

Why addresses & tax lots are holding back your data's potential – and how to unlock it

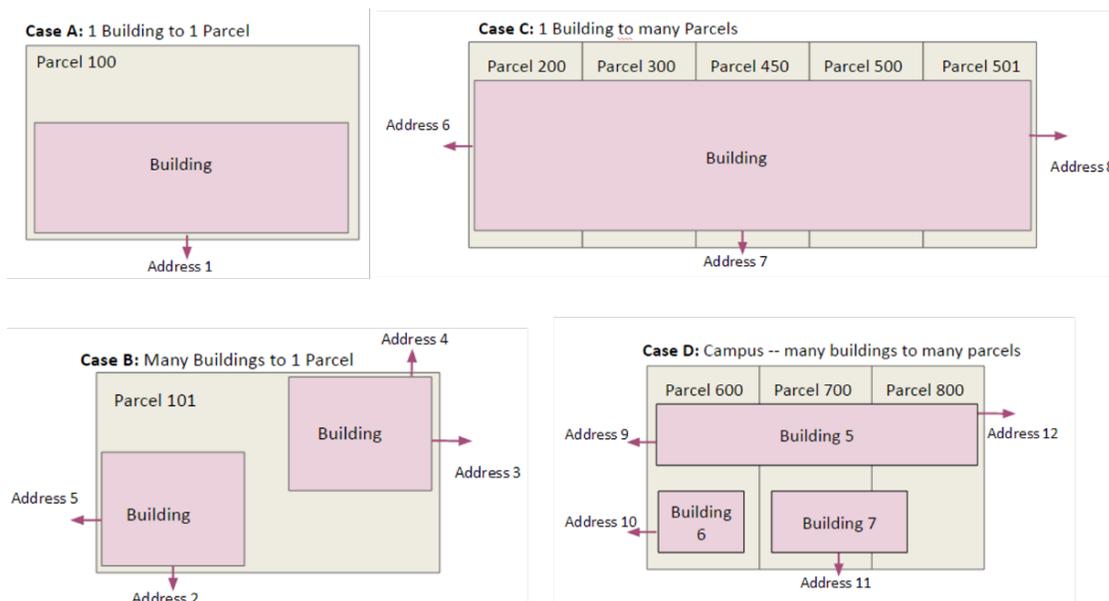
Buildings generate data throughout their lifecycle – about ownership & taxation, usage, zoning, code compliance, energy use, and retrofits. State and local governments collect this data after it flows through growing networks of people and systems. But collecting data is only half the battle; what's really needed is information – the actionable insights that lead to successful policy outcomes.

Until now, a major challenge stymied the conversion of all this data into usable information: How do you connect complementary datasets in order to unlock their full potential?

For data to be useful, its associated objects must be identified consistently. Unfortunately, building-related datasets are notoriously *inconsistent*; the same structure might appear as “123 First Street” in one database, “123 1st St.” in another, and “118 1st St.” (a data entry error) in a third. While a human in the loop can easily see that the first two addresses refer to the same building, a computer would struggle to do so.

Why does it matter if the same object has different names? Suppose the three databases above hold different information about the building at 123 First Street. Because each one calls it something else (one of the three street addresses), the user may never recognize that each database has relevant information about the same building! “Siloed” datasets like this are less valuable and more difficult to maintain than connected datasets – which is why governments have long searched for alternative object IDs.

Many choose to identify buildings by proxy using tax lot numbers, but the relationship between tax lots and individual buildings can be frustratingly ambiguous (see an illustration below). Internal IDs developed by governments themselves are yet another improvement. They're cumbersome to implement, however, and require time-consuming manual updates.



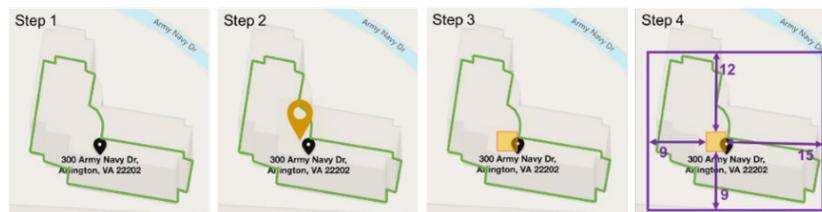
Above: The relationship of buildings to parcels/tax lots can be complicated

To really unlock the full potential of their datasets, public agencies and data aggregators need a tool to generate unique “natural key” IDs for the objects within them (“natural keys” are based on real-world attributes – in this case, the global latitude and longitude grid system). This tool should also assign the same ID to the same object anywhere it appears, whether a list of covered buildings, tax assessment records, or anywhere else. The ideal solution would cost little time and no money – public officials must routinely keep a hundred balls in the air with minimal staff as it is.

Solution

In response to this need, the U.S. Department of Energy (DOE) and Pacific Northwest National Laboratory (PNNL) developed the free and open-source Unique Building Identifier (UBID). UBIDs are unique record numbers (e.g., “849VQGGV+V42-005-004”) which represent the two-dimensional spaces on a map occupied by building footprints, tax lots, or other datapoints of interest.

Assigning IDs to the space an object occupies yields a more effective, consistent, and reliable identifier than building address, parcel IDs, or in-house serial numbers.



Above: UBID in action

Governments that implement UBIDs can streamline systems that track changes to building ownership, tax permit status, or other data across departments, cutting administrative overhead and gaining powerful capabilities. (For example, a building and its annual benchmarking data can be consistently linked across many years – even if it were to change owners, receive a new or modified ENERGY STAR Portfolio Manager ID, or otherwise be subjected to record changes.)

Any two or more datasets which have been assigned UBIDs can be matched automatically: zoning codes from urban planning, special business districts from economic development, demographics from a city census, to take but a few examples. (See an expanded table of potential city datasets in Appendix II).

UBID is incorporated into the [Standard Energy Efficiency Data Platform](#) and [ENERGY STAR Portfolio Manager](#) as a standardized identifier for buildings and land parcels. Sustainability leaders such as Washington, D.C. have assigned UBIDs to their building assets and published them in open access datasets.

“The UBID implementation steps laid out by PNNL were easy to follow, and the process didn’t require anything beyond very simple code. This was easy for me.”

— Adefunke Sonaike, D.C. Department of Energy and Environment (DOEE)

Getting Started

If your organization has a Geographic Information System (GIS) software license and at least one person with basic coding skills, you can implement UBIDs. Take a look at the typical process below and contact BuildingID@pnnl.gov to get started.

▶ **Step 1: Schedule a consultation**

Contact BuildingID@pnnl.gov to develop an implementation plan tailored to your organization's needs. PNNL experts can even work with you to generate and assign UBIDs, verify their accuracy, and link UBID datasets.

▶ **Step 2: Meet with stakeholders**

Convene managers and technical implementers from relevant teams in your organization. A local government might include personnel from their buildings, tax, emergency services, mayoral, public works, and parks departments.

Involving everyone from the beginning makes your data more valuable, since more of it will be linked together with UBIDs. Getting buy-in from key stakeholders also kindles the enthusiasm for a rapid, smooth implementation.

▶ **Step 3: Get footprint Data**

Once you've identified which datasets to update with UBIDs, approach your organization's GIS expert. They'll locate the corresponding "footprint" data (collections of latitude and longitude coordinates) for buildings and other objects.

▶ **Step 4: Cleanse footprint data**

PNNL can work with you to confirm the quality of your footprint data and correct any issues. Once this is done, the data is considered 'cleansed' and is ready to be fed into the UBID generation tool.

▶ **Step 5: Generate & Assign UBIDs**

Your GIS team (or anyone else with basic coding skills) can use the UBID 'pipeline' toolkit (see this and other technical resources in Appendix I) to generate UBIDs and assign them to the records in your dataset(s).

▶ **Step 6: Confirm UBIDs**

Just as footprint data was cleansed before generating UBIDs, you'll want to work with PNNL to confirm that the newly-assigned UBIDs are accurate.

▶ **Step 7: Integrate datasets**

This is where UBIDs really shine. If you've assigned UBIDs to two or more datasets, those can be linked together using the UBID cross-reference tool. Building energy benchmarking programs, for

example, can link a list of covered buildings with parcel and ownership information from tax records. This is just one of many possibilities for unlocking the power of your data with UBIDs.

► **Step 8: Assign new UBIDs**

Repeat the steps above in response to future needs, such as annual new building construction.

UBID Development: Past, Present, and Future

Following initial development of the UBID tools and algorithms by PNNL, a pilot working group and steering committee were convened in 2017. These stakeholders produced a scoping study and successfully demonstrated UBID assignments for selected cities, counties, and states.

To facilitate city and local governments, energy service companies, building data aggregators, and researchers use of UBID, DOE launched the two-year Building Energy Data Analysis (BEDA) Accelerator in 2018, which focused on the testing and applications of UBID to resolve real problems. The Accelerator served as a training and collaboration group for early UBID adopters to develop and share best practices, which are defined and tracked by the DOE Data Tools team as UBID becomes more broadly implemented.

One of the key outputs of this group was experience-driven guidance on best practices for UBID implementations, relevant to various audiences and use cases. Foremost among these in terms of the coming year (2021) is local building energy benchmarking policies, which can be significantly enhanced through the use of UBIDs in relevant datasets. DOE and PNNL plan to promote and support such initiatives, and to this end have put together a comprehensive resource base (see the Technical Resource Directory below).

Another effort under consideration is the development of a Graphical User Interface (GUI) for the UBID tools, which would allow those with more limited coding experience to fully harness the tool.

Learn More

We welcome all participants who are passionate about buildings and/or data. Learn more at BuildingID.pnnl.gov or send us an email at BuildingID@pnnl.gov and let us know how you'd like to be involved.

Appendix I: Technical Resource Directory

[Project Website \(buildingid.pnnl.gov\)](http://buildingid.pnnl.gov)

A PNNL-branded website for the UBID project; it includes links to DOE and BTO websites, the public buildingid Git repository on GitHub, and the new UBID Demonstrator website. The “Download” page links to webinar slide decks and recordings, BEDA Accelerator slide decks, conference posters, and journal articles.

[Original UBID Demonstrator Website \(ubid.pnnl.gov\)](http://ubid.pnnl.gov)

A web application with map-based UBID assignment and data visualization capabilities. It provides open access to UBID-assigned datasets for 4 major U.S. cities: Chicago, New York City, San Francisco, and Washington, D.C. (Map component and aerial photography provided by Microsoft Bing Maps API.)

[New UBID Demonstrator Website \(buildingid.github.io\)](http://buildingid.github.io)

A suite of UBID-related web applications with open source code that is designed to be copied, pasted and extended by third parties; including, but not limited to:

- **Drawing Tools.** Draw an area on the map. Edit and erase the existing shape. Import UBID code strings and Well-Known Text strings.
- **Decode Example.** Decode a UBID code string, then view the shape on the map. View minimum and maximum coordinates for UBID code area.
- **Encode Example.** Encode minimum and maximum coordinates as a UBID code string, then decode and view the shape on the map.
- **Well Known Text Read Example.** Decode a Well Known Text string, encode a UBID code string, then view the shape on the map.

[UBID Demonstrator Source Code \(github.com/buildingid/buildingid.github.io\)](https://github.com/buildingid/buildingid.github.io)

HTML, CSS and JavaScript source code for the new UBID demonstrator website.

[UBID Documentation \(github.com/pnnl/buildingid\)](https://github.com/pnnl/buildingid)

Documentation, including, but not limited to:

- **UBID specification document**
Formal definitions of algorithms for UBID encoding, decoding and validation.
- **UBID visualization guide**
Recommendations and best practices for representation of UBIDs as images.
- **UBID data flow guide**
Overview of information flow in software systems that use UBID. Describes minimum data requirements for UBID assignment and how to convert between UBID and other data types. Uses the [UBID demonstrator website](http://buildingid.pnnl.gov) to provide interactive examples.
- **FAQ**
Technical reference material with in-depth discussion of UBID design rationale and implementation details.

[“Open Spaces” API Description Document \(github.com/pnnl/buildingid-api-docs\)](https://github.com/pnnl/buildingid-api-docs)

An OpenAPI/Swagger description document for a working draft of the “Open Spaces” API, a vendor-neutral specification for exchanging descriptions of spatially extended entities.

[UBID C-sharp API \(github.com/pnnl/buildingid-csharp\)](https://github.com/pnnl/buildingid-csharp)

An API that provides implementations of UBID encoding, decoding and validation algorithms for Microsoft C-sharp .NET programming language.

[UBID JavaScript API \(github.com/pnnl/buildingid-js\)](https://github.com/pnnl/buildingid-js)

An API that provides implementations of UBID encoding, decoding and validation algorithms for JavaScript programming languages.

[UBID Python API & CLI \(github.com/pnnl/buildingid-py\)](https://github.com/pnnl/buildingid-py)

An API that provides implementations of UBID encoding, decoding, and validation algorithms, and CLI that performs UBID assignment for tabular data files and that performs UBID cross-reference for tabular data files by constructing quadtree indices.

[UBID Ruby API \(github.com/pnnl/buildingid-rb\)](https://github.com/pnnl/buildingid-rb)

An API that provides implementations of UBID encoding, decoding, and validation algorithms for Ruby programming language.

Appendix II: Example City Datasets Which Can Be Linked Using UBIDs

Building Performance Information	<p>Benchmarking data, including building characteristics (size and age) and energy and water performance, can be used to identify low performing buildings to target with programs.</p> <p>Audit and Retrocommissioning (RCx) data, which might include equipment and systems information and inventories, recommended energy conservation measures (ECMs) and other energy use-related information, can identify high-priority energy savings opportunities within a city and help in designing programs that can increase the uptake of energy efficient technologies.</p> <p>Building Performance Standard (BPS) data, which include at minimum the energy performance data collected through benchmarking, track compliance of a BPS policy and can be used to target low performers with energy efficiency programs.</p>
Tax Data	<p>Tax lot and assessed value is stored in a dataset containing information for the revenue a city collects for each property and may include other parcel-level data fields.</p>
Permit Data	<p>Permit data for new construction is stored in a dataset containing information on how much, and what type, of additional space is being added to the city.</p> <p>Permit data for renovations records changes in space use over time. Depending on the jurisdiction, these data may also include building systems information. Integrated with UBIDs and benchmarking data, this data could track how building upgrades are affecting energy performance and meeting energy efficiency program goals.</p>

Published January 2021

Pub. No. DOE/EE - 2319