Ontology Editor's Guide

Ontology Terminology and Properties

Basic Terminology

It is helpful to understand that the VIVO web application is built using "triples" consisting of a **subject** (an **individual** in ontology terms, also sometimes referred to informally as an item or entity), a **predicate** (an **object property** or a **data property**) and an **object** (once again, any **individual** in VIVO).

These "triples" are also called **statements** and reflect the structure of a sentence in ordinary language.

Subject-predicate-object triples express the relationships among the individuals in VIVO using **object properties** and support attributes of individuals (e.g., first name, start date) using **data properties**.

Classes

Individuals in VIVO are typed as members of one or more **classes** organized and displayed as a hierarchy. **Child** or sub-classes inherit the **properties** of their **parent** or super-classes. A child class is a specialization of a parent class – if Person is the most general class for people, then Faculty Member and Student are more specialized child classes . The class hierarchy in an ontology acts much like a **taxonomy**, where everything about a parent is also true for its children.

Once classes have been created, they can be populated with **individuals** – as data on an underlying framework.

Properties

A thesaurus extends a taxonomy by providing certain standard relationships among concepts or terms – not only the **broader** term and **narrower** term relationship of the hierarchy itself but also relationships such as **related** term or **see also**. In an ontology, the class hierarchy framework can be extended with more open-ended relationships, called **object** and **data** properties.

Object properties connect two individuals (a **subject** and **object**) with a **predicate**, while with **data properties** the **predicate** connects a single **subject** with some form of attribute data. Data properties have defined **datatypes** including string, integer, date, datetime, or boolean. Both object and data properties are often but not always defined to have a **domain** class, such as Person or Event, specifying the class membership of the **individuals** serving as **subjects** for each object or data property **statement**. Object properties also may have a **range** class that specifies the class membership of the **individuals** serving as **objects** of the property **predicate**.

Property Editing

VIVO has two property editors – one for object properties and another for data properties. Data properties are simpler to create and edit because only a domain class and a datatype need to be selected. Object properties also have a range class and additional decisions must be made about whether or not to create an **inverse property**.

VIVO as an application is designed to create and show object properties from the subject to the object and vice-versa. Bi-directional relationships allow users to navigate from a person to a related department or grant while also supporting lists of department members on department pages or investigators on grant pages. This feels natural, is less work to maintain, and helps assure that an end user arriving at a VIVO page from a Google search can see and navigate easily to contextual information. The properties are typically directional in nature and opposite in meaning – if the Biology Department is **part of** its parent College of Life Sciences, the College **has part** Biology Department.

Object properties can be uni-directional, however, and they can also be defined as **symmetric**. A symmetric property asserts the same relationship in both directions, with the most common example being related terms.

Ontology List

What is the ontology list?

VIVO supports keeping an internal list of ontology namespaces and corresponding prefixes to facilitate using external ontologies as well as to help differentiate local ontology additions from VIVO core.

VIVO uses the common shorthand prefix when referring to an ontology in the ontology editor, such as bibo: (short for Bibontology), foaf: (short for Friend of a Friend), or core: (short for VIVO core). The prefix stands for the fully qualified namespace for the ontology – http://purl.org/ontology/bibo/ for the Bibontology, http://xmlns.com/foaf/0.1/ for FOAF, and http://vivoweb.org/ontology/core# for VIVO core.

When an ontology namespace has not been added to this list, no prefix appears before a class or property name in the ontology editor. If then a class or property name happens to be duplicated from one ontology to another (e.g., bibo:Image and foaf:Image), it's not evident on the VIVO ontology editor's pick lists which name is which.

Adding a prefix

The Site Admin > Ontology List command lists the ontology names, namespaces, and prefixes that your installation of VIVO is currently aware of. If an ontology is listed but no prefix has been specified yet, the ontology name will be used as the prefix in the ontology editor – making the names on pick lists more bulky. To add a prefix, click on the ontology name and then the edit button in the center column of the blue control area on the form that appears. Specify a prefix (without the colon) and click the Submit changes button to save. Note that while the editor allows spaces in the prefix, it is conventional not to use spaces in ontology prefixes.

Adding a new ontology namespace

Adding another ontology namespace to the ontology list will allow VIVO's ontology editor to recognize and include that namespace as a selection option when adding new classes or properties.

Site Admin > Ontology List > Add New Ontology To add a new namespace, click on Add new ontology at the top of the ontology list or on the right on the blue control area of the ontology list editing form. Specify a name, namespace, and prefix for the ontology and then submit the form with the Create New Record button.

The added ontology should be visible in the ontologies listed under "Ontology List".

For example:

- To add the SWRC ontology to VIVO:
- Go to Site Admin > Ontology List > Add New Ontology
    - Ontology Name="SWRC"
    - Namespace URI=http://swrc.ontoware.org/ontology
    - Namespace Prefix="swrc"
- Create New Record button

Keep in mind

- Type namespace URIs carefully and include the final / or # depending on the namespace URI format of the ontology involved
- Adding an ontology namespace to VIVO's ontology list does not import the specified ontology's classes or properties (or data for individuals). To add an ontology with an RDFi file, please see Add/remove RDF
- If local requirements introduce the need to create additional ontology classes and properties for your installation of VIVO, do not create extensions in the VIVO core namespace or the namespace of any other existing ontology. First create a new ontology for your extensions as described above and give it a namespace with the following pattern: http://vivo.mydomain.edu/ontology/vivo-mydomain. For example, Cornell ontology extensions would be in the namespace (http://vivo.cornell.edu/ontology/vivo-cornell/) . Note the trailing right-slash character. Then, when creating classes or properties, be sure to select this new ontology in the "ontology" dropdown list.

Data Input

You use this form to add an individual (aka entity) of a specific class or type. You can see examples of adding content in another section.

Site Admin > Data input - Add individual of this class

- Choose a type of individual you want to create in the drop down menu
- Click on the **Add individual of this class** button > **Individual Editing Form**
- Enter data into the following fields: **Individual Name, Specific Type [new moniker]** and any others you wish.

For example: Add a Faculty Member

Site Admin > Data input - Add individual of this class

- Choose the class "Faculty Member" and click the button **Add Individual of this class > Individual Editing Form**
- **Individual Name**=Jane Faculty
- **Specific Type [new moniker]**=Assistant Professor

NOTE: In Ontology terminology, an "individual" is equivalent to a node. A node is anything from a person to a conference.

Class Management

Individuals and classes

All information in VIVO is stored one way or another as subject-predicate-object statements known as triplesi. The subjects and predicates (and many objects) in these triple statements are references that can be repeated in many statements; references are expressed as identifiers that in the RDFi world take the form of URIs.

All the statements referencing the same identifier as either a subject or object collectively describe an individual in the VIVO system. You may encounter various alternative names for indivdual including resource, entity, item, node, or object. Note, however, that the individual in RDF has no fixed structure, unlike objects in object-oriented programming languages or records in relational databases.

The class hierarchy

Individuals in VIVO are, however, typed (using the rdf:type property) as members of one or more **classes** organized and displayed as a hierarchy. **Child** or sub-classes inherit the **properties** of their **parent** or super-classes. A child class is a specialization of a parent class – if Person is the most general class for people, then Faculty Member and Student are more specialized child classes. The class hierarchy in an ontology acts much like a **taxonomy**, where everything about a parent is also true for its children.

The class hierarchy provides a framework to help identify the different types of individuals modeled in a VIVO application. Class definitions are combined with specifications of the relationships or properties that may be associated with members of the class. Classes and properties together determine what statements can be created to describe individuals (either as assertions by users or imported from other data sources, or as inferences through reasoning).

Classes are much more evident when defining a VIVO ontology or adding content than they are from the public view of a VIVO installation. They do appear in search results and on the Index page as the second-level faceting (or sub-types) under the broad, top-level class groups such as people, activities, events, and organizations.
To add or edit classes, see **Class Hierarchy**.

What are class groups?

Class groups are a VIVO-specific extension to support using VIVO as a public website as well as an ontology and content editor.

Class groups are what the name implies: informal groupings of classes to organize pick lists, search results, and the VIVO Index page. Most often they mirror the top levels of the class hierarchy, with notable exceptions:

- Sometimes it is important to highlight one or two classes that may logically fall one place in the class hierarchy but warrant an independent category. At Cornell, for instance, courses are separated from other activities into their own class group to reflect their prominence in the academic life of the institution.
- Some classes are not intended to serve as public, free-standing data elements such as people, organizations, grants, or publications. Examples include Educational Background, a class to hold information about one degree on a person's list of degrees in the educational background section of their VIVO profile. Information about a single degree earned would make little sense outside of the context of the person's profile and would not be added or modified out of that context.
- A class that is not in a class group *will not appear on the pick list for adding new individuals*on the Site Admin page or on the list of classes (by class group) on the Index page. To add an individual in a class not in a class group, navigate to the class in the class hierarchy and click the Add New Individual in this Class button.

To review, add or edit class groups, see **Class Groups**.

Class Hierarchy

In the Class Hierarchy page, you can edit/add classes, add individuals/entities to a class, and add autolinks .

To edit an existing class

Site Admin > Class Hierarchy

- Click on class to be edited > **Class Control Panel**
- **Edit Class** button > **Class Editing Form**
- Change as desired: **Class name, Class group** from pick list, **Ontology** from pick list (this would typically be your local ontology), and **Internal Name** (all lower case, no spaces between words)
- Note that if the new class is not in a class group, it will not appear on the list of classes when adding individuals on the Site Admin or Individual Control Panel pages
- The **Display level** and **Update level** choices should be left at the least restrictive level of **public**; these annotations on the class are not currently active
- Add a short definition and example if possible
- Ignore (leave blank) **Display Limit** and **Display Rank**, which are currently active on object and data properties only
- Leave any entries for **custom entry form, custom display view, custom short view**, and **custom search view** as is (most frequently blank)
- **Submit changes** - edited class should be visible under appropriate class group from **Site Admin > Class Groups**

To add a new class

Site Admin > Class Hierarchy

Only very seldom will a new class be added at the top level of the class hierarchy. It is much more common to add a class as a subclass of an existing class at the most detailed level of the current hierarchy in order to provide more granularity as needed for a local VIVO installation. This allows the more detailed information to be differentiated locally while still allowing the more general form to be harvested for national-level analysis, searching, and browsing.

Note that in a deep class hierarchy it is possible that not all classes will be displayed from the top level; by clicking on any class partway down the hierarchy (e.g., Person) and then selecting the **Show hierarchy below This Class option**, a browsing view is generated that shows only the hierarchy below the selected starting class.

To add a new subclass in the desired place in the hierarchy, navigate to the desired class via the hierarchy as described above or by selecting* Class Hierarchy > All classes* and searching for the name of the desired class in alphabetical order within the listing.

- Click on the name of the desired class for subdividing and select **Class Control Panel > Add New Subclass of This Class**. (Or in rare cases when adding a top-level class, **Class Hierarchy > Add new class > Class Editing Form**)
- Add **Class name**, choose appropriate Class group from pick list, choose your local Ontology from pick list, and add Internal Name (all lower case, no spaces between words)
- Note that if the new class is not in a class group, it will not appear on the list of classes when adding individuals on the Site Admin or Individual Control Panel pages
- The **Display level** and **Update level** choices should be left at the least restrictive level of **public**; these annotations on the class are not currently active
- Add **short definition** and **example** if possible
- Ignore (leave blank) **Display Limit** and **Display Rank**, which are currently active on object and data properties only
- Leave any entries for **custom entry form, custom display view, custom short view**, and **custom search view** as is (most frequently blank)
- **Submit changes** - the Class Control Panel appears. To get back to the class hierarchy select Class Control Panel > Show Class Hierarchy or Class Control Panel > Show All Classes.

The added class should be visible in the **Site Admin > Class Hierarchy**, via **Site Admin > Class Hierarchy > All classes**, and under the appropriate class group from **Site Admin > Class Groups**

Note also that any class can be positioned at a lower level of the class hierarchy after creation by specifying a **Class Control Panel > New Link to Superclass** from the class, or by specifying Class Control Panel > New Link to Subclass from the desired parent class.

To add a new individual to a class via the class hierarchy

Site Admin > Class Hierarchy

- Click on class to be edited > **Class Control Panel**
- Click on **Add New Individual in This Class > Individual Editing Form**
- Specify appropriate **Individual** name, choose **Specific type** from pick list or add new **Specific type** by choosing [new moniker] and typing it in the field provided, add **URL** and **Anchor Text for URL** associated with the URL if desired, add **Blurb** to describe individual being added if appropriate.
- If appropriate (e.g., if item being added is time-associated) add **Sunrise** (date item first becomes visible) and **Sunset** (date item stops being visible), add **Timekey** if item is to be sorted chronologically in a list of similar items.
- **Create New Record > Class Control Panel** - added item should be visible upon clicking **Show All Individuals in This Class**.

To add a class autolink to a tab and associate all individuals/entities of a certain type with that class

For example: **Add a Class Autolink** to **Librarian** tab to bring items of type **Librarian** into **Librarian** tab)

- Go to tab **People** at top of VIVO screen > click on tab/link **Faculty** > click on tab/link **Librarians**
- At the bottom of this page: "edit tab: **Librarians**" > **Tab Control Panel**
- **Add Class Autolink > Tab**=Librarians; **Vclass**=Librarian
- **Create New Record > Tab Control Panel > Vclass Librarian** should be visible above **Remove Checked Class Autolinks** button

Class Groups

Class groups are a means to organize the classes in VIVO into groups. They represent the facets seen when VIVO is searched (people, activities, events, organizations, etc).

A class that is not in a class group **will not appear on the pick list for adding new individuals** on the Site Admin page or on the list of classes (by class group) on the Index page. To add an individual in a class not in a class group, navigate to the class in the class hierarchy and click the Add New Individual in this Class button.

To edit an existing class group

Site Admin > Class Management - Class Groups

- **Class Groups** screen - click on the class group you would like to edit > **Classgroup Editing Form**
- Change the **Class group name** of the class group as desired
- Change the **Display rank** of the class group to change where it displays in the **Class Groups** screen
- **Submit Changes > Class Groups** screen - edited class group should show up according to the display rank number assigned

To add a new class group

Site Admin > Class Management - Class Groups

- **Class Groups** screen - click on **Add new class group** button
- Add desired **Class group name**
- Change the **Display rank** of the class group from default (-1) to change where it displays in the **Class Groups** screen
- **Create New Record** > Class Groups screen - new class group should show up according to the display rank number assigned

Property Management

Properties

If classes define **what** each **individual** in VIVO is, properties define **how** that **individual relates** to other individuals and allow an **individual** to have attributes of its own.

Object properties connect two **individuals** (a **subject** and **object**) via a **predicate** functioning as the verb phrase in a sentence. For example, the verb phrase "is teacher of" or "teaches" connects a **subject** Person with the **object** Course in an RDFi statement in the same way as the verb connects the subject and the object in the sentence, "Professor Smith teaches Economics 101."

In contrast, a **data property** connects a single **subject** with some form of attribute data. While in theory every bit of information in VIVO (or any RDF database) could be modeled as a full-fledged **individual** member of a class, it is more convenient to store some information in simpler form using defined **datatypes** including string, integer, date, datetime, or boolean. The **objects** of data properties are not shared, nor can they be the **subjects** of other **statements**.

Both object and data properties are defined to have a **domain** class as a way to limit when they apply – if not specified, the most general class of **Thing** implicitly serves as the domain. For example, when a property such as **credit hours** is given a domain class of **Course**, any new **individual** entered into VIVO and typed as a **Course** may have a property listing its credit hours, while that property would not be offered as an option for an **individual** typed as a **Person**.

Object properties also have a **range** class that specifies the class membership of the **individuals** serving as **objects** of the property **predicate**. If no class is specified, the range class is implicitly considered to be **Thing**, meaning that any individual could serve as an object of the property.

Taken together, the **class hierarchy** and the **object properties** and **data properties** that have been defined in an ontology form a framework for adding **statements**. Ontologies can be reused in multiple applications based on different software and can be combined to serve additional purposes. Statements created to populate an ontology can be combined or exchanged among applications sharing the same ontology, independently of the software.

Note that it's possible to introduce inconsistencies in a semantic data model by changing the data or by changing the model after information has been entered. One of the advantages of a semantic web approach to data modeling is that reasoning can help detect inconsistencies when they occur, as well as reduce the likelihood of inconsistencies by limiting the choices made available when users are adding data.

Property Editing

VIVO has two property editors – one for object properties and another for data properties. **Data properties** are simpler to create and edit because only a domain class and a **range datatype** need to be selected (or left with their default values of **Thing** for the class and **untyped** for the datatype).

Object properties are need a **range class** specification instead of a range datatype, and additional decisions must be made about whether or not to create an **inverse property**.

Inverse properties

VIVO was designed to create and show object properties as pairs in opposite directions – from the subject to the object and vice-versa. These bi-directional relationships allow users to navigate from a person to a related department or grant while also supporting lists of department members on department pages or investigators on grant pages. This feels natural, is less work to maintain, and helps assure that an end user arriving at a VIVO page from a Google search can see and navigate easily to contextual information. Paired directional properties are complementary in meaning – if the Biology Department is **part of** its parent College of Life Sciences, the College **has part** Biology Department.

Object properties can be specified to be uni-directional, however, if the complement or inverse would make no sense or if creating and storing the inverse would provide no value to the application.

Symmetric properties

Object properties may also be defined as **symmetric**. A symmetric property asserts the same relationship in both directions, without any notion of complementary or inverse meaning. When two terms are defined as "related" there is no implication that one has precedence over the other in the relationship, whereas if one term is "derived from" another, the second term would have the different and complementary property called "has derivation."

Property Groups

Like class groups, **property groups** are a VIVO-specific extension to support using VIVO as a public website as well as an ontology and content editor. They are stored as RDF within the overall VIVO data model but as annotations on the model do not factor into any reasoning processes.

Property groups are a means to form informal groupings of object and data properties for the purposes of ordering page displays. As discussed in more detail under Verbose Property Listing, property groups have a display rank that determines precedence in page rendering, and object and data properties within a property group each carry a display rank affecting the order they are drawn within their group.

Object Property Hierarchy

Object properties represent "the glue" in the triples that make up VIVO; an object property connects any two entities (also known as items or individuals) in VIVO to complete the triple. Object properies can be created and edited from the **Object Property Hiearchy** screen or from the **Object Properties** alphabetical listing of all object properties. You need to be an admin or a curator to add or edit object properties.

Add New Object Properties

Site Admin > Ontology Editor - Property Management - Object Property Hierarchy

- Click **Add new object property** button > **Property Editing Form**
  For example: Let's create a new object property called knows by filling the following fields.
- **Parent property = none (a root property)**
- **Property Group = affiliation**
- **Display Level = public**
- **Update Level = public**
- **Ontology** = select the local ontology (see **Site Configuration > Ontology Editor > Ontology List**)
- **Local name for property = knows**
- **Label for public display = knows**
- **Inverse property ontology** = select the local ontology again
- **Local name for inverse property = knownBy**
- **Inverse property label for public display = known by**
- **Domain class** = Person
- **Range class = Person**

Fields below this on the form control specialized functionality and may be left blank or in their initial state.

- **Public Description** = Enter any explanatory message for users when adding or editing this property
- **display tier for this property** = leave blank for now. Once you have some examples populated, compare how the order of this property's display vis a vis other properties in the **affiliations** group and adjust this value as necessary to control property display order. You may also wish to move this property to another property group such as **contact** if you wish to have it appear lower down on a person's VIVO page.
- **display tier for this property's inverse** = leave blank for now. Note that an object property and its inverse may be in different groups as well as have different display rank values.
- **related object individuals to display without collapsing = 5.** You may have noticed that VIVO collapses long lists of related individuals to save vertical space on a page. This number controls how many are displayed before the rest of the list are collapsed. This setting is on a per-property basis, meaning that one person can't elect to show 3 friends and another 25.
- **sort related individuals by** = blank. Normally related individuals are sorted in alphabetical order, which works well for names of people and departments but also means that people's names are optimally entered last name first and departments should not all start with "Department of". Seminars may, however, be optimally sorted by the date and time they are to be offered, and news releases by their release dates.
- **sort direction** = blank. Enter "desc" to reverse the normal ascending sort order.
- **data property by which to sort related individuals = ~~select data property~~** The original VIVO options for sorting related entities were driven by a special, pre-defined set of data properties including the individual name, "timekey" (date and time for an event), "sunrise" (a creation date for the individual), and "sunset" (an end date for the individual). This set of options can now be expanded to any data property, and the select list includes a full list of data properties from any ontology in the VIVO system. Note that not all data properties will appear on editing forms due to domain restrictions on data properties and the operative range restrictions on the object property currently being edited. For our example we are dealing with the type Person, so picking a data property from the country names ontology as a sort property would make no sense.
- **transitive** = blank. Under OWL 2.0, properties may be declared transitive, meaning in our example that if person A knows B, and B knows C, A also knows C. This is not necessarily true, so do not check this box. At this point VIVO does nothing with this setting if checked except enable the setting to be passed to reasoners.
- **symmetric** = blank. This setting indicates that if the property is asserted in 1 direction, it should automatically be asserted in the other direction. For our example, if A knows B, B may or may not believe he or she knows A, so leave it unchecked.
- **functional** = blank. This setting would indicate that there can only be one statement of this object property for any one subject individual. VIVO does not yet enforce this setting, either through how it provides editing controls or by any kind of data validation.
- **inverse functional** = blank. This setting provides a further limitation on values by requiring that only one statement for this property be present for each subject individual, but also that the related individual be unique – no other individual could have a property statement with the same range object. VIVO does not yet enforce this setting either.
- **custom entry form** = blank. This is used to provide the name of a JSP file substituted by VIVO for the normal default object property editing form. It should only be populated by your VIVO system administrator, and any entry here not corresponding to a working JSP file in the correct directory on the local VIVO server will cause a system error.
- **delete object when statement deleted?** = blank. As noted on the form itself, this should be checked only if the implcations are fully understood, and refers to the deletion of dependent resources such as Educational Background entries that could only apply to one individual. **Warning: checking this box may cause unintended and unwanted data deletion!**
- **select from existing choices when adding statements?** = checked (this is the default behavior). For most object properties it is natural to offer a list of choices for the object individual based on the range class specified for the property above (or on a **property restriction** added only for members of a certain class). Occasionally this behavior is not desired, as with the educational background example just above, when one person should not pick any individual belonging to any other person. When this box is unchecked, the only option presented when adding a new value of this property is to create a new individual of the appropriate type specified as the range class of the property.
- **offer create option** = blank. This controls whether new individuals may be added when the users thinks no appropriate object individual yet exists, and triggers a picklist of the appropriate classes for object individuals. An example of when this would not be appropriate would be when selecting a country involved in an International Activity – since a standard ontology of country names has been added to VIVO, users should not be free to add additional countries.
- **sort related object individuals of inverse property by** = blank. See similar setting above. This is likely to go away in favor of simpler controls, since object properties and their inverses are now separately editable.
- **inverse sort direction** = blank. Also likely to go away for the same reason.
- **related object individuals of non-inverse property to display without collapsing = 5**. Also likely to go away for the same reason.
- **Example** = Enter a value if you wish to clarify the meaning of a property
- **Description for ontology editors** = Enter information here that will not be displayed to users but is visible to ontology editors and would be exported with any RDFi version of the ontology from this VIVO installation.

Save your work by clicking **Create New Record** and return to the **Object Property Control Panel** screen for the new object property created.

To return to the full list, select **Root Properties** for the **Object Property Hierarchy** screen or **See All Properties** for the **Object Properties** screen listing all object properties in alphabetical order by public display name.

Editing existing object properties

There are no differences between the object property creation form and the editing form, and the above instructions apply.

Data Property Hierarchy

A **data property** connects a single **subject individual** (e.g., a Person or Event) with some form of attribute data. These attributes can be untyped and have a language identifier or can be defined to adhere to one of a number of pre-defined international standard datatypes including string, integer, date, datetime, or boolean.

Data property statements have the same 3 parts as any other RDFi statement – the **subject** or **domain** (an **individual**), the **predicate** indicating which data property is involved, and a datatype value as the **object**. The objects of data properties are not shared, nor can they be the **subjects** of other **statements**.

When the object of a data property is set as **untyped** or **string**, the contents are free text. Other datatypes are constrained to accept values consistent with that datatype.

For example, a data property defined to accept only boolean datatypes allows only 'true' and 'false' and will reject 'yes', 'no', or any other value. Values defined as dates must conform to a standard data format (YYYY-MM-DD). VIVO has only recently begun enforcing these datatypes, and the editing screens do not yet provide calendar widgets or even very clear indications of the required value formats.

Data properies can be created and edited from the Data Property Hiearchy screen. You need to be an admin or a curator to use this screen.

Add New Data Properties

As an example, we will create a new local ID number data property.
All sites should create this new ID property, as well as a new Department ID property, for storing local identification numbers.

Site Admin > Ontology Editor - Property Management - Data Property Hierarchy

- Click the **Add new data property** button > **Datatype Property Editing Form**
- **Property Name**: create a new property that designates a local institution ID (i.e., UFID, WashID, UFDeptID, etc.
- **Property Group**=identity
- **Display Level**: this may vary depending on property. For person ID numbers, choose **self-authenticated** (available for viewing once the person has logged in) for department IDs choose **Curators** and **Admin**, since other users would typically have no need to see them.
- **Update Level**: this may vary depending on the property. For ID numbers we will choose "nobody" (ID numbers should be entered by data ingest from the controlled Human Resource system of record)
- **Ontology**: choose the local ontology (Florida, Cornell, WashU, Scripps, etc.) Adding a local ontology is described in the **Site Administration > Ontology List** section of this user guide.
- **Local Name**: "washId" or "ufId" or "ufDeptId" Should be in camel case – start with lowercase, capitalize new words within it, and not include spaces
- **Domain Class**=Person (Department IDs should use the Organization domain to be applicable across divisions, schools, centers, programs, and other units within universities)
- **Range Datatype**=string (unless your institution's ids are guaranteed to always be an integer value, in which case use "integer between -2147483648 and 2147483647")
- **Functional property (has at most one value for each individual)**= checked. This setting alerts the VIVO reasoner that only one value should be present for this property for any subject individual. Use the separate Display Limit control below to control whether users are presented the option of adding more than one value.
- **Example**=provide several examples of valid local ids for your institution
- **Description for ontology editors**=Provide any information as to why the datatype and range class are specified as they are
- **Public Description**=This text will be inserted at the top of the form users see when they edit an id value – make it clear what you expect the editor or curator to enter, such as "Enter this person's University of Florida ID number"
- **Display Tier**=This number controls the ordering of properties within one property group on a VIVO page. The property group's own display rank value controls ordering of groups. Lower numbers are drawn first.
- **Display Limit=1**. This is a VIVO-specific control that affect the user edit controls but not the logic in the reasoner. Set this to 1 for any property you wish to limit to a single value per subject individual, even if the property itself is not set to be **functional** as part of its definition for reasoning processes.
- custom entry form=blank. VIVO currently has no custom forms for data properties.

When all desired information has been entered, save the new data property using the **Create New Record** button. **The Datatype Property Control Panel** screen is shown with the new property as the active property.

Editing an existing data property

From the **Data Property Hierarchy** listing or the **Data Properties** alphabetical listing of all data properties, click on the desired property to edit. The **Datatype Property Control Panel** screen is displayed; click **Edit this Datatype Property** in the center column of the blue-shaded edit control box.

Property Groups

Property groups are a way to group similar object properties in VIVO.

To edit an existing property group

Site Admin > Property Management - Property Groups

- Click on property group to be edited > **Property Group Editing Form**
- **Property group name=groupname example**
  Public description: fill in as desired to provide a brief explanation of what the property group is intended for--to show up on the list.
- **Display rank**: if desired to change rank of property group display on the list
- **Create New Record** button > **Property Groups** (edited property group should be visible at appropriate display rank)

To add new property group

Site Admin > Property Management - Property Groups

- Click on **Add new property group** button > **Property Group Editing Form**
- **Property group name=groupname example**
- **Public description**: fill in as desired to provide a brief explanation of what the property group is intended for--to show up on the list.
- **Display rank**: if desired to change rank of property group display on the list
- **Create New Record** button > **Property Groups** (new property group should be visible at appropriate display rank)

Verbose Property Listing

Turning verbose property listing on will provide property information as an aid to specifying the desired order of appearance of what can become a large number of object and data properties appearing on a VIVO page.

Note that a user must be logged in and have editing privileges to have the option to turn on verbose property listing.

The order of properties is managed by simple integer rankings on property groups and on properties themselves, with lower numbers appearing first. Group rankings have higher precedence – all the properties included as members of the group ranked first will drawn before any of those in the group ranked second. Properties not included in any property group are drawn last (in the order specified by their own internal display rank value).
Within any property group, properties are drawn in the order specified, again with lower numbers appearing first.

Property groups may include both object properties and data properties.
Managing display order for properties across a whole VIVO installation can be tedious - we recommend using numbers in increments of 10 at first to allow inserting a few properties before or after the initial choices of rankings without having to renumber the entire list of properties.

When a user has activated the verbose property display option for a session, it makes all relevant annotations for the properties on a rendered VIVO page visible to that user. While this is ugly, it greatly facilitates diagnosing the settings interfering with the desired ordering and provides direct links to each property for changing the offending setting.

Note that this is a user-specific setting that does not persist between sessions. It must be turned on each time you log in.