Development and Application of CONTAM APIs

W. Stuart Dols[1], Jialei Shen[2], Brian J. Polidoro[1], David M. Lorenzetti[3], Russell Mills[4], T. S. Cole[5], Michael D. Sohn[3]

1. National Institute of Standards and Technology, Gaithersburg, MD, USA
2. University of Alabama at Birmingham, Birmingham, AL, USA
3. Lawrence Berkeley National Laboratory, Berkeley, CA, USA
4. Riskaware, Bristol, UK
5. Parsons Corporation, Engineered Systems, Chantilly, VA, USA

Published in Building Simulation:
Dols, W. Stuart, Jialei Shen, Brian J. Polidoro, David M. Lorenzetti, Russell Mills, William T. S. Cole, and Michael D. Sohn. 2026. "Development and Application of CONTAM APIs." *Building Simulation* 19(1):257–69. https://doi.org/10.1007/s12273-025-1376-x.

Abstract

A set of application programming interfaces (APIs) has been developed for the whole-building contaminant transport and airflow modeling software program, CONTAM. One API allows for the creation and modification of CONTAM building models to be incorporated into modelling workflows, and another allows other simulation tools and programming environments to dynamically link to and run CONTAM building models, interactively control their execution, and query for simulation results. These new APIs simplify the incorporation of CONTAM within building design and analysis frameworks and co-simulation workflows, and open possibilities for extending the current capabilities of CONTAM. We present an overview and motivation of these APIs and examples of their use, including development of a custom control algorithm, an enhanced sorption model, and integration with a three-dimensional architectural design platform and urban scale analysis tools.

Keywords

Application Programming Interface (API), CONTAM, Co-simulation, Multizone modeling

1 Introduction

CONTAM is a whole-building, multizone airflow and contaminant analysis software tool, which has been under continuous development at the National Institute of Standards and Technology (NIST) since the 1980s (Axley 1987, Walton 1989, Dols et al. 2020). CONTAM enables users to address a broad range of simulation needs, including ventilation analysis, building pressurization, multizone transport of airborne contaminants, smoke distribution associated with building fires, and the design of smoke management systems.

To date, CONTAM has collectively consisted of two programs: the *ContamW* graphical user interface and the *ContamX* simulation engine, both of which are stand-alone executable programs. *ContamW* lets users define a building model, then calls *ContamX* to perform calculations using that model. In addition, *ContamW* saves the building model in a CONTAM project (PRJ) file, which can later be run by *ContamX*. This ability to run a building model independently of its graphical user interface makes CONTAM attractive for incorporating into other building modeling frameworks, performing co-simulation with other analysis tools, e.g., whole-building energy analysis software, and even extending CONTAM's capabilities without direct modification of the source code which can lead to diverging and unsupported code changes. Historically, the ability to integrate CONTAM into other workflows has been difficult in terms of creating/modifying building models and performing co-simulation with other building analysis tools. The CONTAM application programming interfaces (APIs) have been developed to overcome these limitations.

Prior to the development of the CONTAM APIs, the only user-friendly method for creating a CONTAM project was using *ContamW*, which allows users to draw two-dimensional, rectilinear representations of whole-building floor plans. The reliance on *ContamW* restricts the broader application of CONTAM, particularly in integrating with three-dimensional geometry modeling platforms like Rhino/Grasshopper (Robert McNeel & Associates 2024), automating batch processing programs, or performing urban-scale modeling with multiple building representations. Some users developed tools that directly modified specific fields of existing PRJ files, e.g., zone temperature (Herring et al. 2014), which requires detailed knowledge of the PRJ, interdependence of related fields, and is prone to errors due to bypassing the rules of model formation imposed by *ContamW*.

Similarly, prior to the development of the APIs, *ContamX* only provided for direct, co-simulation via a socket-based messaging system, and other integration methods relied on sequential execution of programs and file exchange upon completion of simulations (Bonan et al. 2014, Herring et al. 2014). For example, co-simulation could be useful to provide airflows within a whole-building energy analysis tool such as EnergyPlus or TRNSYS (Dols et al. 2016, Dols et al. 2016) or to estimate the uptake of urban plumes within an atmospheric transport and dispersion tool such as the Hazard Prediction Assessment Capability (HPAC) software (Herring et al. 2014). The application of co-simulation of CONTAM with EnergyPlus and TRNSYS have been applied by users to address a broad range of building design and analysis issues that benefit from the combined energy, airflow, and IAQ approach including the influence of ventilation on IAQ in various building types (Clark et al. 2019, Picard et al. 2022, Tognon et al. 2023, Chang et al. 2024, Walker et al. 2024, Wang et al. 2025).

To date, co-simulation has been cumbersome due to the level of communication needed to effectively couple the models and the detailed nature of the socket-based messaging system, e.g., the specifics of the character-based data exchange. Organizations of some users also raised concerns with the socket-based messaging due to potential security issues associated with network communications. Applications that demonstrate the socket-based approach include those implemented by the developers of CONTAM themselves, namely, ContamFMU and Type 98, which provide for co-simulation of CONTAM with EnergyPlus and TRNSYS, respectively (Dols et al. 2015, Dols et al. 2016), and an earlier version of an urban-scale simulation tool (Walter 2004).

2 CONTAM APIs

Developing the CONTAM APIs required modifications to both *ContamW* and *ContamX*. The key updates were: (1) generating dynamic link libraries (DLLs) (shared objects for Linux) to encapsulate the core functionality; (2) creating functions to *interface* with those libraries, i.e., the APIs; and (3) making all CONTAM code thread-safe to allow co-simulation environments to define, run, and control multiple building models at once.

The APIs are bundled into two libraries: *contamp-lib* for working with CONTAM projects, and *contamx-lib* for performing simulations with the projects. The libraries were developed to enable cross-platform application, e.g., on Windows and Linux operating systems. This section presents an overview of the two APIs. The APIs are written in the C programming language, and various bindings have been created for them, i.e., Python, C#, and Java; some of which will be presented in the

Example Applications section.

CONTAM and related source code are maintained within a private GitLab repository, which incorporates continuous integration and continuous deployment (CI/CD) capabilities. In the case of the APIs, CI/CD pipelines include the ability to automate building, testing, and packaging *contamx-lib* and *contamp-lib* for Windows and Linux operating systems. After successful builds of *ContamX* and the corresponding *contamx-lib*, the CI/CD pipeline runs them through an automated set of regression tests designed to cover a broad range of simulation capabilities, including analytical airflow and contaminant transport cases, where applicable, including some of the test cases presented in (Walton 1989). There are also many examples of empirical validation of CONTAM available within the literature, including (Emmerich 2001, Musser et al. 2001, Dols et al. 2011, Rim et al. 2013, Poppendieck et al. 2016, Kashtan et al. 2024).

2.1 contamp-lib

contamp-lib implements functions to query, create, and modify CONTAM building models, i.e., PRJ files. These functions are categorized and summarized in Table 1. The *CommonState* is used to manage high-level information, including error handling. Once a *CommonState* is obtained, then an associated *ContamProjectState* can be obtained. Functions associated with the C*ontamProjectState* can then be used to create, open, and save PRJ files and populate project-related properties, including project elements (e.g., airflow, source/sink, and schedules), building components (e.g., zones and airflow paths), and project settings (e.g., simulation parameters and file paths of associated boundary condition files, including weather and contaminant files). Similarly, although not presented in Table 1, a C*ontamLibraryState* and associated API functions enable the creation, modification, and usage of CONTAM elements that are shareable via CONTAM library files, such as airflow elements, source/sink elements, and schedules. While Table 1 does not provide the complete set of functions, it presents an overview of the categories of functions, the main capabilities provided by the *contamp-lib* API, and the parameters required when working with the functions.

Table 1 – Summary of functions provided by the *contamp-lib* API

Category	contamp-lib API Functions
Common State	`void* cciGetNewCommonState()`
	`int cciGetLastError(void* cs, char* errorMessage)`
	`int cciPrintToLogFile(void* cs)`
	`int cciDeleteCommonState(void* cs)`
Project State	`void* cpiGetNewProjectState(void* cs)`
	`int cpiOpenProject(void* ps, char* filePath)`
	`int cpiSaveProjectAs(void* ps, char* filePath)`
	`int cpiDoBuildingCheck(void* ps)`
	`int cpiDeleteProjectState(void* ps)`
Project Settings	**`Set and Get functions for Integer, Floating point, and String setting types`**
	`int cpiGetIntegerSetting(void* cs, char* settingId, int* pValue)`
	`int cpiSetIntegerSetting(void* cs, char* settingId, int value)`
Project Elements	**`Element Types: Airflow, Duct, Contaminant, Source/Sink, Filter, Kinetic reaction, Wind pressure coefficient profile, Schedule`**
	`int cpiGetNumberOfElements(void* cs, char* eType)`
	`int cpiAddProjectElement(void* cs, char* eType, char* JSONbuffer)`
	`int cpiGetProjectElementByName(void* cs, char* eType, char* eName, char* JSONbuffer)`
	`int cpiDeleteProjectElement(void* cs, char* eType, char* eName)`
Project Components	**`Building Levels`**
	`int cpiGetNumberOfLevels(void* ps)`
	`int cpiAddLevel(void* ps, char* JSONbuffer, int lNumber)`
	`int cpiGetLevelByName(void* ps, char* lName, char* JSONbuffer)`
	`int cpiDeleteLevel(void* ps, char* lName)`
	`Icon/Building Component Types: Zone, Path, AHS, Duct, Source/Sink, Filter, Occupant, Control, Annotation`
	`int cpiGetNumberOfIcons(void* ps, char* iType)`
	`int cpiAddIcon(void* ps, char* iType, char* JSONbuffer, int* ptrIconNumber)`
	`int cpiGetIcon(void* ps, char* iType, int iconNumber, char* JSONbuffer)`
	`int cpiDeleteIcon(void* ps, char* iType, int iconNumber, char* JSONbuffer)`

C# Bindings

An object-oriented, C# .NET wrapper, referred to as *ContamP.net*, was developed for *contamp-lib*. *ContamP.net* provides the ability to develop CONTAM building models within .NET modeling frameworks like Rhino-Grasshopper (Robert McNeel & Associates 2024). Using these bindings enables CONTAM workflows that are not reliant on *ContamW*, as presented in the

Example Applications section.

2.2 contamx-lib

This library implements functions to run simulations for existing PRJ files and provides similar functionality to the socket-based interface implemented in previous versions of *ContamX* (Dols et al. 2015, Dols et al. 2016). The library is thread-safe, so users can simulate multiple PRJ files at once with a single instance of the library. The API allows for initializing a simulation, obtaining information about a state associated with a PRJ, advancing the simulation by executing consecutive time steps, and setting and getting co-simulation data exchange values at each time step to enable monitoring simulation results, e.g., zone mass fractions, and effecting changes to building components, e.g., air-handling system flows, during simulations. A summary of the functions provided by the *contamx-lib* API is presented in Table 2. C language header files, example driver programs, and associated PRJ files that can be used by developers to generate detailed documentation and demonstration cases will be made available. The

Example Applications section presents workflows that implement *contamx-lib*.

Table 2 – Summary of functions provided by the *contamx-lib* API

Category	contamx-lib API Functions
State and Co-simulation Setup	`void* cxiGetContamState()`
	`int cxiRegisterCallback_PrjDataReady(void* cxs, void(*prjDataReadyFcn(...))`
	`int cxiRegisterCallback_Error(void* cxs, void(*errorAuxFcn(...))`
	`int cxiSetCosimOption(void* cxs, eCosimOption option, int value)`
	`int cxiSetupSimulation(void* cxs, char* prjFilePath)`
Project Information	**`Date and Time`**
	`int cxiGetSimulation[Start|End][Date|Time](void* cxs)`
	`int cxiGetSimulationTimeStep(void* cxs)`
	`Contaminants, Nodes (Zones, Junctions), Links (Airflow paths, Duct terminals and leaks), Simple AHS, Controls`
	`int cxiGetNumZones(void* cxs)`
	`int cxiGetZoneInfo(void* cxs, int zoneNumber, ZONE_COSIM_DSC *pZoneInfo)`
Simulation Status	`int cxiGetCurrent[Date|Time](void* cxs)`
	`int cxiDoCosimStep(void* cxs)`
	`int cxiEndSimulation(void* cxs)`
Set Data Exchange Values	**`Ambient conditions: Temperature, Wind speed and direction, Mass fraction`**
	`int cxiSetAmbt[Temp|WndSpd|WndDir|MF](void* cxs)`
	`Building components: Zones, Junctions, AHS, Controls`
	`int cxiSet[Zone,Junction]Temperature(void* cxs, int zoneNumber, float value)`
	`int cxiSetSupplyReturnPathFlow(void* cxs, int pathNumber, float value)`
	`int cxiSetAHSPercentOA(void* cxs, int ahsNumber, float value)`
	`int cxiSetInputControlValue(void* cxs, int controlNumber, float value)`
Get Data Exchange Values	`int cxiGetZoneMassFraction(void* cxs, int zoneNumber, int ctmNumber, float* pValue)`
	`int cxiGet[Path,Term,Leak]Flow(void* cxs, int pathNumber, float* pValue)`
	`int cxiGetOutputControlValue(void* cxs, int controlNumber, float* pValue)`

Python Bindings

A Python module, referred to as *contamxpy*, has been developed, and an initial release associated with CONTAM version 3.4.1 (which is compatible with PRJ files created with CONTAM version 3.4 that is available on the NIST Multizone Modeling website) is available on the PyPI download site (https://pypi.org/project/contamxpy/). A schematic of *contamxpy* is provided in Figure 1. This schematic provides a subset of the complete module, which includes full documentation and example applications.

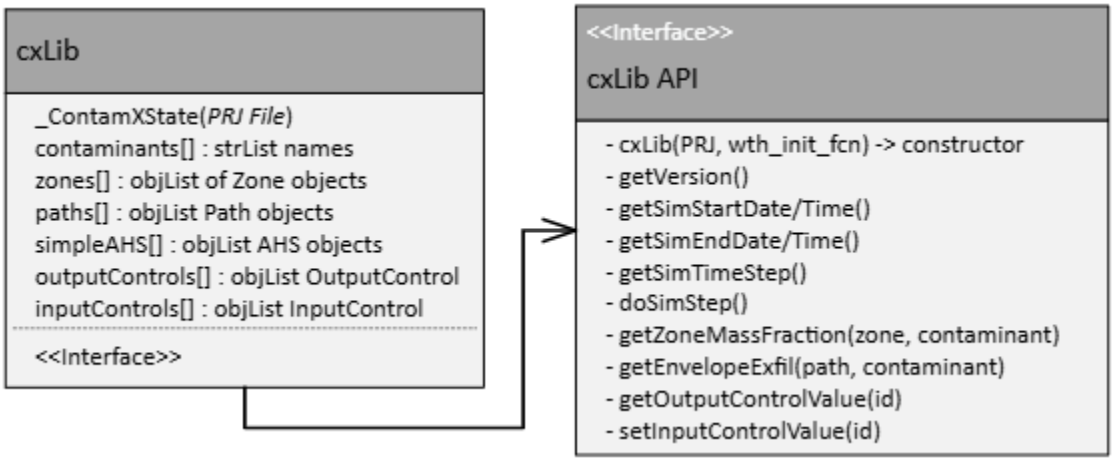

Figure 1 – Schematic of the *cxLib* wrapper class and interface that provides Python bindings to *contamx-lib* (only a subset of the API functions is shown)

JAVA Bindings

Java bindings to *contamx-lib* were implemented using Java Native Access (JNA). The Java bindings are implemented within a single module, *cxLibInterface.java*, providing a wrapper class that contains the JNA-based definitions of the *contamx-lib* API functions. Driver programs can use the *JNA.load* function to associate the native *contamx-lib* DLL (or shared object in Linux) with an instance of the wrapper class.

3 Example Applications

Below are several example applications of the APIs. The first two examples utilize *contamxpy* to extend CONTAM's controls simulation capabilities and pollutant transport modeling, respectively. The third example incorporates *contamp-lib* and *ContamX* into the Rhino-Grasshopper 3D modeling platform. The fourth example presents the use of C# and Java bindings to *contamx-lib* to incorporate CONTAM building modeling into urban-scale transport and dispersion simulation platforms.

3.1 Custom Control Algorithm

This example shows the use of Python bindings to *contamx-lib* to implement a CO_2-based demand-controlled ventilation algorithm. Three control algorithms were compared: simple, basic proportional-integral (PI), and PI with a deadband and anti-integral windup as presented in Pistochini et al. (2023). The first two algorithms utilized pre-programmed CONTAM control elements, but the Pistochini algorithm requires logic that cannot be implemented with existing CONTAM control elements, i.e., complex if-then-else logic and storage of values from previous simulation time steps. Therefore, *contamxpy* was used along with a Python module that implemented the control algorithm, and a driver program to run the co-simulation.

The single-zone case was based on that presented in Pistochini et al. (2023). A simulation period of six hours was run with intermittent occupancy, whereby the CO_2 generation rate alternated between zero and 0.00925 L/s, as shown in Figure 2, which also presents the CO_2 concentration and ventilation rates associated with the three control algorithms. The *Simple* algorithm adjusted the ventilation rate, $\dot{V}_{oa}$, to maintain the indoor concentration at or below a setpoint of 1000 ppmv (1 ppmv = 1 μL/L), the *PI* algorithm adjusted $\dot{V}_{oa}$ according to equation (1) where: $e(t)$ is the difference between the CO_2 setpoint and the concentration at the current simulation time, t; K_p is the proportionality constant (0.076 L/s·ppmv); and τ_I is the integration time constant (4800 s).

$$\dot{V}_{oa} = K_p \left[e(t) + \frac{1}{\tau_I} \int_0^t e(t)\mathrm{dt} \right] \tag{1}$$

The *Pisto* algorithm utilized the same proportional and integration constants as the *PI* algorithm, with the addition of a 25 ppmv deadband and anti-integral windup to reduce overshooting the setpoint. Simulation results match those of the measured values presented in Figure 6, C-PI of (Pistochini et al. 2023).

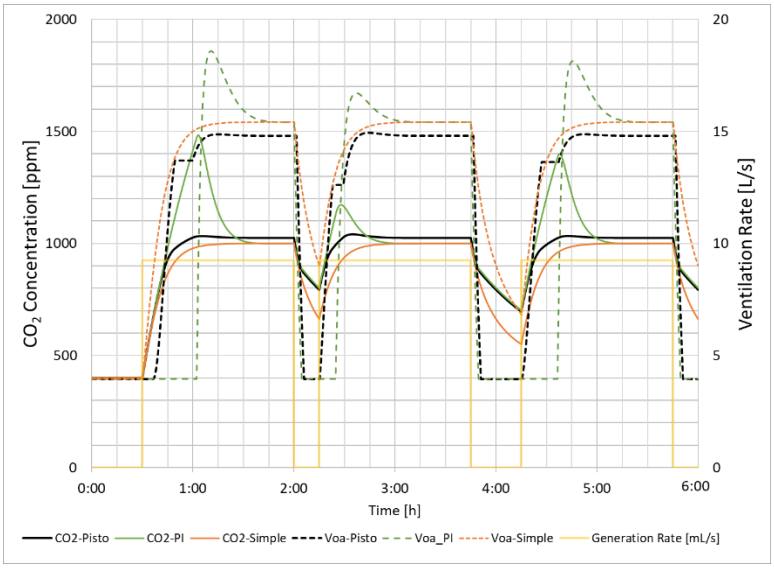

Figure 2 – Transient CO_2 concentrations of simulations comparing three control algorithms

These simulations allow for comparisons between the control algorithms to demonstrate their effectiveness in controlling ventilation rates based on CO_2 concentrations. The box and whisker plots presented in Figure 3 show the effectiveness of these three control algorithms with respect to the mean ventilation rate (an indicator of energy usage), and the range of CO_2 concentrations within the zone. Note that CO_2 concentration does not provide a direct indicator of indoor air quality (IAQ); see Persily et al. (2022) for more information.

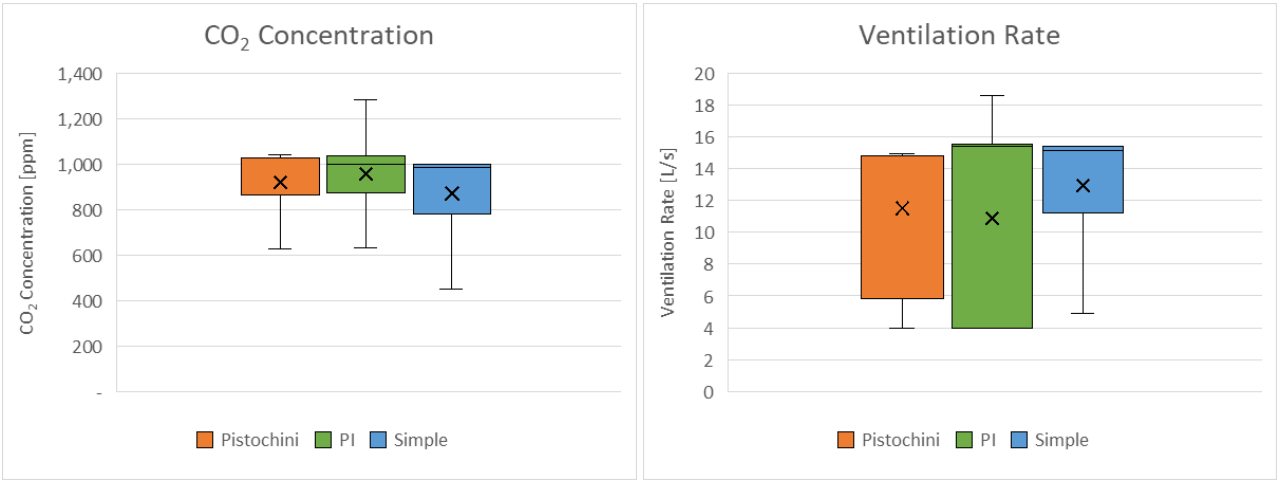

Figure 3 – Box and whisker plots of CO_2 concentration (left) and ventilation rate (right) results for the three control algorithms

3.2 Enhanced IAQ Transport Model

The emission and removal of air pollutants are essential for IAQ analysis. CONTAM simulates these processes using predefined source/sink models, including those with constant rates, constant deposition rates/velocities, pressure-driven rates, cutoff concentrations, exponentially decaying rates, burst mass, and boundary layer diffusion-controlled rates. These models cover most scenarios for air pollutant emissions and decay. However, if a source or sink is affected by varying temperature, humidity, airflow, or other factors, or determined by any physical or chemical processes not predefined in CONTAM, then users cannot directly model these processes. Although additional source/sink models can be added by developers, it is impractical to include every potential model, and users cannot define and test their own mathematical models for describing the emission/removal phenomenon directly within CONTAM.

contamx-lib now enables users to develop customized source/sink models for IAQ analysis by providing access to zone air pollutant concentrations and allowing control of source/sink rates of existing CONTAM models during co-simulation based on user-defined algorithms. This example shows the use of *contamx-lib* to implement a source/sink model that varies depending on environmental conditions or physical properties of building material and species of pollutant at any given time step during co-simulation.

Most building materials, including gypsum, carpet, and furnishings, are porous and have complex emission and adsorption properties, and some materials capable of depositing, adsorbing, and/or accumulating pollutants can influence IAQ during the entire service life of a building. Therefore, the in-material diffusion and storage, and the mass transfer within the boundary layer on the gas and material interface play a dominant role in the determination of pollutant emission, adsorption, desorption, and accumulation for these materials (Xu et al. 2009). Similar to water vapor, air pollutants can also be transmitted between zones through diffusion in the porous media of wall materials. This cross-wall transport can be simulated using hygrothermal software tools such as DELPHIN and CHAMPS-BES (Nicolai et al. 2007), but CONTAM does not directly enable cross-wall transport between zones, only diffusion in materials located within a given zone. Diffusion in porous media enables walls to act as both sinks and sources of pollutants for indoor air, thereby influencing interzonal air pollutant transfer. Many boundary layer transport and in-material diffusion models have been developed, and *contamx-lib* allows users to apply and evaluate these models (e.g., via user-defined algorithms in Python) within the whole-building multizone airflow and pollutant transport model provided by CONTAM.

This example simulates a dual-chamber system to study the transfer and diffusion of formaldehyde (HCHO) from one environmental chamber to another through porous wall material (Xu et al. 2009). As shown in Figure 4(a), two stainless steel chambers of dimensions 0.35 × 0.35 × 0.15 m are partitioned by a test porous wall panel (with a thickness of 0.01 m). Chamber A is supplied with an airflow rate of $Q_A = 6.58 \times 10^{-2}$ m^3/h at a formaldehyde concentration of $C_{A,in} = 372$ μg/m^3. Chamber B is supplied by clean air ($C_{B,in} = 0$) with the same airflow rate ($Q_B = 6.58 \times 10^{-2}$ m^3/h). As the porous material is exposed to the higher concentration in Chamber A and the lower concentration in Chamber B, it absorbs gaseous HCHO from the air in Chamber A. Driven by the concentration

gradient, HCHO diffuses through the material according to Fick's law and is subsequently released into Chamber B. Thus, the material acts as a sink for formaldehyde in Chamber A and a source in Chamber B. The HCHOtransport process can be divided into three stages: convection mass transfer in air, mass transfer at the material-air interface, and in-material diffusion. It is assumed that the air in both chambers is well mixed. The simulation is conducted using *contamxpy*. The governing equation for pollutant (HCHO) concentration in the air of Chambers A and B is:

$$V\frac{dC}{dt} = Q(C_{in} - C) + S \tag{2}$$

where:

- V = chamber volume (V_A for Chamber A, V_B for Chamber B), m^3
- C = concentration of pollutant in the air (C_A for Chamber A, C_B for Chamber B), $\mu g/m^3$
- C_{in} = concentration of pollutant at chamber inlet ($C_{A,in}$ for Chamber A, $C_{B,in}$ for Chamber B), $\mu g/m^3$
- Q = chamber airflow rate (Q_A for Chamber A, Q_B for Chamber B), m^3/h
- S = pollutant source/sink term (S_A for Chamber A, S_B for Chamber B), $\mu g/h$

The pollutant removal/generation rate by the porous material is determined by convective mass transfer between the indoor air concentration (C) and the local gas-phase air pollutant concentration at the material surface ($C_{a,s}$):

$$A_m\frac{dM_s}{dt} = S = A_m h_m\left(C - C_{a,s}\right) \tag{3}$$

$C_{a,s}$ depends on the material surface concentration ($C_{m,s}$) and the partition coefficient (K_{ma}), and it can be estimated by equation 4. In this paper, the local gas-phase pollutant concentration at the material surface in Chamber A is denoted as $C_{A,s}$, while that at the material surface in Chamber B is denoted as $C_{B,s}$.

$$C_{a,s} = \frac{C_{m,s}}{K_{ma}} \tag{4}$$

The in-material diffusion process follows Fick's Law:

$$\frac{\partial C_m}{\partial t} = D_m\frac{\partial^2 C_m}{\partial x^2} \tag{5}$$

With the following boundary conditions:

- At material-air interface on the Chamber A side ($x = 0$): $D_m \cdot \partial C_m/\partial x|_{x=0} = S_A$
- At material-air interface on the Chamber B side ($x = L$): $D_m \cdot \partial C_m/\partial x|_{x=L} = S_B$
- Initial condition in material: $C_m = 0$

where:

M_s = adsorbed mass concentration per unit surface area, µg/m²

A_m = exposed surface area of the porous material, m²

h_m = convective mass transfer coefficient on material surface, m/s

K_{ma} = partition coefficient, dimensionless

$C_{a,s}$ = local gas phase concentration on the material surface, µg/m³

$C_{m,s}$ = concentration on the material surface ($C_m|_{x=0}$ or $C_m|_{x=L}$), µg/m³

C_m = concentration inside the material, µg/m³

D_m = diffusion coefficient within the material, m²/s

Figure 4(a) illustrates the physical process of pollutant transfer described above, along with the expected steady-state concentration profile. The same settings as in (Xu et al. 2009) are adopted for the simulation in this paper (temperature = 23°C, RH = 25 %, K_{ma} = 2574, $D_m = 1.20 \times 10^{-9}$ m²/s). The convective mass transfer coefficient is assumed to be $h_m = 1 \times 10^{-3}$ m/s (negligible convective resistance compared to diffusive resistance). The porous material is discretized into 100 layers. Figure 4(b) presents the use of *contamxpy* to implement the diffusion and source/sink models presented above. *ContamW* was used to create a model of two zones (0.35 × 0.35 × 0.15 m) with air handling units providing supply airflow rates of 6.58×10^{-2} m³/h. A constant concentration of HCHO at the inlet of 372 µg/m³ is supplied to Chamber A, while clean air is supplied to Chamber B. CONTAM control elements are set up to provide the exchange of information between *contamxpy* and the Python module in which the diffusion and source/sink model is implemented. The chamber concentrations were provided as the output control signals from *contamxpy,* and the pollutant removal/generation rates determined were provided as the input control signals to dynamically apply the removal/generation rate to a constant coefficient source/sink element.

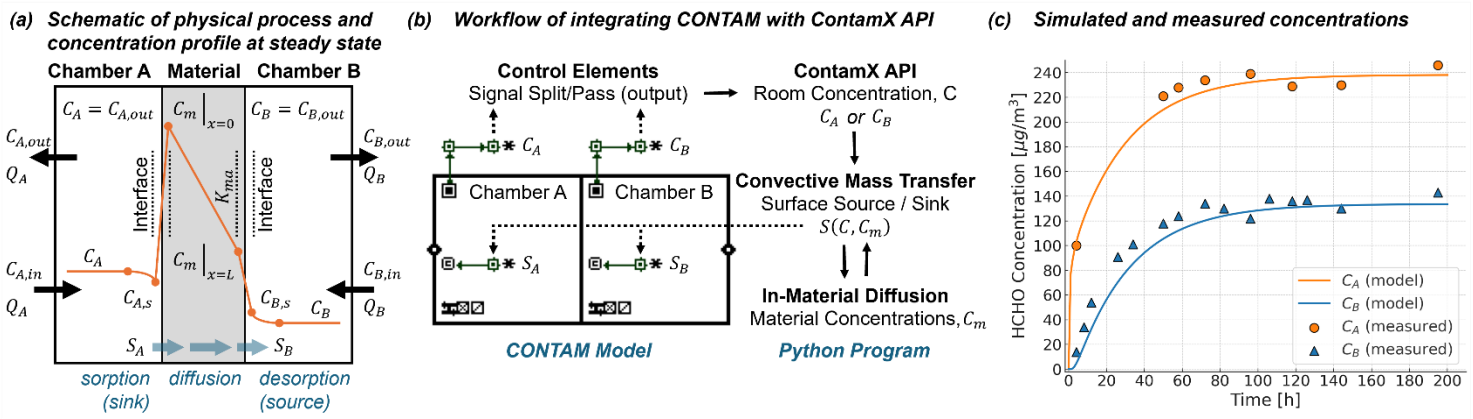

Figure 4 – Simulation of formaldehyde (HCHO) transfer through a porous material in a dual-chamber system (simulation data is compared to the measurement data in Xu, Zhang et al. 2009).

At each time step t, the removal/generation rate was calculated based on equations (4) and (5), using the current indoor air concentration (C) and material surface concentration ($C_{m,s}$) at the previous time step. The concentration within the material was updated according to the material source/sink model and stored for the next time step, and the removal/generation rate was applied to the sink/source term at the current time step. Figure 4(c) presents simulation results of the change of HCHO concentrations in the air of Chamber A and B using the developed approach. The simulation results demonstrate excellent consistency with the measurement data presented by Xu, Zhang et al. 2009.

This example illustrates how the simulation data exchange functions of *contamx-lib* can be used to obtain zone contaminant mass fractions and dynamically adjust source/sink terms based on user-defined models, such as the in-material diffusion model presented here. This approach also offers the potential to integrate CONTAM with other diffusion or hygrothermal modeling tools, such as DELPHIN, CHAMPS-BES (Nicolai et al. 2007), or WUFI (Fraunhofer IBP 2025), enabling more advanced co-simulations between airflow and building material modeling. This approach could also be extended to account for source/sink terms influenced by other environmental factors like temperature or humidity.

3.3 Architectural Design

Rhino is a widely used three-dimensional geometry modeling tool that can be applied to architectural design, and Grasshopper is a graphical algorithm editor. Both Rhino and Grasshopper offer software development kits (SDKs) to enable the creation of plugins, and Grasshopper is provided as a plugin to Rhino. Plugins can access geometries created on the Rhino canvas, enabling the development of building performance analysis tools within the Rhino-Grasshopper platform. Numerous plugins have been developed, including building energy modeling tools like Honeybee and Archsim (built on EnergyPlus and OpenStudio), weather analysis tools such as Ladybug, and computational fluid dynamics (CFD) tools such as Butterfly and Eddy3D (based on OpenFOAM).

A plugin named *ANT*, inspired by "CONTAM-in-ANT", was developed using the Rhino and Grasshopper SDKs and CONTAM APIs (Shen et al. 2024). *ANT* is built on top of *ContamP.net* and includes over ninety CONTAM-related components, organized into twelve categories as presented in Table 3**Error! Reference source not found.**. These components enable users to create CONTAM projects directly from user-defined, three-dimensional building geometry created in Rhino; run simulations using *ContamX*; and visualize results from within the Rhino-Grasshopper environment. The latest version of *ANT* is free to download, along with many other plugins (https://www.food4rhino.com/en/app/ant).

Table 3 – Categories of *ANT* components

Category	Description
Geometry	Create zone and surface elements (e.g., window, door) from Rhino geometry
HVAC	Define simple air handling systems (AHS), supplies, and returns
Filter	Define filter elements
Schedule	Define schedules: dimensionless, temperature, and occupancy
Species	Define contaminant species, kinetic reactions, and particle distributions
Source/Sink	Define source/sink elements for specified contaminants
Occupancy	Define occupant emission, exposure, and health impacts
Airflow	Define airflow paths and associated airflow elements
Library	Create and read CONTAM species, filter, schedule, and wind profile libraries
Ambient	Define weather and contaminant-related ambient boundary conditions
Simulation	Generate PRJ files and configure and run simulations
Results	Read and visualize simulation results

A typical workflow involves drawing a 3D building representation of building zones using the Rhino canvas. *ANT* components are then used to retrieve information from the geometry, including volume, surface area, and orientation, using Rhino and Grasshopper APIs. This information is used to generate CONTAM project components, including zones and airflow paths. *ANT* also enables the creation of simple air-handling systems, contaminant sources, and contaminant removal mechanisms. Once the building is defined, the *Ambient* component can be used to apply boundary conditions, e.g., CONTAM weather (WTH) and contaminant (CTM) files, and the *Simulation* component can be used to save PRJ files and run simulations. Currently, *ANT* simulations can utilize the *ContamX* simulation engine directly. Simulation results can be retrieved using the *Results* components and analyzed via the myriad available Grasshopper plugins and visualized on the Rhino canvas.

Figure 5 presents example applications of *ANT* that were presented in Shen et al. (2024), demonstrating its use in airflow and contaminant transport modeling, with enhanced visualization to incorporate the 3D modeling capabilities of Rhino-Grasshopper to show contours, vectors, and animations (Figure 5). Health outcomes from environmental exposure to air contaminants were quantified using the disability-adjusted life years (DALYs) model available among the *Occupancy* components of *ANT*. Infection risks from airborne viruses were estimated based on the inhaled dose of infectious quanta through the Wells-Riley model (Riley et al. 1978). Integrating *ANT* with other Rhino and Grasshopper plugins will further enhance its functionality (Shen et al. 2024).

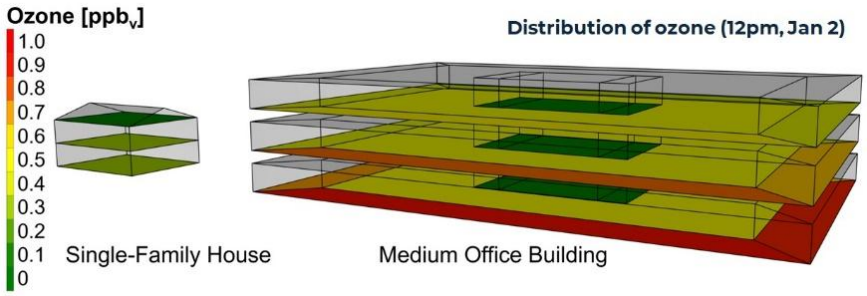

Figure 5 – Example applications of ANT: IAQ transport in residential and commercial buildings, adapted from Shen et al. (2024).

3.4 Urban Scale Analysis

For the purposes presented here, urban-scale analysis refers to IAQ-related simulations over a region that includes multiple buildings. Such simulation capabilities are often needed to estimate community exposures within the built environment based on ambient conditions, building archetypes, and system configuration and operation (Fazli et al. 2018, Underhill et al. 2018, Underhill et al. 2020) or to evaluate exposure to acute, outdoor events (Herring et al. 2014)Prior to the development of the thread-safe contamx-lib, using CONTAM in urban-scale building analysis required users to launch multiple instances of *ContamX*. Recently, multiple urban-scale analysis tools have been developed to take advantage of the CONTAM APIs to simplify and improve co-simulation between ambient airflow and contaminant transport models and health-based exposure models. Three such simulation environments are described here.

Rhino-Grasshopper

The *ANT* plugin for Rhino-Grasshopper can also be used to generate multiple CONTAM projects via *contamp-lib* and simulate them in parallel using *ContamX* while considering outdoor boundary conditions provided by other plugins. For example, a CFD-based plugin can be used to account for an external pollutant source and its transport throughout the urban landscape, including the effects of buildings on the outdoor airflow. This provides the ability to account for spatially and time-varying ambient wind and contaminant conditions, as opposed to the WTH and CTM files that provide spatially uniform, time-varying boundary conditions; greatly enhancing the feasibility of modeling acute health effects associated with IAQ on an urban scale. Other Rhino-Grasshopper plugins are used to obtain building properties from geographical information system (GIS) data to generate building blockages for the external airflow and contaminant transport, e.g., location, shape, and height, and to provide properties to enable *contamp-lib* to generate CONTAM building models, e.g., building volume. Figure 6 presents some example applications of ANT being applied to urban-scale and community-scale simulations.

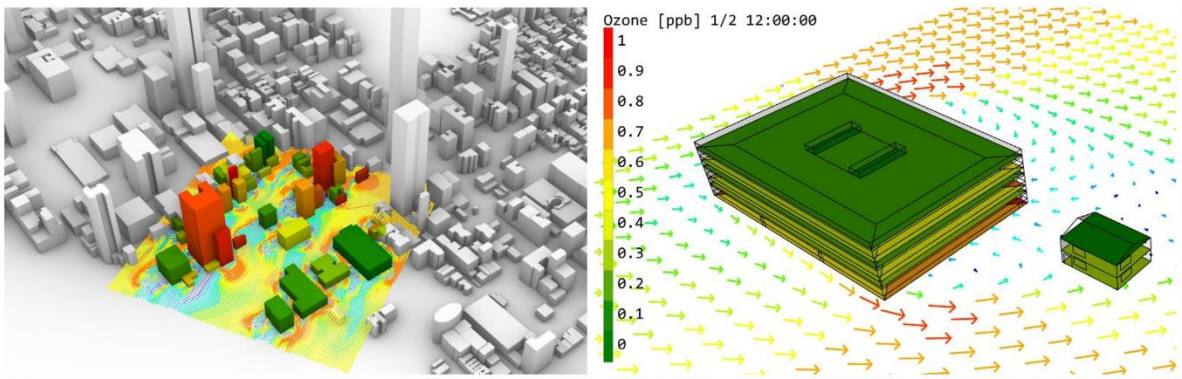

Figure 6 – Application of urban scale simulation with *ANT*

Integrated Urban (IU) and HPAC

IU and HPAC are software tools developed for the US Defense Threat Reduction Agency (DTRA) to predict the indoor and outdoor transport of chemical, biological, radiological, and nuclear (CBRN) material and the resulting health-related impacts (Herring et al. 2014, Sohn et al. 2024). However, each tool is built upon unique frameworks into which CONTAM is integrated using the JNA-based bindings to *contamx-lib*. IU was developed for DTRA and built on several technologies that enable CBRN simulations, including the Common CBRN Modeling Framework (CCMF), which defines protocols by which sub-models communicate. In the UK, two subsystems were integrated with HPAC by Riskaware. These two subsystems are the Internal Building Hazard (IBH) incident source model, which accounts for exfiltration of contaminants from a building to the ambient, and the Urban Subsystem (USS), developed for the UK Defence Science and Technology Laboratory (Dstl), which enables buildings to act as a receptor of ambient contaminants as well as a source of exfiltrated contaminants.

While both IU and USS incorporate their own single-zone building models, CONTAM was added to extend their indoor exposure analysis capabilities to address multizone building models. Each tool can interact with *contamx-lib* to obtain information on the number and location of envelope flow paths through which contaminants may be transported, either into or out of the buildings. Optional run configurations of each tool enable buildings to act as either a receptor or source of outdoor contaminants. Currently, for buildings as a receptor, co-simulation occurs on a sequential basis, i.e., ambient conditions are first simulated via the outdoor modules of the respective platforms, which are then provided to the building models as boundary conditions via the *contamx-lib* API. Because the building simulation library is thread-safe, multiple CONTAM models can be run in parallel by IU and USS.

Building simulation results are then accessed, visualized, and analyzed for health effects in accordance with the respective tools' CCMF and HPAC based requirements. Figure 7 shows an example IU simulation of a hypothetical outdoor plume passing through an urban environment for an outdoor source indicated by the crosshairs and a user-selected set of buildings indicated by the black dots. The illustration shows predictions of adverse health effects for people in outdoor and indoor environments, and the ability to address the importance of capturing the indoor-outdoor interactions (Sohn et al. 2019, Sohn et al. 2020). Figure 8 shows example HPAC-

IBH simulation results, including the contaminant exfiltration over time and a snapshot of an animation of transport within a multizone building. These results are meant to highlight the integration of the APIs with proprietary software and are not meant to provide in-depth results of a particular scenario. The integration efforts of DTRA and Dstl implemented their own in-house verification of the co-simulation using the APIs.

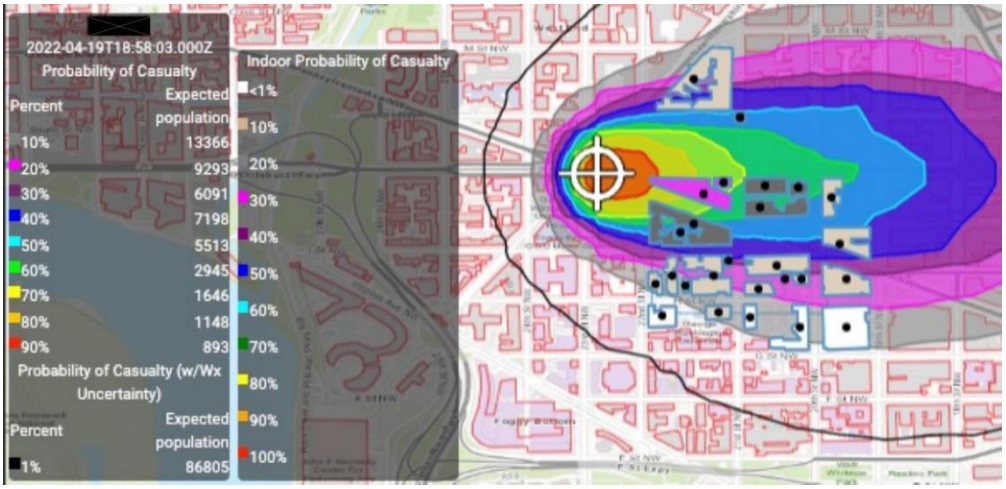

Figure 7 – Result of an IU example case showing probability of casualty for outdoor (colored contours) and indoor (buildings marked by black dots) populations as a result of an outdoor source (located at crosshairs)

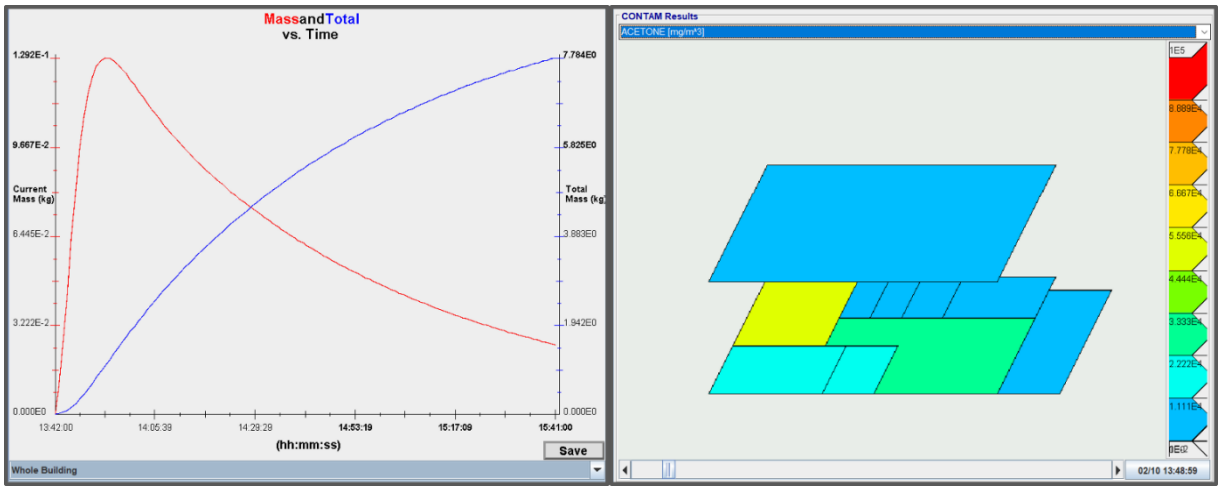

Figure 8 – Whole building and detailed building results from an example simulation performed with HPAC-IBH as presented within the user interface of HPAC-IBH (Mass = mass of contaminant [kg] in the building at the current time, Total = cumulative mass [kg], i.e., exposure over time)

These examples demonstrate the ability to incorporate CONTAM simulation within multiple frameworks. In the case of HPAC integration, CONTAM had been utilized prior to the development of the APIs. Approximately 2500 lines of socket-based CONTAM integration code were removed from the USS codebase and replaced with approximately 200 lines of code to enable the use of the *contamx-lib* API, demonstrating the improvements in the

path to integration of *contamx-lib* over the socket-based approach. Further, the dynamic library-based integration of *contamx-lib* resulted in a 75 % improvement in runtime performance over the socket-based approach.

The IU and HPAC-IBH co-simulation workflows presented here are "sequential" in nature. However, *contamx-lib* can be used within HPAC-USS to perform co-simulation, whereby indoor-outdoor data is exchanged dynamically, i.e., on a per time step basis. This improves the ability to address problems requiring more tightly coupled interaction and run-time, dynamic response of building systems and occupants to contaminant transport-related events. For example, in a densely populated urban area, the possibility exists that pollutant mass entering a building and later exhausting could have significant impacts on the downwind concentrations (reducing downwind concentrations in the short term, but extending the residence time of contaminants within the building). This could meaningfully impact the health consequences for people both outdoors and indoors. In the past, coupled indoor-outdoor simulation operated under the assumption that the mass of the outdoor plume entering buildings did not appreciably reduce the mass outdoors (Sohn et al. 2019, Sohn et al. 2020). Such dynamic co-simulation could also allow analysts to simulate changes in building control mode in response to an outdoor plume. For example, building ventilation systems could be shut down as the plume passes, or ventilation could be increased when outdoor concentrations fall below those indoors.

4 Future Work and Conclusions

The development of the CONTAM APIs opens new opportunities for enhancing CONTAM simulation capabilities and for integrating CONTAM into building design and analysis frameworks and co-simulation workflows. The *contamx-lib* API gives CONTAM users the ability to implement new sub-models, using the time-stepping techniques demonstrated by the Custom Control Algorithm (Section 3.1) and

Enhanced IAQ Transport Model (Section 0) examples. The *contamx-lib* API also allows integration of whole-building airflow and pollutant transport predictions with other tools, as was done for EnergyPlus, TRNSYS, Architectural Design (Section 3.3), and Urban Scale Analysis (Section 3.4) solutions presented above. Many of these developer-supported capabilities will become generally available to the CONTAM community, without requiring individual users to call API functions directly. The examples presented herein involved some level of cooperation between the CONTAM developers and those integrating CONTAM into their workflows and highlight the technology transfer that is a part of the NIST mission. However, this does not preclude others from performing integration without such direct cooperation, and the APIs will make such integration even easier in the future, for example, to incorporate CONTAM into programs, workflows, and frameworks related to IAQ analysis (Bhoonah et al. 2024), smoke control (Thunderhead Engineering 2025), machine learning (Ashayeri et al. 2024), digital twins, AI-based building design and analysis, and GIS-based tools (Usta et al. 2025).

The current version 3.4.1 of the CONTAM simulation engine, *ContamX*, and associated *contamx-lib*, could be further modified to enable the incorporation of other transport-related capabilities. For example, the current detailed zone modeling incorporated into CONTAM, i.e., CFD0 (Wang et al. 2010), is tightly integrated into the simulation code. This limits the ability to extend this capability without significantly affecting the code base. This limitation could be remedied by refactoring *ContamX* to enable CFD co-simulation via *contamx-lib*, thus opening the possibility for integrating with other detailed zone analysis tools or providing for enhancements of existing capabilities while maintaining separate domain expertise and code bases. Design and modification in consideration of such enhancements have yet to be undertaken.

To date, the *contamp-lib* API has only been incorporated into the Rhino-Grasshopper framework. Other frameworks, including those presented here, could also take advantage of the *contamp-lib* API, but may require bindings that have yet to be implemented. Work is proceeding on a solution to provide cross-language and cross-platform implementations of *contamp-lib*. This should reduce the need to generate bindings to native libraries that can suffer from the requirements of targeting specific operating systems and architectures. This will not only enable the CONTAM developers to better maintain and update CONTAM-related utilities, e.g., the CFD0 Editor and CONTAM Results Viewer, but will also extend the ability to integrate CONTAM building model development into other frameworks, as was demonstrated by the Architectural Design example presented above.

Finally, results associated with CONTAM simulations are provided via several files. Some of the results files are binary, and some are text-based. While utilities are provided by the developers of CONTAM to convert binary results files to text files, the results are still generic in nature, i.e., they may require significant post-processing for users to address engineered solutions. One possibility is to develop a results processing library and accompanying APIs to allow CONTAM users to focus on engineered solutions to IAQ and ventilation-related issues as opposed to having to develop detailed results processing tools. Such a results analysis API may also benefit from modifications to the CONTAM building model to facilitate improvements in the results processing experience.

Disclaimer

Certain commercial entities or products may be identified in this document to adequately describe a concept. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities or products are necessarily the best available for the purpose.

References

Ashayeri, M. and N. Abbasabadi (2024). A Hybrid Physics-Based Machine Learning Approach for Integrated Energy and Exposure Modeling. Artificial Intelligence in Performance-Driven Design**:** 57-79.

Axley, J. W. (1987). Indoor Air Quality Modeling: Phase II Report. Gaithersburg, USA, National Institute of Standards and Technology.

Bhoonah, R., M. Mendez and A. Maury-Micolier (2024). "Human health impacts and indoor chemical reactions of VOCs from cleaning products and occupants." Atmospheric Environment **338**.

Bonan, C., M. Nibart, P. Armand and C. Olry (2014). Coupling between PMSS and CONTAM: The Indoor / Outdoor Contaminant Transfer of a Hazardous Release. 16th Conference on Harmonisation. Varna, Bulgaria.

Chang, Y., Z. Ai, P. Wargocki, Y. Liu and Y. Hu (2024). "Design of convertible patient care unit for both non-pandemic and pandemic times: prototype, building spatial layout, and ventilation design." Building and Environment.

Clark, J. D., B. D. Less, S. M. Dutton, I. S. Walker and M. H. Sherman (2019). "Efficacy of occupancy-based smart ventilation control strategies in energy-efficient homes in the United States." Building and Environment.

Dols, W. S., S. J. Emmerich and B. J. Polidoro (2016). "Coupling the Multizone Airflow and Contaminant Transport Software CONTAM with EnergyPlus using Co-simulation." Building Simulation **9**: 469-479.

Dols, W. S., S. J. Emmerich and B. J. Polidoro (2016). "Using coupled energy, airflow and indoor air quality software (TRNSYS/CONTAM) to evaluate building ventilation strategies." Building Services Engineering Research and Technology **37**(2): 163-175.

Dols, W. S., A. K. Persily and J. B. Morrow (2011). Model Development and Validation for Particle Release Experiments in a Two-story Office Building. Gaithersburg, National Institute of Standards and Technology.

Dols, W. S. and B. J. Polidoro (2020). CONTAM User Guide and Program Documentation Version 3.4, National Institute of Standards and Technology.

Dols, W. S., L. Wang, S. J. Emmerich and B. J. Polidoro (2015). "Development and Application of An Updated Whole-building Coupled Thermal, Airflow and Contaminant Transport Simulation Program (TRNSYS/CONTAM)." Journal of Building Performance Simulation **8**(5): 326-337.

Emmerich, S. J. (2001). "Validation of Multizone IAQ Modeling of Residential-scale Buildings: A Review." ASHRAE Transactions **107**(2).

Fazli, T. and B. Stephens (2018). "Development of a nationally representative set of combined building energy and indoor air quality models for U.S. residences." Building and Environment **136**: 198-212.

Fraunhofer IBP. (2025). "WUFI®." Retrieved Sep 2025, from https://wufi.de/en/.

Herring, S., J. England, B. Lingard and S. Parker (2014). Combined Outdoor-Indoor Dispersion Modelling in Urban Areas. 16th International Conference on Harmonisation within Atmospheric Dispersion Modelling for Regulatory Purposes. Varna, Bulgaria.

Kashtan, Y., M. Nicholson, C. J. Finnegan, Z. Ouyang, A. Garg, E. D. Lebel, S. T. Rowland, D. R. Michanowicz, J. Herrera, K. C. Nadeau and R. B. Jackson (2024). "Nitrogen dioxide exposure, health outcomes, and associated demographic disparities due to gas and propane combustion by U.S. stoves." Science Advances **10**(18): eadm8680.

Musser, A., O. Schwabe and S. J. Nabinger (2001). Validation and Calibration of a Multizone Network Airflow Model with Experimental Data. eSim 2001. Ottawa, Canada.

Nicolai, A., J. Grunewald and J. Zhang (2007). Recent improvements in HAM simulation tools: Delphin 5/CHAMPS-BES. Proceedings of 12th Symposium of Building Physics, Technische Universität Dresden Dresden, Germany.

Persily, A. K. and B. J. Polidoro (2022). Indoor Carbon Dioxide Metric Analysis Tool.

Picard, C. F., L. Cony Renaud Salis and M. Abadie (2022). "Home quarantine: A numerical evaluation of SARS-CoV-2 spread in a single-family house." Indoor Air **32**(5).

Pistochini, T., M. Ellis, F. Meyers, A. Frasier, C. Cappa and D. Bennett (2023). "Method of test for CO2-based demand control ventilation systems: Benchmarking the state-of-the-art and the undervalued potential of proportional-integral control." Energy and Buildings **301**.

Poppendieck, D., S. Khurshid, W. S. Dols, L. C. Ng, B. J. Polidoro and S. J. Emmerich (2016). Formaldehyde Concentrations in a Net-Zero Energy House: Real-time Monitoring and Simulation. Indoor Air 2016. Ghent, Belgium.

Riley, E., G. Murphy and R. Riley (1978). "Airborne spread of measles in a suburban elementary school." American journal of epidemiology **107**(5): 421-432.

Rim, D., A. K. Persily, S. J. Emmerich, W. S. Dols and L. Wallace (2013). "Multi-zone modeling of size-resolved outdoor ultrafine particle entry into a test house." Atmospheric Environment **69**(0): 219-230.

Robert McNeel & Associates. (2024). "Rhinoceros." Retrieved 10/25/2024, 2024, from https://www.rhino3d.com/.

Shen, J., W. S. Dols and B. J. Polidoro (2024). ANT: A Multizone Indoor Air Quality (IAQ) and Ventilation Analysis Plug-in for Algorithm Aided Design. Proceedings of SimBuild Conference 2024, IBPSA-USA. **11:** 278-287.

Sohn, M. D., W. W. Delp, R. N. Fry and Y.-S. Kim (2019). "Analysis of a series of urban-scale chlorine dispersion experiments and implications on indoor health consequences." Atmospheric Environment **212**: 83-89.

Sohn, M. D., X. Li and W. R. Chan (2020). "Estimates of pollutant temporal and spatial variability in commercial buildings from the joint urban 2003 field experiments." Indoor Air **30**(2): 335-345.

Sohn, M. D., D. M. Lorenzetti, P. E. Bieringer, S. Kreyenhagen, J. Hurst, S. Runyon, S. Cregan, W. S. Dols and B. J. Polidoro (2024). Integrated Urban: Vision And Results For The Urban HazardAssessment Capability. CBD S&T Conference 2024, Fort Lauderdale, FL, USA, Defense Threat Reduction Agency.

Thunderhead Engineering. (2025). "Ventus." 2025, from https://www.thunderheadeng.com/ventus/.

Tognon, G., M. Marigo, M. De Carli and A. Zarrella (2023). "Mechanical, natural and hybrid ventilation systems in different building types: Energy and indoor air quality analysis." Journal of Building Engineering **76**.

Underhill, L. J., M. P. Fabian, K. Vermeer, M. Sandel, G. Adamkiewicz, J. H. Leibler and J. I. Levy (2018). "Modeling the resiliency of energy-efficient retrofits in low-income multifamily housing." Indoor air **28**(3): 459-468.

Underhill, L. J., C. W. Milando, J. I. Levy, W. S. Dols, S. K. Lee and M. P. Fabian (2020). "Simulation of indoor and outdoor air quality and health impacts following installation of energy-efficient retrofits in a multifamily housing unit." Building and Environment **170**: 106507.

Usta, Y., L. Ng, S. Santantonio and G. Mutani (2025). "Lumped-Parameter Models Comparison for Natural Ventilation Analyses in Buildings at Urban Scale." Energies **18**(9).

Walker, I. S., B. D. Less, C. H. Lozinsky, D. Lorenzetti, N. Casquero-Modrego and M. D. Sohn (2024). "Compartmentalization and ventilation system impacts on air and contaminant transport for multifamily buildings." International Journal of Ventilation: 1-28.

Walter, K. (2004). Virtual Problem Solving for Homeland Security. Science and Technology Review. Livermore, CA, USA, Lawrence Livermore National Laboratory. **September 2004:** 20-21.

Walton, G. N. (1989). AIRNET - A Computer Program for Building Airflow Network Modeling. Gaithersburg, MD, National Institute of Standards and Technology.

Wang, L. L., W. S. Dols and Q. Chen (2010). "Using CFD capabilities of CONTAM 3.0 for simulating airflow and contaminant transport in and around buildings." Hvac&R Research **16**(6): 749-763.

Wang, Y., G. Petrou, P. Symonds, S.-C. Hsu, J. Milner, E. Hutchinson, M. Davies and H. L. Macintyre (2025). "Investigating the Impacts of Home Energy Retrofit on the Indoor Environment through Co-Simulation: A UK Case Study." Journal of Building Engineering: 111794.

Xu, J., J. Zhang, J. Grunewald, J. Zhao, R. Plagge, A. Ouali and F. Allard (2009). "A Study on the Similarities between Water Vapor and VOC Diffusion in Porous Media by a Dual Chamber Method." CLEAN – Soil, Air, Water **37**(6): 444-453.