2018 Building Performance Modeling Conference and
SimBuild co-organized by ASHRAE and IBPSA-USA
Chicago, IL
September 26-28, 2018

READING AND WRITING STANDARDIZED HVAC PERFORMANCE DATA: AN EARLY IMPLEMENTATION OF ASHRAE STANDARD 205P

Neal Kruis, Ph.D., Member[1] and Charles S. Barnaby, BEMP, Life Member[2],
(1) Big Ladder Software, Denver, CO, (2) Retired, Moultonborough, NH

ABSTRACT

ASHRAE Standard 205P recently concluded its first advisory public review. The standard, entitled "Standard Representation of Performance Simulation Data for HVAC&R and Other Facility Equipment", makes the first attempt to standardize the way performance data is conveyed between HVAC equipment manufacturers and energy modelers. The standard defines specific data elements that represent the complete performance of equipment for use in building performance simulation tools.

In parallel with the public review, an initial implementation of the standard was prototyped using Google's FlatBuffer data serialization format. FlatBuffers provides a very computationally and memory efficient format that is also compatible with the more common JSON format. FlatBuffers generates minimal software source code in several languages including C/C++, Python, Java, and JavaScript based on a single schema describing the data elements.

A prototype implementation demonstrates the process of generating an ASHRAE Standard 205P liquid-cooled chiller data file from a manufacturer's spreadsheet using Python, and subsequently reading that same file into a C++ program. This process demonstrates the versatility of the new standardized data format. Any participating manufacturer can use a similar process to provide equipment performance to a user that will represent a specific model in any participating simulation software tool.

ASHRAE STANDARD 205P

ASHRAE Standard 205P is titled "Standard Representation of Performance Simulation Data for HVAC&R and Other Facility Equipment" (ASHRAE, 2017). The stated purpose of ASHRAE Standard 205P is:

> To facilitate sharing of equipment characteristics for performance simulation by defining standard representations such as data models, data formats, and automation interfaces.

The main body of the standard describes the general format of a representation specification (rep spec) for any type of equipment. The majority of the standards normative language resides in separate annexes describing the specific data elements that define the complete performance of a given type of equipment. The current draft standard annexes define three initial rep specs, each given a unique representation specification identifier (RSID) as shown in Table 1

Table 1: Initial Representation Specifications in ASHRAE Standard 205P

RSID	Equipment Type
RS0001	Liquid-Cooled Chillers
RS0002	Unitary Cooling Air-Conditioning Equipment
RS0003	Fan Assemblies

Use cases

ASHRAE Standard 205P is intended to support the following use cases:

- **Data Publication** Data publishers (typically equipment manufacturers) use representation specifications to guide implementation of data writing and validity testing software that produces correctly-formed representation files.
- **Application Development** Application developers use representation specifications to guide implementation of software that correctly reads representation data. Such implementations may include validity tests and developers may use representation specification example data for testing purposes.
- **Data Application** Application users use representation specifications to understand and check representation data. Data exchange will generally be automated but the availability of representation specifications facilitates additional data checking when needed.

The target workflow resulting from compliance with the standard begins with a data publisher (e.g., manufacturer)

populating a data file with values corresponding to the data elements defined by an ASHRAE Standard 205P rep spec for a specific piece of equipment. The data publisher then transmits the corresponding data file to an application user who loads it into a compliant performance simulation software to execute relevant analysis of the equipment in the context of the larger building system(s).

The consensus objective of the standard is to simplify the process as much as possible for application users. The data should require minimal-to-no additional effort on behalf of the application user to correctly represent a manufacturer's equipment in a building simulation.

Related efforts

To date, the only related effort to standardize the exchange of HVAC&R equipment data is the Technology Performance Exchange (TPEx) from the National Renewable Energy Laboratory (NREL) (Studer et al., 2014). TPEx is described as a publicly accessible web-based portal that serves as a centralized repository for users to share and find product-specific energy performance data. Some TPEx data are automatically translated into EnergyPlus (United States Department of Energy (DOE), 2018) input objects and stored in NREL's Building Component Library (NREL, 2018) where they are accessible to users of EnergyPlus-based tools.

ASHRAE Standard 205P differs from TPEx in two notable ways:

1. The standard presumes nothing about where data files are stored or how they are transmitted to the users. Issues of data security and data access may be determined by individual data producers. This does not preclude the possibility of a centralized repository for ASHRAE Standard 205P data files, but this does not fall within the purview of the standard.

2. ASHRAE Standard 205P rep specs are intentionally software agnostic. The Standards Project Committee for 205 (SPC 205) makeup includes a balance of members from all user types (manufacturers, application developers, and application users) according to established ANSI processes. Whereas TPEx is specific to EnergyPlus-based tools, Standard 205P rep specs are not intended to follow the inputs or conventions of any single simulation tool. In fact, as a result of the standards development process, SPC 205 is taking a fresh look at some of the established software models to account for aspects of equipment that are rarely (if ever) handled sufficiently in simulation software tools (e.g., the heat produced by chillers in the mechanical room sometimes needs to be offset by smaller cooling systems).

Representation specification design

Each annex of Standard 205P defines a rep spec for a specific type of equipment. An annex includes a diagram of the equipment and tables (data groups) of equipment attributes (data elements). Each data element that comprises a rep spec defines:

- the data element name,
- a brief description,
- units (for applicable quantities)
- valid ranges,
- minimum precision (in significant digits),
- whether the data element is required, and
- any supplementary notes.

Within a rep spec, data elements may be used to describe:

- Performance characteristics of the equipment (single-entry characteristics that inform aspects of the simulation)
- Supplementary descriptive information about the equipment (not required for simulation)
- Performance values mapped to sets of operating conditions (arrays of values corresponding to different combinations of operating conditions)

Examples of data element names (using the lower camel case convention defined by the standard) included in the RS0001 (Liquid-Cooled Chiller) annex are:

Performance characteristics

evaporatorLiquidType
evaporatorLiquidConcentration condenserLiquidType
condenserLiquidConcentration unitPowerLimit

Supplementary descriptive information

manufacturer modelNumber
nominalVoltage compressorType
refrigerantType

Cooling performance values

inputPower netRefrigeratingCapacity
heatLossFraction
evaporatorLiquidEnteringTemperature
condenserLiquidLeavingTemperature
evaporatorLiquidDifferentialPressure
condenserLiquidDifferentialPressure

where cooling performance values are defined for each operational condition combination of:

evaporatorLiquidVolumeFlowRate
evaporatorLiquidLeavingTemperature
condenserLiquidVolumeFlowRate
condenserLiquidEnteringTemperature netRefrigeratingCapacityFraction

One important conclusion from discussions within the SPC is that data publishers are generally not comfortable with the use of regression coefficients used in many simulation software tools (unless there is a physical basis for the form of the regression equation). The common practice of using polynomial curve coefficients can lead to erroneous results when extrapolating beyond the fitted data. The general approach adopted in Standard 205P is to provide only the raw performance data in the form of a look-up table for the possible range of operation for the equipment. The equipment is assumed to be off (or in a standby mode) for any conditions outside of the data provided.
Data publishers may generate performance data through experiment or from high-fidelity simulation models (as is done with many manufacturer's catalogue data or selection software). As stated in the standard, a sufficient range and number of operational conditions shall be determined by the data producer to capture non-linear performance characteristics (e.g. inflections). This means for equipment with highly non-linear performance, the factorial combination of points within each dimension of the performance map may result in a sizable amount of data.

Data model vs. file format

ASHRAE Standard 205P does not explicitly define a specific file format for exchanging compliant data. Instead, the standard defines a data model that can be characterized in a number of available file formats (as well as future file formats). The standard project committee fully anticipates the industry will informally self-standardize around a specific file format. This allows the file format to evolve over time without requiring changes to the standard.
The data model and the file format are somewhat tied together. Data file formats in the greater software world have more-or-less coalesced around a common data model structure. JSON (JavaScript Object Notation) is a popular example of this coalescence, and describes the structure as follows:

> JSON is built on two structures:
>
> - A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
> - An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.
>
> These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures. (JSON.org, 2018)

Examples of different file format and programming language syntaxes based on these structures are illustrated in Table 2.

Table 2: Examples of the two "universal" data structures

Syntax	Collection of name/value pairs	Ordered list of values
JSON	Object	Array
YAML	Mapping	Sequence
C++	Map	Vector
Python	Dictionary	List
Ruby	Hash Table	Array

In the context of ASHRAE Standard 205P the proposed names for the two fundamental structures are *DataGroups* and *Arrays*, respectively. Using these two concepts one can fully define the data model within an ASHRAE 205P representation specification without locking into a specific file format or syntax.
Notably absent from Table 2 is the XML file format. XML is a markup language that was never intended for strict data exchange. XML was not originally designed to represent ordered lists and does not strictly adhere to the same fundamental data structure as JSON and other data exchange formats. Furthermore, XML tends to be more verbose than other file formats when conveying the same set of data.
So long as the file format uses analogous data structures to those in Table 2 (and nesting thereof), it will be relatively straightforward to transition to other formats as necessary down the road.

PARSER IMPLEMENTATION

Although the standard does not define a specific file format, a file format must be established for any real software applications exchanging Standard 205P compliant data. There are several considerations when selecting a file format for a prototype implementation:

1. The file format must support common data types (e.g., integers, floats, strings, booleans) to define basic data elements.

2. The format must accommodate higher-level structures of name/value pairs and ordered lists (as noted in Table 2). Adhering to this basic data structure will enable simple conversions of the data model between various file formats (including future formats).

3. The file format should have an explicit schema. This allows for the codification of the standard in version-controlled source code. A schema provides a means of consistently validating Standard 205P data across several applications.

4. The file format should be a serialized binary to:

    - minimize size for upload, download, and disk storage considerations, and
    - minimize file read and write time.

5. The file format should be supported in multiple languages (e.g., C, C++, Python, Ruby, Java, C#) to maximize the potential for industry adoption.

The implementation of the parser in any language should be open-source and agnostic towards any simulation software tools. This will minimize duplicate efforts required to read and write Standard 205P data files across different simulation tools and manufacturers.

FlatBuffers

There are now a growing number of serialized binary data formats that could potentially meet the requirements defined in the previous section. Examples include:

- BSON (BSON, 2018)
- Cap'n Proto (Sandstorm Development Group, 2018)
- CBOR (Bormann, 2018)
- FlatBuffers (Google, 2017)
- MessagePack (Furuhashi, 2018)
- Protocol Buffers (Google, 2018)

All of these formats are generally compatible with the data structures defined in Table 2. However, after evaluating each of these formats FlatBuffers appears to have several advantages.

FlatBuffers (Google, 2017) is a computationally fast and memory efficient, cross platform, open-source serialization library. It uses a defined schema to generate minimal source code for several languages including: C/C++, Python, Java, and JavaScript. Most other formats rely on independently developed, generalized parsing libraries.

FlatBuffers was originally developed to improve the performance of client-server communication for performance-critical web applications. FlatBuffers represents hierarchical data in a serialized binary buffer in such a way that it can be accessed directly without decoding or unpacking the file contents. Because the data in the file can be read directly by a FlatBuffer reader, there is no need to copy the data into a native data structure or to deallocate (release from memory) any intermediate forms of the data. This process requires less time and memory to perform than similar processes used to parse other file formats.

There are several advantages to using FlatBuffers:

1. FlatBuffers eliminates some of the more costly I/O processes required for other file formats, making it very fast.

2. The serialized data is similar in size to raw data structures giving it a small memory footprint with zero transient memory allocation.

3. A FlatBuffer file can be directly translated into JSON and vice-versa. FlatBuffers uses *tables* (or *structs*) and *vectors* to represent the data structures defined in Table 2.

4. A FlatBuffer schema can also be directly translated into a JSON file that can be used for automated code and/or documentation generation.

5. FlatBuffers uses a permissive open-source, Apache 2.0 license.

6. Automated source code generation from FlatBuffer schemas allows SPC 205 to concentrate on development of data definitions and schemas without the diversion of creating source code in multiple languages.

7. A FlatBuffers file can be extended beyond the schema using a schema-less extension of the data model called FlexBuffers (Google, 2017). Standard 205P explicitly states that the data can be extended using custom tables.

8. The files generated for language implementations are specific to the data and structure of the schema. The overall dependency is very small relative to that of a more generalized data parser (e.g., for JSON).

A primary component of FlatBuffers is *flatc* – the FlatBuffer schema compiler. From a schema, *flatc* generates minimal source code for a given target language to read and write data consistent with the schema. This means

that, unlike a general purpose interpreter (like most file format parsers), the generated source code only needs to know how to read and write FlatBuffer files defined by the schema that generated it.

Table 3 (interpreted from Google (2017)) illustrates the benefits of FlatBuffers relative to other data parsing approaches. Each approach can be evaluated relative to an ideal case where raw bytes of data are read directly from and written directly to a file. *Note: The alignment and endianness of bytes of raw data is not reliably consistent between machines, making raw data structures an impractical data exchange format.*

Table 3: Serialization format benchmarks implemented in C++ (from Google (2017))

Format	Read [s]	Write [s]	File size [kB]	Temporary memory [kB]	Library size [kB]
Raw data structures	0.02	0.15	0.312	0	0
FlatBuffers (Binary)	0.08	3.2	0.344	0	19
FlatBuffers (JSON)	105	169	1.029	4	47
JSON	583	650	1.475	131	87
XML	196	273	1.137	34	327
Protocol Buffers	302	185	0.228	1	< 3,800

The benchmark results in Table 3 have a limited applicability to Standard 205P. The benchmark test is more representative of web applications where the data files are generally smaller with more frequent exchanges, whereas the anticipated uses of Standard 205P data will likely have relatively few exchanges of larger quantities of data. The read and write speed tests in the benchmark were conducted 1 million times with 312 bytes of data. While this is not a realistic scenario for Standard 205P, FlatBuffers is expected to show similar advantages at other scales as well.

There are other binary serialization formats that are not compared in Table 3. Notably missing are MessagePack and CBOR (both likely similar to Protocol Buffers), and Cap'n Proto (likely similar to FlatBuffers). In fact, Cap'n Proto has a very similar philosophy to FlatBuffers and is another viable file format. A comparison of Cap'n Proto and FlatBuffer capabilities can be found on the Cap'n Proto website (Sandstorm Development Group, 2014).

Considering that memory and disk space are both inexpensive in 2018, and that in most workflows Standard 205P data file will have to be read/written only once, there is perhaps not a strong need for the additional benefits that FlatBuffers would buy over something simpler and/or more common, such as JSON. However, once a parser is developed the construction of the data file will be far enough removed from the user experience that there is almost no drawback from the added complexity of generating a FlatBuffer file. Furthermore, there may be a need for scalability in research and other unforeseen applications of Standard 205P data files, where potentially thousands or even millions of data files are read or generated in a single automated process. It is prudent to use the most efficient technology available.

There are, however, some disadvantages to FlatBuffers:

1. Construction of a FlatBuffer file in source code is somewhat cumbersome as the builder functions generated by the schema cannot easily be automated. This can largely be solved with a small amount of code generation for these kinds of specific applications.

2. FlatBuffers is still a somewhat young technology that relies heavily on contributions from the open-source community for the support of languages beyond C++. There is not always parity in the capabilities among all languages.

ASHRAE 205P FlatBuffer Implementation

Each rep spec annex in ASHRAE Standard 205P has a corresponding FlatBuffer schema file (with the *.fbs extension). For example, the schema file for fan assemblies is RS0003.fbs. Each of these schemas is nested within a top level schema, ASHRAE205.fbs, that is analogous to a base-class for any equipment representation schema. Finally, there are common definitions included in a common.fbs schema file referenced by all representation schemas. An example of a common definition is the enumerated list of valid refrigerant types that could be used in the rep spec for any vapor compression or refrigeration equipment.

Each of the data elements defined in the rep specs are translated into the FlatBuffer schema syntax. Once all components of the schema are defined, *flatc* generates the source code for the necessary languages. In an initial test implementation, only Python and C++ source files were generated.

flatc can also be used to convert ASHRAE 205P FlatBuffer files into their JSON equivalents. In the initial test implementation, it was also used to generate a JSON version of the schema. The JSON schema can be parsed by other tools to help generate code and perform informal validation of data. It is likely that portions of the ASHRAE Standard 205P document will also be generated from the schema to enforce consistency between the implementations and the standard itself.

Basic example end-to-end implementation

A basic example implementation was established to prove the concept of exchanging ASHRAE Standard 205P compliant data written using Python, and read into a compiled C++ program.

A Python script, representing the data publisher, first reads ASHRAE 205P compliant data from a spreadsheet

provided by a chiller manufacturer. The script then builds up the *RS0001* FlatBuffer in memory within Python using the "builder" functions generated by *flatc* using the established schema. Python then writes the FlatBuffer to a file, in this case called chiller.a205 (*.a205 being the Flat- Buffer file extension specific to FlatBuffers conforming to the ASHRAE205.fbs schema).

The chiller.a205 file can hypothetically be emailed to a customer, uploaded to an FTP site, or stored in a database. A hypothetical energy modeler can then load chiller.a205 into a C++ building performance simula- tion tool compiled with the C++ loading functions generated by *flatc* using the ASHRAE205.fbs schema. In this example, the C++ tool simply loads the FlatBuffer into memory and prints out some high-level information about the chiller on the console:

```
EquipmentType: Liquid-CooledChiller RepresentationSpecification
Version: 0.1.0 EquipmentDescription: ExampleChillerfor
                                    ASHRAEStandard205P
Manufacturer: Acme CompressorType:
centrifugal COP: 6.30
IPLV: 9.1
```

This result alone is not necessarily impressive, but all of the data required to simulate the chiller was successfully exchanged between two dissimilar software environments. The data is loaded directly into memory without having to decode, traverse, or copy the file's contents.

NEXT STEPS

As the SPC continues through a publication public review and an eventual release of the final publication, there are several important next steps to support and promote the eventual adoption of ASHRAE Standard 205.

Manufacturer and developer engagement

The exercise of putting together even the simple example presented in this paper required active participation from both a manufacturer and a software developer. This experience provides invaluable insight into an important use case for the standard. It also helps inform the design of the data parser and any required utility tools to help facilitate data exchange.

There is still a need to test data for unitary equipment and fans, however there has been less engagement with manufacturers representing these equipment types. It is possible to generate synthetic data, but that would not demonstrate the process and cooperation between manufacturers and developers that is essential for the success of ASHRAE Standard 205. An important next step is and end-to-end testing where a manufacturer generates an ASHRAE 205P compliant FlatBuffer file and it is loaded into a software program where the represented equipment is simulated.

Beyond, the three established annexes, there is growing interest from both manufacturers, software developers, and energy models to define rep specs for more equipment types. Specifically there are already discussions related to the development of rep specs for:

- Variable refrigerant flow systems
- Air source heat pumps
- Air-cooled chillers
- Cooling towers
- Pumps
- Water heaters
- Fenestration systems

ASHRAE Standard 205P is a developed by volunteers in a committee process. SPC meetings are open to the public for those interested in participating.

ASHRAE Standard 205P toolkit

There is a need for a general toolkit supporting ASHRAE 205P adoption. Some potential capabilities of the toolkit include:

1. *Tool-independent data verification and visualization*

   This would allow users of the data to verify correctness and run standard validity checks. As manufacturers start producing Standard 205P data, they will want an independent verification that the data they produce is valid. An open-source toolkit maintained by the SPC can serve this function.

2. *Conversion between formats (e.g., *.a205, *.json, *.xlsx, *.csv)*

   Many initial applications will begin prototyping data in more common formats. Microsoft Excel is still the tool of choice for many manufacturers and simulation software users. Creating a tool that can translate between ASHRAE 205P FlatBuffers (*.a205) and Excel spreadsheets will greatly reduce the barrier to engage with and understand ASHRAE 205P data. Allowing manufacturers and users to prototype 205P data in a spreadsheet without having to write software will go a long way towards adoption.

   Part of this effort will require some level of standardization around valid *.xlsx and *.csv formats since they are not natively supported by FlatBuffers. This will be the fastest way to initiate engagement with the manufacturers, while they investigate writing to a FlatBuffer directly from their product specification software.

3. *Generation of curve coefficients from ASHRAE 205P data into other existing models*

   This would allow some level of utilization of ASHRAE 205P data before the full integration of the parser in simulation tools as well as testing/comparison of the eventual ASHRAE 205P performance model to the existing performance models in simulation tools.

4. *N-dimensional interpolation utilities*

   A core concept within the current draft of ASHRAE Standard 205P is that performance is characterized as N-dimensional performance maps. The performance of equipment during simulation is determined using interpolation. Interpolation in higher order dimensions can become computationally expensive, and the routines to perform such operations are not readily available in all common programming languages. Lightweight interpolation libraries in various languages can be add to the ASHRAE 205P toolkit to help facilitate the simulation of equipment.

The toolkit can leverage some of the work already completed for the ASHRAE 205P data parser example written in Python. A lightweight, command-line Python application utilizing Pandas (for data management) (AQR Capital Management LLC et al., 2012), SciPy (for interpolation) (SciPy Developers, 2018), and Matplotlib (for plotting) (Hunter, 2007) can be developed very efficiently. If needed, this toolkit can eventually be wrapped in a basic GUI for broader user/manufacturer support.

Much of the capability within the toolkit can be incorporated into C++ routines that can be reused in the various C++ simulation tools (e.g., EnergyPlus, IES-VE, and CSE), or with bindings to other scripting languages.

This toolkit will help facilitate industry adoption of ASHRAE Standard 205P by equipment manufacturers and software vendors alike. Even in the development of the standard, it is very difficult to discuss topics without a common way of visualizing the data. The need for such a toolkit was recognized early in the development of the standard.

CONCLUSIONS

This paper provides an overview of the upcoming ASHRAE Standard 205, and describes an initial implementation of exchanging equipment performance data using FlatBuffers. FlatBuffers is currently a very promising technology and file format for exchanging ASHRAE Standard 205P compliant data.

The standard development process is methodical and sometimes slow. The standard may change between the time this paper is published and when the standard is published. The end result will be the first time the problem of standardized performance data exchange has been solved with participation from all of the stakeholders (manufacturers, software developers, and energy modelers). The outcome from ASHRAE Standard 205P will be more accurate models, more consistent data, and more productive workflows in the context of equipment performance simulation.

REFERENCES

AQR Capital Management LLC, Lambda Foundry Inc., and PyData Development Team (2012). Pandas.

ASHRAE (2017). BSR/ASHRAE Standard 205P: Public Review Draft - Standard Representation of Performance Simulation Data for HVAC&R and Other Facility Equipment.

Bormann, C. (2018). CBOR.

BSON (2018). BSON.

Furuhashi, S. (2018). MessagePack.

Google (2017). FlatBuffers.

Google (2018). Protocol Buffers.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing In Science & Engineering 9*(3), 90–95.

JSON.org (2018). Introducing JSON.

NREL (2018). Building Component Library.

Sandstorm Development Group (2014). Cap'n Proto, FlatBuffers, and SBE.

Sandstorm Development Group (2018). Cap'n Proto.

SciPy Developers (2018). SciPy.

Studer, D., K. Fleming, E. Lee, and W. Livingood (2014). Enabling Detailed Energy Analyses via the Technology Performance Exchange. Technical Report August, NREL, Golden, Colorado.

United States Department of Energy (DOE) (2018). *EnergyPlus: Input Output Reference* (Version 8. ed.). Berkeley, California: The Board of Trustees of the University of Illinois and the Regents of the University of California through the Ernest Orlando Lawrence Berkeley National Laboratory.