Calculated from ABUPS
Output-0200	Reference - service water heating energy type		Inferred	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0201	Reference - service water heating consumption - model output x4 orientatio		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0202	Reference - service water heating demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0203	Reference - IT equipment energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0204	Reference - IT equipment consumption - model output x4 orientation		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0205	Reference - IT equipment demand - model output x4 orientation		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0206	Reference - refrigeration equipment energy type		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0207	Reference - refrigeration equipment consumption - model output x4 orienta		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0208	Reference - refrigeration equipment demand - model output x4 orientation			Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0209	Reference - cooking energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0210	Reference - cooking consumption - model output x4 orientation		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0211	Reference - cooking demand - model output x4 orientation		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0212	Reference - industrial process energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0213	Reference - industrial process consumption - model output x4 orientation		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0214	Reference - industrial process demand - model output x4 orientation		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0215	Reference - elevators and escalators energy type		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0216	Reference - elevators and escalators consumption - model output x4 orienta		N/A	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0217	Reference - elevators and escalators demand - model output x4 orientation		N/A	Reports\PRM\UseDemand	DEUCS: End Uses by Subc
Output-0218	Reference - heat pump supplementary energy type		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0219	Reference - heat pump supplementary consumption - model output x4 orie		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses by Subc
Output-0220	Reference - heat pump supplementary demand - model output x4 orientation			Reports\PRM\UseDemand	ECPDM: Custom Monthly
Output-0221	Reference - Total peak demand summer (kW)			Reports\PRM\UseDemand	Calculated from ECPDM

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0222	Reference - Total peak demand winter (kW)			Reports\PRM\UseDemand	Calculated from ECPDM /
Output-0223	Reference - Peak Electric Demand			Reports\PRM\UseDemand	ECPDM: Custom Monthly
Output-0224	Reference - Hourly electricity consumption data (kWh)				
Output-0225	Reference - HVAC Auxilary (kWh)		BEPS	Reports\PRM\UseDemand	Calculated from ABUPS: E
Output-0226	Reference - Misc Electrical (kWh)		BEPS	Reports\PRM\UseDemand	Calculated from ABUPS: E
Output-0227	Reference - Other Energy Use 1 (kWh)				
Output-0228	Reference - Other Energy Use 2 (kWh)				
Output-0229	Reference - Total - Electric (MJ)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses
Output-0230	Reference - Total - Natural Gas (m3),(MJ)		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses
Output-0231	Reference - Total - Oil / Other fuels (MJ)		N/A	Reports\PRM\UseDemand	ABUPS: End Uses
Output-0232	Reference - Total Annual Energy		BEPS	Reports\PRM\UseDemand	ABUPS: Site and Source E
Output-0233	Reference - Total Annual Energy Cost		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0234	Reference - Total Annual CO2 Emissions		Calculated	Reports\PRM\UseDemand	Calculated from ABUPS: E
Output-0235	Reference - annual energy cost - electricity - model output x4 orientation		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0236	Reference - annual energy cost - natural gas - model output x4 orientation		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0237	Reference - annual energy cost - district cooling - model output x4 orientati		N/A	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0238	Electric GHGs		Calculated	Calculated from Reports\PRM\UseDemand	Calculated from ABUPS: E
Output-0239	Gas GHGs		Calculated	Calculated from Reports\PRM\UseDemand	Calculated from ABUPS: E
Output-0240	Energy Modelling Software	Y			
Output-0241	% Energy Use Reduction		Calculated	Reports\PRM\UseDemand	Calculated
Output-0242	% Energy Cost Reduction		Calculated	Reports\PRM\UseDemand	Calculated
Output-0243	Proposed Electricity Cost		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0244	Proposed Natural Gas Cost		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0245	Proposed Oil / Other Fuel Cost		N/A	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0246	Proposed Renewables Cost		N/A	Reports\PRM\UseDemand	
Output-0247	Reference Electricity Cost		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0248	Reference Natural Gas Cost		ES-D	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0249	Reference Oil / Other fuels Cost		N/A	Reports\PRM\UseDemand	ERSR: Annual Cost
Output-0250	Windows Usi		LV-D	Calculated from Reports\PRM\UseDemand	Calculated from Envelope
Output-0251	Walls (Above Grade) Rsi		LV-D	Reports\PRM\UseDemand	Envelope Summary: Exter
Output-0252	Roof Rsi		LV-D	Calculated from Reports\PRM\UseDemand	Calculated from Envelope
Output-0253	Average Enclosure Above Grade Rsi		LV-D	Calculated from Reports\PRM\UseDemand	Calculated from Envelope
Output-0254	Walls (Below Grade) Rsi		LV-D	Calculated from Reports\PRM\UseDemand	Calculated from Envelope
Output-0255	Window SHGC		LV-D	Reports\PRM\UseDemand	Envelope Summary: Exter
Output-0256	Windows Tvis		LV-D	Reports\PRM\UseDemand	Envelope Summary: Exter
Output-0257	North Surface Area - Walls Including Windows		LV-D	Reports\PRM\UseDemand	IVRS: Window-Wall Ratio
Output-0258	East Surface Area - Walls Including Windows		LV-D	Reports\PRM\UseDemand	IVRS: Window-Wall Ratio

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0259	South Surface Area - Walls Including Windows		LV-D	Reports\PRM\UseDemand	IVRS: Window-Wall Ratio
Output-0260	West Surface Area - Walls Including Windows		LV-D	Reports\PRM\UseDemand	IVRS: Window-Wall Ratio
Output-0261	Roof Surface Area - Roof Including Skylights		LV-D	Reports\PRM\UseDemand	IVRS: Skylight-Roof Ratio
Output-0262	North Surface Area - Windows		LV-D	Reports\PRM\UseDemand	IVRS: Window-Wall Ratio
Output-0263	East Surface Area - Windows		LV-D	Reports\PRM\UseDemand	IVRS: Window-Wall Ratio
Output-0264	South Surface Area - Windows		LV-D	Reports\PRM\UseDemand	IVRS: Window-Wall Ratio
Output-0265	West Surface Area - Windows		LV-D	Reports\PRM\UseDemand	IVRS: Window-Wall Ratio
Output-0266	Roof Surface Area - Skylights		LV-D	Reports\PRM\UseDemand	IVRS: Skylight-Roof Ratio
Output-0267	Window to Wall Ratio		Calculated	Reports\PRM\UseDemand	IVRS: Window-Wall Ratio
Output-0268	Volume		LV-B Building Total	Reports\PRM\UseDemand	IVRS: Zone Summary
Output-0269	Underground Wall (m2)		LV-D	Reports\PRM\UseDemand	IVRS: Zone Summary
Output-0270	Soffits (m2)				
Output-0271	Interior LPD (W/m2)		LV-B	Reports\PRM\UseDemand	IVRS: Zone Summary
Output-0272	Total People #		LV-B		Calculated from IVRS: Zor
Output-0273	Equipment Load Density (W/m2)		LV-B	Reports\PRM\UseDemand	IVRS: Zone Summary
Output-0274	Max cooling load excluding OA		Calculated		
Output-0275	Max heating load excluding OA		Calculated		
Output-0276	Max cooling load including OA		Calculated		
Output-0277	Max heating load including OA		Calculated		
Output-0278	Heating System Level - Capacity (kW)		Calculated		Equipment Summary: He
Output-0279	Heating System Level - Efficiency (%)		Calculated	Reports\PRM\VistaReport (Load/fuel)	Equipment Summary: He
Output-0280	Heating Plant Level - Capacity (kW)		Calculated		Equipment Summary: Cer
Output-0281	Heating Plant Level - Efficiency (%)		Calculated	Reports\PRM\VistaReport (Load/fuel)	Equipment Summary: Cer
Output-0282	Weighted Average Heating Efficiency		Calculated		
Output-0283	Cooling System Level - Capacity (kW)		Calculated		Equipment Summary: Coo
Output-0284	Cooling System Level - Efficiency (%)		Calculated	Reports\PRM\VistaReport (Load/fuel)	Equipment Summary: Coo
Output-0285	Cooling Plant Level - Capacity (kW)		Calculated		Equipment Summary: Cer
Output-0286	Cooling Plant Level - Efficiency (%)		Calculated	Reports\PRM\VistaReport (Load/fuel)	Equipment Summary: Cer
Output-0287	Weighted Average Cooling Efficiency		Calculated		
Output-0288	Primary Heating Source	Y			
Output-0289	Primary Cooling Source	Y			
Output-0290	Primary Ventilation System Type	Y			
Output-0291	Total Supply Air (m3/s)		SV-A		
Output-0292	Total Outside Air (m3/s)		SV-A		
Output-0293	Total Fan Load (kW)		SV-A	Calculated from Reports\PRM\UseDemand	
Output-0294	VSD on Fans	Y			
Output-0295	Economizer	Y			

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0296	VSD on Pumps	Y			
Output-0297	Heat Recovery	Y			
Output-0298	Heat Recovery (% effectiveness)	Y			
Output-0299	Lighting Occupancy Sensors	Y			
Output-0300	Lighting Daylight Sensors	Y			
Output-0301	Primary Lighting Fixture Type	Y			
Output-0302	Seconadary Lighting Fixture Type	Y			
Output-0303	LEED Version	Y			
Output-0304	Level of Certification	Y			
Output-0305	Certified	Y			
Output-0306	Total LEED points achieved	Y			
Output-0307	Total LEED points available	Y			
Output-0308	EAc1 Reference Standard	Y			
Output-0309	% MRc2	Y			
Output-0310	% MRc4	Y			
Output-0311	% MRc5	Y			
Output-0312	EAc5 Pursued	Y			
Output-0313	EAc1 Points Available	Y			
Output-0314	EAc1 Points Achieved	Y			
Output-0315	% Irrigation Reduction (WEc1)	Y			
Output-0316	% In-Building Reduction (WEc3)	Y			
Output-0317	Total In-Building Potable Water Use (L/yr)	Y			
Output-0318	Total In-Building Grey Water Use (L/yr)	Y			
Output-0319	Incremental cost of equipment	Y			
Output-0320	Incremental cost of labour	Y			
Output-0321	Incremental cost of energy assessments	Y			
Output-0322	Incremental cost of design	Y			
Output-0323	Incremental cost of project management	Y			
Output-0324	Incremental cost of shipping	Y			
Output-0325	Incremental cost of documentation	Y			
Output-0326	LLT Option followed	Y			
Output-0327	Floor Area - Assembly	Y		Reports\PRM\UseDemand	
Output-0328	Floor Area - Health/Institutional	Y		Reports\PRM\UseDemand	
Output-0329	Floor Area - Hotel/Motel	Y		Reports\PRM\UseDemand	
Output-0330	Floor Area - Light Manufacturing	Y		Reports\PRM\UseDemand	
Output-0331	Floor Area - Multifamily	Y		Reports\PRM\UseDemand	
Output-0332	Floor Area - Office	Y		Reports\PRM\UseDemand	

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0333	Floor Area - Restaurant	Y		Reports\PRM\UseDemand	
Output-0334	Floor Area - Retail	Y		Reports\PRM\UseDemand	
Output-0335	Floor Area - School	Y		Reports\PRM\UseDemand	
Output-0336	Floor Area - Warehouse	Y		Reports\PRM\UseDemand	
Output-0337	Floor Area - Other	Y		Reports\PRM\UseDemand	
Output-0338	Proposed Building Description	Y			
Output-0339	HVAC System Description - Reference Bldg Design	Y			
Output-0340	HVAC System Description - Proposed Design	Y			
Output-0341	Energy Efficiency Features in Proposed Design	Y			
Output-0342	Select Fuel - Service water heating		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses
Output-0343	Select Fuel - Space cooling		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses
Output-0344	Select Fuel - Space heating fuel		BEPS	Reports\PRM\UseDemand	ABUPS: End Uses
Output-0345	Select Fuel - Renewable energy fuel		N/A	Reports\PRM\UseDemand	N/A
Output-0346	Renewable energy costs (MJ)		N/A	Reports\PRM\UseDemand	N/A
Output-0347	Renewable REC system	Y			
Output-0348	Select Fuel - Energy credit		N/A	N/A	N/A
Output-0349	Energy credit costs (MJ)		N/A	N/A	N/A
Output-0350	Energy credit REC system	Y			
Output-0351	Electricity consumption units	Y			
Output-0352	Electricity demand units	Y			
Output-0353	Electricity Utility Rate Name	Y			
Output-0354	Electricity Utility Rate Structure	Y			
Output-0355	Natural Gas consumption units	Y			
Output-0356	Natural Gas demand units	Y			
Output-0357	Natural Gas Utility Rate Name	Y			
Output-0358	Natural Gas Utility Rate Structure	Y			
Output-0359	District Cooling consumption units	Y			
Output-0360	District Cooling demand units	Y			
Output-0361	District Cooling Utility Rate Name	Y			
Output-0362	District Cooling Utility Rate Structure	Y			
Output-0363	Site energy consumption unit used	Y			
Output-0364	Source energy consumption unit used	Y			
Output-0365	On-site renewable energy systems? (Y/N)	Y			
Output-0366	Renewable System - type	Y			
Output-0367	Renewable - Offset energy type	Y			
Output-0368	Renewable - Rated capacity	Y			
Output-0369	Renewable - Annual site energy generated	Y			

Field Label	Field Description	Manual entry	eQuest DOE-2	IES	EnergyPlus
Output-0370	Renewable - Annual energy cost offset ($/year)	Y			
Output-0371	Exceptional methods? (Y/N)	Y			
Output-0372	Exceptional method - calculation ID	Y			
Output-0373	Exceptional method - Energy type				
Output-0374	Exceptional method - Add to baseline / subtract from proposed				
Output-0375	Exceptional method - Site energy				
Output-0376	Exceptional method - Annual energy cost difference				
Output-0377	Interior lighting unregulated?	Y			
Output-0378	Exterior lighting unregulated?	Y			
Output-0379	Space heating unregulated?	Y			
Output-0380	Space cooling unregulated?	Y			
Output-0381	Pumps unregulated?	Y			
Output-0382	Heat rejection unregulated?	Y			
Output-0383	Fans - interior ventilation unregulated?	Y			
Output-0384	Fans - parking garage unregulated?	Y			
Output-0385	Service water heating unregulated?	Y			
Output-0386	Receptacle equipment unregulated?	Y			
Output-0387	IT equipment unregulated?	Y			
Output-0388	Interior lighting - process unregulated?	Y			
Output-0389	Refrigeration equipment unregulated?	Y			
Output-0390	Fans - kitchen ventilation unregulated?	Y			
Output-0391	Cooking unregulated?	Y			
Output-0392	Industrial process unregulated?	Y			
Output-0393	Elevators and escalators unregulated?	Y			
Output-0394	Heat pump supplementary unregulated?	Y			
Output-0395	Reference - reporting strategy	Y			
Output-0396	Reference - unmet heating hours			DetailedTables	ABUPS: Comfort and Setp
Output-0397	Proposed - unmet heating hours			DetailedTables	ABUPS: Comfort and Setp
Output-0398	Reference - unmet cooling hours			DetailedTables	ABUPS: Comfort and Setp
Output-0399	Proposed - unmet cooling hours			DetailedTables	ABUPS: Comfort and Setp
Output-0400	TEDI				
Output-0401	Infiltration				
Output-0402	EUI				

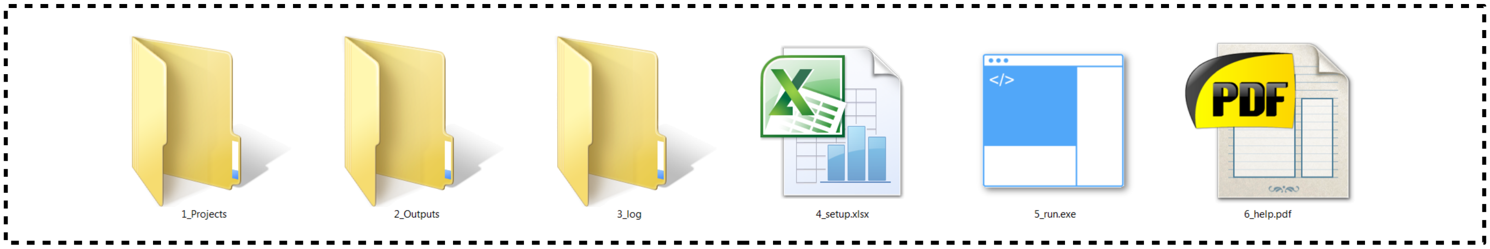
APPENDIX B:

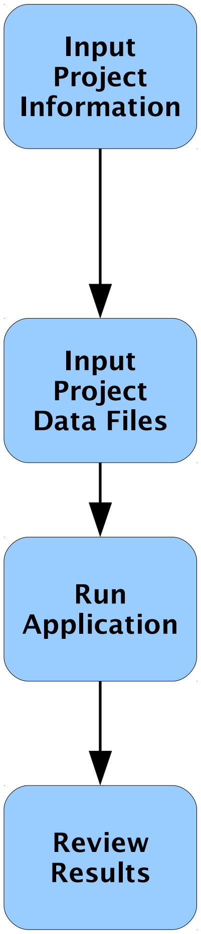
USE CASE DIAGRAM AND SYSTEM FLOWCHART

Energy Model Data Harvester – Use Case Diagram

2017-05-01

Application Folder





Input Project Information

- Edit the *Project Setup Workbook* “4_setup.xlsx”
  - Add each project to be harvested
  - Select desired *Program Output Forms*
  - Enter *Project Background Information* (manual entry items from matrix)
- Save and exit

Input Project Data Files

- Create a folder for each project inside of the “1_Projects” folder.
- Insert required energy model files into project folders

Run Application

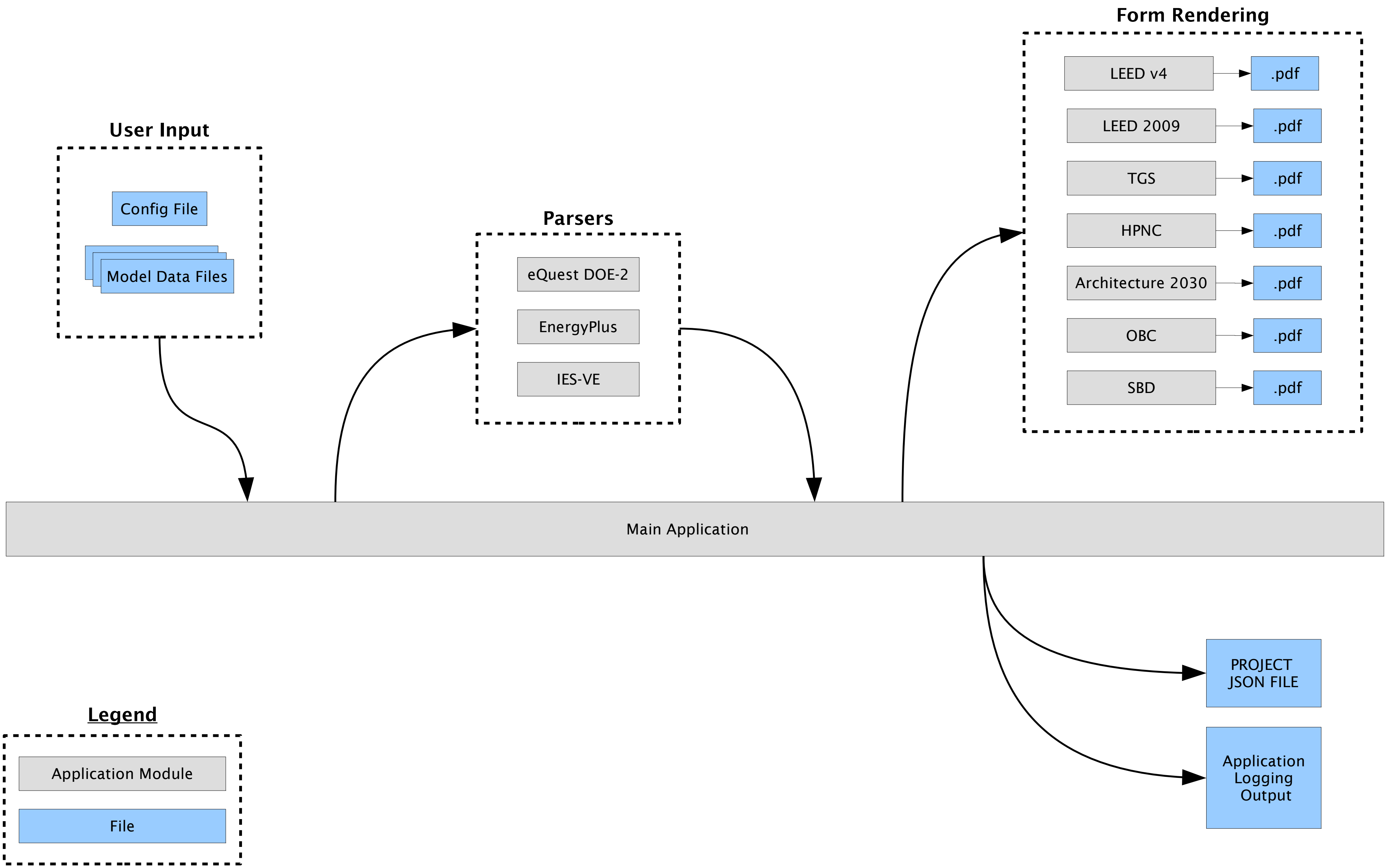
- Run the “5_run.exe” executable
  - First, this will read the Project Setup Workbook to determine the projects to be analyzed and the desired outputs
  - Project data files will be parsed and requested output files will be created

Review Results

- A log containing all messages, warnings, and errors will be created in the “3_log” folder
- A file containing a summary of all projects will be created in the “2_Outputs” folder
- A subfolder for each project will be created in the “2_Outputs” folder that that will contain the requested output files

Energy Model Data Harvester - System Flowchart

2017-05-01



APPENDIX C:

SOURCE CODE

```python
import fdfgen
import subprocess
import os
import tempfile


def fdf(template, fields, output):

    fdf_file = fdfgen.forge_fdf('', fields)

    handle, tmp_file = tempfile.mkstemp()
    f = open(tmp_file, 'wb')
    f.write(fdf_file)
    f.close()

    subprocess.run(["pdftk", "templates/"+template, "fill_form", tmp_file, "output", output])

    os.close(handle)
    os.remove(tmp_file)

```

```python
from bs4 import BeautifulSoup
from units import Area, Energy, Power, Currency


def splitUp(table):
    output = []
    outputRow = []
    for row in table.findAll('tr'):
        for column in row.findAll(['th','td']):
            outputRow.append(' '.join(column.findAll(text=True)))
        output.append(outputRow)
        outputRow = []
    return output


def spaceSummaryParse(table):
    output = {}

    #Check units
    header = table[0]
    for column in header[1:]:
        if column[-3:] != 'm 2': print('spaceSummaryParse area units error!:',column)

    for row in table[2:]:
        output[row[0]] = {'conditioned':Area(row[1],'m2'),
                          'unconditioned':Area(row[2],'m2'),
                          'total':Area(row[3],'m2')}
    return output


def prmParse(table):

    energy = []
    demand = []

    body = table[1:-2]

    for row in body[:-2]:
        if ' '.join(row[3].split(' ')[:-1]).strip() == 'Energy use':
            energy.append(row)
        elif ' '.join(row[3].split(' ')[:-1]).strip() == 'Demand':
            demand.append(row)

    energyOutput = {}
    demandOutput = {}

    for row in energy:
        process = row[1] == 'Yes'
        units = row[3].split(' ')[-1]

        energyOutput[row[0]] = {'process':process,
                                'type':row[2],
                                'proposed':Energy(row[4],units),
                                'reference':Energy(row[6],units),}

    for row in demand:
         units = row[3].split(' ')[-1]

         demandOutput[row[0]] = {'process':process,
                                 'type':row[2],
                                 'proposed':Power(row[4],units),
                                 'reference':Power(row[6],units),}

     return energyOutput,demandOutput


 def economicsParse(table):

     energy = {}
     cost = {}

     costUnits = table[1][1][4:].strip('( )')
     for row in table[2:]:
         cost[row[0]] = {
             'proposed': Currency(row[3],costUnits),
             'reference':Currency(row[5],costUnits),
         }

     for row in table[2:]:
         if row[1]: energyUnits = row[1]
         else: energyUnits = 'kwh'

         energy[row[0]] = {
             'proposed': Energy(row[2],energyUnits),
             'reference':Energy(row[4],energyUnits),
         }

     return energy, cost


 def isIES(file):
     with open (file) as f:
         html_doc = f.read()

     soup = BeautifulSoup(html_doc, 'html.parser')

     if soup.find(class_='iestext'):
         return True
     else: return False


 def iesParse(file):
     with open(file) as f:
         html_doc = f.read()

     soup = BeautifulSoup(html_doc, 'html.parser')

     spaceSummary = soup.findAll(text='1.2 - Space Summary')[-1].parent.parent.parent.next_sibling.find('table')
     prm = soup.find(id='tableprmcompliance')
     economics = soup.find(text='1.8.2 (b) Energy Cost & Consumption by energy Type - PRM Compliance').parent.parent.parent.next_sibling.find('table')
```

```python

    spaceSummary = splitUp(spaceSummary)[:-1]
    prm = splitUp(prm)
    economics = splitUp(economics)

    spaceSummary = spaceSummaryParse(spaceSummary)
    prmEnergy,prmDemand = prmParse(prm)
    economicsEnergy, economicsCost = economicsParse(economics)

    return {'spaceSummary':spaceSummary, 'prmEnergy':prmEnergy, 'prmDemand':prmDemand,'economicsEnergy':economicsEnergy,'economicsCost':economicsCost}

```

```python
import fdf

def obc(p,output_file_path):
    remaining = [
        ('Interior Lighting', 'test7'),
        ('Other_2', 'test8'),
        ('1_3', 'test9'),
        ('2_3', 'test10'),

        ('Building Energy 7', 'test17'),
        ('Building Energy 8', 'test18'),
        ('Units 7', 'test25'),
        ('Units 8', 'test26'),

        ('', 'test30'),
        ('2_6', 'test33'),
        ('Percentage less CO2e emissions by proposed building', 'test34'),
        ('undefined_7', 'test35'),
        ('undefined_9', 'test36'),

    ]

    fields = []

    if p.prop_int_lights_total: fields.append(('Building Energy 6', '{:,.0f}'.format(p.prop_int_lights_total.get('kwh'))))
    if p.prop_space_heat_total: fields.append(('Building Energy 1','{:,.0f}'.format(p.prop_space_heat_total.get('kwh'))))
    if p.prop_space_cool_total: fields.append(('Building Energy 2','{:,.0f}'.format(p.prop_space_cool_total.get('kwh'))))
    if p.prop_vent_fans_total: fields.append(('Building Energy 3','{:,.0f}'.format(p.prop_vent_fans_total.get('kwh'))))
    if p.prop_misc_total: fields.append(('Building Energy 4','{:,.0f}'.format(p.prop_misc_total.get('kwh'))))
    if p.prop_dhw_total: fields.append(('Building Energy 5','{:,.0f}'.format(p.prop_dhw_total.get('kwh'))))
    if p.prop_energy_total: fields.append(('undefined_5','{:,.0f}'.format(p.prop_energy_total.get('kwh'))))
    if p.prop_carbon_total: fields.append(('1_6','{:,.0f}'.format(p.prop_carbon_total.get('kg'))))
    if p.prop_cost_total: fields.append(('fill_63','{:,.0f}'.format(p.prop_cost_total)))
    # if p.: fields.append(('','{:,.0f}'.format(p..get('kwh'))))

    if p.ref_int_lights_total : fields.append(('Energy 6', '{:,.0f}'.format(p.ref_int_lights_total.get('kwh'))))
    if p.ref_space_heat_total: fields.append(('Energy 1','{:,.0f}'.format(p.ref_space_heat_total.get('kwh'))))
    if p.ref_space_cool_total: fields.append(('Energy 2','{:,.0f}'.format(p.ref_space_cool_total.get('kwh'))))
    if p.ref_vent_fans_total: fields.append(('Energy 3','{:,.0f}'.format(p.ref_vent_fans_total.get('kwh'))))
    if p.ref_misc_total: fields.append(('Energy 4','{:,.0f}'.format(p.ref_misc_total.get('kwh'))))
    if p.ref_dhw_total: fields.append(('Energy 5','{:,.0f}'.format(p.ref_dhw_total.get('kwh'))))
    if p.ref_energy_total: fields.append(('undefined_4','{:,.0f}'.format(p.ref_energy_total
.get('kwh'))))
    if p.ref_energy_total: fields.append(('Percentage less energy used', '{:,.1%}'.format(1-p.prop_energy_total.get()/p.ref_energy_total.get())))
    if p.ref_carbon_total: fields.append(('Total Annual CO2e Emissions','{:,.0f}'.format(p.ref_carbon_total.get('kg'))))
    if p.ref_cost_total: fields.append(('fill_62','{:,.0f}'.format(p.ref_cost_total)))
    # if p.: fields.append(('','{:,.0f}'.format(p..get('kwh'))))



    fields.append(('Units 1', 'kWh'))
    fields.append(('Units 2', 'kWh'))
    fields.append(('Units 3', 'kWh'))
    fields.append(('Units 4', 'kWh'))
    fields.append(('Units 5', 'kWh'))
    fields.append(('Units 6', 'kWh'))
    fields.append(('undefined_6', 'kWh'))
    fields.append(('undefined_10b', 'kg'))



    fdf.fdf('OBC_SB-10.pdf', fields, output_file_path)

```

```python
import openpyxl

from openpyxl.chart import (
    PieChart,
    Reference
)

def sbd(p,output_filepath):

    wb = openpyxl.load_workbook('templates/sbd.xlsx')
    ws = wb.active


    ws['C4'] = p.prop_area_lights_total.get('kwh')
    ws['D4'] = p.prop_misc_total.get('kwh')
    ws['E4'] = p.prop_space_heat_total.get('kwh')
    ws['F4'] = p.prop_space_cool_total.get('kwh')
    ws['G4'] = p.prop_vent_fans_total.get('kwh')
    ws['H4'] = p.prop_dhw_total.get('kwh')
    ws['L3'] = p.prop_energy_total.get('kwh')
    ws['L4'] = p.prop_carbon_total.get('kg')

    if p.reference:
        ws['C3'] = p.ref_area_lights_total.get('kwh')
        ws['D3'] = p.ref_misc_total.get('kwh')
        ws['E3'] = p.ref_space_heat_total.get('kwh')
        ws['F3'] = p.ref_space_cool_total.get('kwh')
        ws['G3'] = p.ref_vent_fans_total.get('kwh')
        ws['H3'] = p.ref_dhw_total.get('kwh')

        ws['K3'] = p.ref_energy_total.get('kwh')
        ws['K4'] = p.ref_carbon_total.get('kg')




    pie = PieChart()
    pie.width = 18.8468
    pie.height = 7.62

    labels = Reference(ws,min_col=3,max_col=8,min_row=2)
    data = Reference(ws,min_col=3,max_col=8,min_row=3)
    pie.add_data(data,from_rows=True)
    pie.set_categories(labels)
    pie.title = "SB-10 Reference Building"

    ws.add_chart(pie,"B6")

    pie2 = PieChart()
    pie2.width = 18.8468
    pie2.height = 7.62
    data2 = Reference(ws,min_col=3,max_col=8,min_row=4)
    pie2.add_data(data2,from_rows=True)
    pie2.set_categories(labels)
    pie2.title = "SBD Building"

    ws.add_chart(pie2,"B22")

    wb.save(output_filepath)



```

```python
import fdf, datetime, pprint


def tgs(p, output_filepath):

    fields = []

    remaining = [
        ('prop_summer_demand', 'test35'),
        ('prop_winter_demand', 'test39'),

        ('ref_pump_elec', 'test55'),
        ('ref_pump_eui', 'test56'),
        ('ref_pump_gas', 'test57'),

        ('ref_summer_demand', 'test58'),

        ('ref_winter_demand', 'test62'),
        ('savings_energy_percent', 'test63'),
        ('savings_energy_total', 'test64'),
        ('savings_summer_demand', 'test65'),
        ('savings_winter_demand', 'test66'),
    ]

    if p.spaNumber: fields.append(('spa_number', p.spaNumber))
    if p.architectCompany: fields.append(('architect_company', p.architectCompany))
    if p.architectEmail: fields.append(('architect_email', p.architectEmail))
    if p.architectName: fields.append(('architect_name', p.architectName))
    if p.architectPhone: fields.append(('architect_phone', p.architectPhone))
    if p.architectTitle: fields.append(('architect_title', p.architectTitle))
    if p.buildingType: fields.append(('building_type', p.buildingType))
    if p.codeCompliancePath: fields.append(('code_compliance_path', p.codeCompliancePath))
    if p.modellerCompany: fields.append(('modeller_company', p.modellerCompany))
    if p.modellerEmail: fields.append(('modeller_email', p.modellerEmail))
    if p.modellerName: fields.append(('modeller_name', p.modellerName))
    if p.modellerPhone: fields.append(('modeller_phone', p.modellerPhone))
    if p.modellerTitle: fields.append(('modeller_title', p.modellerTitle))
    if p.buildingArea: fields.append(('building_area', '{:,.0f}'.format(p.buildingArea.get('m2'))+' m²'))
    if p.projectAddress: fields.append(('project_address', p.projectAddress))

    if p.prop_dhw_elec: fields.append(('prop_dhw_elec', '{:,.0f}'.format(p.prop_dhw_elec.get('kwh'))))
    if p.prop_dhw_total: fields.append(('prop_dhw_eui', '{:,.2f}'.format(p.prop_dhw_total.get('kwh')/p.buildingArea.get('m2'))))
    if p.prop_dhw_gas: fields.append(('prop_dhw_gas', '{:,.0f}'.format(p.prop_dhw_gas.get('kwh'))))
    if p.prop_area_lights_elec: fields.append(('prop_lights_elec', '{:,.0f}'.format(p.prop_area_lights_elec.get('kwh'))))
    if p.prop_area_lights_total: fields.append(('prop_lights_eui', '{:,.2f}'.format(p.prop_area_lights_total.get('kwh')/p.buildingArea.get('m2'))))
    if p.prop_area_lights_gas: fields.append(('prop_lights_gas', '{:,.0f}'.format(p.prop_area_lights_gas.get('kwh'))))
    if p.prop_vent_fans_elec: fields.append(('prop_fans_elec', '{:,.0f}'.format(p.prop_vent_fans_elec.get('kwh'))))
    if p.prop_vent_fans_total: fields.append(('prop_fans_eui', '{:,.2f}'.format(p.
 prop_vent_fans_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.prop_vent_fans_gas: fields.append(('prop_fans_gas', '{:,.0f}'.format(p.prop_vent_fans_gas.get('kwh'))))
     if p.prop_space_heat_elec: fields.append(('prop_heat_elec', '{:,.0f}'.format(p.prop_space_heat_elec.get('kwh'))))
     if p.prop_space_heat_total: fields.append(('prop_heat_eui', '{:,.2f}'.format(p.prop_space_heat_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.prop_space_heat_gas: fields.append(('prop_heat_gas', '{:,.0f}'.format(p.prop_space_heat_gas.get('kwh'))))
     if p.prop_space_cool_elec: fields.append(('prop_cool_elec', '{:,.0f}'.format(p.prop_space_cool_elec.get('kwh'))))
     if p.prop_space_cool_total: fields.append(('prop_cool_eui', '{:,.2f}'.format(p.prop_space_cool_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.prop_space_cool_gas: fields.append(('prop_cool_gas', '{:,.0f}'.format(p.prop_space_cool_gas.get('kwh'))))
     if p.prop_misc_elec: fields.append(('prop_misc_elec', '{:,.0f}'.format(p.prop_misc_elec.get('kwh'))))
     if p.prop_misc_total: fields.append(('prop_misc_eui', '{:,.2f}'.format(p.prop_misc_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.prop_misc_gas: fields.append(('prop_misc_gas', '{:,.0f}'.format(p.prop_misc_gas.get('kwh'))))
     if p.prop_elec_total: fields.append(('prop_total_elec', '{:,.0f}'.format(p.prop_elec_total.get('kwh'))))
     if p.prop_energy_total: fields.append(('prop_total_eui', '{:,.2f}'.format(p.prop_energy_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.prop_gas_total: fields.append(('prop_total_gas', '{:,.0f}'.format(p.prop_gas_total.get('kwh'))))
     if p.prop_pumps_elec: fields.append(('prop_pump_elec', '{:,.0f}'.format(p.prop_pumps_elec.get('kwh'))))
     if p.prop_pumps_total: fields.append(('prop_pump_eui', '{:,.2f}'.format(p.prop_pumps_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.prop_pumps_gas: fields.append(('prop_pump_gas', '{:,.0f}'.format(p.prop_pumps_gas.get('kwh'))))
     # if p.: fields.append(('', p.))

     if p.ref_dhw_elec: fields.append(('ref_dhw_elec', '{:,.0f}'.format(p.ref_dhw_elec.get('kwh'))))
     if p.ref_dhw_total: fields.append(('ref_dhw_eui', '{:,.2f}'.format(p.ref_dhw_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.ref_dhw_gas: fields.append(('ref_dhw_gas', '{:,.0f}'.format(p.ref_dhw_gas.get('kwh'))))
     if p.ref_area_lights_elec: fields.append(('ref_lights_elec', '{:,.0f}'.format(p.ref_area_lights_elec.get('kwh'))))
     if p.ref_area_lights_total: fields.append(('ref_lights_eui', '{:,.2f}'.format(p.ref_area_lights_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.ref_area_lights_gas: fields.append(('ref_lights_gas', '{:,.0f}'.format(p.ref_area_lights_gas.get('kwh'))))
     if p.ref_vent_fans_elec: fields.append(('ref_fans_elec', '{:,.0f}'.format(p.ref_vent_fans_elec.get('kwh'))))
     if p.ref_vent_fans_total: fields.append(('ref_fans_eui', '{:,.2f}'.format(p.ref_vent_fans_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.ref_vent_fans_gas: fields.append(('ref_fans_gas', '{:,.0f}'.format(p.ref_vent_fans_gas.get('kwh'))))
     if p.ref_space_heat_elec: fields.append(('ref_heat_elec', '{:,.0f}'.format(p.ref_space_heat_elec.get('kwh'))))
     if p.ref_space_heat_total: fields.append(('ref_heat_eui', '{:,.2f}'.format(p.
 ref_space_heat_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.ref_space_heat_gas: fields.append(('ref_heat_gas', '{:,.0f}'.format(p.ref_space_heat_gas.get('kwh'))))
     if p.ref_space_cool_elec: fields.append(('ref_cool_elec', '{:,.0f}'.format(p.ref_space_cool_elec.get('kwh'))))
     if p.ref_space_cool_total: fields.append(('ref_cool_eui', '{:,.2f}'.format(p.ref_space_cool_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.ref_space_cool_gas: fields.append(('ref_cool_gas', '{:,.0f}'.format(p.ref_space_cool_gas.get('kwh'))))
     if p.ref_misc_elec: fields.append(('ref_misc_elec', '{:,.0f}'.format(p.ref_misc_elec.get('kwh'))))
     if p.ref_misc_total: fields.append(('ref_misc_eui', '{:,.2f}'.format(p.ref_misc_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.ref_misc_gas: fields.append(('ref_misc_gas', '{:,.0f}'.format(p.ref_misc_gas.get('kwh'))))
     if p.ref_elec_total: fields.append(('ref_total_elec', '{:,.0f}'.format(p.ref_elec_total.get('kwh'))))
     if p.ref_energy_total: fields.append(('ref_total_eui', '{:,.2f}'.format(p.ref_energy_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.ref_gas_total: fields.append(('ref_total_gas', '{:,.0f}'.format(p.ref_gas_total.get('kwh'))))
     if p.ref_pumps_elec: fields.append(('ref_pump_elec', '{:,.0f}'.format(p.ref_pumps_elec.get('kwh'))))
     if p.ref_pumps_total: fields.append(('ref_pump_eui', '{:,.2f}'.format(p.ref_pumps_total.get('kwh')/p.buildingArea.get('m2'))))
     if p.ref_pumps_gas: fields.append(('ref_pump_gas', '{:,.0f}'.format(p.ref_pumps_gas.get('kwh'))))
     # if p.: fields.append(('', p.))


     fields.append(('modelling_software', p.proposed.type))
     fields.append(('date', datetime.datetime.today().strftime('%d/%m/%Y')))


     fdf.fdf('TGS.pdf', fields, output_filepath)

```

```
class Measure:
    conversion = {}
    default = None

    def __init__(self, quantity, units):

        self.units = units.lower()

        self.checkUnits(self.units)



        self.value = float(str(quantity).replace(',',''))*self.conversion[self.units]

    def __add__(self,other):
        if self.__class__ == other.__class__:
            return self.__class__(self.get()+other.get())
        elif other in [0,None]:
            return self
        else:
            return NotImplemented

    def __radd__(self,other):
        if other in [0,None]:
            return Energy(self.get())
        else:
            return NotImplemented

    def get(self,units=None):
        if units == None:
            units = self.default
        self.checkUnits(units)
        return self.value/self.conversion[units.lower()]

    def checkUnits(self,units):
        if units.lower() not in self.conversion.keys():
            print(units,"not in conversion dictionary")



    @classmethod
    def units(cls):
        return cls.conversion.keys()



class Energy(Measure):
    conversion = {
        'kwh':0.0036,
        'therm':0.10548,
        'gj':1,
        'mj':0.001,
        'mbtu':1.05505585
        }
    default = 'gj'
    def __init__(self, quantity, units=default):
```

```python
        super().__init__(quantity,units)

class Power(Measure):
    conversion = {
        'mw':1000000,
        'kw':1000,
        'w':1,
        'btu/h': 0.29307107,
        'kbtu/h': 293.07107
        }
    default = 'w'
    def __init__(self, quantity, units=default):
        super().__init__(quantity,units)

class Area(Measure):
    conversion = {
        'ft2':0.092903,
        'm2':1
        }
    default = 'm2'
    def __init__(self, quantity, units=default):
        super().__init__(quantity,units)

class Volume(Measure):
    conversion = {
        'ft3':0.092903,
        'm3':1
        }
    default = 'm3'
    def __init__(self, quantity, units=default):
        super().__init__(quantity,units)

class Flow(Measure):
    conversion = {
        'cfm':0.471947,
        'l/s': 1,
        'm3/s':1,
    }
    default = 'l/s'

    def __init__(self, quantity, units=default):
        super().__init__(quantity,units)


class Currency(Measure):
    conversion = {
        '£': 1.69,
        'cad': 1,
    }
    default = 'cad'

    def __init__(self, quantity, units=default):
        super().__init__(quantity,units)

class Mass(Measure):
    conversion = {
        'tonnes':10^6,
        'kg':1000,
        'g': 1,
    }

    default = 'g'

    def __init__(self, quantity, units=default):
        super().__init__(quantity, units)
```

```python
import math
from units import Energy, Area, Volume, Power, Flow


class SimFile:
    def __init__(self, file):
        self.fullFile = None
        self.reports = None
        self.equestVersion = None

        self.fullFile = self.load(file)
        self.reports = self.parse(self.fullFile)
        self.doeVersion = self.getDoeVersion()

    def load(self, file):
        with open(file, 'r') as f:
            return list(f)

    def parse(self, data):
        reports = []
        report = []
        for row in data:
            if row[0] != '\x0c':
                report.append(row)
            else:
                if report != []:
                    reports.append(Report(report))
                report = [row]
        reports.append(Report(report))
        return reports

    def getDoeVersion(self):

        familiarVersions = ['DOE-2.1Ec133', 'DOE-2.2-47h2', 'DOE-2.2-48y']
        usedVersions = []

        for report in self.reports:
            if report.header:

                if report.doeVersion not in familiarVersions:
                    print('Unfimiliar DOE version:', report.doeVersion)
                    break
                if report.doeVersion not in usedVersions:
                    usedVersions.append(report.doeVersion)

        if len(usedVersions) > 1:
            print('Multiple eQuest Versions found:', usedVersions)
            return None

        elif len(usedVersions) == 1:
            return usedVersions[0]

        elif len(usedVersions) == 0:
            print('eQuest Version not found')
            return None

```

```python
class Report:
    def __init__(self, content):

        self.content = content
        self.header = None
        self.title = None
        self.body = None
        self.doeVersion = None
        self.runDatetime = None

        if self.content[0].strip() == 'DOE-2   UNITS   TABLE':
            self.title = 'DOE-2 UNITS TABLE'
            self.body = self.content[1:]
        elif self.content[0].strip() == '':
            if self.content[0].strip() == 'MESSAGE LIST FROM SYSTEMS     PROGRAM':
                self.title = 'MESSAGE LIST FROM SYSTEMS PROGRAM'
                self.body = self.content[3:]
        else:
            self.header = self.content[:4]
            self.title = self.header[2][8:12].strip()
            self.body = self.content[4:]

        if self.header:
            self.doeVersion = self.header[0][82:94].strip()

    def mergeHeaders(self, reports):
        header = reports[0].header
        if len(reports) > 1:
            for report in reports:
                if report.header[:3] != header[:3]:
                    print("Header Mismatch")
        return header

    def mergeBodies(self, reports):
        merged = []
        [merged.extend(report.body) for report in reports]
        return merged

    def filterReports(self, reports, title):
        return [report for report in reports if report.title == title]

    def checkHeader(self, header, schema, reportTitle):
        for column in schema:
            combined = ' '.join([row[column[2]:column[3]].strip() for row in header]).strip()
            if combined != column[0]: print(reportTitle, "Header Schema Error! Expected:", column[0], "Actual:",
                                            combined)

    def trimAtRepeatedNewlines(self,lines,count):
        newLineCount = 0
        output = []

        for line in lines:
            if line == '\n':
                newLineCount += 1
            else:
                newLineCount = 0
                output.append(line)
            if newLineCount == count:
                break
        return output

    def removeBlankLines(self, lines):
        return [line for line in lines if line != '\n']

class LSC(Report):
    def __init__(self, simFile):
        self.simFile = simFile
        reports = simFile.reports
        self.lscReports = self.filterReports(reports, 'LS-C')

        if len(self.lscReports) > 1: print("Multiple LSC reports")

        self.header = self.lscReports[0].header
        self.body = self.lscReports[0].body

        self.coolingLoad, self.heatingLoad = self.lscParse()

    def lscParse(self):

        row = self.body[45]

        if row[0:26].strip() != 'TOTAL LOAD': print('LSC Header Error')
        if row[56:64].strip() != 'KW': print('LSC Units Error')
        if row[103:108].strip() != 'KW': print('LSC Units Error')

        coolingLoad = Power(row[46:56],'kw')
        heatingLoad = Power(row[93:103],'kw')

        return coolingLoad, heatingLoad

class SSD(Report):
    def __init__(self, simFile):
        self.simFile = simFile
        reports = simFile.reports
        self.ssdReports = self.filterReports(reports, 'SS-D')

        if len(self.ssdReports) > 1: print("Multiple SSD reports")

        self.header = self.ssdReports[0].header
        self.body = self.ssdReports[0].body

        self.coolingLoad, self.heatingLoad = self.ssdParse()

    def ssdParse(self):

        row = self.body[39]

        if row[0:3].strip() != 'MAX': print('SSD Header Error')

        coolingLoad = Power(row[38:52],'kbtu/h')
        heatingLoad = Power(row[89:103],'kbtu/h')

        return coolingLoad, heatingLoad

class PSC(Report):
    def __init__(self, simFile):
        self.simFile = simFile
        reports = simFile.reports
        self.pscReports = self.filterReports(reports,'PS-C')

        self.header = self.mergeHeaders(self.pscReports)
        self.body = self.mergeBodies(self.pscReports)

        self.boilers, self.chillers = self.pscParse()

        self.boilerLoad = sum([boiler[1] for boiler in self.boilers])
        self.boilerElectricity = sum([boiler[2] for boiler in self.boilers])
        self.boilerFuel = sum([boiler[3] for boiler in self.boilers])

        if self.chillers:
            self.chillerLoad = sum([chiller[1] for chiller in self.chillers])
            self.chillerElectricity = sum([chiller[2] for chiller in self.chillers])
            self.chillerFuel = sum([chiller[3] for chiller in self.chillers])

        else:
            self.chillerLoad = Energy(0,'mbtu')
            self.chillerElectricity = Energy(0,'kwh')
            self.chillerFuel = Energy(0,'mbtu')

    def pscParse(self):
        table = self.trimAtRepeatedNewlines(self.body, 3)

        header = table[:3]
        body = table[4:]

        output = []
        outputRow = []

        if self.simFile.doeVersion[4:7] == '2.1':
            headerSchema = [
                ('ANNUAL LOAD (MBTU)','',90,99),
                ('ELEC USED (KWH)', '',108,117),
                ('THERMAL USED (MBTU)', '',117,127),
            ]
            boilerList = ['HW-BOILER']
            chillerList = ['OPEN-CENT-CHLR','OPEN-REC-CHLR','DBUN-CHLR']
            ignoreList = ['COOLING-TWR']
            bodySchema = headerSchema[:]
            bodySchema.insert(0, ('Name','',0,16))

            self.checkHeader(header,headerSchema,'PS-C')

            boilers = []
            chillers = []

            for i,v in enumerate(body):
                if i % 2 == 0:
                    for i2,v2 in enumerate(bodySchema):
                        value = v[v2[2]:v2[3]]
                        if i2 in [1,3]:
                            outputRow.append(Energy(value,'mbtu'))
                        elif i2 == 2:
                            outputRow.append(Energy(value,'kwh'))
                        elif i2 == 0:
                            outputRow.append(value.strip())
                    if outputRow[0] in boilerList:
                        boilers.append(outputRow)
                    elif outputRow[0] in chillerList:
                        chillers.append(outputRow)
                    elif outputRow[0] in ignoreList:
                        pass
                    else:
                        print("Unrecognized Equipment:",outputRow[0])
                    outputRow = []

            output = (boilers,chillers)

        elif self.simFile.doeVersion[4:7] == '2.2':
            headerSchema = [
                ('COOL LOAD (MBTU) (KBTU/HR)', '', 11, 21),
                ('HEAT LOAD (MBTU) (KBTU/HR)', '', 23, 33),
                ('ELEC USE (KWH) (KW)', '', 35, 45),
                ('FUEL USE (MBTU) (KBTU/HR)', '', 47, 57),
            ]
            bodySchema = headerSchema[:]
            boilerList = ['boiler']
            chillerList = ['chiller']
            ignoreList = ['tower','cooling','dhw','wh','dw']

            self.checkHeader(header,headerSchema,'PS-C')

            boilers = []
            chillers = []

            for i,v in enumerate(body):
                mod = i % 4
                if mod == 0:
                    outputRow.append(v[0:32].strip())
                elif mod == 1:
                    for column in bodySchema:
                        if v[column[2]:column[3]].strip() == '':
                            outputRow.append(0)
                        else:
                            outputRow.append(abs(float(v[column[2]:column[3]])))
                    output.append(outputRow)
                    outputRow = []
                continue

            output = [row for row in output if row[1] or row[2] ] #remove pumps (equipment with no loads)

            for item in output:
                identified = False
                name = item[0].lower()

                for boiler in boilerList:
                    if boiler in name:
                        boilers.append(item)
                        identified = True
                        break

                for chiller in chillerList:
                    if chiller in name:
                        chillers.append(item)
                        identified = True
                        break

                for ignore in ignoreList:
                    if ignore in name:
                        identified = True
                        break

                if not identified:
                    if not item[1]:
                        boilers.append(item)
                        identified = True
                        continue

                if not identified:
                    if float(item[2]) == 0:
                        chillers.append(item)
                        identified = True
                        continue

            boilers = [[boiler[0],Energy(boiler[1]+boiler[2],'mbtu'),Energy(boiler[3],'kwh'),Energy(boiler[4],'mbtu')] for boiler in boilers]
            chillers = [[chiller[0],Energy(chiller[1]+chiller[2],'mbtu'), Energy(chiller[3],'kwh'), Energy(chiller[4],'mbtu')] for chiller in chillers]

            output = (boilers, chillers)

        return output

class SSH(Report):
    def __init__(self, simFile):
        self.simFile = simFile
        reports = simFile.reports
        self.sshReports = self.filterReports(reports,'SS-H')

        self.parsed = self.sshParse()

        # for system in self.parsed:

    def sshParse(self):

        header1Schema = [
            ('- -F A N   E L E C- - -','',0,29),
            ('- -F U E L   H E A T- -','',32,55),
            ('- -F U E L   C O O L- -','',58,81),
            ('-E L E C   H E A T- -','',84,105),
            ('-E L E C   C O O L- -','',108,129)
        ]
        header2Schema = [
            ('MAXIMUM FAN LOAD (KW)','',17,29),
            ('MAXIMUM GAS OIL LOAD (KBTU/HR)','',42,55),
            ('MAXIMUM GAS OIL LOAD (KBTU/HR)','',68,81),
            ('MAXIMUM ELECTRIC LOAD (KW)','',93,105),
            ('MAXIMUM ELECTRIC LOAD (KW)','',117,129),
        ]
        bodySchema = header2Schema[:]

        if self.simFile.doeVersion[4:7] == '2.1':
            titleSlice = slice(46,92)
        elif self.simFile.doeVersion[4:7] == '2.2':
            titleSlice = slice(42,92)

        output = []

        for system in self.sshReports:

            outputRow = []

            outputRow.append(system.header[2][titleSlice].strip())

            table = self.removeBlankLines(system.body)
            self.checkHeader([table[0]],header1Schema,'SS-H')
            self.checkHeader(table[1:5],header2Schema,'SS-H')

            maxRow = table[19]

            for i,v in enumerate(bodySchema):
                value = maxRow[v[2]:v[3]]
                if i in [0,3,4]: outputRow.append(Power(value, 'kw'))
                elif i in [1,2]: outputRow.append(Power(value,'kbtu/h'))

            output.append(outputRow)

        return output

class SVA(Report):
    def __init__(self,simFile):
        self.simFile = simFile
        reports = simFile.reports
        self.svaReports = self.filterReports(reports,'SV-A')

        self.parsed = self.svaParse()

    def svaParse(self):

        header1Schema = [
            ('CAPACITY (CFM )', '',9,19),
        ]

        header2Schema = [
            ('OUTSIDE AIR RATIO', '', 41,52 ),
            ('COOLING EIR (BTU/BTU)', '',85,96),
            ('HEATING EIR (BTU/BTU)', '',96,107),
        ]

        body1Schema = header1Schema[:]
        body2Schema = header2Schema[:]

        if self.simFile.doeVersion[4:7] == '2.1':
            titleSlice = slice(38, 92)
        elif self.simFile.doeVersion[4:7] == '2.2':
            titleSlice = slice(41, 92)

        output = []
        parsedSystems = []

        for system in self.svaReports:

            outputRow = []

            table = self.removeBlankLines(system.body)
            contentRow1 = table[7]
            contentRow2 = table[3]

            title = system.header[2][titleSlice].strip()

            if title in parsedSystems:
                parsedSystems.append(title)
                continue

            if contentRow2[0:8].strip() in ['FPH','SUM']:
                continue

            parsedSystems.append(title)

            outputRow.append(title)


            self.checkHeader(table[4:7], header1Schema, 'SV-A')
            self.checkHeader(table[0:3], header2Schema, 'SV-A')



            for i, v in enumerate(body1Schema):
                value = contentRow1[v[2]:v[3]]
                outputRow.append(Flow(value,'cfm'))

            for i,v in enumerate(body2Schema):
                value = contentRow2[v[2]:v[3]]
                outputRow.append(value)

            output.append(outputRow)

        return output

class LVB(Report):
    def __init__(self, simFile):
        self.simFile = simFile
        reports = simFile.reports
        self.lvbReports = self.filterReports(reports, 'LV-B')

        self.header = self.mergeHeaders(self.lvbReports)
        self.body = self.mergeBodies(self.lvbReports)

        self.parsed, self.totals = self.parseLVB()

        self.people, self.grossArea, self.grossVolume = self.totals

        self.parsedExcluded = [row for row in self.parsed if not self.excludePlenums(row)]

        self.netArea = Area(sum([row[1] * row[9] for row in self.parsedExcluded]), 'ft2')
        self.netVolume = Volume(sum([row[1] * row[10] for row in self.parsedExcluded]), 'ft3')

        self.lightingPower = Power(sum([row[1] * row[4] * row[9] for row in self.parsedExcluded]), 'w')
        self.equipmentPower = Power(sum([row[1] * row[6] * row[9] for row in self.parsedExcluded]), 'w')

    def parseLVB(self):

        headerSchema21 = [
            ('SPACE', 'Space Name', 0, 12),
            ('SPACE*FLOOR MULTIPLIER', 'Multiplier', 12, 23),
            ('SPACE TYPE', 'Space Type', 26, 31),
            ('AZIMUTH', 'Azimuth', 33, 40),
            ('LIGHTING (WATT / SQFT )', 'Lighting Intensity', 41, 49),
            ('PEOPLE', 'People', 52, 58),
            ('EQUIP (WATT / SQFT )', 'Equipment Intensity', 61, 68),
            ('INFILTRATION METHOD', 'Infiltration Method', 70, 82),
            ('AIR CHANGES PER HOUR', 'Air Changes', 84, 95),
            ('AREA (SQFT )', 'Area', 95, 109),
            ('VOLUME (CUFT )', 'Volume', 109, 124)
        ]

        headerSchema22 = [
            ('SPACE', 'Space Name', 0, 29),
            ('SPACE*FLOOR MULTIPLIER', 'Multiplier', 29, 40),
            ('SPACE TYPE', 'Space Type', 41, 46),
            ('AZIM', 'Azimuth', 46, 53),
            ('LIGHTS (WATT / SQFT )', 'Lighting Intensity', 53, 61),
            ('PEOPLE', 'People', 61, 68),
            ('EQUIP (WATT / SQFT )', 'Equipment Intensity', 68, 76),
            ('INFILTRATION METHOD', 'Infiltration Method', 76, 89),
            ('ACH', 'Air Changes', 89, 95),
            ('AREA (SQFT )', 'Area', 95, 107),
            ('VOLUME (CUFT )', 'Volume', 107, 120)
        ]

        bodySchema21, bodySchema22 = headerSchema21[:], headerSchema22[:]

        bodySchema21[0] = (bodySchema22[0][0], bodySchema22[0][1], 0, 16)
        bodySchema21[1] = (bodySchema22[1][0], bodySchema22[1][1], 16, bodySchema21[1][3])

        bodySchema22[0] = (bodySchema22[0][0], bodySchema22[0][1], 0, 32)
        bodySchema22[1] = (bodySchema22[1][0], bodySchema22[1][1], 32, bodySchema22[1][3])

        if self.simFile.doeVersion[4:7] == '2.1':
            headerSchema, bodySchema = headerSchema21, bodySchema21
        elif self.simFile.doeVersion[4:7] == '2.2':
            headerSchema, bodySchema = headerSchema22, bodySchema22

        table = self.removeBlankLines(self.body)
        tableHeader = table[1:4]
        tableBody = table[4:-2]

        self.checkHeader(tableHeader, headerSchema, 'LV-B')

        output = []
        outputRow = []

        for line in tableBody:

            if line[0:16] == 'Spaces on floor:': continue

            for i, v in enumerate(bodySchema):
                value = line[v[2]:v[3]].strip()
                if i in [1, 3, 4, 5, 6, 8, 9, 10]:
                    value = float(value)
                outputRow.append(value)
            output.append(outputRow)
            outputRow = []

        totals = table[-1]

        totalSchema = [v for i, v in enumerate(bodySchema) if i in [5, 9, 10]]

        totalOutput = []

        for i, v in enumerate(totalSchema):
            totalOutput.append(float(totals[v[2]:v[3]]))

        return output, totalOutput

    def excludePlenums(self, row):

        plenums = ['plenum', 'plnm']

        for plenum in plenums:
            if plenum in row[0].lower():
                # print("Assumed Plenum (plenum in name):",row[0],)
                return True

        if row[10] / row[9] < 6:
            # print("Assumed Plenum (too short):",row[0], row[10]/row[9])
            return True

        return False
```

```python

class LVD(Report):
    def __init__(self, simFile):
        reports = simFile.reports
        self.lvdReports = self.filterReports(reports, 'LV-D')

        self.summary = self.lvdReports[-1]

        self.summaryHeader = self.summary.header
        self.summaryBody = self.summary.body

        self.parsed = self.parseLVD()

        self.northWallArea = sum([row[5] for row in self.parsed if row[0] in ['NORTH', 'NORTH-EAST']])
        self.northWindowArea = sum([row[4] for row in self.parsed if row[0] in ['NORTH', 'NORTH-EAST']])
        self.eastWallArea = sum([row[5] for row in self.parsed if row[0] in ['EAST', 'SOUTH-EAST']])
        self.eastWindowArea = sum([row[4] for row in self.parsed if row[0] in ['EAST', 'SOUTH-EAST']])
        self.southWallArea = sum([row[5] for row in self.parsed if row[0] in ['SOUTH', 'SOUTH-WEST']])
        self.southWindowArea = sum([row[4] for row in self.parsed if row[0] in ['SOUTH', 'SOUTH-WEST']])
        self.westWallArea = sum([row[5] for row in self.parsed if row[0] in ['WEST', 'NORTH-WEST']])
        self.westWindowArea = sum([row[4] for row in self.parsed if row[0] in ['WEST', 'NORTH-WEST']])

        self.roofArea = [row[5] for row in self.parsed if row[0] == "ROOF"][0]
        self.skylightArea = [row[4] for row in self.parsed if row[0] == "ROOF"][0]

        self.undergroundWallArea = sum([row[5] for row in self.parsed if row[0] == "UNDERGRND"])

        self.windowU = [row[1] for row in self.parsed if row[0] == "WALLS+ROOFS"][0]
        self.wallAboveU = [row[2] for row in self.parsed if row[0] == "ALL WALLS"][0]
        self.wallBelowU = sum([row[2] for row in self.parsed if row[0] == "UNDERGRND"])
        self.roofU = [row[2] for row in self.parsed if row[0] == "ROOF"][0]
        self.enclosureAboveU = [row[3] for row in self.parsed if row[0] == "WALLS+ROOFS"][0]

    def parseLVD(self):

        headerSchema = [
            ('AVERAGE U-VALUE/WINDOWS (BTU/HR-SQFT-F)', 'Window U-Value', 0, 35),
            ('AVERAGE U-VALUE/WALLS (BTU/HR-SQFT-F)', 'Wall U-Value', 40, 55),
            ('AVERAGE U-VALUE WALLS+WINDOWS (BTU/HR-SQFT-F)', 'Window+Wall U-Value', 60, 75),
            ('WINDOW AREA (SQFT)', 'Window Area', 77, 90),
            ('WALL AREA (SQFT)', 'Wall Area', 91, 105),
            ('WINDOW+WALL AREA (SQFT)', 'Window+Wall Area', 106, 125)
        ]

        bodySchema = headerSchema[:]
        bodySchema[0] = (bodySchema[0][0], bodySchema[0][1], 20, 35)
        bodySchema.insert(0, ('Classifier', 'Classifier', 0, 20))

        table = self.removeBlankLines(self.summaryBody)
        tableHeaders = table[:3]
        tableBody = table[3:]

        for column in headerSchema:
            combined = ' '.join([row[column[2]:column[3]].strip() for row in tableHeaders]).strip()
            if combined != column[0]: print("LV-D Header Schema Error! Expected:", column[0], "Actual:", combined)

        output = []
        outputRow = []

        for line in tableBody:
            for i, v in enumerate(bodySchema):
                value = line[v[2]:v[3]].strip()
                if i != 0:
                    value = float(value)
                outputRow.append(value)
            output.append(outputRow)
            outputRow = []

        return output

class BEPU(Report):
    def __init__(self, simFile):
        reports = simFile.reports
        self.bepureports = self.filterReports(reports, 'BEPU')

        if len(self.bepureports) > 1:
            print("Multiple BEPU reports")

        self.header = self.bepureports[0].header
        self.body = self.bepureports[0].body

        self.lightsElec = None
        self.lightsGas = None
        self.equipElec = None
        self.equipGas = None
        self.spaceHeatElec = None
        self.spaceHeatGas = None
        self.spaceCoolElec = None
        self.spaceCoolGas = None
        self.heatRejectElec = None
        self.heatRejectGas = None
        self.pumpsElec = None
        self.pumpsGas = None
        self.fansElec = None
        self.fansGas = None
        self.dhwElec = None
        self.dhwGas = None

        if simFile.doeVersion[4:7] == '2.1':
            self.parsed = self.parse21()

            self.lightsElec = Energy(self.parsed[0][2], self.parsed[0][1])
            self.lightsGas = Energy(self.parsed[1][2], self.parsed[1][1])
            self.equipElec = Energy(self.parsed[0][3], self.parsed[0][1])
            self.equipGas = Energy(self.parsed[1][3], self.parsed[1][1])
            self.spaceHeatElec = Energy(self.parsed[0][4], self.parsed[0][1])
            self.spaceHeatGas = Energy(self.parsed[1][4], self.parsed[1][1])
            self.spaceCoolElec = Energy(self.parsed[0][5], self.parsed[0][1])
            self.spaceCoolGas = Energy(self.parsed[1][5], self.parsed[1][1])
            self.heatRejectElec = Energy(self.parsed[0][6], self.parsed[0][1])
            self.heatRejectGas = Energy(self.parsed[1][6], self.parsed[1][1])
            self.pumpsElec = Energy(self.parsed[0][7], self.parsed[0][1])
            self.pumpsGas = Energy(self.parsed[1][7], self.parsed[1][1])
            self.fansElec = Energy(self.parsed[0][8], self.parsed[0][1])
            self.fansGas = Energy(self.parsed[1][8], self.parsed[1][1])
            self.dhwElec = Energy(self.parsed[0][9], self.parsed[0][1])
            self.dhwGas = Energy(self.parsed[1][9], self.parsed[1][1])

        elif simFile.doeVersion[4:7] == '2.2':

            self.parsed = self.parse22()

            self.areaLightsElec = sum([Energy(row[3], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.areaLightsGas = sum([Energy(row[3], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.taskLightsElec = sum([Energy(row[4], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.taskLightsGas = sum([Energy(row[4], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.equipElec = sum([Energy(row[5], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.equipGas = sum([Energy(row[5], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.spaceHeatElec = sum([Energy(row[6], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.spaceHeatGas = sum([Energy(row[6], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.spaceCoolElec = sum([Energy(row[7], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.spaceCoolGas = sum([Energy(row[7], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.heatRejectElec = sum([Energy(row[8], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.heatRejectGas = sum([Energy(row[8], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.pumpsElec = sum([Energy(row[9], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.pumpsGas = sum([Energy(row[9], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.fansElec = sum([Energy(row[10], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.fansGas = sum([Energy(row[10], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.refrigDisplayElec = sum([Energy(row[11], row[2]) for row in self.parsed
if row[1] == 'ELECTRICITY'])
            self.refrigDisplayGas = sum([Energy(row[11], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.htPumpElec = sum([Energy(row[12], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.htPumpGas = sum([Energy(row[12], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.dhwElec = sum([Energy(row[13], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.dhwGas = sum([Energy(row[13], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.extElec = sum([Energy(row[14], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.extGas = sum([Energy(row[14], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])

    def parse21(self):

        table = self.removeBlankLines(self.body)

        expectedHeadings = [
            (0, "ENERGY TYPE:", "Energy Type"),
            (1, "SITE UNITS:", "Site Units"),
            (4, "AREA LIGHTS", "Area Lights"),
            (5, "MISC EQUIPMT", "Miscellaneous Equipment"),
            (6, "SPACE HEAT", "Space Heating"),
            (7, "SPACE COOL", "Space Cooling"),
            (8, "HEAT REJECT", "Heat Rejection"),
            (9, "PUMPS & MISC", "Pumps and Miscellaneous"),
            (10, "VENT FANS", "Vent Fans"),
            (11, "DOMHOT WATER", "Domestic Hot Water"),
            (13, "TOTAL", "Total"),
        ]

        for i in expectedHeadings:
            if table[i[0]][:47].strip() != i[1]: print(i[2], "not found in BEPU:",
                                                       '"' + table[i[0]][:47].strip() + '" vs.', '"' + i[1] + '"')

        numberOfColumns = math.floor((len(table[0]) - 46) / 14)

        if numberOfColumns > 3:
            print("More than 3 columns in BEPU")

        output = []
        outputRow = []

        WIDTH = 14

        for column in range(numberOfColumns):
            start = 46 + WIDTH * column
            if column == numberOfColumns - 1:
                end = len(table[row[0]])
            else:
                end = 46 + WIDTH * (column + 1)

            for row in expectedHeadings:
                outputRow.append(table[row[0]][start:end].strip())
            output.append(outputRow)
            outputRow = []

        if output[0][0] != 'ELECTRICITY': print("Electricity Column missing")
        if output[1][0] != 'NATURAL-GAS': print("Natural Gas Column missing")
        if numberOfColumns > 2:
            if output[2][0] != 'RECOVERED': print("Recovered Column missing")

        return output

    def parse22(self):

        table = self.trimAtRepeatedNewlines(self.body,3)

        expectedHeadings = [
            "LIGHTS",
            "TASK LIGHTS",
            "MISC EQUIP",
            "SPACE HEATING",
            "SPACE COOLING",
            "HEAT REJECT",
            "PUMPS & AUX",
            "VENT FANS",
            "REFRIG DISPLAY",
            "HT PUMP SUPPLEM",
            "DOMEST HOT WTR",
            "EXT USAGE",
            "TOTAL"
        ]

        meterUnits = ["ELECTRICITY", "NATURAL-GAS"]

        WIDTH = 9
        line1 = table[0][12:]
        line2 = table[1][12:]

        for i, v in enumerate(expectedHeadings):
            combined = (
            line1[i * WIDTH:(i + 1) * WIDTH].strip() + ' ' + line2[i * WIDTH:(i + 1) * WIDTH].strip()).strip()
            if combined != v: print("Heading Error! Expected:", v, "Actual:", combined)

        tableInterior = table[3:-2]

        output = []
        outputRow = []

        for i, v in enumerate(tableInterior):
            if i % 2 == 0:
                outputRow.append(v[:5].strip())
                if v[5:16] not in meterUnits:
                    print("Unrecognized meter units", v[5:16])
                else:
                    outputRow.append(v[5:16].strip())
            else:

                unit = v[:12].strip().lower()
                if unit not in Energy.units(): print("Invalid unit in BEPS", unit)
                outputRow.append(unit)
                rest = v[12:]
                for i, v in enumerate(expectedHeadings[:-1]):
                    outputRow.append(float(rest[i * WIDTH:(i + 1) * WIDTH].strip()))
                outputRow.append(rest[(len(expectedHeadings) - 1) * WIDTH:].strip())

                output.append(outputRow)
                outputRow = []

        return output

class BEPS(Report):
    def __init__(self, simFile):
        reports = simFile.reports
        self.bepsreports = self.filterReports(reports, 'BEPS')

        if len(self.bepsreports) > 1:
            print("Multiple BEPS reports")

        self.header = self.bepsreports[0].header
        self.body = self.bepsreports[0].body

        self.areaLightsElec = None
        self.areaLightsGas = None
        self.equipElec = None
        self.equipGas = None
        self.spaceHeatElec = None
        self.spaceHeatGas = None
        self.spaceCoolElec = None
        self.spaceCoolGas = None
        self.heatRejectElec = None
        self.heatRejectGas = None
        self.pumpsElec = None
        self.pumpsGas = None
        self.fansElec = None
        self.fansGas = None
        self.dhwElec = None
        self.dhwGas = None
        self.taskLightsElec = None
        self.taskLightsGas = None
        self.refrigDisplayElec = None
        self.refrigDisplayGas = None
        self.htPumpElec = None
        self.htPumpGas = None
        self.extElec = None
        self.extGas = None
        self.elecTotal = None
        self.gasTotal = None

        if simFile.doeVersion[4:7] == '2.1':
            self.parsed = self.parse21()

```

```python
            self.areaLightsElec = Energy(self.parsed[0][2], 'mbtu')
            self.areaLightsGas = Energy(self.parsed[1][2], 'mbtu')
            self.equipElec = Energy(self.parsed[0][3], 'mbtu')
            self.equipGas = Energy(self.parsed[1][3], 'mbtu')
            self.spaceHeatElec = Energy(self.parsed[0][4], 'mbtu')
            self.spaceHeatGas = Energy(self.parsed[1][4], 'mbtu')
            self.spaceCoolElec = Energy(self.parsed[0][5], 'mbtu')
            self.spaceCoolGas = Energy(self.parsed[1][5], 'mbtu')
            self.heatRejectElec = Energy(self.parsed[0][6], 'mbtu')
            self.heatRejectGas = Energy(self.parsed[1][6], 'mbtu')
            self.pumpsElec = Energy(self.parsed[0][7], 'mbtu')
            self.pumpsGas = Energy(self.parsed[1][7], 'mbtu')
            self.fansElec = Energy(self.parsed[0][8], 'mbtu')
            self.fansGas = Energy(self.parsed[1][8], 'mbtu')
            self.dhwElec = Energy(self.parsed[0][9], 'mbtu')
            self.dhwGas = Energy(self.parsed[1][9], 'mbtu')
            self.elecTotal = Energy(self.parsed[0][10], 'mbtu')
            self.gasTotal = Energy(self.parsed[1][10], 'mbtu')

        elif simFile.doeVersion[4:7] == '2.2':
            self.parsed = self.parse22()

            self.areaLightsElec = sum([Energy(row[3], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.areaLightsGas = sum([Energy(row[3], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.taskLightsElec = sum([Energy(row[4], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.taskLightsGas = sum([Energy(row[4], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.equipElec = sum([Energy(row[5], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.equipGas = sum([Energy(row[5], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.spaceHeatElec = sum([Energy(row[6], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.spaceHeatGas = sum([Energy(row[6], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.spaceCoolElec = sum([Energy(row[7], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.spaceCoolGas = sum([Energy(row[7], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.heatRejectElec = sum([Energy(row[8], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.heatRejectGas = sum([Energy(row[8], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.pumpsElec = sum([Energy(row[9], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.pumpsGas = sum([Energy(row[9], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.fansElec = sum([Energy(row[10], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.fansGas = sum([Energy(row[10], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.refrigDisplayElec = sum([Energy(row[11], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.refrigDisplayGas = sum([Energy(row[11], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.htPumpElec = sum([Energy(row[12], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.htPumpGas = sum([Energy(row[12], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.dhwElec = sum([Energy(row[13], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.dhwGas = sum([Energy(row[13], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.extElec = sum([Energy(row[14], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.extGas = sum([Energy(row[14], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])
            self.elecTotal = sum([Energy(row[15], row[2]) for row in self.parsed if row[1] == 'ELECTRICITY'])
            self.gasTotal = sum([Energy(row[15], row[2]) for row in self.parsed if row[1] == 'NATURAL-GAS'])

    def parse21(self):

        table = self.removeBlankLines(self.body)

        expectedHeadings = [
            (0, "ENERGY TYPE:", "Energy Type"),
            (1, "UNITS: MBTU", "UNITS: MBTU"),
            (4, "AREA LIGHTS", "Area Lights"),
            (5, "MISC EQUIPMT", "Miscellaneous Equipment"),
            (6, "SPACE HEAT", "Space Heating"),
            (7, "SPACE COOL", "Space Cooling"),
            (8, "HEAT REJECT", "Heat Rejection"),
            (9, "PUMPS & MISC", "Pumps and Miscellaneous"),
            (10, "VENT FANS", "Vent Fans"),
            (11, "DOMHOT WATER", "Domestic Hot Water"),
            (13, "TOTAL", "Total"),
        ]

        for i in expectedHeadings:
            if table[i[0]][:47].strip() != i[1]: print(i[2], "not found in BEPS:",
                                                       '"' + table[i[0]][:47].strip() + '" vs.', '"' + i[1] + '"')

        numberOfColumns = math.floor((len(table[0]) - 46) / 14)

        if numberOfColumns > 3:
            print("More than 3 columns in BEPU")

        output = []
        outputRow = []

        WIDTH = 14

        for column in range(numberOfColumns):
            start = 46 + WIDTH * column
            if column == numberOfColumns - 1:
                end = len(table[row[0]])
            else:
                end = 46 + WIDTH * (column + 1)

            for row in expectedHeadings:
                outputRow.append(table[row[0]][start:end].strip())
            output.append(outputRow)
            outputRow = []

        if output[0][0] != 'ELECTRICITY': print("Electricity Column missing")
        if output[1][0] != 'NATURAL-GAS': print("Natural Gas Column missing")
        if numberOfColumns > 2:
            if output[2][0] != 'RECOVERED': print("Recovered Column missing")

        return output

    def parse22(self):

        table = self.trimAtRepeatedNewlines(self.body, 3)


        expectedHeadings = [
            "LIGHTS",
            "TASK LIGHTS",
            "MISC EQUIP",
            "SPACE HEATING",
            "SPACE COOLING",
            "HEAT REJECT",
            "PUMPS & AUX",
            "VENT FANS",
            "REFRIG DISPLAY",
            "HT PUMP SUPPLEM",
            "DOMEST HOT WTR",
            "EXT USAGE",
            "TOTAL"
        ]

        meterUnits = ["ELECTRICITY", "NATURAL-GAS"]

        WIDTH = 9
        line1 = table[0][12:]
        line2 = table[1][12:]

        for i, v in enumerate(expectedHeadings):
            combined = (
            line1[i * WIDTH:(i + 1) * WIDTH].strip() + ' ' + line2[i * WIDTH:(i + 1) * WIDTH].strip()).strip()
            if combined != v: print("Heading Error! Expected:", v, "Actual:", combined)

        tableInterior = table[3:-2]

        output = []
        outputRow = []

        for i, v in enumerate(tableInterior):
            if i % 2 == 0:
                outputRow.append(v[:5].strip())
                if v[5:16] not in meterUnits:
                    print("Unrecognized meter units", v[5:16])
                else:
                    outputRow.append(v[5:16].strip())
            else:

                unit = v[:12].strip().lower()
                if unit not in Energy.units(): print("Invalid unit in BEPS", unit)
                outputRow.append(unit)
                rest = v[12:]
                for i, v in enumerate(expectedHeadings[:-1]):
                    outputRow.append(float(rest[i * WIDTH:(i + 1) * WIDTH].strip()))
                outputRow.append(rest[(len(expectedHeadings) - 1) * WIDTH:].strip())

                output.append(outputRow)
                outputRow = []

        return output

class ESD(Report):
    def __init__(self, simFile):
        self.simFile = simFile
        reports = simFile.reports

        self.esdreports = self.filterReports(reports, 'ES-D')

        if len(self.esdreports) > 1:
            print("Multiple ESD reports")

        self.header = self.esdreports[0].header
        self.body = self.esdreports[0].body

        self.electricityCost = None
        self.gasCost = None
        self.totalCost = None

        self.parsed = self.parseESD()

        self.electricityCost = sum([row[5] for row in self.parsed if row[1] == "ELECTRICITY"])
        self.gasCost = sum([row[5] for row in self.parsed if row[1] == "NATURAL-GAS"])
        self.totalCost = self.electricityCost + self.gasCost

    def parseESD(self):

        table = self.trimAtRepeatedNewlines(self.body, 2)

        headerSchema21 = [
            ('UTILITY-RATE', 'Utility Rate Name', 0, 16),
            ('RESOURCE', 'Fuel Type', 20, 36),
            ('METERS', 'Meter Name', 40, 51),
            ('METERED ENERGY UNITS/YR', 'Metered Energy', 55, 74),
            ('TOTAL CHARGE ($)', 'Total Charge', 78, 88),
            ('VIRTUAL RATE ($/UNIT)', 'Virtual Rate', 92, 102),
            ('RATE USED ALL YEAR?', 'Rate Used All Year', 106, 115)
        ]
        headerSchema22 = [
            ('UTILITY-RATE', 'Utility Rate Name', 0, 32),
            ('RESOURCE', 'Fuel Type', 35, 51),
            ('METERS', 'Meter Name', 54, 65),
            ('METERED ENERGY UNITS/YR', 'Metered Energy', 68, 87),
            ('TOTAL CHARGE ($)', 'Total Charge', 90, 100),
            ('VIRTUAL RATE ($/UNIT)', 'Virtual Rate', 103, 113),
            ('RATE USED ALL YEAR?', 'Rate Used All Year', 116, 125)
        ]

        bodySchema21 = headerSchema21[:]
        bodySchema21[3] = ('METERED ENERGY UNITS/YR', 'Metered Energy Value', 55, 65)
        bodySchema21.insert(4, ('METERED ENERGY UNITS/YR', 'Metered Energy Units', 65, 74))

        bodySchema22 = headerSchema22[:]
        bodySchema22[3] = ('METERED ENERGY UNITS/YR', 'Metered Energy Value', 68, 78)
        bodySchema22.insert(4, ('METERED ENERGY UNITS/YR', 'Metered Energy Units', 78, 87))

        if self.simFile.doeVersion[4:7] == '2.1':
            self.headerSchema, self.bodySchema = headerSchema21, bodySchema21

        elif self.simFile.doeVersion[4:7] == '2.2':
            self.headerSchema, self.bodySchema = headerSchema22, bodySchema22

        for row in self.headerSchema:
            combined = ' '.join([i[row[2]:row[3]].strip() for i in table[:3]]).strip()
            if combined != row[0]: print("ESD Header Schema Error! Expected:", row[0], " Actual:", combined)

        output = []
        outputRow = []

        meterUnits = ["ELECTRICITY", "NATURAL-GAS"]
        for row in table[4:-2]:
            for i, v in enumerate(self.bodySchema):

                value = row[v[2]:v[3]].strip()
                if i in [3, 5, 6]:
                    value = float(value)
                if i == 1:
                    if value not in meterUnits: print("ESD: Unrecognized meter units", row[1])
                outputRow.append(value)
            output.append(outputRow)
            outputRow = []

        return output

def equest(p):
    sim = SimFile(p)

    return {
        'LV-B':LVB(sim),
        'BEPS':BEPS(sim),
        'ES-D':ESD(sim)
    }
```

```python
import openpyxl

def leed2009(p,output_filepath):

    wb = openpyxl.load_workbook('templates/leed2009.xlsx')
    ws = wb.active

    # ws['H55'] = 'text1'
    # ws['H56'] = 'text2'
    # ws['H57'] = 'text3'
    # ws['H58'] = 'text4'
    # ws['H59'] = 'text5'
    ws['I49'] = 'Natural gas'
    ws['I50'] = 'Electric'
    ws['I53'] = 'Natural gas'
    # ws['I55'] = 'Natural gas'
    # ws['I56'] = 'Natural gas'
    # ws['I57'] = 'Natural gas'
    # ws['I58'] = 'Natural gas'
    # ws['I59'] = 'Natural gas'

    ws['J48'] = p.prop_area_lights_elec.get('mj')
    ws['J49'] = p.prop_space_heat_gas.get('mj')
    ws['J50'] = p.prop_space_cool_elec.get('mj')
    ws['J51'] = p.prop_pumps_elec.get('mj')
    ws['J52'] = p.prop_vent_fans_elec.get('mj')
    ws['J53'] = p.prop_dhw_gas.get('mj')
    ws['J54'] = p.prop_plugs_elec.get('mj')
    # ws['J55'] = 1008
    # ws['J56'] = 1009
    # ws['J57'] = 1010
    # ws['J58'] = 1011
    # ws['J59'] = 1012

    ws['I65'] = p.prop_elec_total.get('mj')
    ws['I66'] = p.prop_gas_total.get('mj')
    # ws['I67'] = 1027
    ws['J65'] = p.prop_cost_elec
    ws['J66'] = p.prop_cost_gas
    # ws['J67'] = 1030

    if p.reference:
        ws['M48'] = p.ref_area_lights_elec.get('mj')
        ws['M49'] = p.ref_space_heat_gas.get('mj')
        ws['M50'] = p.ref_space_cool_elec.get('mj')
        ws['M51'] = p.ref_pumps_elec.get('mj')
        ws['M52'] = p.ref_vent_fans_elec.get('mj')
        ws['M53'] = p.ref_dhw_gas.get('mj')
        ws['M54'] = p.ref_plugs_elec.get('mj')
        # ws['M55'] = 1020
        # ws['M56'] = 1021
        # ws['M57'] = 1022
        # ws['M58'] = 1023
        # ws['M59'] = 1024

        ws['L65'] = p.ref_elec_total.get('mj')
        ws['L66'] = p.ref_gas_total.get('mj')
        # ws['L67'] = 1033
        ws['M65'] = p.ref_cost_elec
        ws['M66'] = p.ref_cost_gas
        # ws['M67'] = 1036

    wb.save(output_filepath)
```

```python
import os, os.path, datetime, openpyxl
import energyplus, ies, equest
import tgs, obc, leed2009, sbd
from units import Energy, Mass

PWB = '4_setup.xlsx'
PF = '1_projects'
OF = '2_outputs'


class Project:
    def __init__(self, projectName=None):
        self.projectName = projectName
        self.proposed = None
        self.reference = None
        self.projectAddress = None
        self.spaNumber = None
        self.architectName = None
        self.architectPhone = None
        self.architectEmail = None
        self.architectTitle = None
        self.architectCompany = None
        self.modellerName = None
        self.modellerPhone = None
        self.modellerEmail = None
        self.modellerTitle = None
        self.modellerCompany = None
        self.buildingType = None
        self.buildingArea = None
        self.codeCompliancePath = None

        self.prop_area_lights_elec = None
        self.prop_int_lights_elec = None
        self.prop_misc_elec = None
        self.prop_space_heat_elec = None
        self.prop_space_cool_elec = None
        self.prop_vent_fans_elec = None
        self.prop_dhw_elec = None
        self.prop_pumps_elec = None
        self.prop_plugs_elec = None
        self.prop_area_lights_gas = None
        self.prop_int_lights_gas = None
        self.prop_misc_gas = None
        self.prop_space_heat_gas = None
        self.prop_space_cool_gas = None
        self.prop_vent_fans_gas = None
        self.prop_dhw_gas = None
        self.prop_pumps_gas = None
        self.prop_plugs_gas = None
        self.prop_area_lights_total = None
        self.prop_int_lights_total = None
        self.prop_misc_total = None
        self.prop_space_heat_total = None
        self.prop_space_cool_total = None
        self.prop_vent_fans_total = None
        self.prop_dhw_total = None
        self.prop_pumps_total = None
        self.prop_plugs_total = None
        self.prop_elec_total = None
        self.prop_gas_total = None
        self.prop_energy_total = None
        self.prop_carbon_elec = None
        self.prop_carbon_gas = None
        self.prop_carbon_total = None
        self.prop_cost_elec = None
        self.prop_cost_gas = None
        self.prop_cost_total = None


        self.ref_area_lights_elec = None
        self.ref_int_lights_elec = None
        self.ref_misc_elec = None
        self.ref_space_heat_elec = None
        self.ref_space_cool_elec = None
        self.ref_vent_fans_elec = None
        self.ref_dhw_elec = None
        self.ref_pumps_elec = None
        self.ref_plugs_elec = None
        self.ref_area_lights_gas = None
        self.ref_int_lights_gas = None
        self.ref_misc_gas = None
        self.ref_space_heat_gas = None
        self.ref_space_cool_gas = None
        self.ref_vent_fans_gas = None
        self.ref_dhw_gas = None
        self.ref_pumps_gas = None
        self.ref_plugs_gas = None
        self.ref_area_lights_total = None
        self.ref_int_lights_total = None
        self.ref_misc_total = None
        self.ref_space_heat_total = None
        self.ref_space_cool_total = None
        self.ref_vent_fans_total = None
        self.ref_dhw_total = None
        self.ref_pumps_total = None
        self.ref_plugs_total = None
        self.ref_elec_total = None
        self.ref_gas_total = None
        self.ref_energy_total = None
        self.ref_carbon_elec = None
        self.ref_carbon_gas = None
        self.ref_carbon_total = None
        self.ref_cost_elec = None
        self.ref_cost_gas = None
        self.ref_cost_total = None

    def addName(self,name):
        self.projectName = name

    def addProposed(self, modelPath):
        if modelPath:
            self.proposed = Model(modelPath)
```

```python

            if self.proposed.type == 'IES':
                iesParsed = ies.iesParse(self.proposed.filepath)
                self.addReference(modelPath)

                self.buildingArea = iesParsed['spaceSummary']['Totals']['total']

                lights = [iesParsed['prmEnergy'][light] for light in ['Exterior Lighting', 'Interior Lighting Process','Internal Lighting']]
                self.prop_area_lights_elec = Energy(sum([light['proposed'].get() for light in lights if light['type']=='Electricity']))
                self.prop_area_lights_gas = Energy(sum([light['proposed'].get() for light in lights if light['type'] == 'Gas']))
                self.prop_area_lights_total = self.prop_area_lights_elec+self.prop_area_lights_gas

                intLights = [iesParsed['prmEnergy'][light] for light in
                          ['Interior Lighting Process', 'Internal Lighting']]
                self.prop_int_lights_elec = Energy(
                    sum([light['proposed'].get() for light in lights if light['type'] == 'Electricity']))
                self.prop_int_lights_gas = Energy(
                    sum([light['proposed'].get() for light in lights if light['type'] == 'Gas']))
                self.prop_int_lights_total = self.prop_int_lights_elec + self.prop_int_lights_gas

                dhws = [iesParsed['prmEnergy'][dhw] for dhw in ['Service Water Heating (Fossil Fuel)','Service Water Heating']]
                self.prop_dhw_elec = Energy(
                    sum([dhw['proposed'].get() for dhw in dhws if dhw['type'] == 'Electricity']))
                self.prop_dhw_gas = Energy(
                    sum([dhw['proposed'].get() for dhw in dhws if dhw['type'] == 'Gas']))
                self.prop_dhw_total = self.prop_dhw_elec + self.prop_dhw_gas

                coolings = [iesParsed['prmEnergy'][cooling] for cooling in ['Space Cooling','Heat Rejection']]
                self.prop_space_cool_elec = Energy(
                    sum([cooling['proposed'].get() for cooling in coolings if cooling['type'] == 'Electricity']))
                self.prop_space_cool_gas = Energy(
                    sum([cooling['proposed'].get() for cooling in coolings if cooling['type'] == 'Gas']))
                self.prop_space_cool_total = self.prop_space_cool_elec + self.prop_space_cool_gas

                heatings = [iesParsed['prmEnergy'][heating] for heating in ['Space Heating','Space Heating (Fossil Fuel)']]
                self.prop_space_heat_elec = Energy(
                    sum([heating['proposed'].get() for heating in heatings if heating['type'] == 'Electricity']))
                self.prop_space_heat_gas = Energy(
                    sum([heating['proposed'].get() for heating in heatings if heating['type'] == 'Gas']))
                self.prop_space_heat_total = self.prop_space_heat_elec + self.
```

```
prop_space_heat_gas

                miscs = [iesParsed['prmEnergy'][misc] for misc in ['Receptacle Equipment', 'Refrigeration', 'Data Centre Equipment', 'Elevators Escalators']]
                self.prop_misc_elec = Energy(
                    sum([misc['proposed'].get() for misc in miscs if misc['type'] == 'Electricity']))
                self.prop_misc_gas = Energy(
                    sum([misc['proposed'].get() for misc in miscs if misc['type'] == 'Gas']))
                self.prop_misc_total = self.prop_misc_elec + self.prop_misc_gas

                fans = [iesParsed['prmEnergy'][fan] for fan in ['Fans Interior', 'Fans Parking Garage']]
                self.prop_vent_fans_elec = Energy(
                    sum([fan['proposed'].get() for fan in fans if fan['type'] == 'Electricity']))
                self.prop_vent_fans_gas = Energy(
                    sum([fan['proposed'].get() for fan in fans if fan['type'] == 'Gas']))
                self.prop_vent_fans_total = self.prop_vent_fans_elec + self.prop_vent_fans_gas

                self.prop_elec_total = iesParsed['economicsEnergy']['Electricity']['proposed']
                self.prop_gas_total = iesParsed['economicsEnergy']['Gas']['proposed']
                self.prop_energy_total = iesParsed['economicsEnergy']['Subtotal (Model Outputs):']['proposed']

                self.prop_carbon_elec = Mass(self.prop_elec_total.get('kwh')*50,'g')
                self.prop_carbon_gas = Mass(self.prop_gas_total.get('kwh')*182,'g')
                self.prop_carbon_total = self.prop_carbon_elec + self.prop_carbon_gas

            elif self.proposed.type == 'eQuest':
                equestParsed = equest.equest(modelPath)
                beps = equestParsed['BEPS']
                esd = equestParsed['ES-D']
                self.buildingArea = equestParsed['LV-B'].netArea

                self.prop_area_lights_elec = beps.areaLightsElec + beps.taskLightsElec
                self.prop_int_lights_elec = beps.areaLightsElec + beps.taskLightsElec
                self.prop_misc_elec = beps.equipElec
                self.prop_space_heat_elec = beps.spaceHeatElec
                self.prop_space_cool_elec = beps.spaceCoolElec + beps.heatRejectElec
                self.prop_vent_fans_elec = beps.fansElec
                self.prop_dhw_elec = beps.dhwElec
                self.prop_pumps_elec = beps.pumpsElec
                self.prop_plugs_elec = beps.equipElec
                self.prop_area_lights_gas = beps.areaLightsGas + beps.taskLightsGas
                self.prop_int_lights_gas = beps.areaLightsGas + beps.taskLightsGas
                self.prop_misc_gas = beps.equipGas
                self.prop_space_heat_gas = beps.spaceHeatGas
                self.prop_space_cool_gas = beps.spaceCoolGas + beps.heatRejectGas
                self.prop_vent_fans_gas = beps.fansGas
                self.prop_dhw_gas = beps.dhwGas
                self.prop_pumps_gas = beps.pumpsGas
                self.prop_plugs_gas = beps.equipGas
                self.prop_area_lights_total = self.prop_area_lights_elec + self.prop_area_lights_gas
                self.prop_int_lights_total = self.prop_int_lights_elec + self.prop_int_lights_gas
                self.prop_misc_total = self.prop_misc_elec + self.prop_misc_gas
                self.prop_space_heat_total = self.prop_space_heat_elec + self.prop_space_heat_gas
                self.prop_space_cool_total = self.prop_space_cool_elec + self.prop_space_cool_gas
                self.prop_vent_fans_total = self.prop_vent_fans_elec + self.prop_vent_fans_gas
                self.prop_dhw_total = self.prop_dhw_elec + self.prop_dhw_gas
                self.prop_pumps_total = self.prop_pumps_elec + self.prop_pumps_gas
                self.prop_plugs_total = self.prop_plugs_elec + self.prop_plugs_gas
                self.prop_elec_total = beps.elecTotal
                self.prop_gas_total = beps.gasTotal
                self.prop_energy_total = self.prop_elec_total + self.prop_gas_total
                self.prop_carbon_elec = Mass(self.prop_elec_total.get('kwh')*50,'g')
                self.prop_carbon_gas = Mass(self.prop_elec_total.get('kwh')*50,'g')
                self.prop_carbon_total = self.prop_carbon_elec + self.prop_carbon_gas
                self.prop_cost_elec = esd.electricityCost
                self.prop_cost_gas = esd.gasCost
                self.prop_cost_total = esd.totalCost

            elif self.proposed.type == 'energyPlus':
                pass

    def addReference(self,modelPath):
        if modelPath:
            self.reference = Model(modelPath)

            if self.reference.type == 'IES':
                iesParsed = ies.iesParse(self.reference.filepath)

                lights = [iesParsed['prmEnergy'][light] for light in ['Exterior Lighting','Interior Lighting Process','Internal Lighting']]
                self.ref_area_lights_elec = Energy(sum([light['reference'].get() for light in lights if light['type']=='Electricity']))
                self.ref_area_lights_gas = Energy(sum([light['reference'].get() for light in lights if light['type'] == 'Gas']))
                self.ref_area_lights_total = self.ref_area_lights_elec + self.ref_area_lights_gas

                intLights = [iesParsed['prmEnergy'][light] for light in
                          ['Interior Lighting Process', 'Internal Lighting']]
                self.ref_int_lights_elec = Energy(
                    sum([light['reference'].get() for light in lights if light['type'] == 'Electricity']))
                self.ref_int_lights_gas = Energy(
                    sum([light['reference'].get() for light in lights if light['type'] == 'Gas']))
                self.ref_int_lights_total = self.ref_int_lights_elec + self.ref_int_lights_gas

                dhws = [iesParsed['prmEnergy'][dhw] for dhw in ['Service Water Heating (Fossil Fuel)','Service Water Heating']]
                self.ref_dhw_elec = Energy(
                    sum([dhw['reference'].get() for dhw in dhws if dhw['type'] == 'Electricity']))
                self.ref_dhw_gas = Energy(
                    sum([dhw['reference'].get() for dhw in dhws if dhw['type'] == 'Gas']))
                self.ref_dhw_total = self.ref_dhw_elec + self.ref_dhw_gas

                coolings = [iesParsed['prmEnergy'][cooling] for cooling in ['Space Cooling','Heat Rejection']]
                self.ref_space_cool_elec = Energy(
                    sum([cooling['reference'].get() for cooling in coolings if cooling['type'] == 'Electricity']))
                self.ref_space_cool_gas = Energy(
                    sum([cooling['reference'].get() for cooling in coolings if cooling['type'] == 'Gas']))
                self.ref_space_cool_total = self.ref_space_cool_elec + self.ref_space_cool_gas

                heatings = [iesParsed['prmEnergy'][heating] for heating in ['Space Heating','Space Heating (Fossil Fuel)']]
                self.ref_space_heat_elec = Energy(
                    sum([heating['reference'].get() for heating in heatings if heating['type'] == 'Electricity']))
                self.ref_space_heat_gas = Energy(
                    sum([heating['reference'].get() for heating in heatings if heating['type'] == 'Gas']))
                self.ref_space_heat_total = self.ref_space_heat_elec + self.ref_space_heat_gas

                miscs = [iesParsed['prmEnergy'][misc] for misc in ['Receptacle Equipment', 'Refrigeration', 'Data Centre Equipment', 'Elevators Escalators']]
                self.ref_misc_elec = Energy(
                    sum([misc['reference'].get() for misc in miscs if misc['type'] == 'Electricity']))
                self.ref_misc_gas = Energy(
                    sum([misc['reference'].get() for misc in miscs if misc['type'] == 'Gas']))
                self.ref_misc_total = self.ref_misc_elec + self.ref_misc_gas

                fans = [iesParsed['prmEnergy'][fan] for fan in ['Fans Interior', 'Fans Parking Garage']]
                self.ref_vent_fans_elec = Energy(
                    sum([fan['reference'].get() for fan in fans if fan['type'] == 'Electricity']))
                self.ref_vent_fans_gas = Energy(
                    sum([fan['reference'].get() for fan in fans if fan['type'] == 'Gas']))
                self.ref_vent_fans_total = self.ref_vent_fans_elec + self.ref_vent_fans_gas

                self.ref_elec_total = iesParsed['economicsEnergy']['Electricity']['reference']
                self.ref_gas_total = iesParsed['economicsEnergy']['Gas']['reference']
                self.ref_energy_total = iesParsed['economicsEnergy']['Subtotal (Model Outputs):']['reference']

                self.ref_carbon_elec = Mass(self.ref_elec_total.get('kwh')*50,'g')
                self.ref_carbon_gas = Mass(self.ref_gas_total.get('kwh')*182,'g')
                self.ref_carbon_total = self.ref_carbon_elec + self.ref_carbon_gas

            elif self.reference.type == 'eQuest':
                equestParsed = equest.equest(modelPath)
                esd = equestParsed['ES-D']
                beps = equestParsed['BEPS']

                self.ref_area_lights_elec = beps.areaLightsElec + beps.taskLightsElec
                self.ref_int_lights_elec = beps.areaLightsElec + beps.taskLightsElec
                self.ref_misc_elec = beps.equipElec
                self.ref_space_heat_elec = beps.spaceHeatElec
                self.ref_space_cool_elec = beps.spaceCoolElec + beps.heatRejectElec
                self.ref_vent_fans_elec = beps.fansElec
                self.ref_dhw_elec = beps.dhwElec
                self.ref_pumps_elec = beps.pumpsElec
                self.ref_plugs_elec = beps.equipElec
                self.ref_area_lights_gas = beps.areaLightsGas + beps.taskLightsGas
                self.ref_int_lights_gas = beps.areaLightsGas + beps.taskLightsGas
                self.ref_misc_gas = beps.equipGas
                self.ref_space_heat_gas = beps.spaceHeatGas
                self.ref_space_cool_gas = beps.spaceCoolGas + beps.heatRejectGas
                self.ref_vent_fans_gas = beps.fansGas
                self.ref_dhw_gas = beps.dhwGas
                self.ref_pumps_gas = beps.pumpsGas
                self.ref_plugs_gas = beps.equipGas
                self.ref_area_lights_total = self.ref_area_lights_elec + self.ref_area_lights_gas
                self.ref_int_lights_total = self.ref_int_lights_elec + self.ref_int_lights_gas
                self.ref_misc_total = self.ref_misc_elec + self.ref_misc_gas
                self.ref_space_heat_total = self.ref_space_heat_elec + self.ref_space_heat_gas
                self.ref_space_cool_total = self.ref_space_cool_elec + self.ref_space_cool_gas
                self.ref_vent_fans_total = self.ref_vent_fans_elec + self.ref_vent_fans_gas
                self.ref_dhw_total = self.ref_dhw_elec + self.ref_dhw_gas
                self.ref_pumps_total = self.ref_pumps_elec + self.ref_pumps_gas
                self.ref_plugs_total = self.ref_plugs_elec + self.ref_plugs_gas
                self.ref_elec_total = beps.elecTotal
                self.ref_gas_total = beps.gasTotal
                self.ref_energy_total = self.ref_elec_total + self.ref_gas_total
                self.ref_carbon_elec = Mass(self.ref_elec_total.get('kwh')*50,'g')
                self.ref_carbon_gas = Mass(self.ref_elec_total.get('kwh')*50,'g')
                self.ref_carbon_total = self.ref_carbon_elec + self.ref_carbon_gas
                self.ref_cost_elec = esd.electricityCost
                self.ref_cost_gas = esd.gasCost
                self.ref_cost_total = esd.totalCost

            elif self.reference.type == 'energyPlus':
                pass

    def addByPWB(self,pd):
        self.projectName = pd['Project Name']
        self.addProposed(os.path.join(PF,pd['Proposed File']))
        self.addReference(pd['Reference File'])
        self.projectAddress = pd['Project Address']
        self.spaNumber = pd['SPA Number']
        self.architectName = pd['Architect Name']
        self.architectPhone = pd['Architect Phone']
        self.architectEmail = pd['Architect Email']
        self.architectTitle = pd['Architect Title']
        self.architectCompany = pd['Architect Company']
        self.modellerName = pd['Energy Modeller Name']
        self.modellerPhone = pd['Energy Modeller Phone']
        self.modellerEmail = pd['Energy Modeller Email']
        self.modellerTitle = pd['Energy Modeller Title']
        self.modellerCompany = pd['Energy Modeller Company']
        self.buildingType = pd['Building Type']
        self.buildingArea = pd['Building Area']
        self.codeCompliancePath = pd['Code Compliance Path']


class Model:
    def __init__(self,filepath):
        self.filepath = filepath
        self.type = None

        if os.path.splitext(filepath)[1].lower() == '.sim':
            self.type = 'eQuest'
        elif ies.isIES(filepath):
            self.type = 'IES'

def readPWB(pwb):

    projects = []

    if not os.path.isfile(PWB):
        print('Project workbook: "' + PWB + '" not found')
        return projects

    wb = openpyxl.load_workbook(PWB)
    ws = wb.active

    pwbProjects = []
    pwbProjectValues = {}


    for i,column in enumerate(ws.columns):
        if i == 0:
            headings = []
            for row in column:
                headings.append(row.value)
        else:
            for i2, row in enumerate(column):
                pwbProjectValues[headings[i2]] = row.value
            pwbProjects.append(pwbProjectValues)
            pwbProjectValues = {}

    for entry in pwbProjects:
        project = Project()
```

```python
        project.addByPWB(entry)
        projects.append(project)

    return projects

def scanPF(pf):
    fileNames = []
    projectNames = {}
    projects = []

    for entry in os.listdir(pf):
        filepath = os.path.join(pf,entry)
        if os.path.isfile(filepath):
            fileNames.append(entry)

    for fileName in fileNames:
        root = os.path.splitext(fileName)[0]
        if root.endswith('_prop'):
            withoutSuffix = root[:-5]
            if withoutSuffix not in projectNames.keys(): projectNames[withoutSuffix] = {}
            projectNames[withoutSuffix]['proposed'] = os.path.join(pf,fileName)
        elif root.endswith('_ref'):
            withoutSuffix = root[:-4]
            if withoutSuffix not in projectNames.keys(): projectNames[withoutSuffix] = {}
            projectNames[withoutSuffix]['reference'] = os.path.join(pf,fileName)
        else:
            withoutSuffix = root
            if withoutSuffix not in projectNames.keys(): projectNames[withoutSuffix] = {}
            projectNames[withoutSuffix]['untagged'] = os.path.join(pf,fileName)

    for project in projectNames.keys():
        if len(projectNames[project].keys()) == 3:
            print('3 files provided for the same project')
        else:
            newProject = Project()
            if 'proposed' in projectNames[project].keys():
                newProject.addName(project)
                newProject.addProposed(projectNames[project]['proposed'])
                if 'reference' in projectNames[project].keys():
                    newProject.addReference(projectNames[project]['reference'])
                elif 'untagged' in projectNames[project].keys():
                    newProject.addReference(projectNames[project]['untagged'])

            elif 'untagged' in projectNames[project].keys():
                newProject.addName(project)
                newProject.addProposed(projectNames[project]['untagged'])
                if 'reference' in projectNames[project].keys():
                    newProject.addReference(projectNames[project]['reference'])
            else:
                newProject.addName(project)
                newProject.addProposed(projectNames[project]['reference'])

            projects.append(newProject)

    return projects

```

```

def joinPWBandPF(pwb,pf):
    return pwb+pf


def harvest():
    runTime = datetime.datetime.now().strftime('%Y%m%d %H%M%S')
    outFolderBase = os.path.join(OF,runTime)

    #open project setup workbook
    # projectsInPWB = readPWB(PWB)
    #scan project in project folder
    projectsInPF = scanPF(PF)

    #join project lists
    # projects = joinPWBandPF(projectsInPWB,projectsInPF)
    projects = projectsInPF

    # return projects

    # pass project details to output modules

    os.mkdir(outFolderBase)
    # projects = projects[9:15]
    # projects = projects[:2]
    for project in projects:
        print('Creating output for:', project.projectName)
        outFolder = os.path.join(outFolderBase,project.projectName)
        os.mkdir(outFolder)
        tgs.tgs(project,os.path.join(outFolder,'tgs.pdf'))
        obc.obc(project,os.path.join(outFolder,'obc.pdf'))
        sbd.sbd(project,os.path.join(outFolder,'sbd.xlsx'))
        leed2009.leed2009(project,os.path.join(outFolder,'leed2009.xlsx'))

    print('Output creation complete')

    return projects


if __name__ == '__main__':
    projects = harvest()
    # projects = scanPF(PF)
    # test = equest.equest(projects[0].proposed.filepath)

```

APPENDIX D:

SAMPLE SUBMISSION REPORTS FROM TEST RUNS

LEED Canada for New Construction and Major Renovations 2009

Project Number:
0

EA Prerequisite 2: MINIMUM ENERGY PERFORMANCE & EA Credit 1: OPTIMIZE ENERGY PERFORMANCE

New Construction

Please select only ONE of the following options:

Whole Building Energy Simulation

□ **OPTION 1. (PATH 1): Model National Energy Code For Buildings 1997 (MNECB)**

□ **OPTION 1. (PATH 2): ASHRAE 90.1-2007, Energy Standard for Buildings Except Low-Rise Residential Buildings**

Prescriptive Compliance Path

□ **OPTION 2. (PATH 1): ASHRAE Advanced Energy Design Guide for Small Office Buildings 2004**
Less than 1,860 square metres (20,000 square feet) with office occupancy.

□ **OPTION 2. (PATH 2): ASHRAE Advanced Energy Design Guide for Small Retail Buildings 2006**
Less than 1,860 square metres (20,000 square feet) with retail occupancy.

□ **OPTION 2. (PATH 3): ASHRAE Advanced Energy Design Guide for Small Warehouses and Self-Storage Buildings 2008**
Less than 4,645 square metres (50,000 square feet) with warehouse or self-storage occupancy.

□ **OPTION 2. (PATH 4): ASHRAE Advanced Energy Design Guide for K-12 School Buildings**
Less than 18,600 square metres (200,000 square feet) with K-12 school occupancy.

□ **OPTION 3: Advanced Buildings™ Core Performance™ Guide**
Less than 9,290 square metres (100,000 square feet).

□ **Special Circumstances or Alternative Compliance Path**

OPTION 1: Whole Building Energy Simulation (1 to 19 Points)

PATH

Complete the following tables to support the selected option:

Table: Energy Cost and Consumption by Energy Type

Energy Summary by End Use		Energy Type	Proposed Building [MJ]	Intensity [kWh/m2]	Reference Building [MJ]	Intensity [kWh/m2]	Energy Savings [%]
Lighting		Electric	678,401	-	944,697	-	28%
Space Heating		Natural gas	7,669,306	-	13,380,218	-	43%
Space Cooling		Electric	299,108	-	527,211	-	43%
Pumps		Electric	171,341	-	497,037	-	66%
Fans		Electric	2,253,599	-	2,714,764	-	17%
Service Water Heating		Natural gas	191,704	-	248,677	-	23%
Plug Loads		Electric	285,182	-	285,182	-	0%
Other:	Enter End Use	Select a fuel	0	-	0	-	0%
Other:	Enter End Use	Select a fuel	0	-	0	-	0%
Other:	Enter End Use	Select a fuel	0	-	0	-	0%
Other:	Enter End Use	Select a fuel	0	-	0	-	0%
Other:	Enter End Use	Select a fuel	0	-	0	-	0%
Subtotal			**11,548,641**	**0.0**	**18,597,786**	**0.0**	**38%**

Total Energy Summary		Proposed Building Energy [MJ]	Cost [$]	Reference Building Energy [MJ]	Cost [$]	Percent Savings Energy [%]	Cost [%]
Electricity		3,687,631	$116,038	4,968,997	$156,352	26%	26%
Natural Gas		7,861,010	$12,473	13,628,895	$21,625	42%	42%
Oil / Other Fuels		0	$0	0	$0	0%	0%
Total		**11,548,641**	**$128,511**	**18,597,891**	**$177,977**	**38%**	**28%**
Subtotal Energy Costs		**11,548,641**	**$128,511**	**(DEC')**	**$177,974**	**(ECB')**	
Renewable	Select a fuel	0	$0	(REC1)	Enter REC System 1		**(REC')**
Energy Credit	Select a fuel	0	$0	(REC2)	Enter REC System 2		**$0**
Net Total		**11,548,641**	**$128,511**				

Percent Savings = 100 x (ECB' $ - DEC' $ + REC' $)/ECB' $ = 27%
Points Awarded 0

Declare that:

□ The project complies with the mandatory provisions of MNECB 1997 / ASHRAE 90.1-2007.

Provide the following to support the selected option:

□ Proof of the installation of an energy meter(s) that measures all energy use, for both building and site energy uses, as well as proof of calibration of any meters owned by the building owner, management organization or tenant.

Provide ONE the following to support the selected option:

□ A compliance report from an acceptable independent third party agency (such as Natural Resources Canada) and a modelling report that details any changes between the compliance report and the LEED Canada Energy Modelling Rules (e.g., changes made for LEED compliance).

OR

□ A compliance report from an acceptable independent third party (individual on CaGBC's Experienced Modellers List), and signed compliance documentation for MNECB 1997 / ASHRAE 90.1-2007 mandatory provisions.

OR

□ A modelling report, copies of computer simulation output files, and signed compliance documentation for MNECB 1997 / ASHRAE 90.1-2007.
Required elements of a compliant modelling report are outlined in the Guidance for Energy Modelling Compliance Documentation in LEED Canada (released 2013).

OPTION 2: Prescriptive Compliance Path (1 Point)

PATH

Provide the following to support the selected option:

□ Proof of the installation of an energy meter(s) that measures all energy use, for both building and site energy uses, as well as proof of calibration of any meters owned by the building owner, management organization or tenant.

□ A compliance report from acceptable independent third party (individual on CaGBC’s Experienced Modellers List).
NOTE: Individuals on CaGBC’s Experienced Modellers List qualify to conduct their own reports and do not require a third party review.

OPTION 3: Prescriptive Compliance Path: Advanced Buildings™ Core Performance™ Guide (1 to 3 Points)

Declare that:

□ Project complies with Section 1: Design Process Strategies.

□ Project complies with Section 2: Core Performance Requirements.

Indicate the qualifying Section 3: Enhanced Performance strategies that were implemented (1 point for every 3 strategies) :

□ Daylighting and controls

□ Additional lighting power reductions

□ Plug loads, appliance efficiency

□ Supply air temperature reset (VAV)

□ Indirect evaporative cooling

□ Heat recovery

□ Premium economizer performance

□ Variable speed control

□ Demand-responsive buildings (peak power reduction)

□ On-site supply of renewable energy

□ Fault detection and diagnostics

Provide the following to support the selected option:

□ Proof of the installation of an energy meter(s) that measures all energy use, for both building and site energy uses, as well as proof of calibration of any meters owned by the building owner, management organization or tenant.

□ A compliance report from acceptable independent third party (individual on CaGBC’s Experienced Modellers List).
NOTE: Individuals on CaGBC’s Experienced Modellers List qualify to conduct their own reports and do not require a third party review.

Special Circumstances or Alternative Compliance Path **Select Option**

Special circumstances preclude documentation of credit compliance with the submittal requirements outlined in this form or the project team is using an alternative compliance path in lieu of standard submittal paths.

Provide the following to support the selected option:

□ A narrative describing the special circumstances or alternative compliance path and any supporting alternate documentation.
(The narrative must include justification that the credit intent and requirements are met and reference the alternate documentation provided.
Non-standard documentation will be considered upon its merits.)

Credit Interpretation Request (CIR) applied to credit:

EA Prerequisite 2: Minimum Energy Performance

	Prerequisite Documented
OPTION 1: Whole Building Energy Simulation:	NO
OPTION 2: Prescriptive Compliance Path:	NO
OPTION 3: Prescriptive Compliance Path: Advanced Buildings™ Core Performance™ Guide	NO
Special Circumstances or Alternative Compliance Path	NO

EA Credit 1: Optimize Energy Performance

	Points Documented
OPTION 1: Whole Building Energy Simulation: (1 to 19 points)	0
OPTION 2: Prescriptive Compliance Path: (1 point)	0
OPTION 3: Prescriptive Compliance Path: Advanced Buildings™ Core Performance™ Guide (1 to 3 points)	0
Special Circumstances or Alternative Compliance Path **select option**	0

The signature below constitutes a declaration that the project meets the credit intent and the requirements of the option selected above and that the submitted documents accurately represent the project.

Name: ____________ 0

Organization: ____________ 0

Role in project: ____________ Mechanical Engineer

Signature: ____________

Date: ____________

OBC SB-10 COMPLIANCE (1) EXCEED MNECB BY NOT LESS THAN o5% (2) EXCEED ASHRAE 90.1-2010 BY NOT LESS THAN MT¢5% FORM A

Please select which of the two options pursued for compliance:

PROPOSED BUILDING IS SHOWN TO CONSUME AT LEAST 35% LESS ENERGY (GJ or kWh) ANNUALLY THAN THE MNECB REFERENCE BUILDING. ENERGY CONSUMPTION VALUES ARE DETERMINED ACCORDING TO THE MODELLING PROCEDURES IDENTIFIED IN PART 8 OF THE MNECB.	□ YES
PROPOSED BUILDING IS SHOWN TO CONSUME AT LEAST 17.5% LESS ENERGY (GJ or kWh) ANNUALLY THAN THE ASHRAE 90.1-2010 REFERENCE BUILDING. ENERGY CONSUMPTION VALUES ARE DETERMINED ACCORDING TO THE MODELLING PROCEDURES OUTLINED IN CHAPTER 11 OF ASHRAE 90.1-2010.	□ YES

Project:

Modeller Name:

Occupancies	**Floor Area**
□ Assembly	
□ Health/Institutional	
□ Hotel/Motel	
□ Light Manufacturing	
□ Multifamily	
□ Office	
□ Restaurant	
□ Retail	
□ School	
□ Warehouse	
□ Other	
Total	

□ **Proposed Building Description**

Annual Energy Summary [1]

Annual Consumption Summary	**Reference Building Energy**	**Proposed Building Energy**	**Units**
Space Heating	3,716,727	2,130,363	kWh
Space Cooling	146,448	83,086	kWh
HVAC Auxiliary	754,101	626,000	kWh
Misc. Electrical	79,217	79,217	kWh
Service Hot Water	69,077	53,251	kWh
Interior Lighting	262,416	188,445	kWh
Other			
Other			
Total Annual Energy	5,166,081 >	3,207,956	kWh

Percentage less energy used by proposed building: 37.9%

Total Annual CO_2e Emissions 138,028 > 102,434

Percentage less CO_2e emissions by proposed building ……………………….

Peak Electric Demand ………………… > ………………………. □ YES or

Building components specified in Sentence 1.1.2.3.(2) of Chapter 1 of Division 3 of SB-10 comply with the prescriptive requirements of ASHRAE 90.1-2010 □ YES

Reference Building Energy and Proposed Building Energy Consumptions are calculated by:

Please specify modelling software: ____________________

HVAC System Descriptions

Reference Building Design

Proposed Building Design

Energy Efficiency Features in Proposed Building Design [2]

The reference building and proposed building design are modelled in accordance with the requirements of the SB-10 and the applicable standard specified above □ Yes

The information submitted above is accurate to the best of my knowledge.

Signature:

Name/Title:

Notes: (1) A full modelling report is required to be submitted.

(2) Explain major energy saving features utilized to achieve modelled savings.

ASHRAE 90.1-2010 & SB-10 ENERGY COST BUDGET (ECB) COMPLIANCE REPORT FORM 11

Project:

Designer Name:

Occupancies	Floor Area
□ Assembly	
□ Health/Institutional	
□ Hotel/Motel	
□ Light Manufacturing	
□ Multifamily	
□ Office	
□ Restaurant	
□ Retail	
□ School	
□ Warehouse	
□ Other	
Total	

□ **Proposed Building Description**

Annual Consumption Summary[1]	Reference Building Energy		Proposed Building Energy	Units
Space Heating	3,716,727		2,130,363	kWh
Space Cooling	146,448		83,086	kWh
HVAC Auxiliary	754,101		626,000	kWh
Misc. Electrical	79,217		79,217	kWh
Service Hot Water	69,077		53,251	kWh
Interior Lighting	262,416		188,445	kWh
Other				
Other				
Total Annual Energy	5,166,081		3,207,956	kWh
Total Annual Energy Cost	$ 177,977	>	$ 128,511	
Total Annual CO_2e Emissions	138,028	>	102,434	kg
Peak Electric Demand		>		□ YES or

Building components specified in Sentence 1.1.2.3.(2) of Chapter 1 of Division 3 of SB-10 comply with the prescriptive requirements of ASHRAE 90.1-2010 □ YES

Reference Building Energy and Proposed Building Energy Consumptions are calculated by:

Please specify modelling software:

HVAC System Descriptions

Reference Building Design

Proposed Building Design

Energy Efficiency Features in Proposed Building Design[2]

Building is in compliance with mandatory requirements of sections 5.4, 6.4, 7.4, 8.4, 9.4, and 10.4 □ YES

Compliance Result

The design detailed in the above referenced plans complies with the mandatory requirements of the ASHRAE 90.1-2010 Standard and the additional requirements of Supplementary Standard SB-10. The calculated proposed building energy cost (design energy cost), CO_2 emissions and peak electric demand do-not exceed the calculated reference building energy cost (energy cost budget) CO_2 emissions and peak electric demand. Therefore, this design **DOES COMPLY** with the ASHRAE 90.1-2010 ECB compliance methodology and the additional requirements of Supplementary Standard SB-10.

The information submitted above is accurate to the best of my knowledge.

Signature:

Name/Title:

Notes: (1) Verify with building official whether full modelling report is required to be submitted.
(2) Explain major energy saving features utilized to achieve modelled savings.

Note: Numbering is based on SI edition of ASHRAE 90.1-2010.

Appendix-A

Better Buildings Partnership - New Construction

Energy Modeling Report Summary

PROJECT INFORMATION

Date (dd/mm/yyyy): 04/07/2017

Project Address:

Building Type:

SPA-Number:

Building Area: 2,894 m²

Energy Modeller Name:

Architect Name:

Energy Modeller Telephone:

Architect Telephone:

Energy Modeller E-Mail:

Architect E-Mail:

Modelling Software Used: eQuest

Code Compliance Path:

	Reference Building					Proposed Building					Energy Savings			
Energy End Use	Electrical Annual Consumption (kWh)	Natural Gas Annual Consumption (kWh)	Energy Use Intensity (kWh/m2.yr)	Peak Demand Summer (kW)	Peak Demand Winter (kW)	Electrical Annual Consumption (kWh)	Natural Gas Annual Consumption (kWh)	Energy Use Intensity (kWh/m2.yr)	Peak Demand Summer (kW)	Peak Demand Winter (kW)	Peak Demand Summer (kW)%	Peak Demand Winter (kW)%	Annual Consumption (kWh)	Energy Efficiency Above Base Case%
Lights	262,416	0	90.69			188,445	0	65.13						
Misc. Equipment	79,217	0	27.38			79,217	0	27.38						
Space Heating	0	3,716,727	1,284.48			0	2,130,363	736.24						
Space Cooling	146,448	0	50.61			83,086	0	28.71						
Pumps	138,066	0	47.71			47,595	0	16.45						
Fans	754,101	0	260.61			626,000	0	216.34						
Service Hot Water	0	69,077	23.87			0	53,251	18.40						
Totals	1,380,277	3,785,804	1,785.37			1,024,342	2,183,614	1,108.65						

I herby certify that the energy demand and consumption are properly representative of the energy modelling report submitted for the above project.

Energy Modeler Name:

Architect Name:

Title:

Title:

Company:

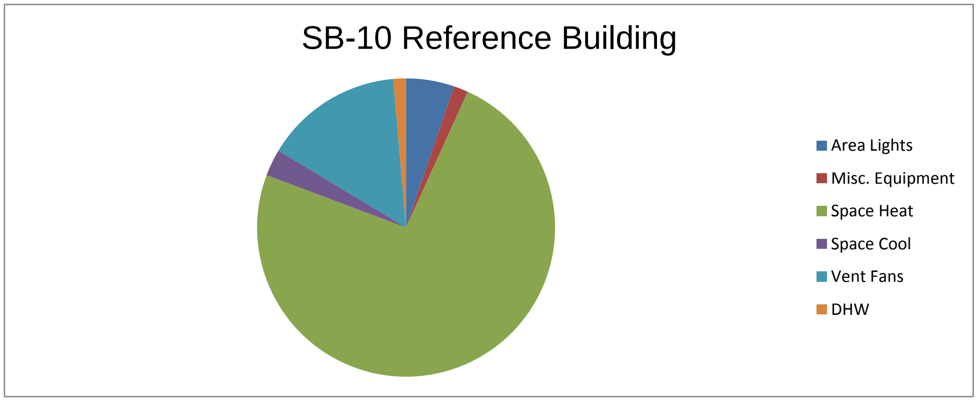
Company:

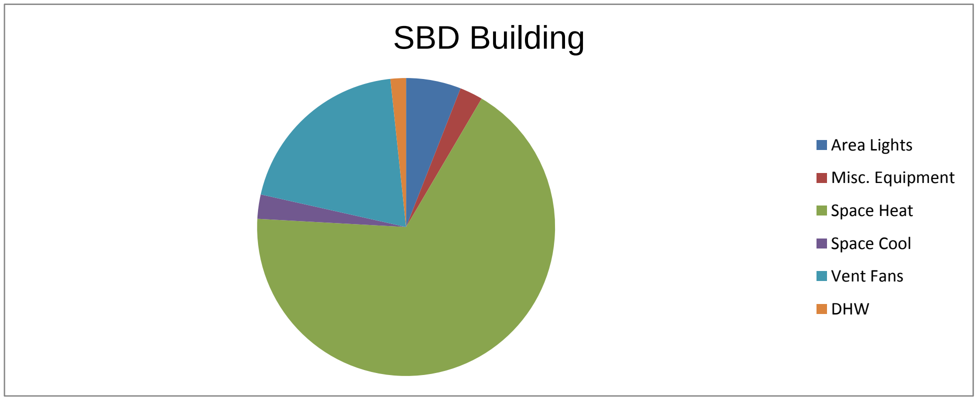
Signature:

Signature:

Energy (ekWh/yr)	Area Lights	Misc. Equipment	Space Heat	Space Cool	Vent Fans	DHW
SB-10 Reference Building	262,416	79,217	3,716,727	146,448	754,101	69,077
SBD Building	188,445	79,217	2,130,363	83,086	626,000	53,251

Energy	Reference Building	Savings By Design Building	Cumulative Reduction
Annual Consumption (ekWh/yr)	5,166,081	3,207,956	37.9%
GHG Emissions (kg CO2eq)	138,028	102,434	25.8%


SB-10 Reference Building
Area Lights
Misc. Equipment
Space Heat
Space Cool
Vent Fans
DHW


SBD Building
Area Lights
Misc. Equipment
Space Heat
Space Cool
Vent Fans
DHW