Building Geodatabases with CASE Tools

GIS by ESRI™

Building geodatabases with CASE tools

IN THIS DOCUMENT

- **What are CASE tools?**
- **Designing the object model in Microsoft Visio**
- **Generating schema using CASE tools**

The geodatabase brings the physical representation of geographic features closer to their actual real-world counterparts. It is possible to create hydrants, mains, and valves and to define a number of characteristics for each, such as fields, validation rules, relationships, and subtypes. The Computer-Aided Software Engineering (CASE) tools subsystem lets you create blueprints of the structure of the geodatabase using a graphical language—the Unified Modeling Language (UML). Using class diagrams, you can represent geodatabase elements, such as feature datasets or geometric networks, and clearly see the relationships among them.

This document discusses how you can use Microsoft® Visio® to construct your UML and how you can use CASE tools in ESRI® ArcCatalog™ to generate the schema for your geodatabase.

What are CASE tools?

As discussed in *Building a Geodatabase*, there are three general strategies to creating geodatabases. The first two strategies—migrating existing databases to the geodatabase and using tools in ArcCatalog and ArcToolbox™ to create the schema for your geodatabase design—are discussed in *Building a Geodatabase*. This document will discuss the third strategy: using UML and the CASE tools subsystem of ArcGIS™ to generate the schema for your geodatabase.

UML models are created using specialized tools, such as Microsoft Visio or Rational Software Corporation's Rational Rose®, then exported to an intermediate format: the Microsoft Repository or an XML Metadata Interchange (XMI) file. The CASE tools will read the model and allow you to create the geodatabase schema and, optionally, generate code to define custom behavior.

XMI is an Object Management Group (OMG) standard that specifies how to store a UML model in an XML file. ArcGIS now reads models in XMI files as well as models stored in the Microsoft Repository.

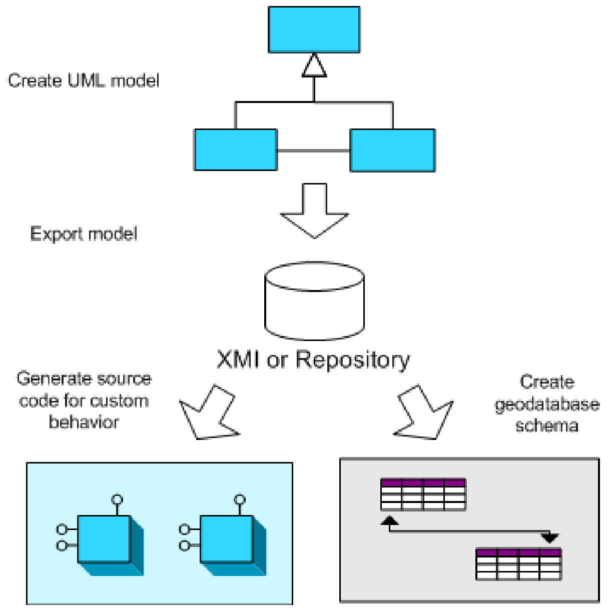
Using CASE for schema design and generation

The general strategy for using UML and CASE tools to design and create your geodatabase involves using UML to define all of the schema for the geodatabase, generating that schema, then populating the schema with data. The steps for accomplishing this are outlined below:

1. Create your geodatabase design in UML.
2. Export your UML model to XMI or Microsoft Repository.
3. Use the Schema Wizard in ArcCatalog to create the schema in your geodatabase from your UML model.

Once you have generated the schema, you may want to start directly editing that schema to build your database, but typically you will have existing data with which you want to populate that schema. There are a number of things that can impact performance when loading data into a geodatabase schema, especially when working with network data.

There is more than one strategy for loading data into an existing database schema. Each strategy has its limitations and affects performance of the database. The different strategies for loading your existing data into that schema and the performance considerations of each are outlined in *Building a Geodatabase*.


Create UML model
Export model
XMI or Repository
Generate source code for custom behavior
Create geodatabase schema

The general strategy for using UML and CASE tools to design and create your geodatabase involves using UML to define all of the schema for the geodatabase, generating that schema, then populating the schema with data.

Modeling database structure

Geodatabase elements, such as tables, feature classes, and relationship classes, follow certain rules that dictate where these elements are stored in the geodatabase relative to each other. For example, feature datasets are defined under the geodatabase, while geometric networks are defined inside feature datasets. Some elements, such as feature classes and relationship classes, can be inside or outside feature datasets.

These aspects of a geodatabase's structure are defined in UML through the use of packages to represent the geodatabase and feature datasets. The UML classes that are used to define feature classes, relationship classes, tables, and geometric networks are defined under these packages.

Tagged values

Tagged values are used to set additional properties of UML elements. For example, you can set the length (in characters) of a string field by using a tagged value.

Tagged values are recognized on several other UML elements: class, attribute, associations, and so on. The following table summarizes the tagged values used for geodatabase schema elements.

Table 1: Tagged values for geodatabase UML elements

Tagged value name	Values/Remarks
Fields:	
Precision	Integer value *Integer fields: number of digits* *Double fields: total number of digits*
Scale	Integer value *Number of decimal places in single and double fields*
Length	Integer value *Width of character fields*
AllowNulls	True/False
Alias	String *The alias name for the field*
Feature class/Object class:	
Geometry type	esriGeometryPoint esriGeometryPolygon esriGeometryPolyline esriGeometryMultipoint *Valid only for feature classes*
Ancillary role	esriNCARNone esriNCARSourceSink *Valid only for junction feature classes*
ConfigKeyword	String value

HasM	True/False *Valid only for feature classes*
HasZ	True/False *Valid only for feature classes*
CLSID	GUID in registry format *If specified, the Schema Wizard will look for a COM class identified by the CLSID in the system registry.* *If not specified, the appropriate ESRI COM class will be used instead. This tagged value will be overwritten by the Code Generation Wizard if code is generated for the custom feature.* *This tagged value can be specified for UML classes representing class extensions as well.*
Alias	String *The alias name for the field*

Geometric network:

GNConfigKeyword	String

Relationship class:

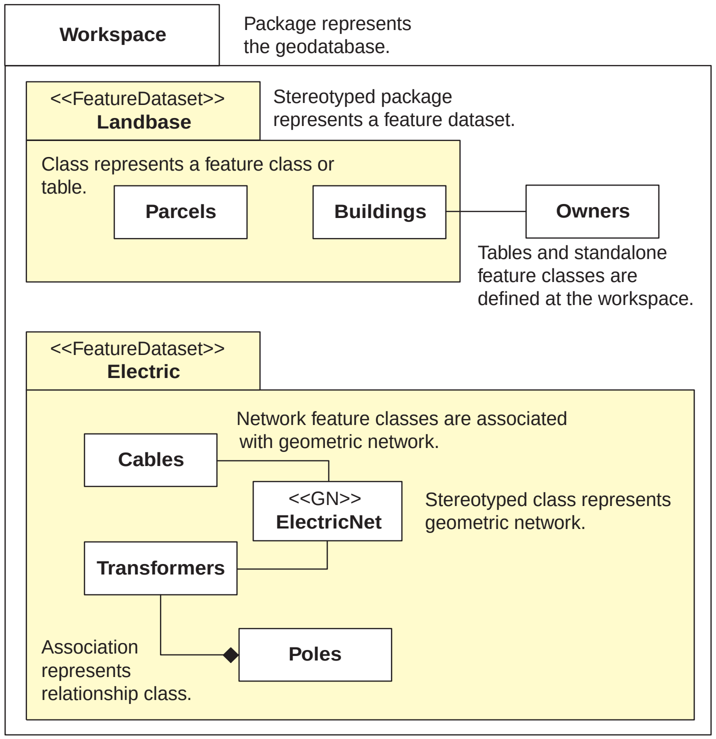
Notification	esriRelNotificationBackward esriRelNotificationBoth esriRelNotificationForward esriRelNotificationNone
IsAttributed	True/False
OriginClass	Name of the origin class
OriginPrimaryKey	Name of the primary key field of the origin class
OriginForeignKey	Name of the foreign key field of the origin class *For 1-1 and 1-M relationship classes, this field lives in the destination class. For 1-M/attributed relationship classes, this field lives in the auxiliary <<RelationshipClass>> class.*
DestinationPrimaryKey	Name of the primary key field of the destination class *Valid for M-M/attributed relationship classes only.*
DestinationForeignKey	Name of the foreign key field of the destination class *Valid for M-M/attributed relationship classes only. This field lives in the auxiliary <<RelationshipClass>> class.*

Domain:

Description	String value

Feature datasets

Feature datasets are modeled in UML as stereotyped packages that you create under the geodatabase workspace package. A feature dataset package cannot be created under another feature dataset package; however, other packages created for organizational purposes can. For example, a package that holds all of the subtypes for a particular class can be created under a feature dataset package.



Workspace
Package represents the geodatabase.
<<FeatureDataset>> Landbase
Stereotyped package represents a feature dataset.
Class represents a feature class or table.
Parcels
Buildings
Owners
Tables and standalone feature classes are defined at the workspace.
<<FeatureDataset>> Electric
Network feature classes are associated with geometric network.
Cables
<<GN>> ElectricNet
Stereotyped class represents geometric network.
Transformers
Poles
Association represents relationship class.

Geodatabase elements, such as tables, feature classes, and relationship classes, follow certain rules that dictate where these elements are stored in the geodatabase relative to each other. These aspects of a geodatabase's structure are defined in UML through the use of packages to represent the geodatabase and feature datasets.

Feature datasets have a spatial reference associated with them. Spatial references are not modeled in UML. Instead, the spatial reference for a feature dataset is set when generating the schema in ArcCatalog.

Tables, feature classes, and geometric networks

UML classes are used to model feature classes and tables. When schema is generated from the UML model, one table or feature class is created for each class in the model. Each property of the object is mapped to a field of the table or feature class. Required fields are included as properties of the base class and need not be repeated in any inherited classes. For example, you may have a class called Pipes that has the properties Material and Diameter. If you then create a new class called Mains that is inherited from Pipes, then Material and Diameter are also properties of the Mains class but do not need to be repeated in the Mains class in the UML model.

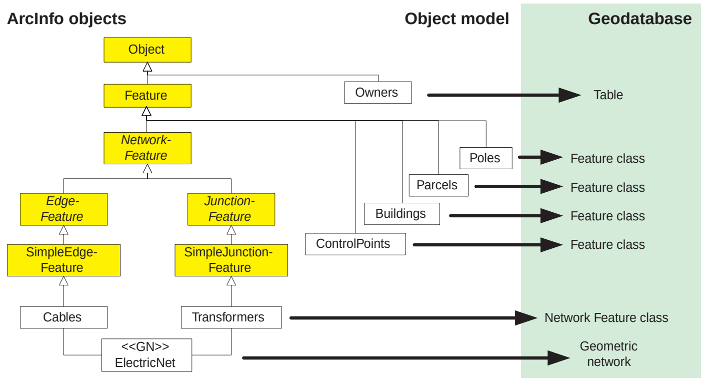
When schema is generated, each property is mapped as a field on the table or feature class. You can specify the length, scale, and precision of the fields that correspond with these properties in the UML diagram by setting tagged values. You can also use tagged values to set properties for feature classes, class extensions, relationship classes, and interfaces.

When generating schema, all feature classes the wizard creates use the coordinate system of the target feature dataset. Object classes (tables) are created at the workspace level.

The default grid size is 1,000. Grid size is not stored as part of the model because the coordinate system of target feature datasets might be different. You can specify a grid size for each feature class in the Schema Wizard.

Tables are created for those objects that inherit from the Object class, feature classes are created for those that inherit from the Feature class, and simple or complex network feature classes are created for those that inherit from the Network feature classes.

All network feature classes must be associated with a geometric network. Geometric networks (GN) are modeled in UML as special classes. The geometric network and its associated network feature classes must be created in the same feature dataset package. A junction feature class is created for all features that inherit from Simple Junction or Complex Junction. For more information on geometric networks, see *Building a Geodatabase*.

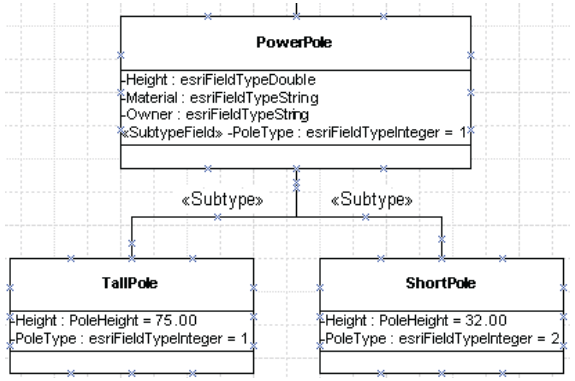


Tables are created for objects that inherit from the Object class, feature classes are created for objects that inherit from the Feature class, and network feature classes are created for objects that inherit from the Network feature class.

Subtypes and domains

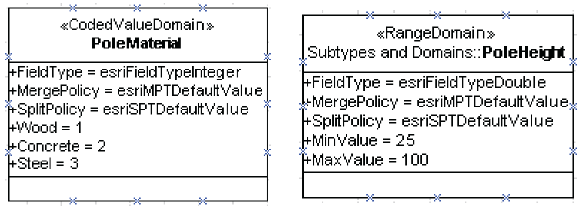
Attribute domains are modeled in UML as special classes. When the CASE tool generates the schema from the UML model, these domain classes are stored in the geodatabase as domains. Coded value and range domains, along with their valid values, split policies, and merge policies, are all modeled in this way.

If your tables or feature classes require subtypes, these can also be modeled in UML and automatically generated when the tables or feature classes are created in the database. Subtypes are modeled in UML as classes related to the parent class through an association stereotyped as Subtype. The Subtype field is specified in the parent class as a stereotype of a UML attribute.

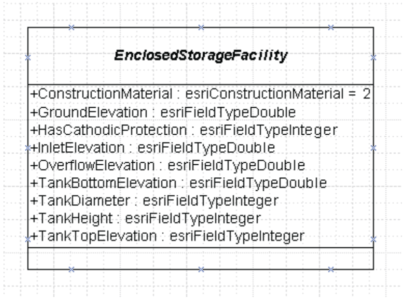

PowerPole
-Height : esriFieldTypeDouble
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
«SubtypeField» -PoleType : esriFieldTypeInteger = 1
«Subtype»
«Subtype»
TallPole
-Height : PoleHeight = 75.00
-PoleType : esriFieldTypeInteger = 1
ShortPole
-Height : PoleHeight = 32.00
-PoleType : esriFieldTypeInteger = 2

Subtypes are modeled in UML as classes related to the parent class through an association stereotyped as Subtype. You can specify domains and default values for the fields for that subtype.

You can specify default values and domains for each field for a particular feature class or subtype. To associate a domain with a particular field, simply specify the name of the domain as the field type. Similarly, the field's default value can be supplied for feature classes or subtypes by setting the initial value in the UML attribute representing the field.


«CodedValueDomain»
PoleMaterial
+FieldType = esriFieldTypeInteger
+MergePolicy = esriMPTDefaultValue
+SplitPolicy = esriSPTDefaultValue
+Wood = 1
+Concrete = 2
+Steel = 3
«RangeDomain»
Subtypes and Domains::PoleHeight
+FieldType = esriFieldTypeDouble
+MergePolicy = esriMPTDefaultValue
+SplitPolicy = esriSPTDefaultValue
+MinValue = 25
+MaxValue = 100

Attribute domains are modeled in UML as special classes.


EnclosedStorageFacility
+ConstructionMaterial : esriConstructionMaterial = 2
+GroundElevation : esriFieldTypeDouble
+HasCathodicProtection : esriFieldTypeInteger
+InletElevation : esriFieldTypeDouble
+OverflowElevation : esriFieldTypeDouble
+TankBottomElevation : esriFieldTypeDouble
+TankDiameter : esriFieldTypeInteger
+TankHeight : esriFieldTypeInteger
+TankTopElevation : esriFieldTypeInteger

If the feature class or table you are modeling does not have subtypes, but you still want to associate domains and default values to fields, you can do so directly on the class object itself.

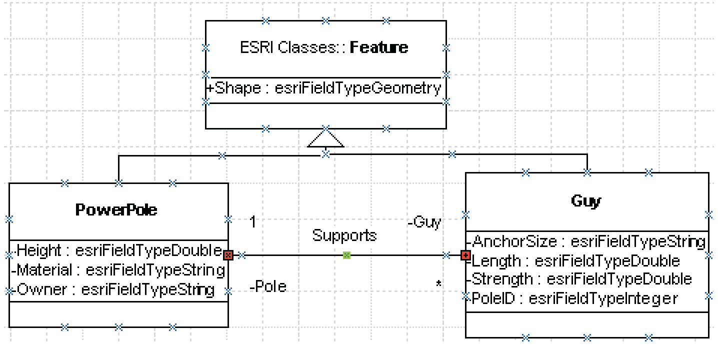
Relationships

Associations between objects in your UML model are created in the geodatabase as relationship classes. The cardinality of those associations is reflected in the cardinality of the relationship class. The name of the relationship class is the name of the association. The primary and foreign keys are specified directly in the UML model as tagged values of the UML association.

Attributed *relationships* are modeled as classes with the same name as the relationship class, stereotyped as a relationship class. The fields for the relationship class's attributes are modeled in the same way as the attributes of any other class.

Notification direction—the direction messages are passed—for the relationship class can be modeled in UML using tagged values.

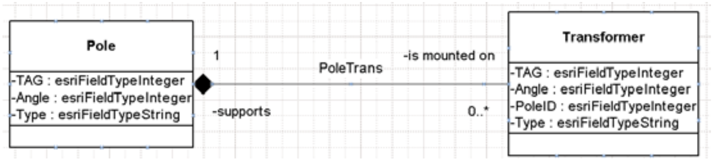
Relationship rules apply to subtypes in feature classes and tables. In UML, these relationship rules are modeled as associations between subtypes of the classes participating in the relationship class.


ESRI Classes:: Feature
+Shape : esriFieldTypeGeometry
PowerPole
-Height : esriFieldTypeDouble
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
1
Supports
-Guy
-Pole
*
Guy
-AnchorSize : esriFieldTypeString
-Length : esriFieldTypeDouble
-Strength : esriFieldTypeDouble
-PoleID : esriFieldTypeInteger

Relationship classes are modeled in UML as associations. The cardinality of the association is reflected as the cardinality of the relationship class. Attributed relationships are modeled as classes with the same name as the relationship class.

Composite relationships are modeled in UML as *aggregation*. In a composite relationship, the origin class is always the parent in the aggregation. Composite relationship classes always have a one-to-many relationship.

To learn more about relationship classes, see *Building a Geodatabase*.

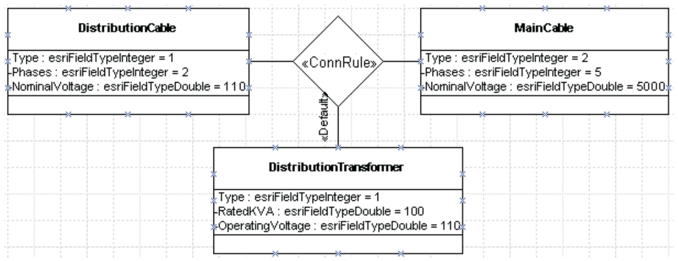

Pole
-TAG : esriFieldTypeInteger
-Angle : esriFieldTypeInteger
-Type : esriFieldTypeString
1
PoleTrans
-is mounted on
-supports
0..*
Transformer
-TAG : esriFieldTypeInteger
-Angle : esriFieldTypeInteger
-PoleID : esriFieldTypeInteger
-Type : esriFieldTypeString

Composite relationship classes are modeled in UML as aggregations between classes.

Connectivity rules

Connectivity rules apply to network feature classes and subtypes of network features participating in the same geometric network. These connectivity rules are modeled in UML as a special association stereotyped as *ConnRule*.

For *edge–junction connectivity rules,* the ConnRule association is between the edge subtype and the junction subtype. The junction in one of the edge–junction rules can be set as the default junction by stereotyping the association end as Default. *Edge–Edge rules* are represented by a UML N-ary association that involves two edge subtypes and any number of junction subtypes. One of the junction subtypes must be marked as the *default junction* by stereotyping the association end as Default.


DistributionCable
Type : esriFieldTypeInteger = 1
Phases : esriFieldTypeInteger = 2
NominalVoltage : esriFieldTypeDouble = 110
«ConnRule»
MainCable
Type : esriFieldTypeInteger = 2
Phases : esriFieldTypeInteger = 5
NominalVoltage : esriFieldTypeDouble = 5000
«Default»
DistributionTransformer
Type : esriFieldTypeInteger = 1
RatedKVA : esriFieldTypeDouble = 100
OperatingVoltage : esriFieldTypeDouble = 110

Edge–Edge rules are represented by a UML N-ary association. One of the junction classes or subtypes must be marked as the default junction by stereotyping the association end as Default.

CASE tools

The CASE tools subsystem of ArcGIS 8 has two parts: the Code Generation Wizard and the Schema Wizard. The remainder of this document discusses how you can create each component of your geodatabase schema in UML using Visio Enterprise and how you can use the CASE tools in ArcCatalog to generate the schema for your UML design.

For more information on the ESRI object model and on generating code for your custom objects using the Code Generation Wizard, see *Modeling Our World* and *Exploring ArcObjects*.

The ArcInfo UML Model diagram

When you are ready to begin creating your UML model, you will start with one of the ArcInfo™ UML Model diagrams that were installed with ArcGIS. These diagrams are Visio Drawing templates—ArcInfo UML Model(Ent).vst for Visio Enterprise or ArcInfo UML Model (Pro).vst for Visio Professional. These Visio Drawing templates are located under your ArcGIS installation in the casetools\UML Models directory.

The ArcInfo UML Model diagram contains the object model required for using UML to model your geodatabase. The object model has five packages:

- Logical View
- ESRI Classes
- ESRI Interfaces
- ESRI Network
- Workspace

These UML packages act as directories where different parts of the entire object model are maintained. The Logical View package is the root level and contains the other three packages. Database designers and developers can use the Workspace package to create their object and database designs. It is possible to create more packages if the complexity of the model requires you to do so.

The ESRI Classes package contains the portion of the GeoData Access Components necessary to create object models. Classes in this package represent components that are used to access spatial data sources, including geodatabases. Feature classes and object classes in your object models will inherit from these classes. The ESRI Interfaces package contains the definition of the interfaces implemented by the components shown in the ESRI Classes package. The interfaces are used only for code generation when creating custom objects.

The tasks in this document demonstrate how you can use the ArcInfo UML Model diagram to model the pieces of your geodatabase design. All examples given are for Visio Professional.

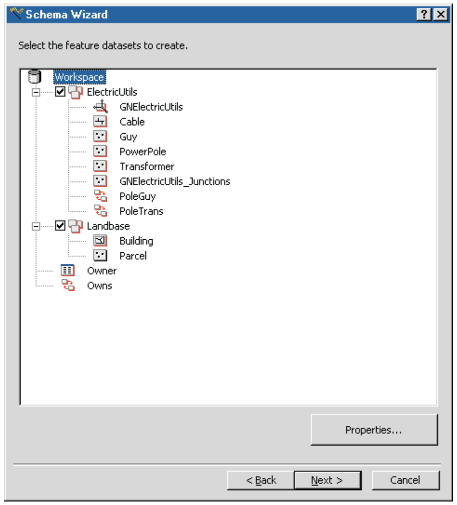
Semantics checker

The CASE tools expect UML models to be created following a set of modeling rules. For example, a network feature class must be associated to a geometric network. The semantics checker can be used to verify that a model stored in the Microsoft Repository or XMI has been correctly defined. It will produce a report with the list of errors encountered in the model. You should use the semantics checker before running the CASE tools wizards. You can run the semantics checker from within Visio when the template diagram is loaded.

Applying your model to existing data

You can update the schema of a geodatabase with information stored in an object model. For example, you can import data to a geodatabase, then apply a model to add subtypes, relationships, and other elements. Alternatively, your current schema could have been created previously with the Schema Wizard based on a model. Since then, the model might have changed, and you may want to update the database.

You can use the Schema Wizard to modify an existing geodatabase, even if data has already been loaded. To do so, select the target geodatabase and run the Schema Wizard. When the wizard starts, some elements in the model are searched for and matched to objects in the database—feature classes and tables, for example. If found, the wizard will modify them. The Schema Wizard will show the matched elements with a red shadow in the tree view.


Schema Wizard

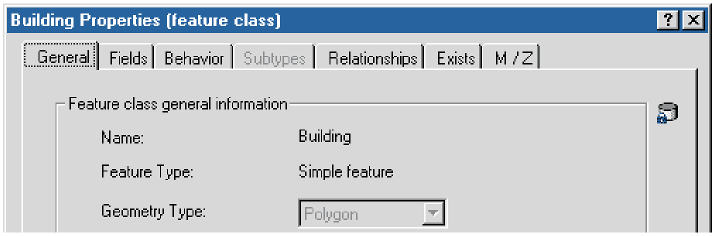
Select the feature datasets to create.

Workspace
- ElectricUtils
  - GNElectricUtils
  - Cable
  - Guy
  - PowerPole
  - Transformer
  - GNElectricUtils_Junctions
  - PoleGuy
  - PoleTrans
- Landbase
  - Building
  - Parcel
- Owner
- Owns

The Schema Wizard will show matched elements with a red shadow in the tree view.

For a number of reasons, your database may contain feature classes and fields whose names do not correspond exactly to the names in the UML model. Perhaps you have applications written previously that rely on the existing field's name being different. For example, a field could be named "Hgt" in the database and "Height" in the model. The geodatabase lets you assign a model name to feature classes and fields, and the CASE tools use the model name when matching them to UML elements. Some properties of the matched object may be read-only—for example, the spatial reference of an existing feature dataset.

Conversely, other properties of the existing object will be changed based on the information in the model, such as the domain assigned to a field. In the former case, a locked database icon will appear in the form or tab displaying the particular property. In the latter case, a model icon will be used instead.

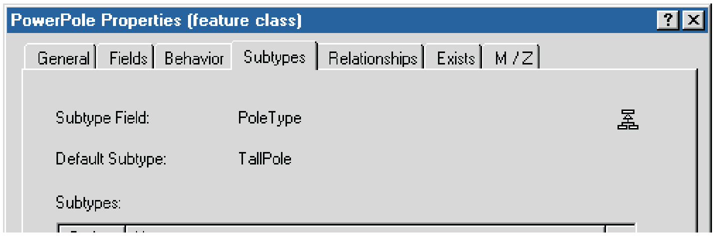


Feature class general information

Name: Building

Feature Type: Simple feature

Geometry Type: Polygon



Subtype Field: PoleType

Default Subtype: TallPole

Subtypes:

Those properties that cannot be changed when reapplying the model will have a locked database icon on their properties dialog box, while those that can be modified will have a model icon.

Since the schema of matched elements is modified when a model is applied, exclusive schema locks are acquired for the elements that will be modified. These locks can be established only if there are no other users connected to the database and the current user has the right permissions. Since the schema is modified, you should back up your database before applying the changes.

Each geodatabase element supported by the Schema Wizard follows a set of rules when the model is reapplied. The following is a description of these rules:

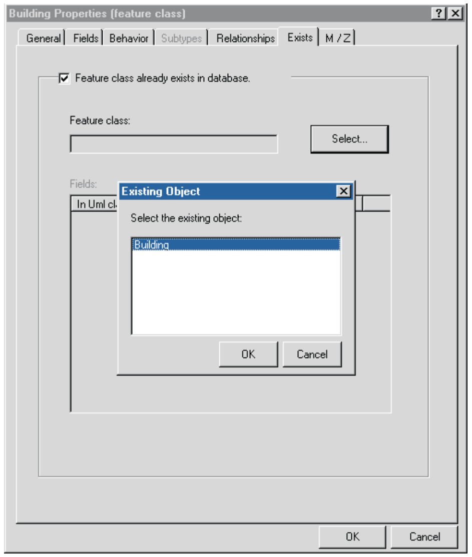
Feature datasets: The Schema Wizard uses the name of the UML package representing the feature dataset to search for an existing feature dataset in the target database. The spatial reference and spatial domain properties become read-only.

Other model elements are defined inside feature datasets, such as geometric networks and feature classes. These elements will be found in the existing database only if the feature dataset itself is found.

Feature classes: The name of the UML class representing the table or feature class will be used to search for an existing object in the geodatabase. The comparison is first made against the model names of the existing feature classes. If no match is found, the comparison is made against the names of the existing feature classes. If still no match is found, you can manually set the match using the Exists tab in the feature class's properties dialog box.

Only feature classes with the same feature type can be used when matching manually. Also, only feature classes in the same location are available for matching—for example, the feature classes under the matched feature dataset.

When the Schema Wizard is run, it updates the model name of both existing and new feature classes and fields. This ensures the matching will occur automatically the next time the model is applied.


Building Properties (feature class)

General | Fields | Behavior | Subtypes | Relationships | Exists | M / Z

☑ Feature class already exists in database.

Feature class:

Fields:

In Uml cl

Existing Object

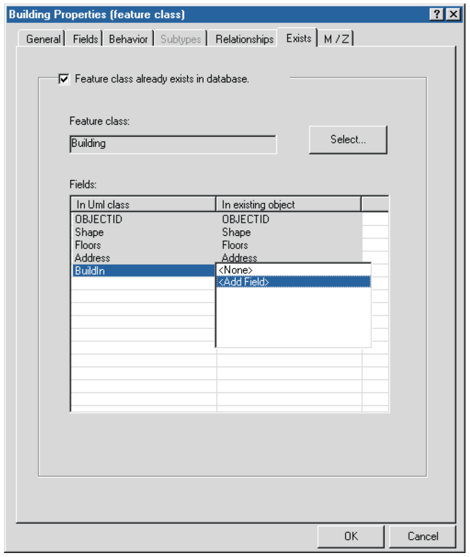
Select the existing object:

Building

If no match between the model and the database is found for a feature class or table, you can manually set the match using the Exists tab in the feature class's properties dialog box.

Fields: Like feature classes, fields are matched based on model name, and a manual match can be done. The UML class may also contain additional fields. These can be marked for addition in the Exists tab of the feature class properties dialog box. Existing fields not matched are left untouched. Reapplying the model to existing data does not drop fields.

For matched fields, the field type, length, precision, and scale are read-only properties. However, the domain and initial value will be updated if they have been set in the model (the domain must have the same field type).


Building Properties (feature class)

General | Fields | Behavior | Subtypes | Relationships | Exists | M / Z

☑ Feature class already exists in database.

Feature class: Building

Fields:

In Uml class	In existing object
OBJECTID	OBJECTID
Shape	Shape
Floors	Floors
Address	Address
BuildIn	<None>
	<Add Field>

If the UML class contains additional fields, these can be marked for addition in the Exists tab of the feature class properties dialog box.

Domains: Domains are matched based on the name of the UML class and the existing domain. If found, the domain will be altered with modifications made to the UML class. For example, new codes can be added to a coded value domain.

Subtypes: Subtypes of matched feature classes are deleted and re-created using the subtypes in the model. Because connectivity and relationship rules are created among subtypes, they are deleted and re-created as well. Whenever you use ArcCatalog to add a subtype or rule, you should make sure the model is updated accordingly in case you ever reapply it.

Relationship classes: The name of the relationship class in the model will be used to search for an existing relationship class in the target database. Existing relationship classes that are not matched will be left untouched. The policies on modifying vary, depending on the type of relationship class.

Nonattributed relationship classes are deleted and re-created. Along with the relationship class, the relationship rules are deleted and re-created as well.

Internally, attributed relationship classes are implemented by creating an extra table in the database. This table holds attributes of the relationships and keys to the rows in the related tables. Because this table may be holding data already, only the relationship class's rules, not the relationship class itself, are deleted and re-created.

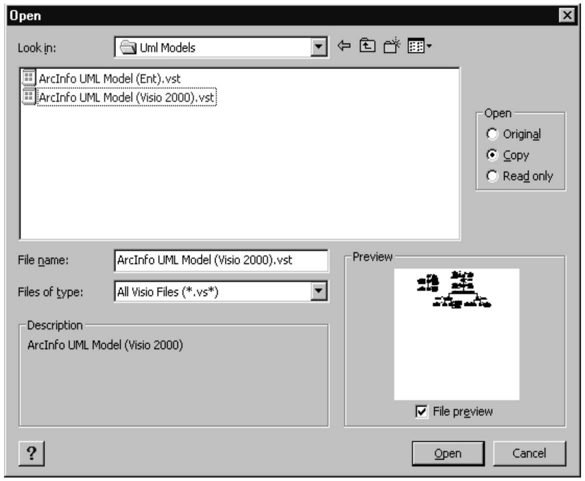
Geometric networks: The name of the class representing the network will be used to search for an existing geometric network in the target database. As mentioned before, a geometric network is matched only if the feature dataset it belongs to is matched as well. All existing connectivity rules are deleted and re-created.

Designing the object model in Microsoft Visio

This document discusses how to create models using Microsoft Visio and how you can use CASE tools in ArcCatalog to generate the schema for your geodatabase. You will learn how to:

- Create UML packages and static structure diagrams.
- Create feature datasets.
- Create feature classes.
- Create relationship classes.
- Create domains.
- Create subtypes.
- Create geometric networks.
- Create connectivity rules.
- Extend classes with custom behavior.
- Export your UML model.
- Generate schema.

1. Start Visio.
2. Click File and click Open.
3. Browse to C:\arcgis\arcexe82\CaseTools\UML Models, which is the default installation path, and double-click the ArcInfo UML Model template file.


Open

Look in: Uml Models

ArcInfo UML Model (Ent).vst
ArcInfo UML Model (Visio 2000).vst

Open: Original [✓ Copy] Read only

File name: ArcInfo UML Model (Visio 2000).vst

Files of type: All Visio Files (*.vs*)

Description: ArcInfo UML Model (Visio 2000)

Preview

[✓] File preview

Creating UML packages and static structure diagrams

Packages are a convenient way to organize your UML model. They act as folders where you can group model elements. Just like with folders on a disk, you can create a hierarchy of packages in your model.

You can create as many packages as you want. For example, you could have a model that has the following packages: Domains, ElectricUtils, and Landbase.

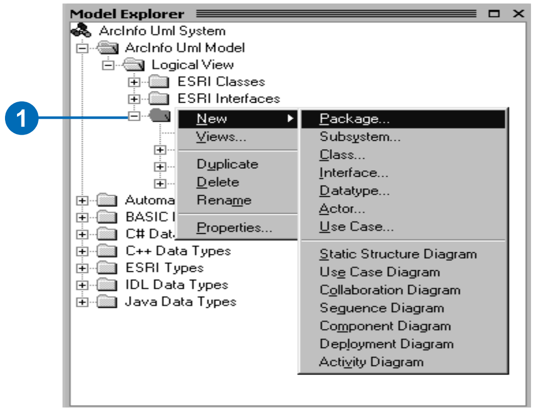
Before you begin, create a new diagram using the ArcInfo UML Model Visio templates. These templates are located under your ArcGIS installation in the casetools\UML Models directory.

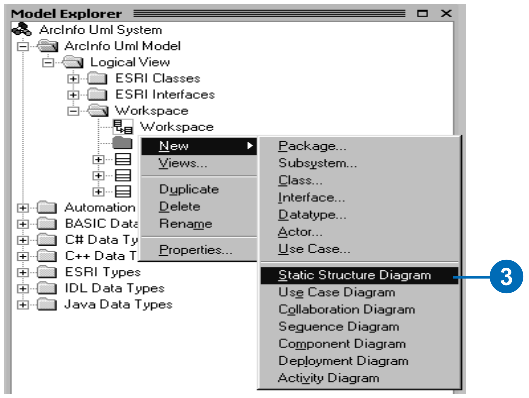
All of the tasks for creating UML diagrams that are discussed here are performed within the Visio application.

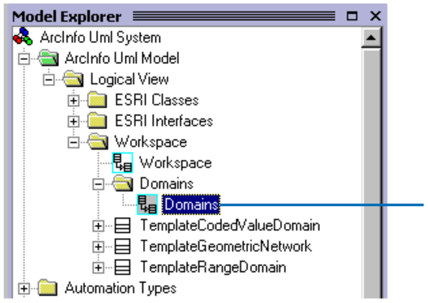
See Also

Special packages are used to model feature datasets. See 'Creating feature datasets' later in this document.

1. In the Model Explorer tree, right-click Workspace, point to New, then click Package.
2. Type the name of the package and click OK.
3. Right-click the new package. Point to New and click Static Structure Diagram to create a new class diagram.







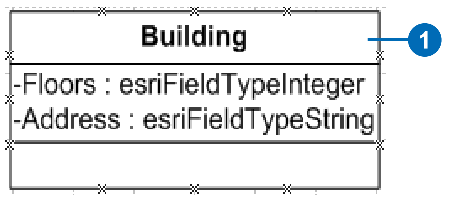
A new class diagram is created in the Model Explorer.

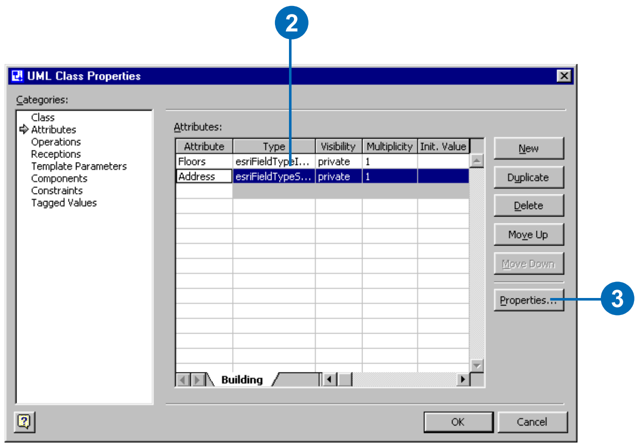
Setting tagged values

Tagged values are used to set additional properties of UML elements. For example, you can set the length (in characters) of a string field by using a tagged value.

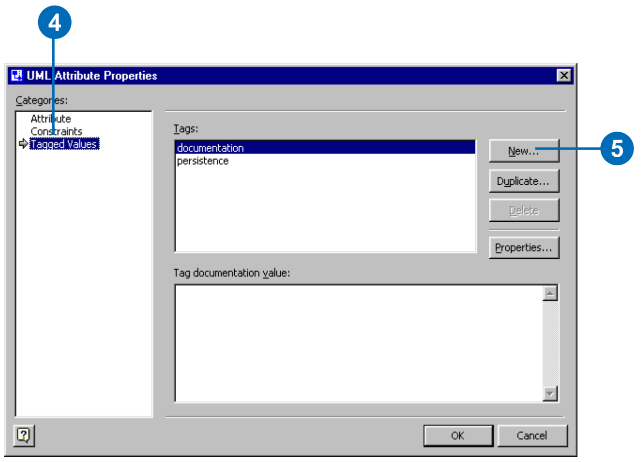
Tagged values are recognized on several other UML elements: class, attribute, associations, and so on.

1. In the Visio diagram, double-click the UML class.
2. Click Attributes in the Categories window and double-click a string-typed attribute.
3. Click Properties.
4. Click Tagged Values in the Categories window.
5. Click New to create a new tagged value. ▸

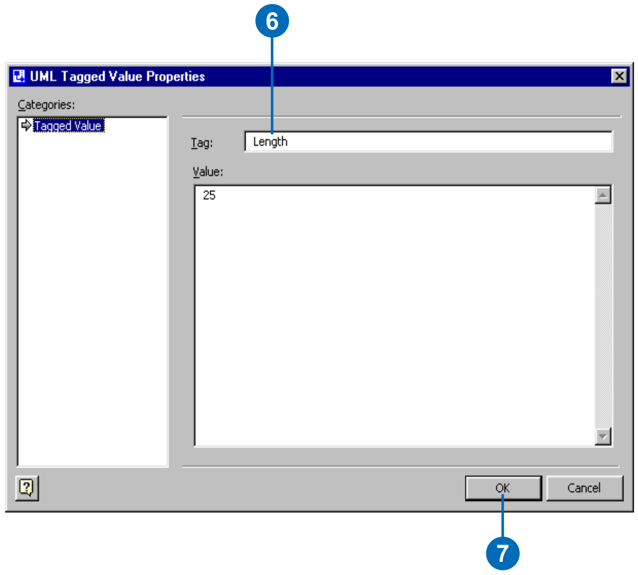

Building
-Floors : esriFieldTypeInteger
-Address : esriFieldTypeString
1



Attribute	Type	Visibility	Multiplicity	Init. Value
Floors	esriFieldTypeI…	private	1	
Address	esriFieldTypeS…	private	1	



6. Type “Length” in the Tag text box and type the length of the field in the Value text box.
7. Click OK.



Tag: Length

Value: 25

Creating feature datasets

Feature datasets are modeled in UML as stereotyped packages. Other geodatabase elements (feature classes, for example) defined under the package will be created under the feature dataset. The name of the UML package will become the name of the feature dataset.

The spatial reference of a feature dataset is set while running the Schema Wizard.

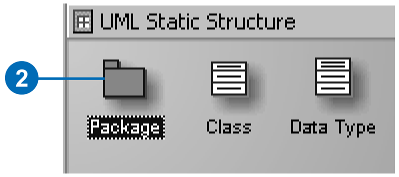
Tip

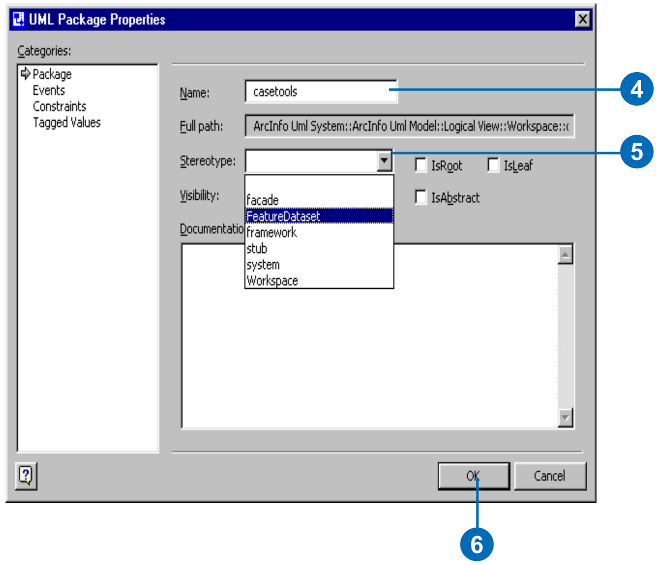
Class diagrams

Creating several static structure diagrams can help you reduce diagram cluttering. A package can contain many static structure diagrams, and UML elements can appear in any diagram.

1. In the Model Explorer, double-click the Workspace diagram to open it.
2. In the UML Static Structure stencil, click Package and drag it onto the diagram.
3. Double-click the new package.
4. Type a name for the package.
5. Click the Stereotype dropdown arrow and click FeatureDataset.
6. Click OK.

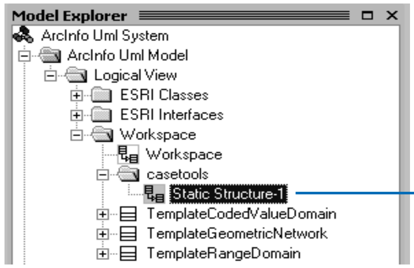
   A new package and a new drawing are created in the Model Explorer.





Name: casetools

Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::c



A new package and a new drawing are created in the Model Explorer.

Creating feature classes

Feature classes are represented by UML classes in the model. You can model fields, geometry type, and other characteristics of the feature class in the UML class.

Feature classes can be created in the Workspace UML package or in a feature dataset package.

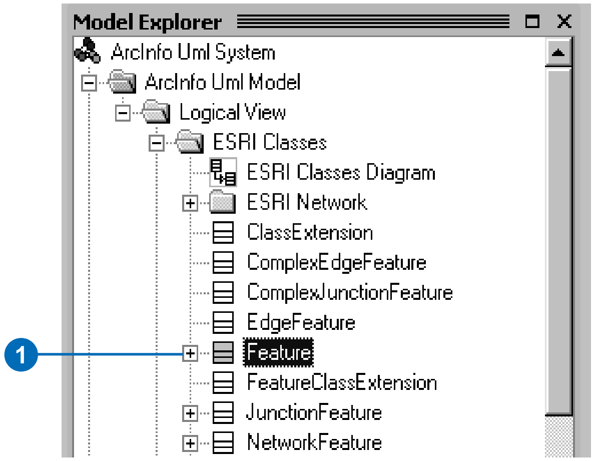
You can add fields to a feature class by adding attributes to the UML class.

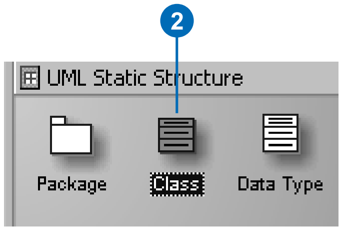
The field type is one of the values of the esriFieldType enumeration—for example, esriFieldTypeInteger.

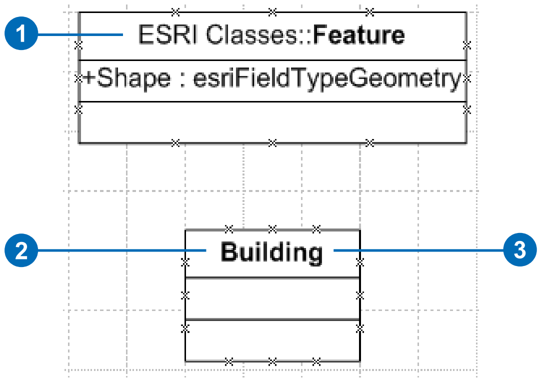
A domain created beforehand can be used as the attribute type as well.

Creating a feature class

1. In the Model Explorer tree under ESRI Classes, click the parent class and drag and drop it onto the diagram.
2. In the UML Static Structure stencil, click Class and drag and drop a new UML class onto the diagram.
3. In the diagram, double-click the new class. ▸







Tip

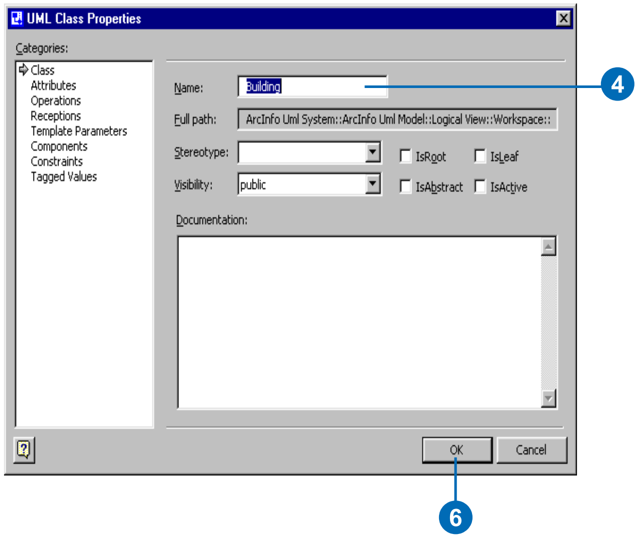
Abstract classes

Your model may include abstract classes. Fields in abstract classes are inherited by the class descendants. Check the IsAbstract check box on the Class tab on the Class Properties dialog box to mark a class as abstract.

See Also

Tagged values allow you to fully specify UML elements in the model. See 'Setting tagged values' in this document.

4. Type the name of the feature class.
5. To set tagged values, follow steps 2 through 6 of 'Setting tagged values' in this document; otherwise, skip to step 6.
6. Click OK to accept the changes.
7. In the UML Static Structure stencil, click Generalization and drag and drop it onto the diagram.
8. Drag the ends of the generalization arrow and connect the new class with its parent.


UML Class Properties

Categories:
- Class
- Attributes
- Operations
- Receptions
- Template Parameters
- Components
- Constraints
- Tagged Values

Name: Building

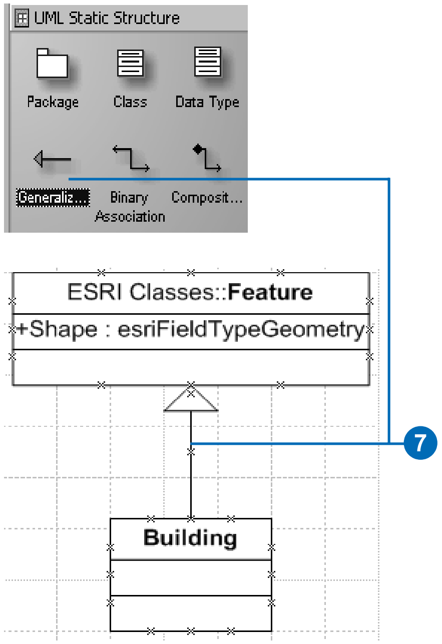
Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::

Stereotype:

Visibility: public

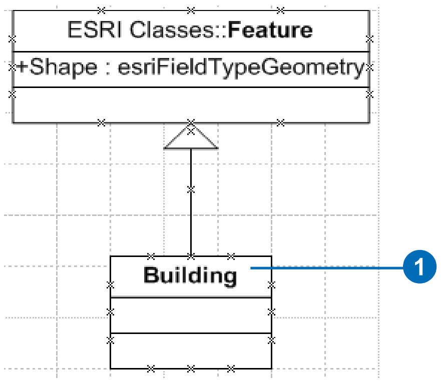
IsRoot, IsLeaf, IsAbstract, IsActive

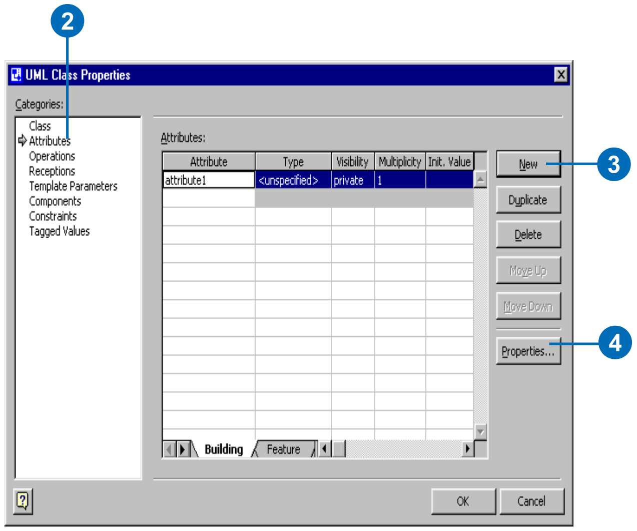
Documentation:



Adding fields to a feature class

1. In the diagram, double-click the class.
2. Click Attributes in the Categories window and double-click a string-typed attribute.
3. Click New to add a new attribute.
4. Click Properties to edit the field property. ▸





Attribute	Type	Visibility	Multiplicity	Init. Value
attribute1	<unspecified>	private	1	

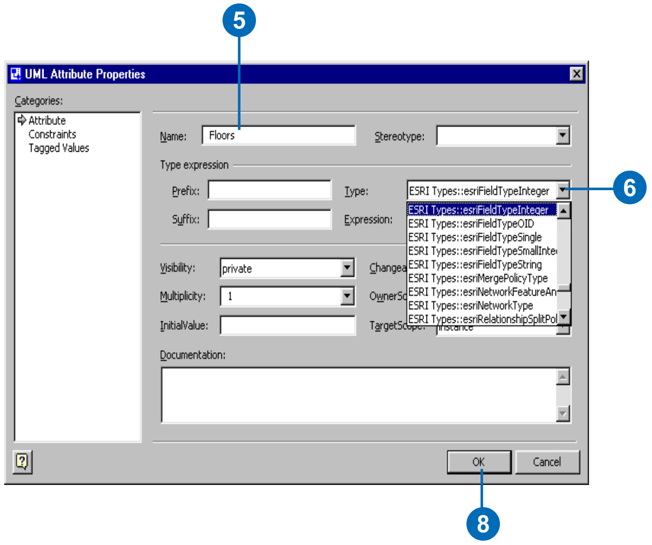
Tip

Default values

You can set the default for a field by entering the value in the Initial Value text box on the Attribute tab of the UML Property Editor dialog box. Default values are optional.

5. Type a name for the new field.
6. Click the Type dropdown arrow and click the field type.
7. To set tagged values, follow steps 2 through 6 of ‘Setting tagged values’ in this document; otherwise, skip to step 8.
8. Click OK.
9. Repeat steps 3 through 8 until you have added all the fields for your new class.

   The fields appear in the diagram as attributes of the class.


UML Attribute Properties

Categories: Attribute, Constraints, Tagged Values

Name: Floors

Stereotype:

Type expression

Prefix:

Suffix:

Type: ESRI Types::esriFieldTypeInteger

Expression:

ESRI Types::esriFieldTypeInteger
ESRI Types::esriFieldTypeOID
ESRI Types::esriFieldTypeSingle
ESRI Types::esriFieldTypeSmallInte
ESRI Types::esriFieldTypeString
ESRI Types::esriMergePolicyType
ESRI Types::esriNetworkFeatureAn
ESRI Types::esriNetworkType
ESRI Types::esriRelationshipSplitPol

Visibility: private

Multiplicity: 1

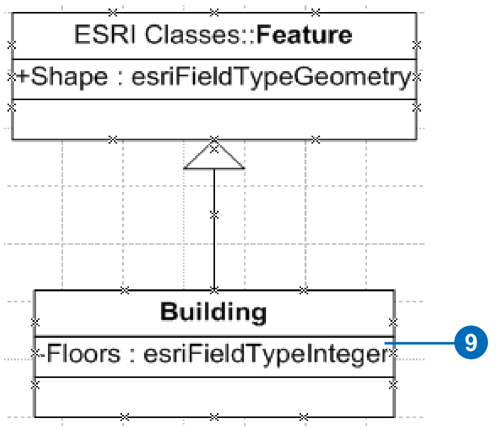
InitialValue:

Changea

OwnerSc

TargetScope: instance

Documentation:


ESRI Classes::Feature
+Shape : esriFieldTypeGeometry
Building
-Floors : esriFieldTypeInteger
9

Creating relationship classes

UML associations represent relationship classes among feature classes and object classes (tables).

Relationship classes can be created only among leaf classes. Primary and foreign key fields must be present in the classes.

Tagged values for relationship classes include:

- OriginClass: the name of the origin class

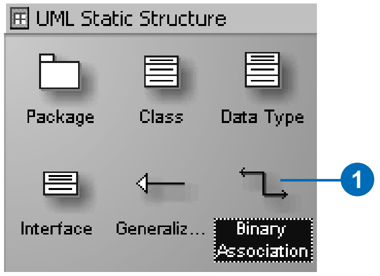
  Example: PowerPole
- OriginPrimaryKey: the name of the primary key field in the origin class

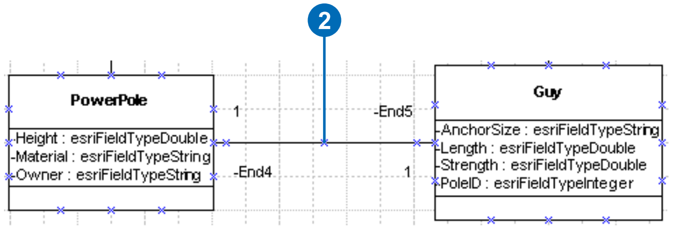
  Example: OriginPrimaryKey=OBJECTID
- OriginForeignKey: the name of the foreign key field in the destination class

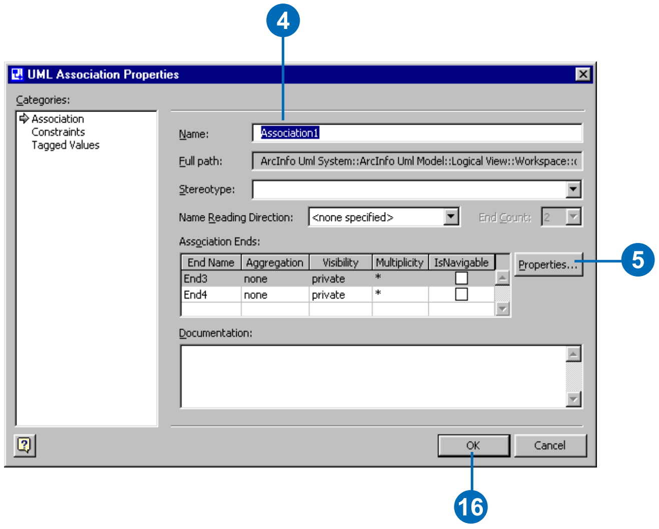
  Example: OriginForeignKey=PoleID ▸

Creating nonattributed relationship classes

1. In the UML Static Structure stencil, click Binary Association and drag and drop it onto the diagram.
2. Connect the two classes. The left end of the association is the origin class, and the right end is the destination class.
3. Double-click the association.
4. Type a name for the association.
5. Click one of the association ends and click Properties. ▸




2
PowerPole
-Height : esriFieldTypeDouble
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
1
-End5
-End4
1
Guy
-AnchorSize : esriFieldTypeString
-Length : esriFieldTypeDouble
-Strength : esriFieldTypeDouble
-PoleID : esriFieldTypeInteger



UML Association Properties

Categories:
- Association
- Constraints
- Tagged Values

Name: Association1

Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::(

Stereotype:

Name Reading Direction: <none specified>   End Count: 2

Association Ends:

End Name	Aggregation	Visibility	Multiplicity	IsNavigable
End3	none	private	*	☐
End4	none	private	*	☐

Properties...

Documentation:

OK   Cancel

- Notification: one of the values of the esriRelNotification enumeration

  Example: Notification=esriRelNotificationBoth

The cardinality of the relationship class is derived from the multiplicity of both association ends. Because the cardinality of a relationship class can be only 1-1, 1-M, or M-N, the only valid multiplicity values are 1 and * (many). The name of an association end becomes a relationship class path label.

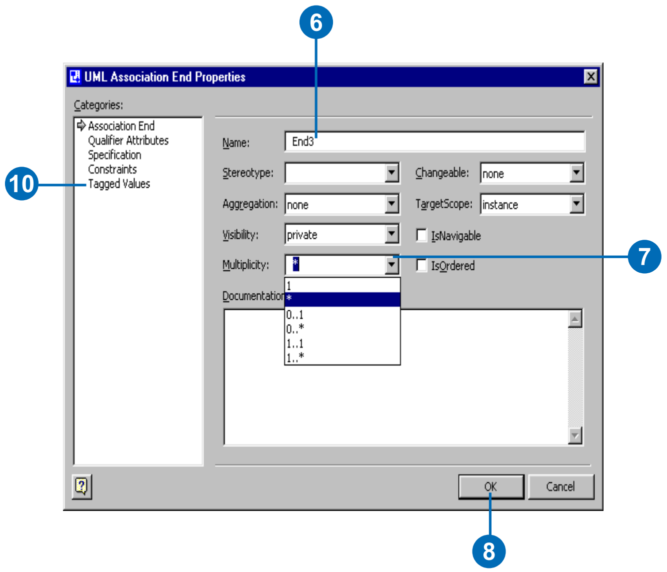
A relationship class can have its own set of attributes. You can model such relationship classes by adding a UML class named after the relationship and stereotyped as RelationshipClass. The fields of the relationship class are modeled in the same way as fields in any other class. Foreign keys must be included in the class representing the attributed relationship. Many-to-many relationships are always attributed.

The following tagged values are recognized for attributed relationship classes:

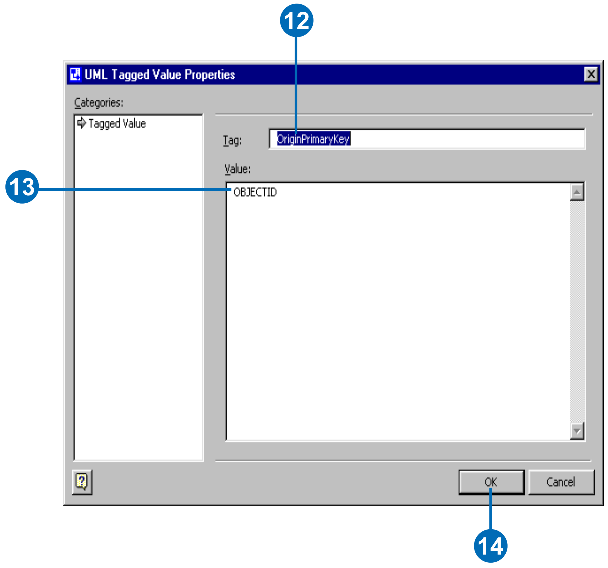
- IsAttributed: should be True ►

6. Type the association end name.
7. Click the Multiplicity dropdown arrow and click the association end multiplicity.
8. Click OK.
9. Repeat steps 5 through 8 to set the name and multiplicity of the second end.

   The keys for the relationship class are set in UML using tagged values.
10. Click Tagged Values in the Categories window.
11. Click New.
12. Type “OriginPrimaryKey” for the tag name.
13. Type the name of the origin primary key field.
14. Click OK.
15. Repeat steps 11 through 14 to set the origin foreign key and notification tags. ►



Name: End3
Stereotype:
Changeable: none
Aggregation: none
TargetScope: instance
Visibility: private
IsNavigable: ☐
Multiplicity: *
IsOrdered: ☐



Tag: OriginPrimaryKey
Value: OBJECTID

- OriginPrimaryKey: the name of the primary key field in the origin class

  Example:
  OriginPrimaryKey=OBJECTID
- OriginForeignKey: the name of the foreign key field in the destination class

  Example:
  OriginForeignKey=OwnerID
- DestinationPrimaryKey: the name of the primary key field in the destination class

  Example:
  DestinationPrimaryKey=OBJECTID
- DestinationForeignKey: the name of the foreign key field in the attributed relationship

  Example:
  DestinationForeignKey=FittingID
- Notification: one of the values of the esriRelNotification enumeration

  Example:
  Notification=esriRelNotificationBoth

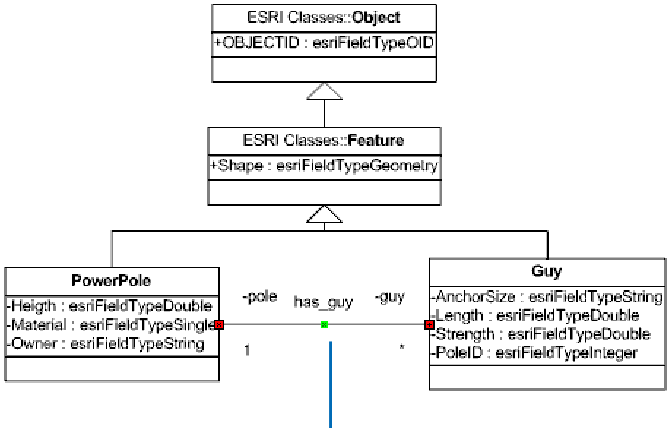
Tip

Using OBJECTID as a key field

OBJECTID is defined in the Object class and is inherited by all other classes. If a field typed as OID is used as the primary key, then the foreign key must be typed as esriFieldTypeInteger.

16. Click OK.

    The relationship class is shown as an association between two classes. The name of the association is the name of the relationship class.


ESRI Classes::Object
+OBJECTID : esriFieldTypeOID
ESRI Classes::Feature
+Shape : esriFieldTypeGeometry
PowerPole
-Heigth : esriFieldTypeDouble
-Material : esriFieldTypeSingle
-Owner : esriFieldTypeString
-pole
has_guy
-guy
1
*
Guy
-AnchorSize : esriFieldTypeString
-Length : esriFieldTypeDouble
-Strength : esriFieldTypeDouble
-PoleID : esriFieldTypeInteger

The relationship class is shown as an association between classes.

Tip

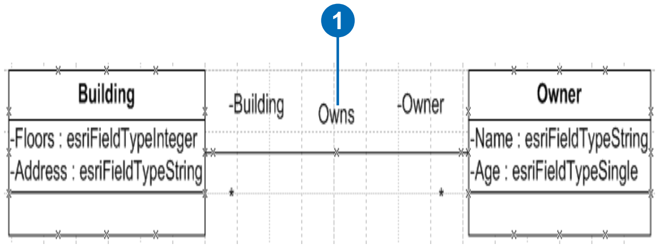
Composite relationships

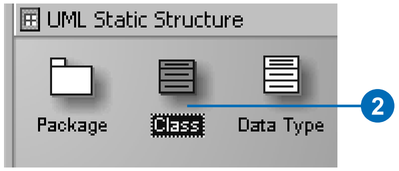
In a composite relationship, one of the objects controls the lifetime of the associated objects. You can create a composite relationship by using a Composition instead of a binary association.

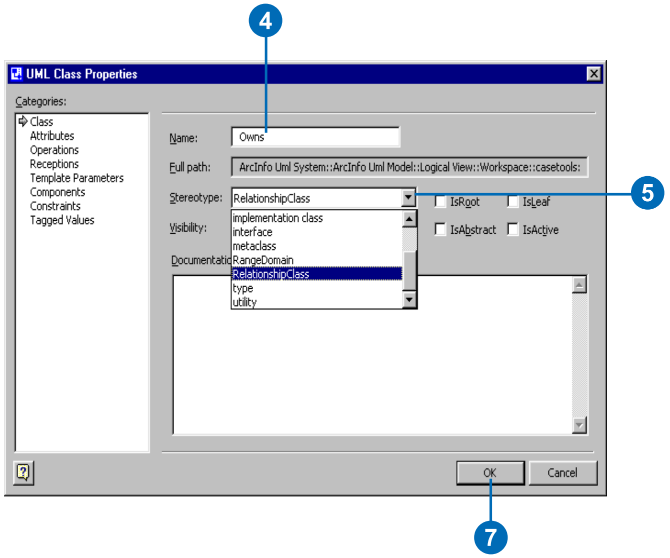
Creating attributed relationship classes

1. Create the UML association representing the relationship class by following steps 1 through 16 of 'Creating nonattributed relationship classes'.
2. In the UML Static Structure stencil, click Class and drag and drop a new UML class onto the diagram.
3. Double-click the new class.
4. Type the name of the association as the name of the new class.
5. Click the Stereotype dropdown arrow and click RelationshipClass.
6. Follow steps 1 through 8 of 'Adding fields to a feature class' in this document to add fields to the relationship class.
7. Click OK.

   The keys for the relationship class are set in UML using tagged values. ▸


1
Building
-Floors : esriFieldTypeInteger
-Address : esriFieldTypeString
-Building
Owns
-Owner
Owner
-Name : esriFieldTypeString
-Age : esriFieldTypeSingle
*
*





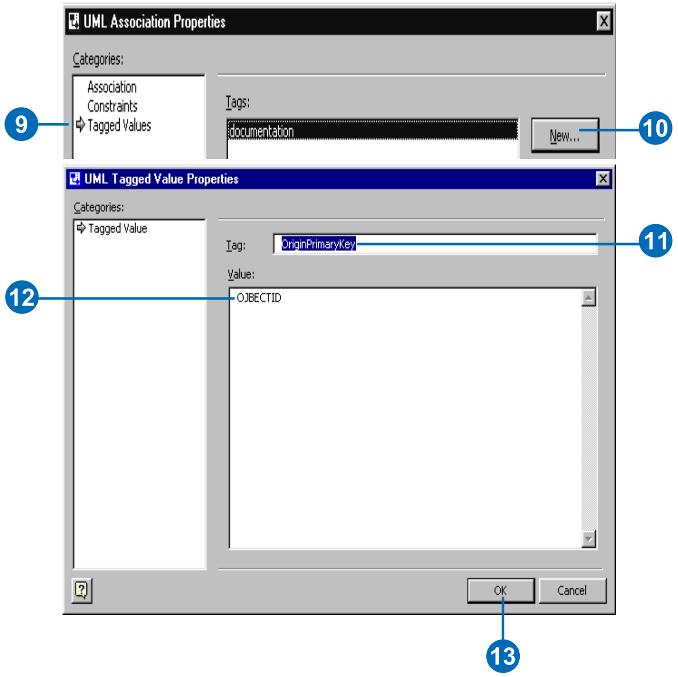
Tip

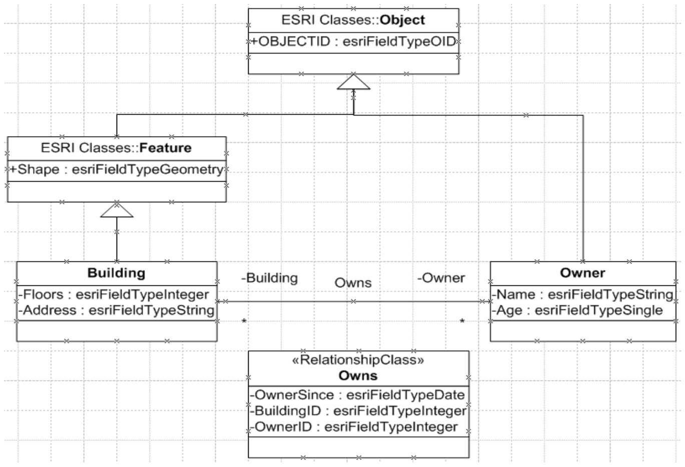
Association tagged values

The tagged values should be in the association and not in the class with the attributes of the relationship.

8. Double-click the association.
9. Click Tagged Values in the Categories window.
10. Click New.
11. Type "OriginPrimaryKey" for the tag name.
12. Type the name of the origin primary key field.
13. Click OK.
14. Repeat steps 10 through 13 for the destination primary key, the origin and destination foreign keys, and the notification tags.
15. Click OK.

    The attributed relationship is represented by a class with the same name as the relationship association. Its attributes appear as properties of the class.




ESRI Classes::Object
+OBJECTID : esriFieldTypeOID
ESRI Classes::Feature
+Shape : esriFieldTypeGeometry
Building
-Floors : esriFieldTypeInteger
-Address : esriFieldTypeString
-Building
Owns
-Owner
*
*
Owner
-Name : esriFieldTypeString
-Age : esriFieldTypeSingle
«RelationshipClass»
Owns
-OwnerSince : esriFieldTypeDate
-BuildingID : esriFieldTypeInteger
-OwnerID : esriFieldTypeInteger

Creating domains

Domains are modeled in UML as stereotyped classes. Domains can be used as the type of a field to define the type and valid values for the field.

The first three attributes in the class define the field type, the *merge policy*, and the *split policy*. The type for any of these attributes is not important and can be left unspecified. The initial value, however, is the actual setting. For example, the following is a valid MergePolicy:

- Attribute name: MergePolicy
- Attribute type: <unspecified>
- Attribute Initial Value: esriMPTDefaultValue

The valid initial values for FieldType, MergePolicy, and SplitPolicy are taken from the esriFieldType, esriMergePolicy, and esriSplitPolicy enumerations, respectively.

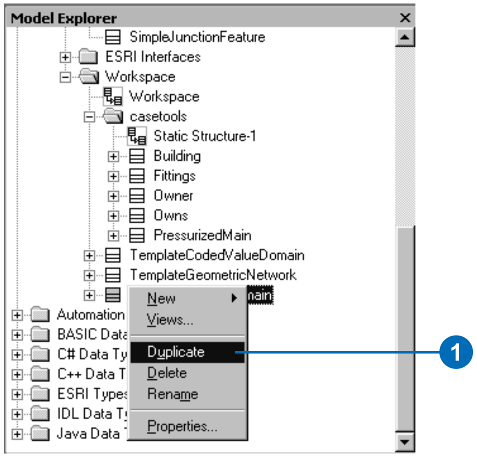
In addition to the standard attributes of domains (FieldType, MergePolicy, and SplitPolicy), range domains have MinValue and MaxValue. ►

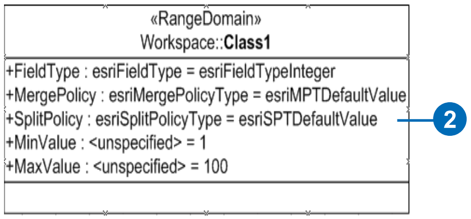
Creating a range domain

1. In the Model Explorer, under Workspace package, right-click TemplateRangeDomain and click Duplicate.

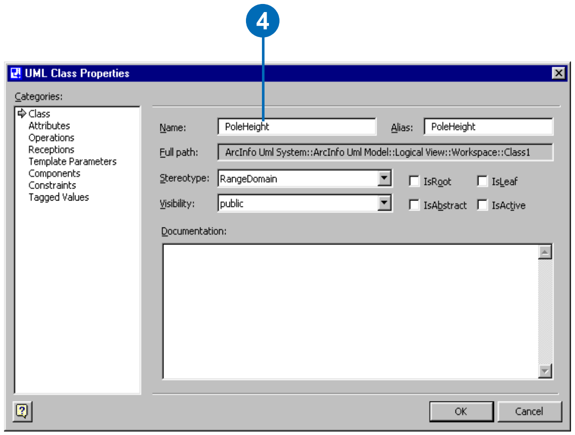
   A copy of the TemplateRangeDomain is created under the Workspace package.

2. Drag and drop it on the diagram.
3. Double-click the new class.
4. Type the name of the domain. ►




«RangeDomain»
Workspace::**Class1**

+FieldType : esriFieldType = esriFieldTypeInteger
+MergePolicy : esriMergePolicyType = esriMPTDefaultValue
+SplitPolicy : esriSplitPolicyType = esriSPTDefaultValue
+MinValue : <unspecified> = 1
+MaxValue : <unspecified> = 100

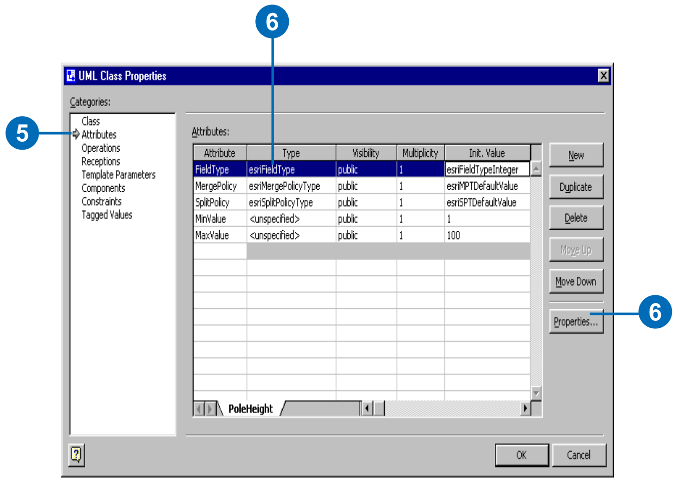

UML Class Properties

Name: PoleHeight
Alias: PoleHeight
Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::Class1
Stereotype: RangeDomain
Visibility: public
Documentation:

The initial values in MinValue and MaxValue define the actual range. The type of these attributes is not important and can be left unspecified.

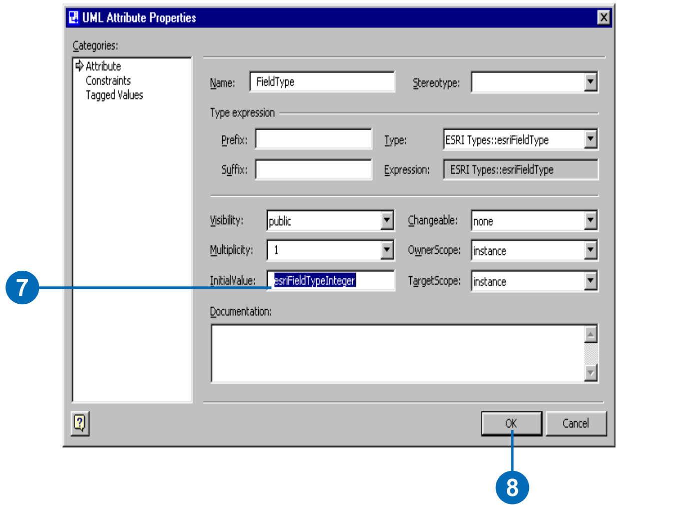
Coded value domains can have any number of UML attributes representing the set of permissible values. The initial value defines the valid code, and the name of the attribute is the name of the code. The type of these attributes is not important and can be left unspecified.

5. Click Attributes in the Categories window.
6. Click FieldType and click Properties.
7. Type the type of field with which this domain will be associated in the InitialValue text box.
8. Click OK.
9. Click MergePolicy and click Properties.
10. Type the merge policy in the InitialValue text box.
11. Click OK.
12. Click SplitPolicy and click Properties.
13. Type the split policy in the InitialValue text box.
14. Click OK. ▸



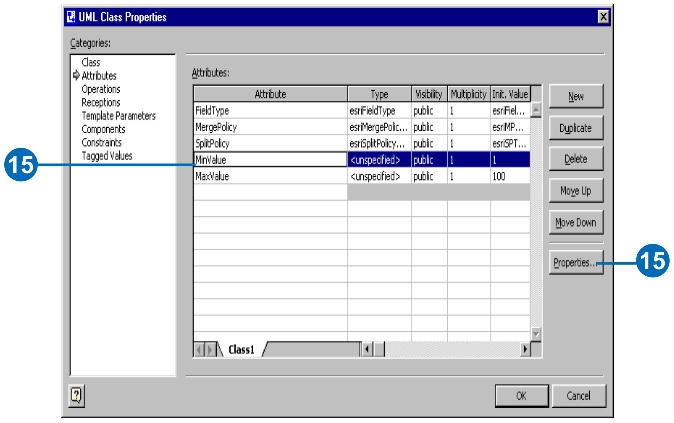
Attributes:

Attribute	Type	Visibility	Multiplicity	Init. Value
FieldType	esriFieldType	public	1	esriFieldTypeInteger
MergePolicy	esriMergePolicyType	public	1	esriMPTDefaultValue
SplitPolicy	esriSplitPolicyType	public	1	esriSPTDefaultValue
MinValue	<unspecified>	public	1	1
MaxValue	<unspecified>	public	1	100



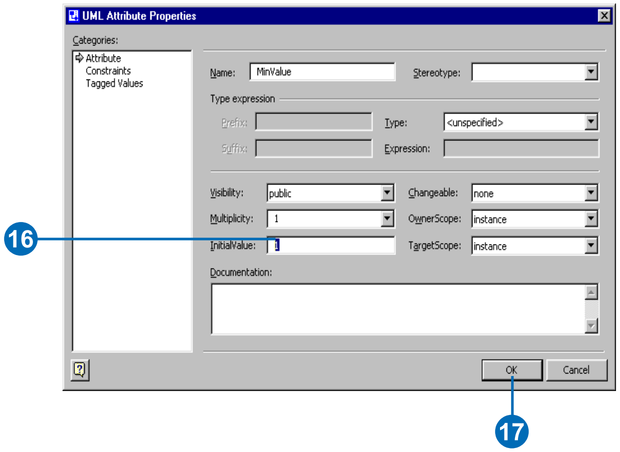
Name: FieldType
Stereotype:
Prefix:
Suffix:
Type: ESRI Types::esriFieldType
Expression: ESRI Types::esriFieldType
Visibility: public
Multiplicity: 1
InitialValue: esriFieldTypeInteger
Changeable: none
OwnerScope: instance
TargetScope: instance

15. Click the MinValue attribute and click Properties.
16. Type the minimum value for the range domain in the InitialValue text box.
17. Click OK.
18. Click the MaxValue attribute and click Properties.
19. Type the maximum value for the range domain in the InitialValue text box.
20. Click OK.
21. Right-click the domain in the diagram and click Shape Display Options. Click the Attribute check box to hide the attribute types.

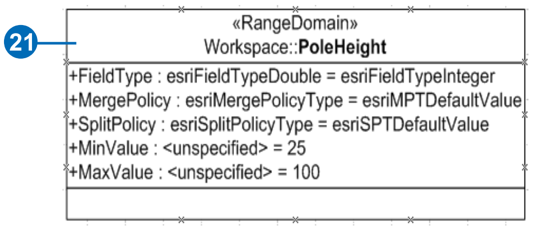


Attributes:

Attribute	Type	Visibility	Multiplicity	Init. Value
FieldType	esriFieldType	public	1	esriFiel...
MergePolicy	esriMergePolic...	public	1	esriMP...
SplitPolicy	esriSplitPolicy...	public	1	esriSPT...
MinValue	<unspecified>	public	1	1
MaxValue	<unspecified>	public	1	100



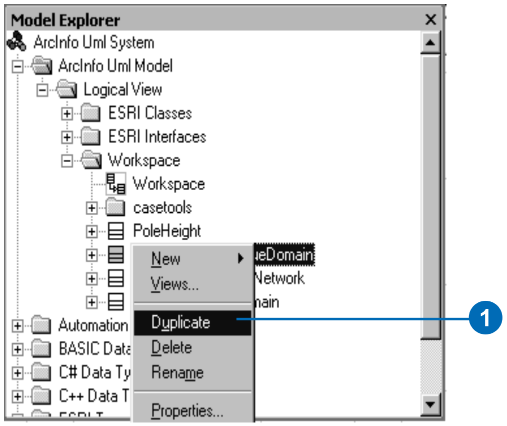
Name: MinValue
Stereotype:
Type: <unspecified>
Visibility: public
Changeable: none
Multiplicity: 1
OwnerScope: instance
InitialValue: 1
TargetScope: instance

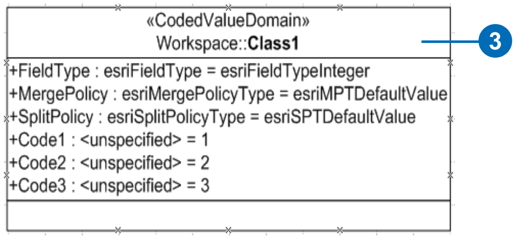


«RangeDomain»
Workspace::**PoleHeight**
+FieldType : esriFieldTypeDouble = esriFieldTypeInteger
+MergePolicy : esriMergePolicyType = esriMPTDefaultValue
+SplitPolicy : esriSplitPolicyType = esriSPTDefaultValue
+MinValue : <unspecified> = 25
+MaxValue : <unspecified> = 100

Creating coded value domains

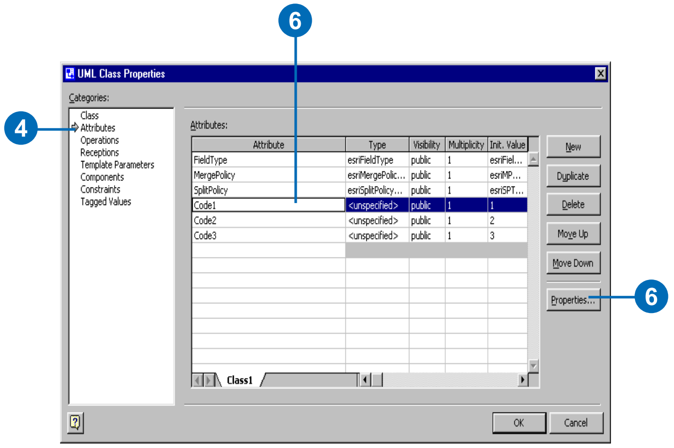
1. In the Model Explorer, under the Workspace package, navigate to and right-click TemplateCodedValueDomain. Click Duplicate.
2. A copy of the TemplateCodedValueDomain is created under the Workspace package. Drag and drop it on the diagram.
3. Double-click the domain in the diagram and type the name of the domain.
4. Click Attributes in the Categories window.
5. Follow steps 6 through 14 of ‘Creating a range domain’ in this document to set the field type and the split and merge policies.
6. Click the Code1 attribute and click Properties. ▸





«CodedValueDomain»
Workspace::**Class1**

+FieldType : esriFieldType = esriFieldTypeInteger
+MergePolicy : esriMergePolicyType = esriMPTDefaultValue
+SplitPolicy : esriSplitPolicyType = esriSPTDefaultValue
+Code1 : <unspecified> = 1
+Code2 : <unspecified> = 2
+Code3 : <unspecified> = 3



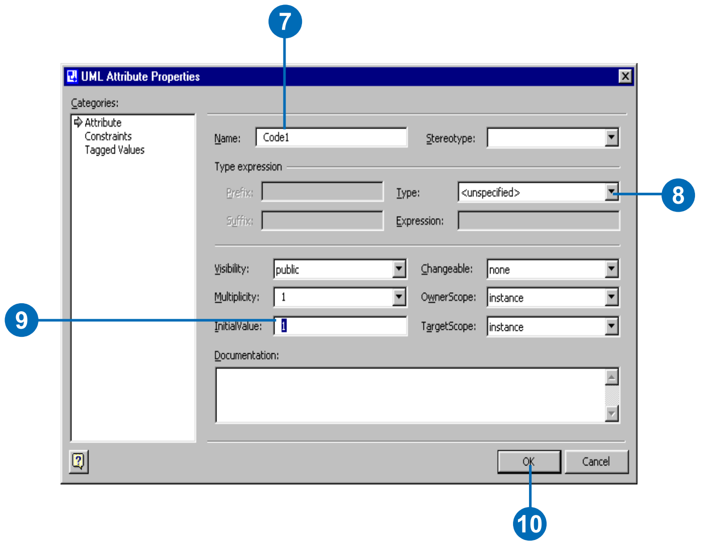
Attribute	Type	Visibility	Multiplicity	Init. Value
FieldType	esriFieldType	public	1	esriFiel...
MergePolicy	esriMergePolic...	public	1	esriMP...
SplitPolicy	esriSplitPolicy...	public	1	esriSPT...
Code1	<unspecified>	public	1	1
Code2	<unspecified>	public	1	2
Code3	<unspecified>	public	1	3

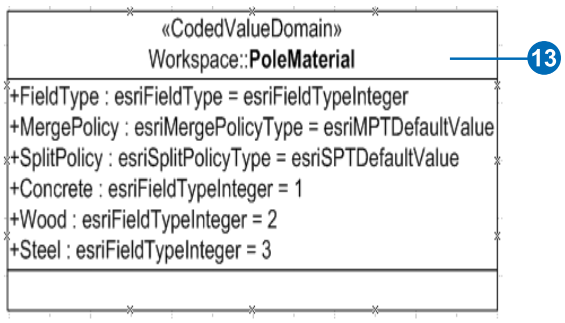
Tip

Adding additional codes

In the UML Class Properties Editor, you can add additional codes by clicking Attributes in the Categories window. Click New and click Properties for the attribute and set the name and initial value for the code.

7. Type the code name.
8. Click the Type dropdown arrow and select the type of field.
9. Type the code in the InitialValue text box.
10. Click OK.
11. Repeat steps 6 through 10 for the other codes.
12. Click OK.
13. Right-click the domain in the diagram and click Shape Display Options. Click the Attribute check box to hide the attribute types.





«CodedValueDomain»
Workspace::**PoleMaterial**

+FieldType : esriFieldType = esriFieldTypeInteger
+MergePolicy : esriMergePolicyType = esriMPTDefaultValue
+SplitPolicy : esriSplitPolicyType = esriSPTDefaultValue
+Concrete : esriFieldTypeInteger = 1
+Wood : esriFieldTypeInteger = 2
+Steel : esriFieldTypeInteger = 3

Creating subtypes

Subtypes are modeled in UML as classes related to the parent class through an association stereotyped as Subtype. You can specify domains and default values for the fields in the subtype.

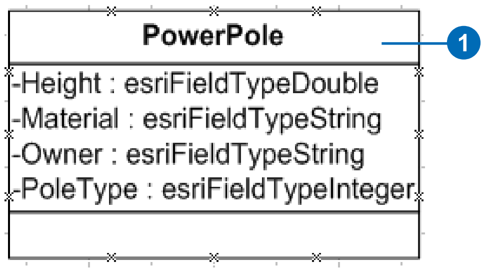
A field in the parent class must be stereotyped as SubtypeField. Its initial value is the subtype code of the default subtype. The subtype field must be typed as "esriFieldTypeInteger" in the parent class.

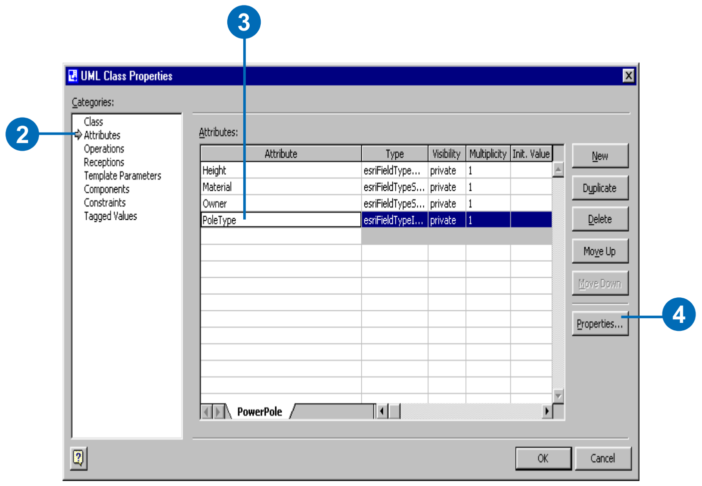
All subtypes must have a unique value for the subtype field—this unique value is its subtype code. Fields in the subtype must match those of the parent class in name and type, but not all the fields of the parent class have to be present in the subtype. The subtype field, however, is a required field in the subtype.

Initial values must be type-compatible with the type of the field. The type of a field can be a domain previously created.

Defining the subtype field for the feature class

1. In the diagram, double-click the class for which you want to create a subtype.
2. Click Attributes in the Categories window.
3. Click the attribute that will be the subtype field.
4. Click Properties. ▸

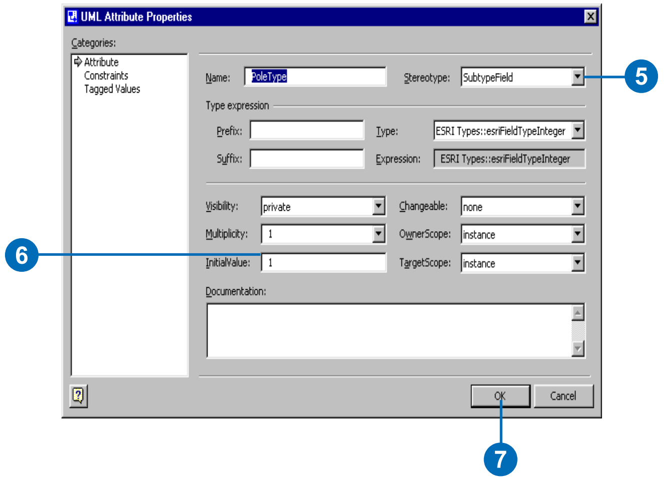

PowerPole
-Height : esriFieldTypeDouble
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
-PoleType : esriFieldTypeInteger
1



Attribute	Type	Visibility	Multiplicity	Init. Value
Height	esriFieldType...	private	1	
Material	esriFieldTypeS...	private	1	
Owner	esriFieldTypeS...	private	1	
PoleType	esriFieldTypeI...	private	1	

5. Click the Stereotype dropdown arrow and click SubtypeField.
6. Type the subtype code of the default subtype in the InitialValue text box.
7. Click OK.

   The subtype field appears as a property of the class with a stereotype of SubtypeField.


UML Attribute Properties

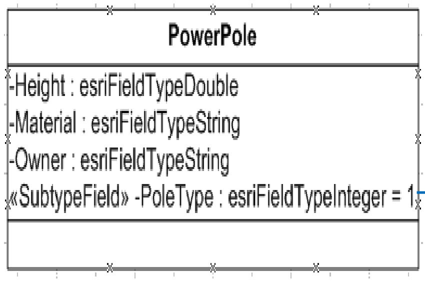
Categories: Attribute, Constraints, Tagged Values

Name: PoleType
Stereotype: SubtypeField

Type expression
Prefix:
Type: ESRI Types::esriFieldTypeInteger
Suffix:
Expression: ESRI Types::esriFieldTypeInteger

Visibility: private
Changeable: none
Multiplicity: 1
OwnerScope: instance
InitialValue: 1
TargetScope: instance

Documentation:



PowerPole
-Height : esriFieldTypeDouble
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
«SubtypeField» -PoleType : esriFieldTypeInteger = 1

The subtype field is stereotyped as SubtypeField.

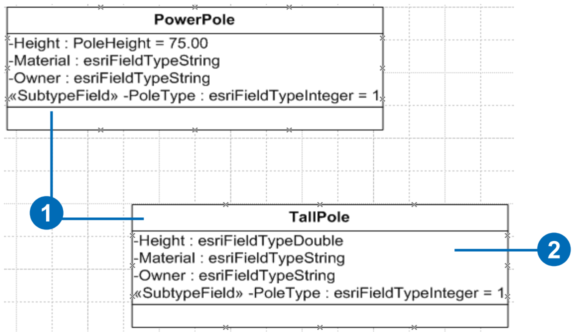
Tip

When to include a field in a subtype

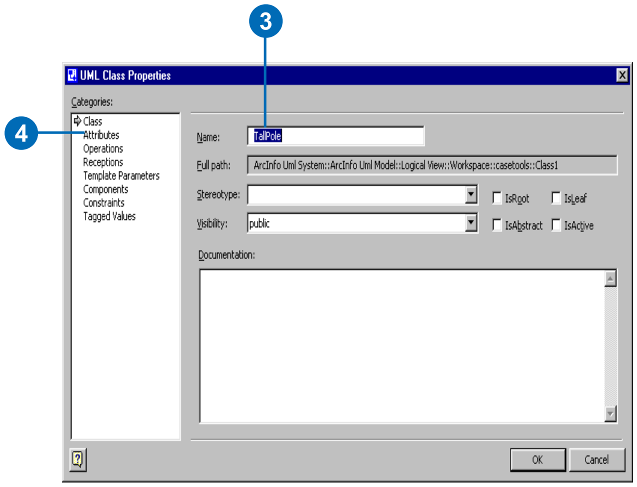
A field should be included in a subtype only when you want to associate a default value or domain with it. Fields inherited from abstract classes can be included in subtypes.

Creating a subtype

1. In the diagram, copy the class for which you want to create a subtype. Paste it on the diagram.
2. Double-click the new class.
3. Type a name for the subtype.
4. Click Attributes in the Categories window.
5. Click each field for which you don't want to associate a default value or domain for this subtype, then click Delete to remove each one. ▸



PowerPole
-Height : PoleHeight = 75.00
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
«SubtypeField» -PoleType : esriFieldTypeInteger = 1
1
TallPole
-Height : esriFieldTypeDouble
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
«SubtypeField» -PoleType : esriFieldTypeInteger = 1
2



Name: TallPole

Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::casetools::Class1

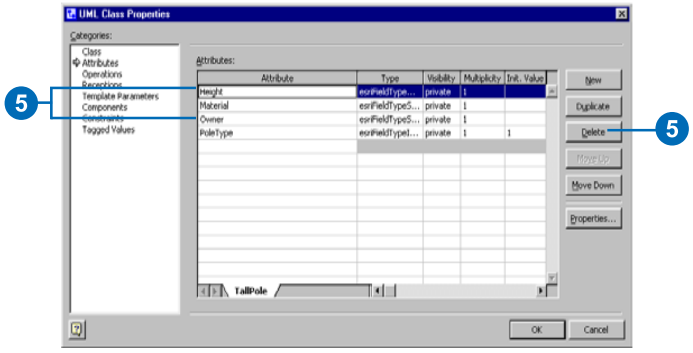
Stereotype:

☐ IsRoot ☐ IsLeaf

Visibility: public

☐ IsAbstract ☐ IsActive

Documentation:



Attributes:

Attribute	Type	Visibility	Multiplicity	Init. Value
Height	esriFieldType...	private	1	
Material	esriFieldTypeS...	private	1	
Owner	esriFieldTypeS...	private	1	
PoleType	esriFieldTypeI...	private	1	1

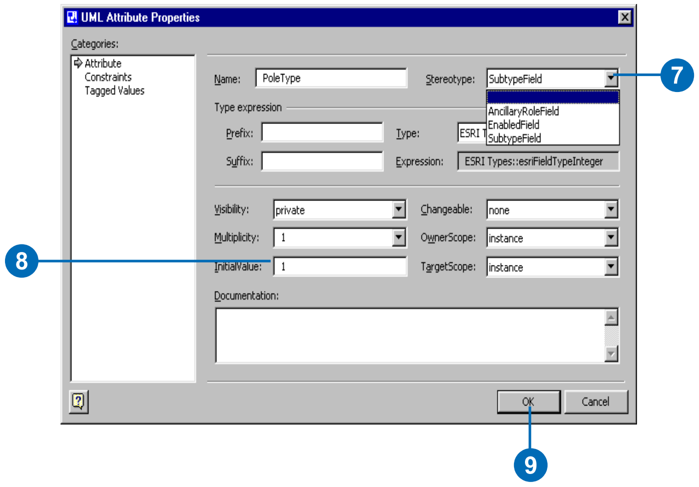
6. Click the subtype field and click Properties.

7. Click the Stereotype dropdown arrow and click the blank stereotype to set its stereotype to nothing.

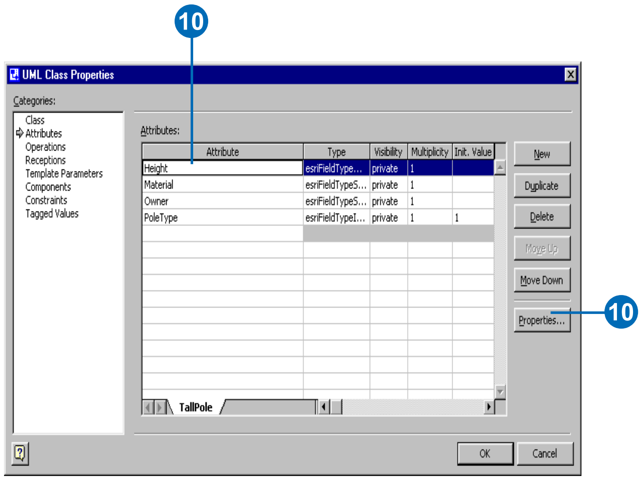
8. Type the code for this subtype in the InitialValue text box.

9. Click OK.

10. To set default values and domains for a field, click the attribute and click Properties. ▸



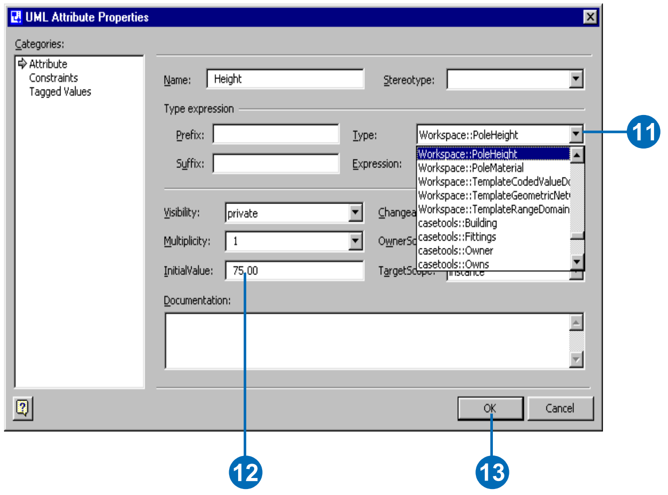
Name: PoleType
Stereotype: SubtypeField
Prefix:
Type: ESRI T
Suffix:
Expression: ESRI Types::esriFieldTypeInteger
Visibility: private
Changeable: none
Multiplicity: 1
OwnerScope: instance
InitialValue: 1
TargetScope: instance
Documentation:



Attribute	Type	Visibility	Multiplicity	Init. Value
Height	esriFieldType...	private	1	
Material	esriFieldTypeS...	private	1	
Owner	esriFieldTypeS...	private	1	
PoleType	esriFieldTypeI...	private	1	1

11. Click the Type dropdown arrow and click the domain you want to associate with the attribute as the type.
12. To associate a default value with this attribute, type the default value in the InitialValue text box.
13. Click OK.
14. Repeat steps 6 through 9 until you have associated default values and domains with all the attributes.
15. Click OK.

    The field appears in the class with its domain as its type and its default value as its initial value.


UML Attribute Properties

Categories:
- Attribute
- Constraints
- Tagged Values

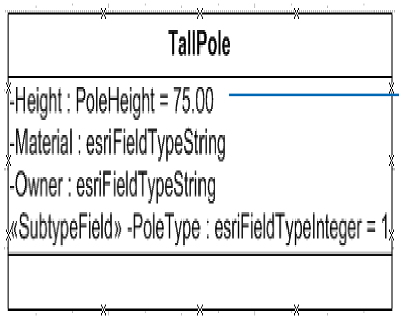
Name: Height
Stereotype:

Type expression
Prefix:
Suffix:
Type: Workspace::PoleHeight
Expression:

- Workspace::PoleHeight
- Workspace::PoleMaterial
- Workspace::TemplateCodedValueDo
- Workspace::TemplateGeometricNet
- Workspace::TemplateRangeDomain
- casetools::Building
- casetools::Fittings
- casetools::Owner
- casetools::Owns

Visibility: private
Multiplicity: 1
InitialValue: 75.00

Documentation:


TallPole

-Height : PoleHeight = 75.00
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
«SubtypeField» -PoleType : esriFieldTypeInteger = 1

The domain is displayed as the field type and the default value as the initial value.

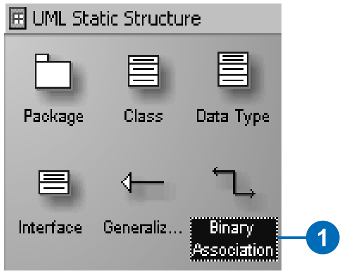
Tip

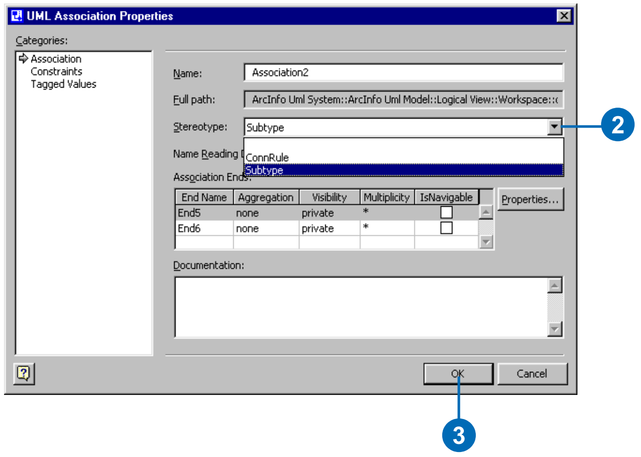
Setting the initial value for the subtype field

Be very careful when setting the initial value for the subtype field in your subtypes. Its value has to be unique across all the subtypes of a class.

Associating the subtype with its parent class

1. In the UML Static Structure stencil, click Binary Association and drag and drop it on the diagram. Connect the parent and subtype classes.
2. In the diagram, double-click the association. Click the Stereotype dropdown arrow and click Subtype.
3. Click OK.
4. Right-click the association and click the Shape Display Options to show its name and stereotype and to hide the end names and cardinalities.





Categories:
Association
Constraints
Tagged Values

Name: Association2

Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::(

Stereotype: Subtype

ConnRule
Subtype

Name Reading

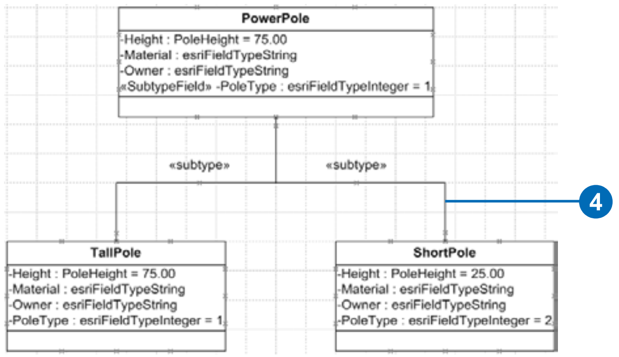
Association Ends:

End Name	Aggregation	Visibility	Multiplicity	IsNavigable
End5	none	private	*	☐
End6	none	private	*	☐

Properties...

Documentation:

OK Cancel


PowerPole
-Height : PoleHeight = 75.00
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
«SubtypeField» -PoleType : esriFieldTypeInteger = 1
«subtype»
«subtype»
TallPole
-Height : PoleHeight = 75.00
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
-PoleType : esriFieldTypeInteger = 1
ShortPole
-Height : PoleHeight = 25.00
-Material : esriFieldTypeString
-Owner : esriFieldTypeString
-PoleType : esriFieldTypeInteger = 2
4

Creating relationship rules

Relationship classes can have a set of relationship rules. These rules control which subtypes of objects from the origin class can be related to which subtypes of the destination class.

Relationship rules are represented by a UML association between subtypes. Multiplicity in association ends is used to set the cardinality of the relationship rule.

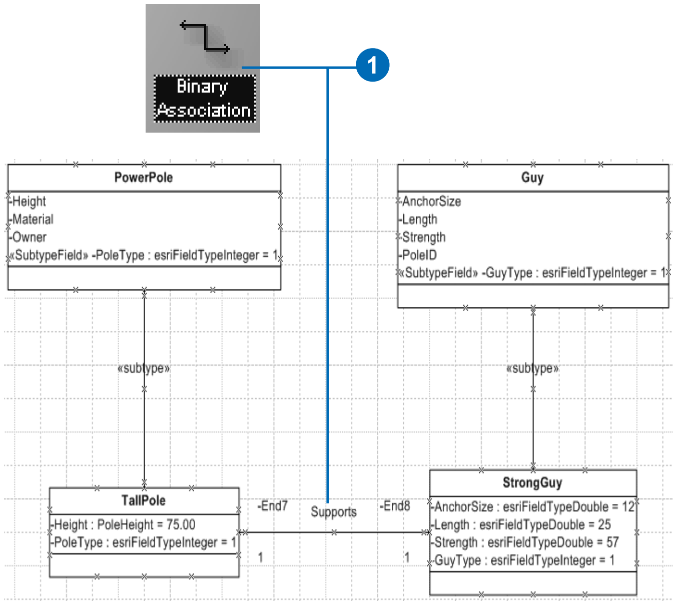
A value of 1..3 means at least one and no more than three objects of this subtype can be related.

The cardinality range of both ends must be compatible with the cardinality of the parent relationship class (in the example, 1-1..3 is compatible with 1-M).

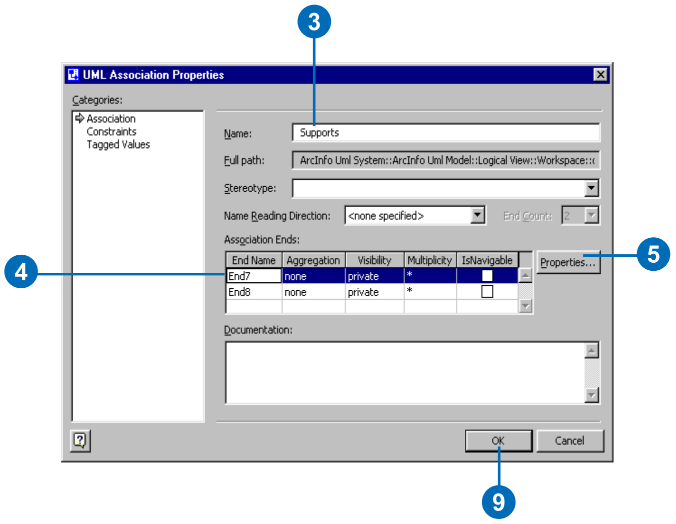
See Also

For more information on relationship rules, see Building a Geodatabase.

1. In the UML Static Structure stencil, click Binary Association and drag and drop it on the diagram. Connect the two subtypes between which you want to create a relationship rule.
2. Double-click the association.
3. Type the name of the relationship between the parent classes as the name for this association.
4. Click one of the association ends.
5. Click Properties. ▸



Binary Association
1
PowerPole
-Height
-Material
-Owner
«SubtypeField» -PoleType : esriFieldTypeInteger = 1
Guy
-AnchorSize
-Length
-Strength
-PoleID
«SubtypeField» -GuyType : esriFieldTypeInteger = 1
«subtype»
«subtype»
TallPole
-Height : PoleHeight = 75.00
-PoleType : esriFieldTypeInteger = 1
-End7
Supports
-End8
1
1
StrongGuy
-AnchorSize : esriFieldTypeDouble = 12
-Length : esriFieldTypeDouble = 25
-Strength : esriFieldTypeDouble = 57
-GuyType : esriFieldTypeInteger = 1



UML Association Properties

Categories:
- Association
- Constraints
- Tagged Values

Name: Supports

Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::(

Stereotype:

Name Reading Direction: <none specified>   End Count: 2

Association Ends:

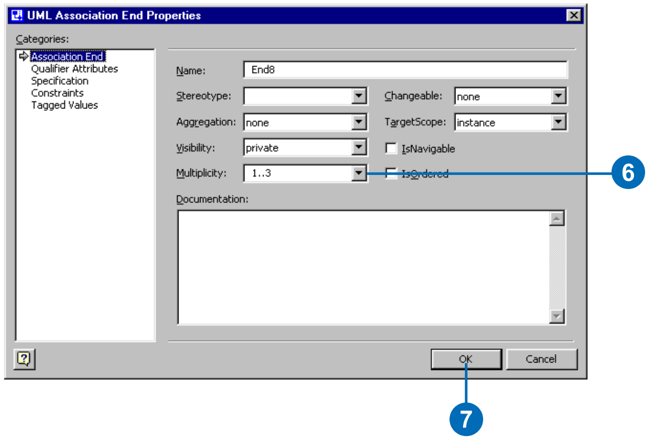
End Name	Aggregation	Visibility	Multiplicity	IsNavigable
End7	none	private	*	☐
End8	none	private	*	☐

Properties...

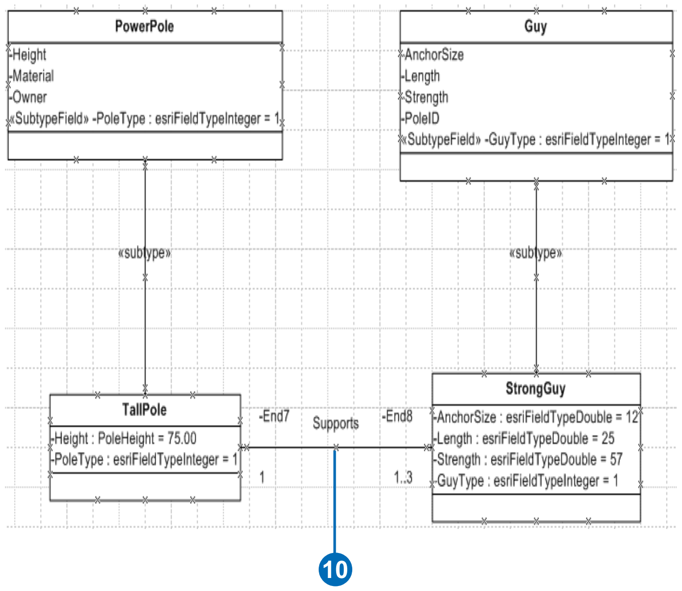
Documentation:

OK   Cancel

6. Type the valid cardinality as the multiplicity of the association end.
7. Click OK.
8. Repeat steps 4 through 7 for the other association end.
9. Click OK.
10. Right-click the association and click the Shape Display Options to show its name and stereotype and to hide the end names and cardinalities.



Name: End8
Stereotype:
Changeable: none
Aggregation: none
TargetScope: instance
Visibility: private
IsNavigable
Multiplicity: 1..3
IsOrdered


PowerPole
-Height
-Material
-Owner
«SubtypeField» -PoleType : esriFieldTypeInteger = 1
Guy
-AnchorSize
-Length
-Strength
-PoleID
«SubtypeField» -GuyType : esriFieldTypeInteger = 1
«subtype»
«subtype»
TallPole
-Height : PoleHeight = 75.00
-PoleType : esriFieldTypeInteger = 1
StrongGuy
-AnchorSize : esriFieldTypeDouble = 12
-Length : esriFieldTypeDouble = 25
-Strength : esriFieldTypeDouble = 57
-GuyType : esriFieldTypeInteger = 1
-End7
Supports
-End8
1
1..3
10

Creating geometric networks

A geometric network is modeled with a UML class stereotyped as GeometricNetwork. UML binary associations are used to associate the network feature classes in the model with the geometric network.

The geometric network UML class and all associated feature classes must be created under the same feature dataset UML package.

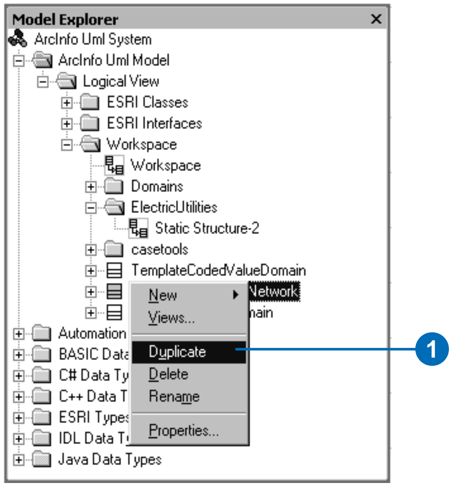
The only attribute in a geometric network class defines the network type. When you copy the template, the attribute is already in the class.

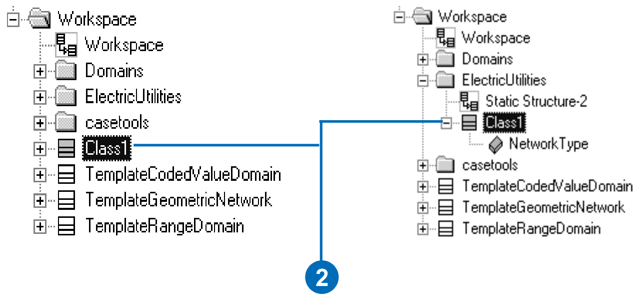
Tip

Feature datasets

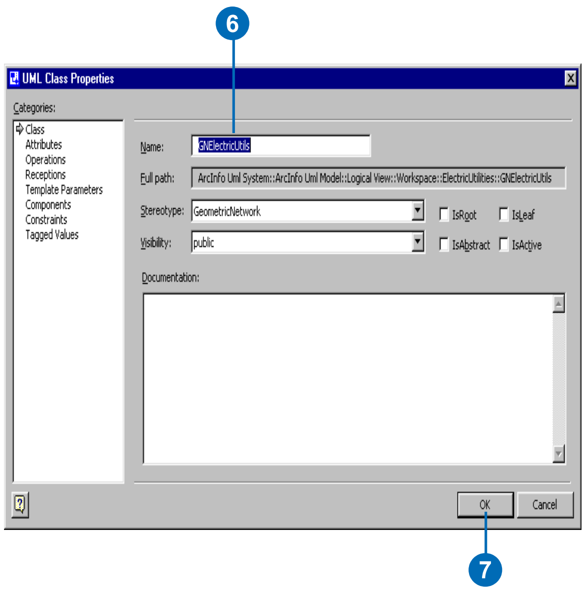
Make sure you have created a feature dataset before creating a geometric network. A geometric network can exist only inside a feature dataset.

1. In the Model Explorer, under the Workspace package, right-click TemplateGeometricNetwork and click Duplicate.
2. A copy of the TemplateGeometricNetwork is created under the Workspace package. In the Model Explorer, drag and drop it inside the package representing the feature dataset. ▸





3. Double-click the feature dataset diagram to open it.
4. Drag and drop the new geometric network on the diagram.
5. Double-click the geometric network to open its properties.
6. Type a name for the geometric network.
7. Click OK.
8. In the Model Explorer, click Binary Association and drag and drop it on the diagram. Connect the network feature classes to the geometric network.



Name: GNElectricUtils

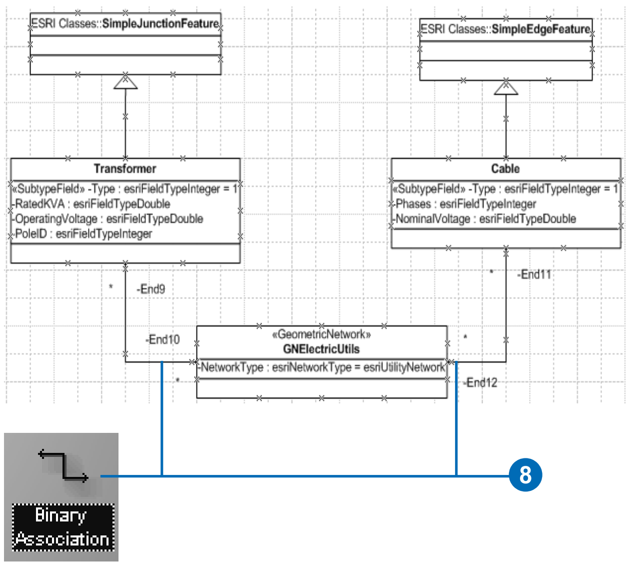
Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::ElectricUtilities::GNElectricUtils

Stereotype: GeometricNetwork

Visibility: public

☐ IsRoot ☐ IsLeaf ☐ IsAbstract ☐ IsActive

Documentation:


ESRI Classes::SimpleJunctionFeature
ESRI Classes::SimpleEdgeFeature
Transformer
«SubtypeField» -Type : esriFieldTypeInteger = 1
-RatedKVA : esriFieldTypeDouble
-OperatingVoltage : esriFieldTypeDouble
-PoleID : esriFieldTypeInteger
Cable
«SubtypeField» -Type : esriFieldTypeInteger = 1
-Phases : esriFieldTypeInteger
-NominalVoltage : esriFieldTypeDouble
*
-End9
*
-End11
-End10
«GeometricNetwork»
GNElectricUtils
-NetworkType : esriNetworkType = esriUtilityNetwork
*
*
-End12
Binary Association
8

Creating connectivity rules

Connectivity rules can be defined among subtypes of network feature classes modeled in a geometric network. UML associations are used to define connectivity rules. Two types of connectivity rules can be created: edge–edge rules and edge–junction rules.

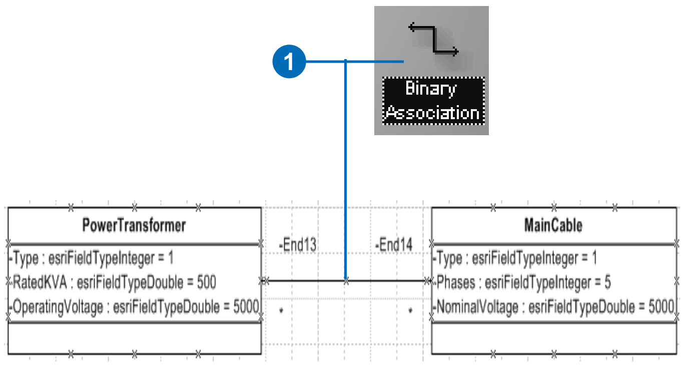
Edge–Junction connectivity rules can have specific cardinalities for each subtype involved. In this example, one power transformer can be connected to up to two main cables. One of the junction subtypes can be marked as the default junction by stereotyping the association end as Default.

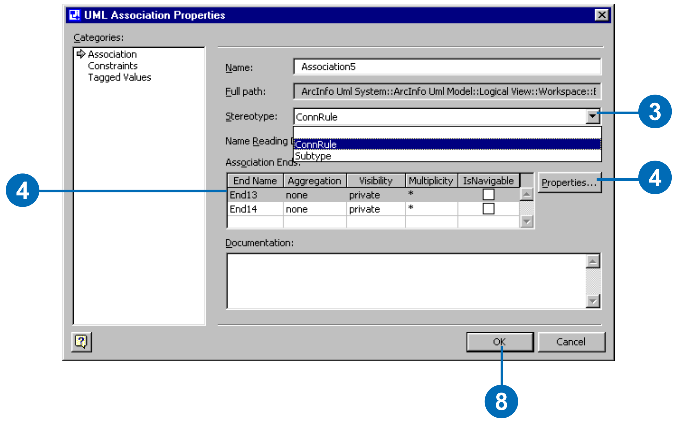
Edge–Edge rules are represented by a UML N-ary association that involves two edges and any number of junctions. One of the junction subtypes must be marked as the default junction by stereotyping the association end as Default.

The rule in this example can be read as "Distribution cables can be connected to main cables through distribution transformers". ▸

Creating an edge–junction rule

1. In the UML Static Structure stencil, click Binary Association and drag and drop it on the diagram. Connect the edge subtypes and junction subtype.
2. Double-click the association.
3. Click the Stereotype dropdown arrow and click ConnRule.
4. Click one of the association ends and click Properties. ▸


1
Binary Association
PowerTransformer
-Type : esriFieldTypeInteger = 1
-RatedKVA : esriFieldTypeDouble = 500
-OperatingVoltage : esriFieldTypeDouble = 5000
-End13
-End14
MainCable
-Type : esriFieldTypeInteger = 1
-Phases : esriFieldTypeInteger = 5
-NominalVoltage : esriFieldTypeDouble = 5000



UML Association Properties

Categories:
Association
Constraints
Tagged Values

Name: Association5

Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::E

Stereotype: ConnRule

Name Reading

ConnRule
Subtype

Association Ends:

End Name	Aggregation	Visibility	Multiplicity	IsNavigable
End13	none	private	*	☐
End14	none	private	*	☐

Properties...

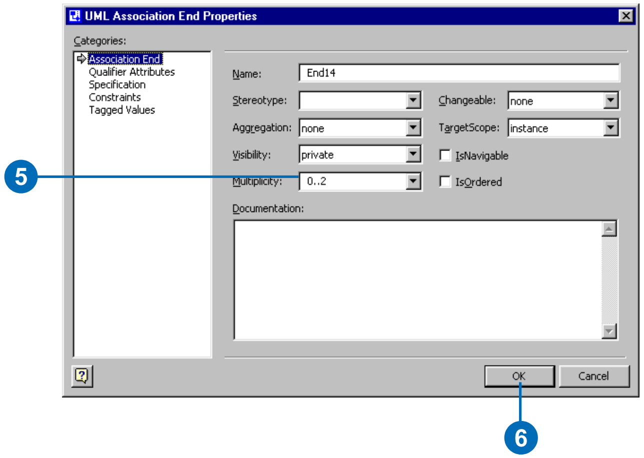
Documentation:

OK Cancel

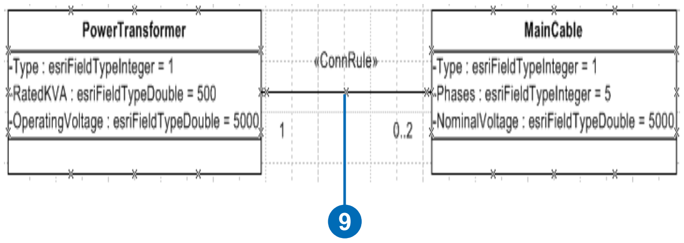
All geometric networks have a default or generic junction subtype, also called the orphan junction type. You can create connectivity rules that include the generic junction.

Both edge–junction and edge–edge rules can involve the generic junction.

5. Type the cardinality for the subtype.
6. Click OK.
7. Repeat steps 4 through 6 to set the cardinality for the second subtype.
8. Click OK.
9. Right-click the association and click the Shape Display Options to show its name and stereotype and to hide the end names and cardinalities.



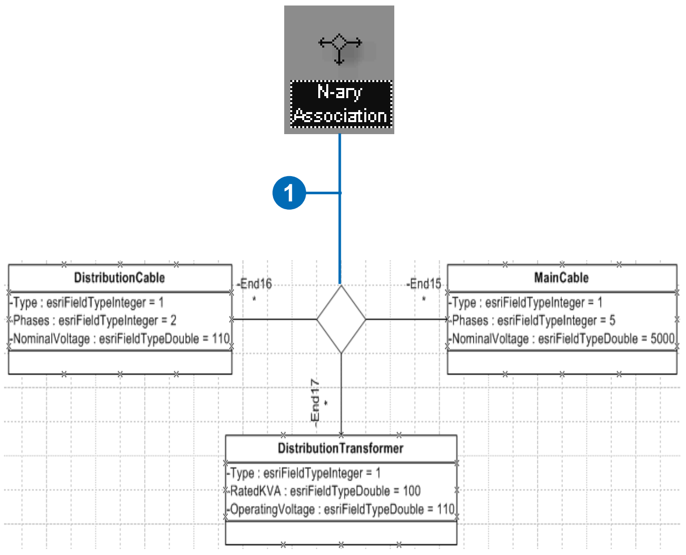
Name: End14
Stereotype:
Changeable: none
Aggregation: none
TargetScope: instance
Visibility: private
IsNavigable
Multiplicity: 0..2
IsOrdered

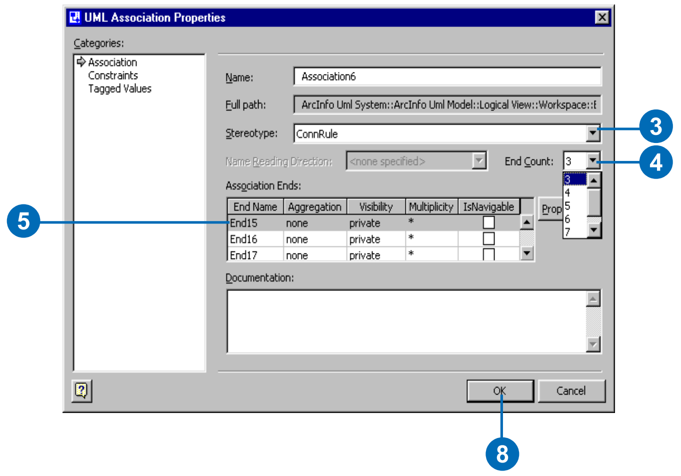


PowerTransformer
-Type : esriFieldTypeInteger = 1
-RatedKVA : esriFieldTypeDouble = 500
-OperatingVoltage : esriFieldTypeDouble = 5000
«ConnRule»
1
0..2
MainCable
-Type : esriFieldTypeInteger = 1
-Phases : esriFieldTypeInteger = 5
-NominalVoltage : esriFieldTypeDouble = 5000
9

Creating an edge–edge rule

1. In the UML Static Structure stencil, click N-ary Association and drag and drop it on the diagram. Connect two edge subtypes and a junction subtype.
2. Double-click the association.
3. Click the Stereotype dropdown arrow and click ConnRule.
4. If there is more than one junction subtype for this edge–edge rule, click the End Count dropdown arrow and click the number of junction subtypes.
5. Click one of the association ends and click Properties. ▸


N-ary Association
1
DistributionCable
-Type : esriFieldTypeInteger = 1
-Phases : esriFieldTypeInteger = 2
-NominalVoltage : esriFieldTypeDouble = 110
-End16
*
-End15
*
MainCable
-Type : esriFieldTypeInteger = 1
-Phases : esriFieldTypeInteger = 5
-NominalVoltage : esriFieldTypeDouble = 5000
-End17
*
DistributionTransformer
-Type : esriFieldTypeInteger = 1
-RatedKVA : esriFieldTypeDouble = 100
-OperatingVoltage : esriFieldTypeDouble = 110



Name: Association6

Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::

Stereotype: ConnRule

Name Reading Direction: <none specified>

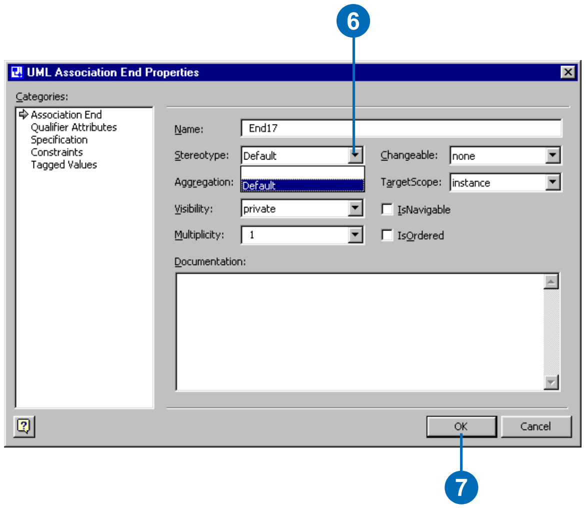
End Count: 3 (3, 4, 5, 6, 7)

Association Ends:

End Name	Aggregation	Visibility	Multiplicity	IsNavigable
End15	none	private	*	☐
End16	none	private	*	☐
End17	none	private	*	☐

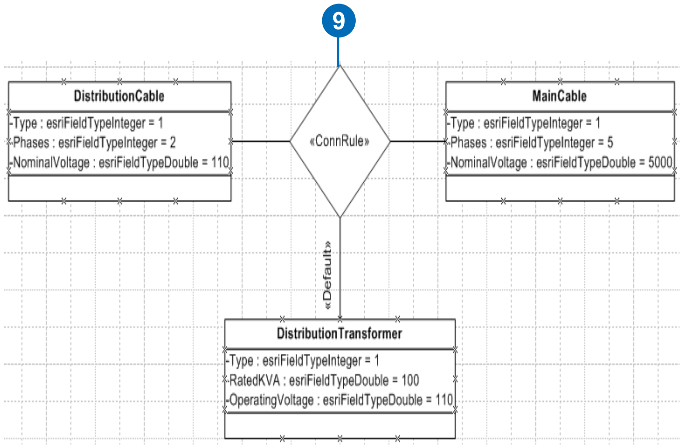
Documentation:

6. Click the Stereotype dropdown arrow and click Default to mark the junction subtype as the default junction subtype in the edge–edge rule.
7. Click OK.
8. Click OK.
9. Right-click the association and click the Shape Display Options to show its name and stereotype and to hide the end names and cardinalities.



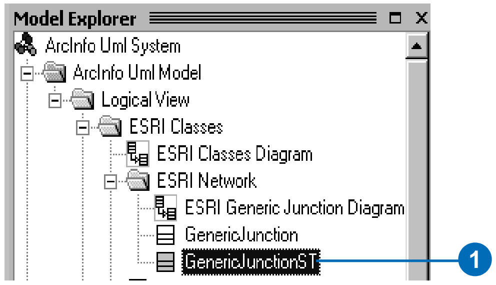
Categories:
- Association End
- Qualifier Attributes
- Specification
- Constraints
- Tagged Values

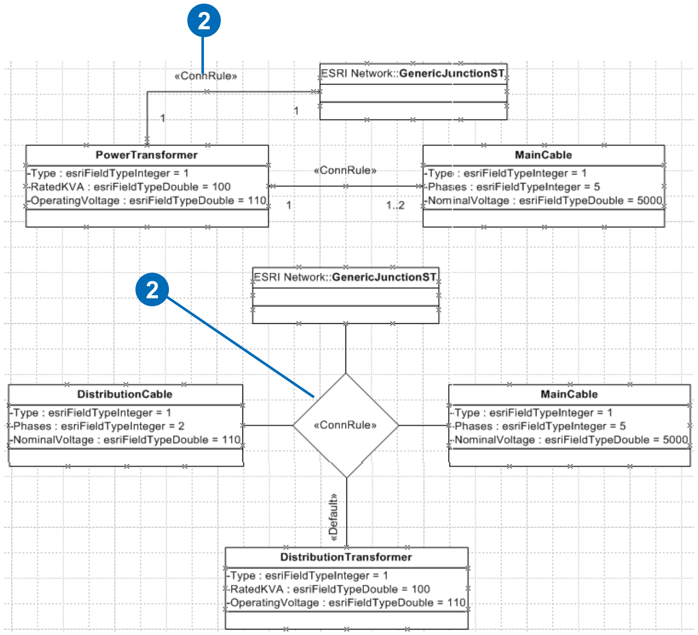
Name: End17
Stereotype: Default
Aggregation: Default
Visibility: private
Multiplicity: 1
Changeable: none
TargetScope: instance
IsNavigable
IsOrdered
Documentation:


9
DistributionCable
Type : esriFieldTypeInteger = 1
Phases : esriFieldTypeInteger = 2
NominalVoltage : esriFieldTypeDouble = 110
«ConnRule»
MainCable
Type : esriFieldTypeInteger = 1
Phases : esriFieldTypeInteger = 5
NominalVoltage : esriFieldTypeDouble = 5000
«Default»
DistributionTransformer
Type : esriFieldTypeInteger = 1
RatedKVA : esriFieldTypeDouble = 100
OperatingVoltage : esriFieldTypeDouble = 110

Using the generic junction subtype

1. In the Model Explorer, navigate to and click GenericJunctionST. Drag and drop it on the diagram.
2. Follow steps 1 through 9 of 'Creating an edge–junction rule' or steps 1 through 9 of 'Creating an edge–edge rule', in this document, using the generic junction subtype.




2
«ConnRule»
ESRI Network::GenericJunctionST
1
1
PowerTransformer
-Type : esriFieldTypeInteger = 1
-RatedKVA : esriFieldTypeDouble = 100
-OperatingVoltage : esriFieldTypeDouble = 110
«ConnRule»
1
1..2
MainCable
-Type : esriFieldTypeInteger = 1
-Phases : esriFieldTypeInteger = 5
-NominalVoltage : esriFieldTypeDouble = 5000
ESRI Network::GenericJunctionST
2
DistributionCable
-Type : esriFieldTypeInteger = 1
-Phases : esriFieldTypeInteger = 2
-NominalVoltage : esriFieldTypeDouble = 110
«ConnRule»
MainCable
-Type : esriFieldTypeInteger = 1
-Phases : esriFieldTypeInteger = 5
-NominalVoltage : esriFieldTypeDouble = 5000
«Default»
DistributionTransformer
-Type : esriFieldTypeInteger = 1
-RatedKVA : esriFieldTypeDouble = 100
-OperatingVoltage : esriFieldTypeDouble = 110

Extending classes with custom behavior

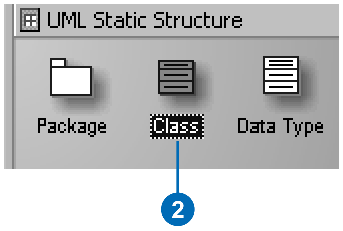
In addition to designing schema, you can use UML and CASE tools to generate code to create custom feature behavior. There are two modeling tasks associated with this: creating new interfaces and creating class extensions. It is important to understand that interfaces and class extensions in UML models are used only when code is generated for the model.

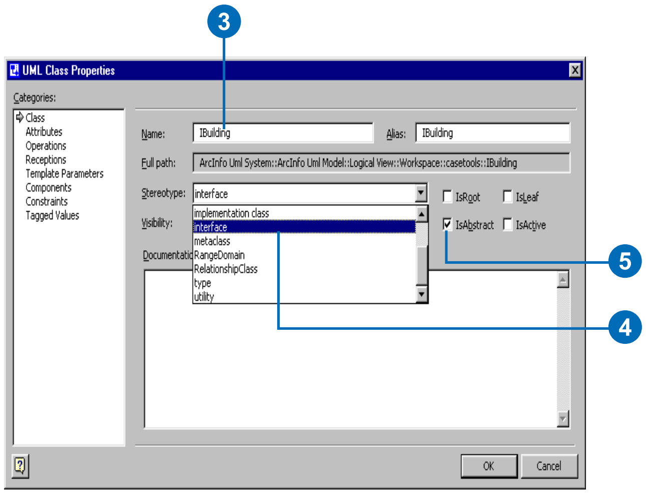
Interfaces are a set of related methods that a custom feature agrees to implement. A custom feature may implement any number of interfaces. Interfaces are inherited. An interface implemented by a parent class is also implemented by its children.

Types used in interfaces should be either other interfaces or automation-compatible types, such as long, double, DATE, and VARIANT_BOOL. UML attributes will become properties. They can be used as Object.Prop = value or var = Object.Prop. ▶

Creating an interface

1. In the UML Static Structure stencil, click Class and drag and drop it on the diagram.
2. Double-click the class.
3. Type the name of the interface in the Name box.
4. Click the Stereotype dropdown arrow and click interface.
5. Check the IsAbstract check box. ▶





Name: IBuilding
Alias: IBuilding
Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::casetools::IBuilding
Stereotype: interface
Visibility:
☐ IsRoot ☐ IsLeaf
☑ IsAbstract ☐ IsActive

UML operations will become methods. They can be used as Object.Method(argument:type).

To create a read-only property, create an operation with the prefix get_. Likewise, use the prefixes put_ or putref_ to create write-only properties.

You can model class extensions when you want custom behavior associated with the feature class. Class extensions require a naming convention. The name must be formed by concatenating the name of the feature class or table with ClassExtension.

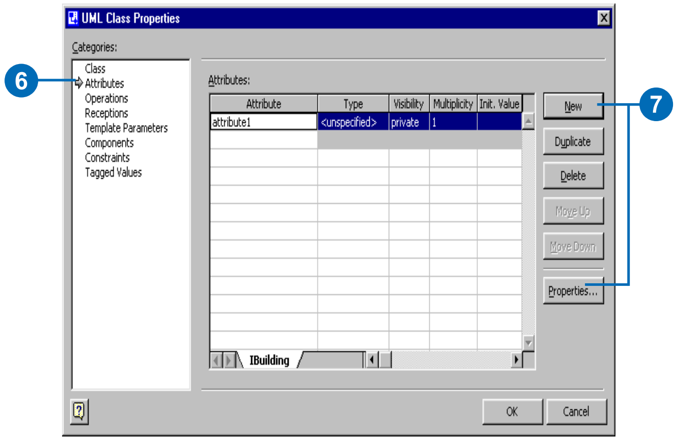
Class extensions for tables are derived from ObjectClassExtension. Likewise, class extensions for feature classes are derived from FeatureClassExtension.

Class extensions may implement optional interfaces, such as IObjectInspector. In addition, they may implement their own interfaces (for example, IBuildingClassExtension).

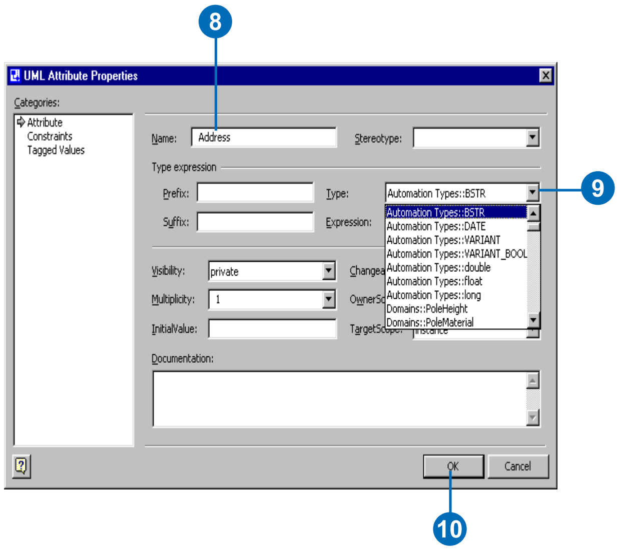
See Also

For a detailed explanation about creating custom features and class extensions, see Exploring ArcObjects.

6. Click Attributes in the Categories window.
7. Click New to add a new attribute and click Properties.
8. Type a name for the new property.
9. Click the Type dropdown arrow and click the field type.
10. Click OK.
11. Repeat steps 7 through 10 until you have added all the properties of the new interface. ▸



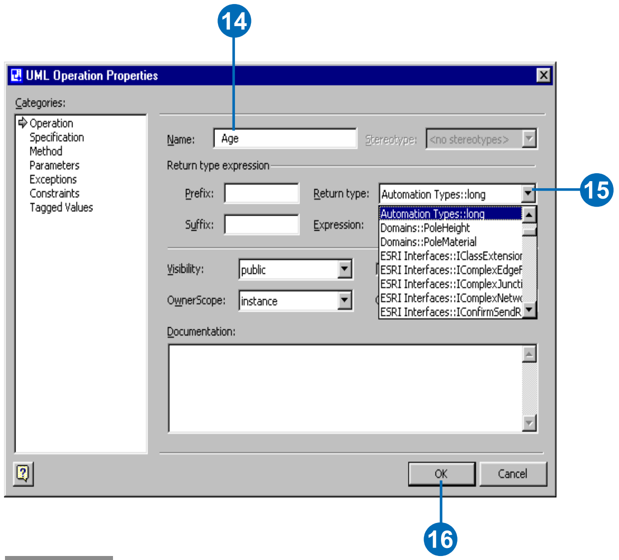
Attribute	Type	Visibility	Multiplicity	Init. Value
attribute1	<unspecified>	private	1	



Name: Address
Type: Automation Types::BSTR
Visibility: private
Multiplicity: 1
InitialValue:

Automation Types::BSTR
Automation Types::DATE
Automation Types::VARIANT
Automation Types::VARIANT_BOOL
Automation Types::double
Automation Types::float
Automation Types::long
Domains::PoleHeight
Domains::PoleMaterial

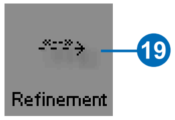
12. Click Operation in the Categories window.
13. Click New to add a new method and click Properties.
14. Type the name of the method.
15. Click the Return type dropdown arrow and click the method type if it returns something.
16. Click OK.
17. Repeat steps 13 through 16 until you have added all the methods of the new interface.
18. Click OK.
19. In the UML Static Structure stencil, click Refinement and drag and drop it on the diagram.
20. Connect the custom feature and the interface.

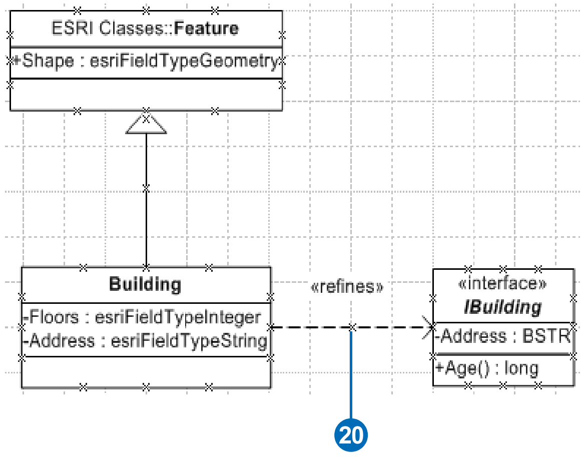

UML Operation Properties

Categories:
Operation
Specification
Method
Parameters
Exceptions
Constraints
Tagged Values

Name: Age
Stereotype: <no stereotypes>
Return type expression
Prefix:
Suffix:
Return type: Automation Types::long
Expression:
Automation Types::long
Domains::PoleHeight
Domains::PoleMaterial
ESRI Interfaces::IClassExtension
ESRI Interfaces::IComplexEdgeF
ESRI Interfaces::IComplexJuncti
ESRI Interfaces::IComplexNetw
ESRI Interfaces::IConfirmSendR
Visibility: public
OwnerScope: instance
Documentation:

OK
Cancel




ESRI Classes::Feature
+Shape : esriFieldTypeGeometry
Building
-Floors : esriFieldTypeInteger
-Address : esriFieldTypeString
«refines»
«interface»
IBuilding
-Address : BSTR
+Age() : long
20

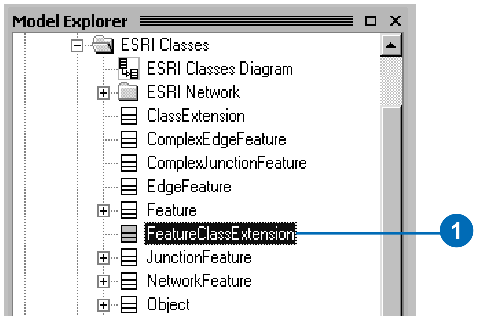
Tip

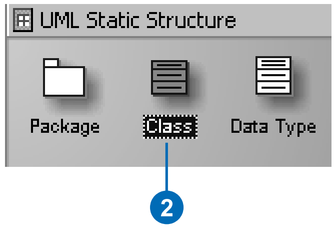
Class extension naming convention

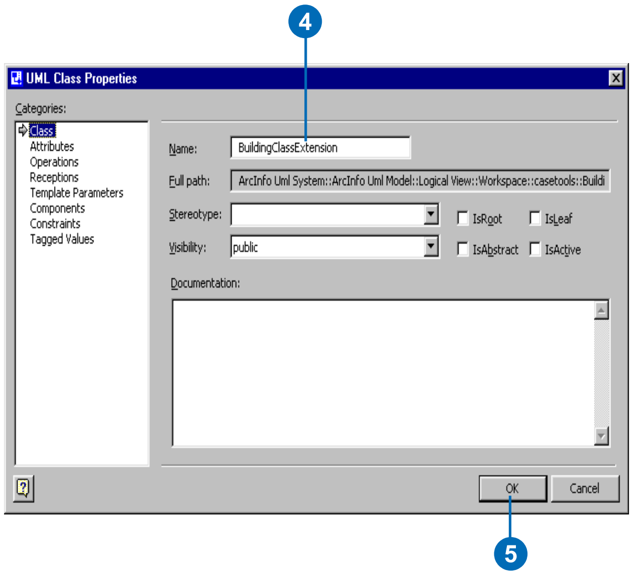
Class extensions must have the name <class>ClassExtension. For example, a class extension for a Parcels class would be ParcelsClassExtension.

Creating a class extension

1. In the Model Explorer tree under ESRI Classes, click FeatureClassExtension and drag and drop it onto the diagram.
2. In the UML Static Structure stencil, click Class and drag and drop a new UML class onto the diagram.
3. In the diagram, double-click the new class.
4. Type the name of the class following the class extension naming convention.
5. Click OK. ►







Name: BuildingClassExtension

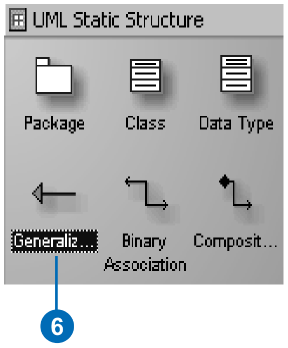
Full path: ArcInfo Uml System::ArcInfo Uml Model::Logical View::Workspace::casetools::Buildi

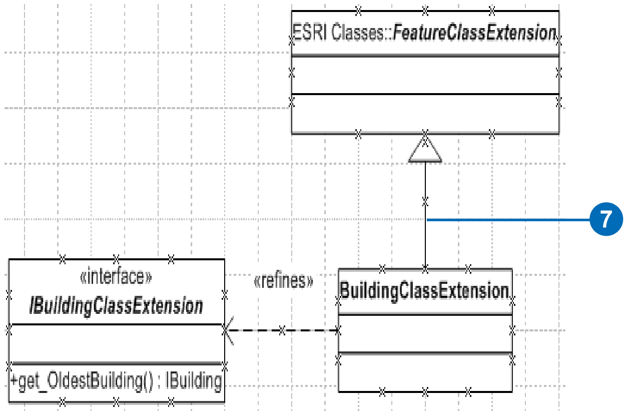
Stereotype:

Visibility: public

6. In the UML Static Structure stencil, click Generalization and drag and drop it onto the diagram.
7. Drag the ends of the generalization arrow and connect the new class with its parent.

To add your own interfaces, follow the instructions for creating an interface. There are a number of predefined optional interfaces you can implement in your class extension. ▶



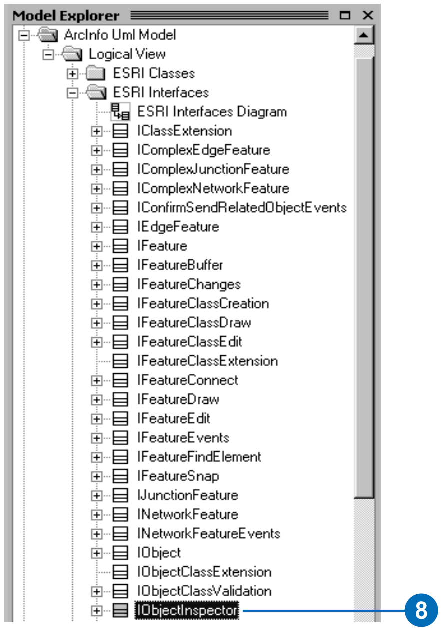

ESRI Classes::FeatureClassExtension
«interface»
IBuildingClassExtension
+get_OldestBuilding() : IBuilding
«refines»
BuildingClassExtension
7

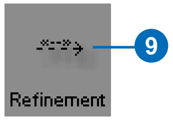
Tip

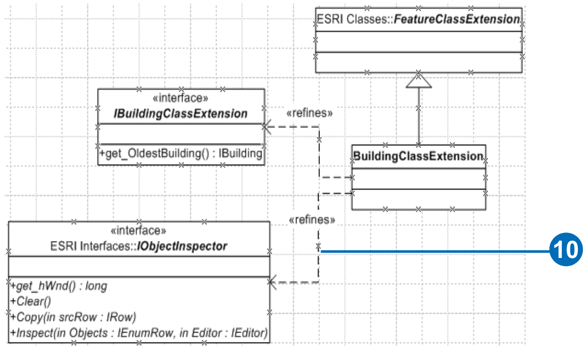
Optional interfaces

Your class extension can implement all or none of the optional class extension interfaces.

8. In the Model Explorer tree under ESRI Interfaces, click the optional interface and drag and drop it onto the diagram.
9. In the UML Static Structure stencil, click Refinement and drag and drop it onto the diagram.
10. Drag the ends of the refinement and connect the class extension with the interface.
11. Repeat steps 8 through 10 until you have added all of the optional interfaces you want to implement.






ESRI Classes::FeatureClassExtension
«interface»
IBuildingClassExtension
+get_OldestBuilding() : IBuilding
«refines»
BuildingClassExtension
«refines»
«interface»
ESRI Interfaces::IObjectInspector
+get_hWnd() : long
+Clear()
+Copy(in srcRow : IRow)
+Inspect(in Objects : IEnumRow, in Editor : IEditor)
10

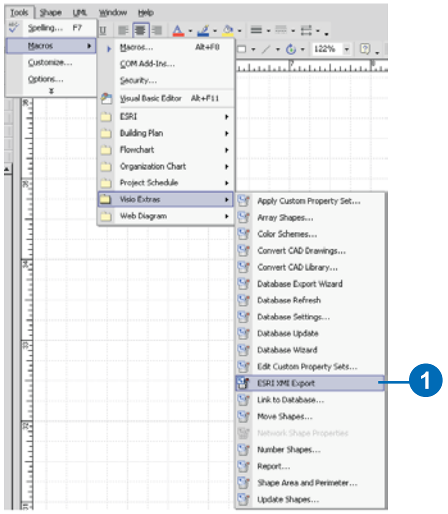
Exporting a model to XMI

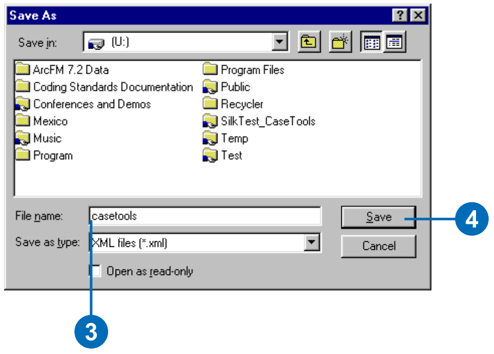
Before you can generate your geodatabase schema from your UML model, you must first export it to XMI.

The tools to export your UML model to XMI are contained within Visio.

Exporting to XMI

1. In Visio, click Tools, point to Macros, point to Visio Extras, then click ESRI XMI Export.
2. Browse to navigate to a directory where you will save your XMI file.
3. Type the name of the XMI file.
4. Click Save to export the UML model.
5. Click OK.





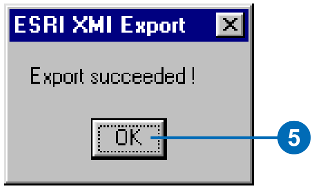
Save in: (U:)

ArcFM 7.2 Data
Coding Standards Documentation
Conferences and Demos
Mexico
Music
Program
Program Files
Public
Recycler
SilkTest_CaseTools
Temp
Test

File name: casetools

Save as type: XML files (*.xml)

Open as read-only



Export succeeded!

Checking your model for errors

The semantics checker can be used to make sure your models are valid. If errors are found, a report will be automatically created. The report can be printed or exported to a number of formats.

The semantics checker works on models already exported to the Microsoft Repository or XMI.

You should run the semantics checker before using the Schema Wizard or Code Generation Wizard.

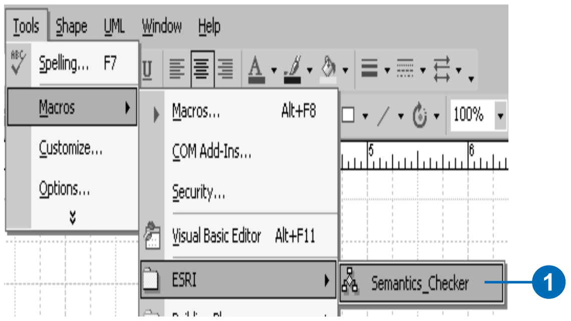
Tip

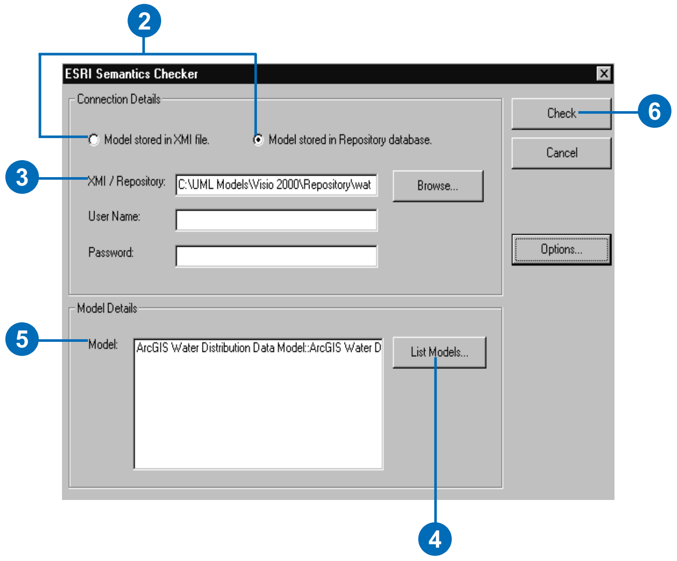
Options

Changing the options for the report allows you to modify the error report. By default the first two options will be checked.

Running the semantics checker

1. In Visio, click Tools, point to Macros, point to ESRI, then click Semantics_Checker.
2. Select the source of your model. Models can be stored in XMI files or repository databases.
3. Type the path to the XMI file or repository database or click Browse to navigate to it.
4. If you are using a repository, click List Models and select a model from the list.
5. Select the model you want to verify. ▸





ESRI Semantics Checker

Connection Details

Model stored in XMI file. [✓ Model stored in Repository database.]

XMI / Repository: C:\UML Models\Visio 2000\Repository\wat

User Name:

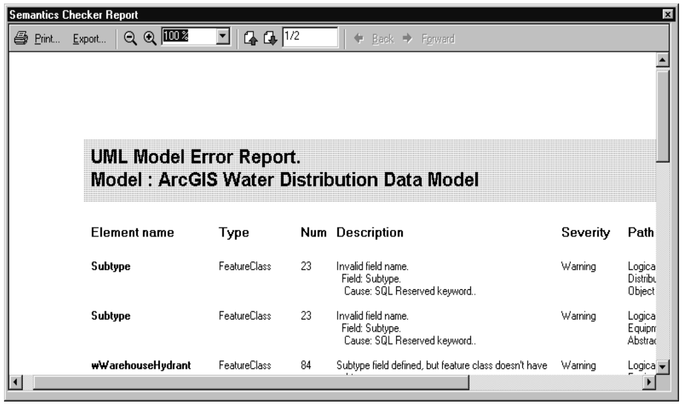
Password:

Model Details

Model: ArcGIS Water Distribution Data Model::ArcGIS Water D

6. Click Check.

   If errors are found, a report is generated listing all of the modeling errors.



UML Model Error Report.
Model : ArcGIS Water Distribution Data Model

Element name	Type	Num	Description	Severity	Path
Subtype	FeatureClass	23	Invalid field name. Field: Subtype. Cause: SQL Reserved keyword..	Warning	Logica Distribu Object
Subtype	FeatureClass	23	Invalid field name. Field: Subtype. Cause: SQL Reserved keyword..	Warning	Logica Equipm Abstrac
wWarehouseHydrant	FeatureClass	84	Subtype field defined, but feature class doesn't have	Warning	Logica

A report is generated listing all of the modeling errors.

Generating schema from an XMI or Microsoft Repository

ArcCatalog contains tools to read the XMI or Microsoft Repository database you created using the UML modeling software. The Schema Wizard guides you through the process of creating new feature classes, tables, and other pieces of your geodatabase.

Although all of the required information for the geodatabase schema can be read directly from the XMI or Microsoft Repository, you can change certain information. Once the wizard is finished, you will have the schema for your design ready to be populated with data.

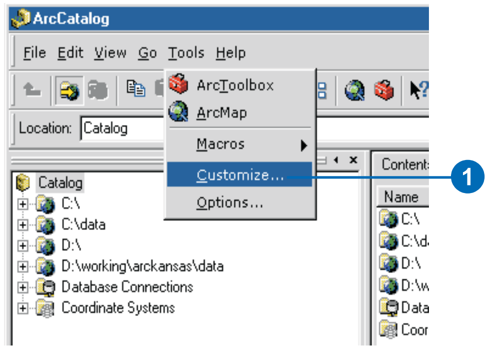
During the schema generation process, you will be presented with a tree view of the feature datasets, tables, feature classes, network feature classes, and geometric networks in the model.

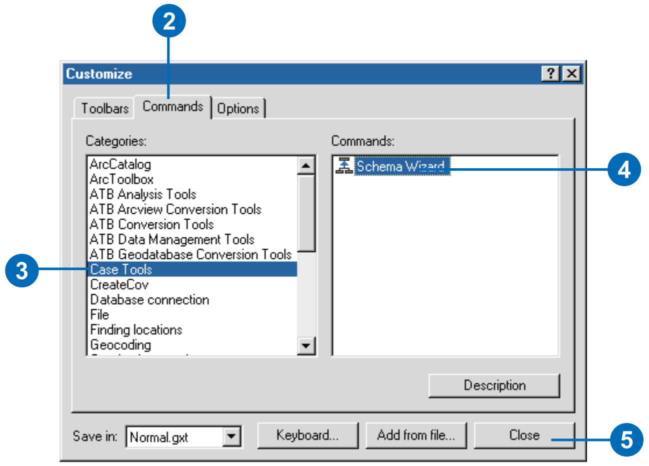
The examples here involve objects, features, and network features. Many of these objects and features contain subtypes with attribute domains and default values. The examples ▸

Adding the CASE Tools wizard to ArcCatalog

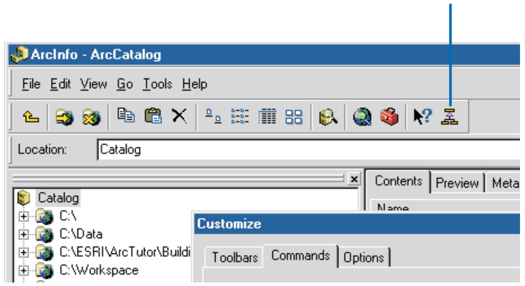
1. In ArcCatalog, click Tools and click Customize.
2. Click the Commands tab in the Customize dialog box.
3. Click Case Tools.
4. Drag the Schema Wizard command from the Commands list and drop it on the Standard toolbar.

   The command appears on the toolbar.
5. Click Close on the Customize dialog box.





The command appears on the toolbar.



also include relationship classes between some of the object and feature classes. However, a UML model can be as simple as containing a single feature or object class.

If the schema you are generating contains attribute domains, you can view the properties for these domains, but you cannot modify them.

Tip

Add the Schema Wizard to an existing toolbar

You can also add the Schema Wizard from the Commands list onto an existing toolbar.

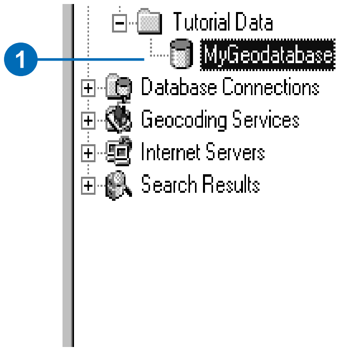
See Also

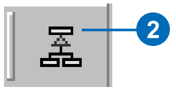
For more information on how to customize ArcCatalog, see Using ArcCatalog *and* Exploring ArcObjects.

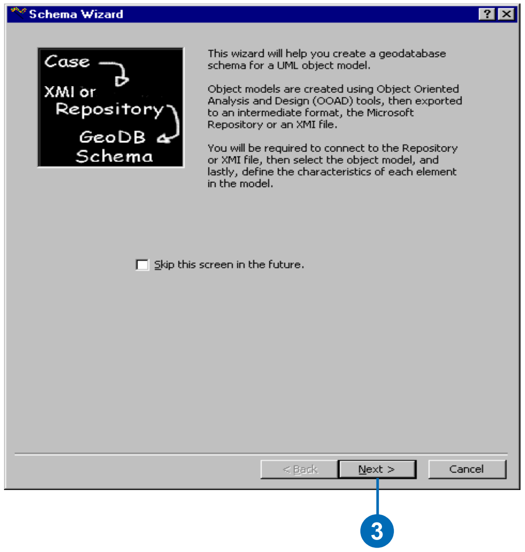
Connecting to an XMI file or repository

1. In the ArcCatalog tree, click the geodatabase in which you will create the schema.
2. Click the Schema Wizard button to start the Case Schema Creation Wizard.
3. A brief introduction to the wizard appears. Click Next.

   Clicking the check box in the introduction dialog box will allow you to skip the introduction dialog box.







This wizard will help you create a geodatabase schema for a UML object model.

Object models are created using Object Oriented Analysis and Design (OOAD) tools, then exported to an intermediate format, the Microsoft Repository or an XMI file.

You will be required to connect to the Repository or XMI file, then select the object model, and lastly, define the characteristics of each element in the model.

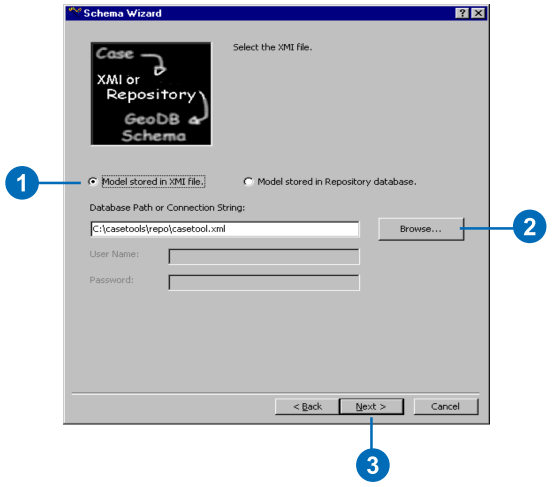
☐ Skip this screen in the future.

Selecting an XMI file

1. Click Model stored in XMI file in the Schema Wizard.
2. Type the name of the XMI file or click Browse to navigate to the XMI file you created.
3. Click Next.

   A tree view of the schema represented in the model is displayed.

   This may take several minutes, depending on the size of your object model.



Schema Wizard

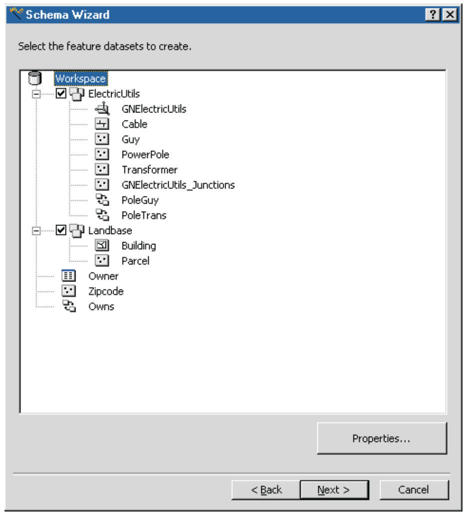
Select the XMI file.

(•) Model stored in XMI file. ( ) Model stored in Repository database.

Database Path or Connection String: C:\casetools\repo\casetool.xml

User Name:

Password:



Schema Wizard

Select the feature datasets to create.

```
Workspace
  [✓] ElectricUtils
        GNElectricUtils
        Cable
        Guy
        PowerPole
        Transformer
        GNElectricUtils_Junctions
        PoleGuy
        PoleTrans
  [✓] Landbase
        Building
        Parcel
      Owner
      Zipcode
      Owns
```

The geodatabase schema represented in the model is displayed in a tree view.

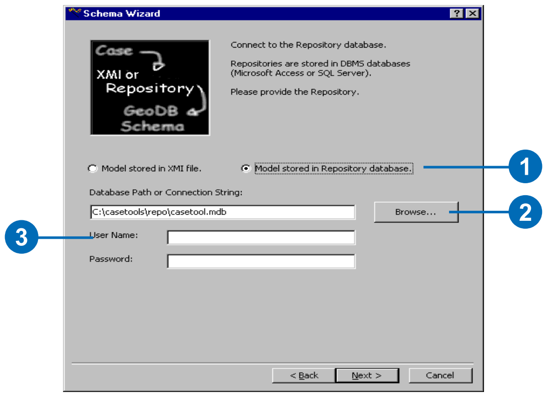
Tip

Microsoft Repository

If you are connecting to a repository stored in Microsoft SQL Server™, type the name of the open database communication (ODBC) data source that stores the information describing how to connect to the SQL Server database—for example, "dsn=casetools".

Connecting to a repository

1. Click Model stored in Repository database.
2. Type the name of the database or click Browse and navigate to the repository you created. To connect to a repository stored in SQL Server, type the ODBC data source.
3. In most cases you will not have to enter a username and password unless you specified one while exporting the UML model.
4. Click Next.



Connect to the Repository database.

Repositories are stored in DBMS databases (Microsoft Access or SQL Server).

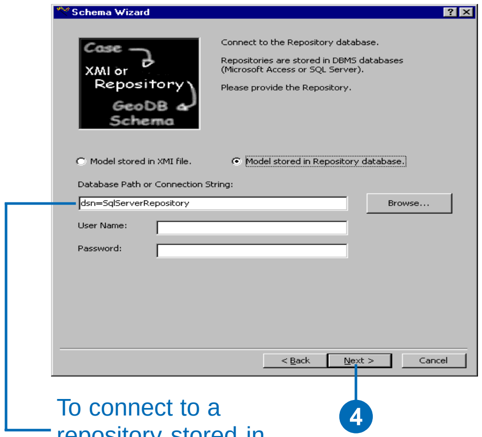
Please provide the Repository.

Model stored in XMI file. [✓ Model stored in Repository database.]

Database Path or Connection String: C:\casetools\repo\casetool.mdb

User Name:

Password:



Connect to the Repository database.

Repositories are stored in DBMS databases (Microsoft Access or SQL Server).

Please provide the Repository.

Model stored in XMI file. [✓ Model stored in Repository database.]

Database Path or Connection String: dsn=SqlServerRepository

User Name:

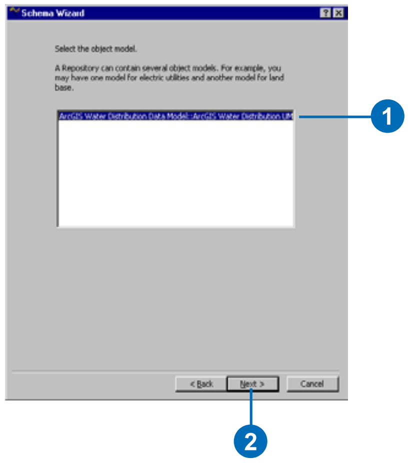
Password:

To connect to a repository stored in SQL Server, type the ODBC data source.

Selecting an object model

1. If you are using a Repository database, the wizard will let you select the model you want to create the schema for. Select the model.
2. Click Next.

   The wizard reads the model. This may take several minutes, depending on the size of your object model.



Select the object model.

A Repository can contain several object models. For example, you may have one model for electric utilities and another model for land base.

ArcGIS Water Distribution Data Model::ArcGIS Water Distribution UM

Setting feature dataset properties

The Schema Wizard dialog box displays your objects in a tree view. All of the datasets in your object model are selected and have check marks beside them by default. You will need to select each dataset in the object tree and specify the Spatial Reference.

Tip

Selecting feature datasets

When you select a feature dataset, the schema for all of the feature classes, geometric networks, and relationship classes contained in that feature dataset is also created.

Tip

Spatial reference

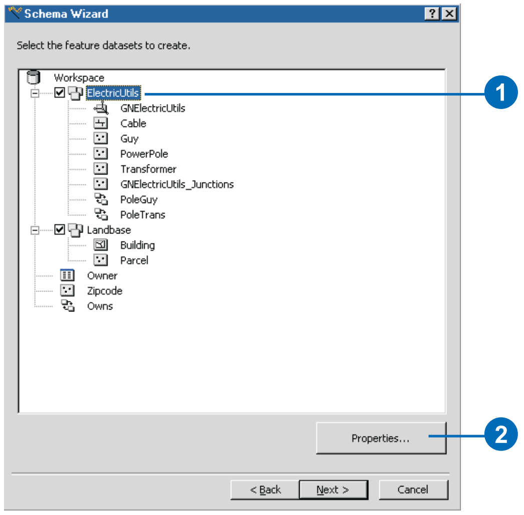
When you select a feature dataset and assign it a spatial reference, all of the feature classes within that feature dataset share the same spatial reference. The exception is the M-domain. Feature classes in the same feature dataset may have different M-domains.

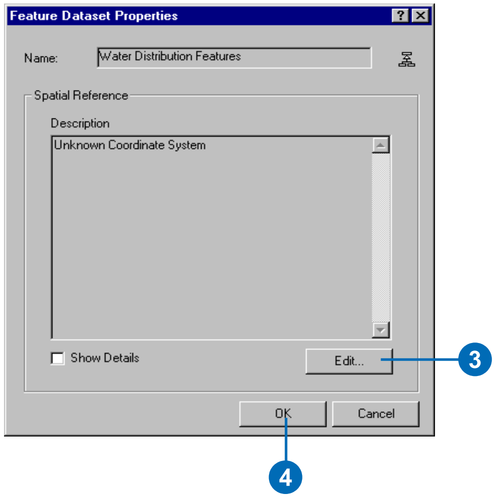
See Also

For a detailed explanation of spatial references, see Building a Geodatabase.

1. Check the feature dataset for which you want to generate schema.
2. Click the Properties button to set the properties for this feature dataset.

   Since you cannot model spatial references in UML, you will have to set the spatial reference for this feature dataset.
3. Click Edit to modify the spatial reference of the feature dataset.
4. Click OK once you have set the spatial reference.





Name: Water Distribution Features

Description: Unknown Coordinate System

Setting properties for object classes (tables)

After connecting to the Microsoft Repository or XMI file and selecting your model, you can set properties for the object classes (tables) from your model.

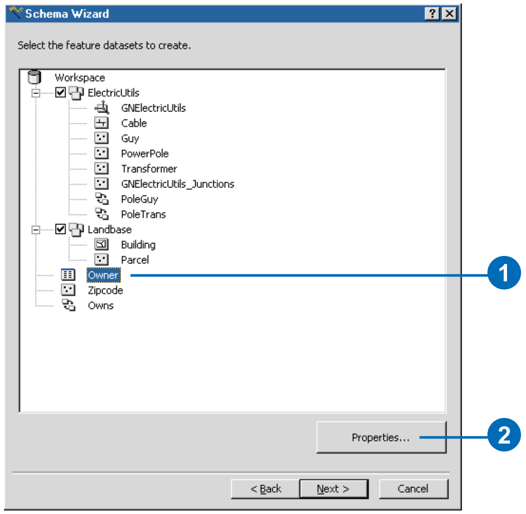
Tip

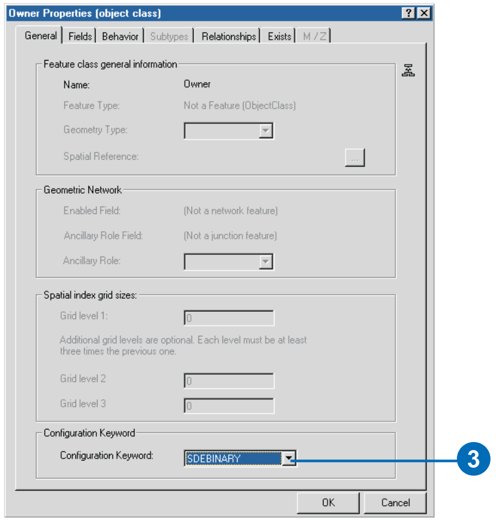
Classes that exist in a geodatabase

Classes in the object tree displayed in red indicate existing classes within the geodatabase.

1. Click the table whose properties you want to set.
2. Click Properties.

   Since this class does not store features, a number of properties are unavailable.
3. If you are generating schema for an ArcSDE™ geodatabase, click the dropdown arrow and click a storage configuration keyword. If you do not select a keyword, DEFAULTS will be used. ▸





Feature class general information
Name: Owner
Feature Type: Not a Feature (ObjectClass)
Geometry Type:
Spatial Reference:

Geometric Network
Enabled Field: (Not a network feature)
Ancillary Role Field: (Not a junction feature)
Ancillary Role:

Spatial index grid sizes:
Grid level 1: 0
Additional grid levels are optional. Each level must be at least three times the previous one.
Grid level 2 0
Grid level 3 0

Configuration Keyword
Configuration Keyword: SDEBINARY

Tip

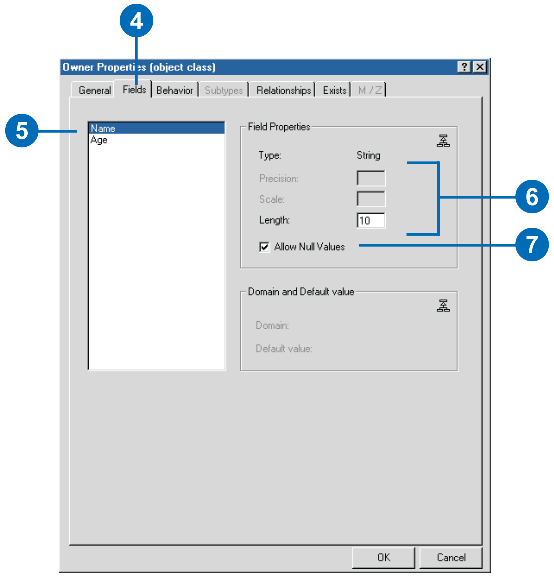
Related objects

If a class you select participates in a relationship class with another class, the related class is automatically selected in the Catalog tree.

See Also

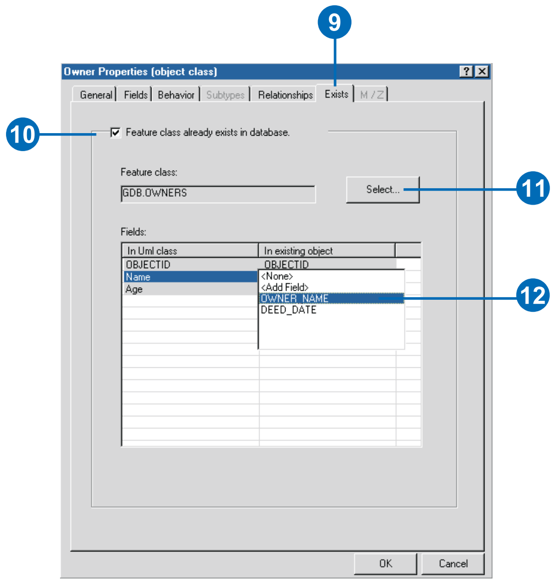
For more information about relationship class properties, see 'Setting properties for relationship classes'.

4. If the table in which you want to store this class does not already exist in the database, click the Fields tab. If the table does exist, skip to step 10.
5. Click a field in the list of fields. If you used tagged values in your UML model to set the field properties, skip to step 8.
6. Type its precision if it is an integer field, its precision and scale if it is a float field, or its length if it is a text field.
7. Check Allow Null Values if you want *null values* to be permitted in this field.
8. Repeat steps 5 through 7 for each field in the table, then skip to step 13.
9. Click the Exists tab if the table in which you want to store this class already exists in the database.
10. Check the check box to indicate that the table already exists.
11. Click Select to select the table from a list of tables in the database.
12. For each property in the UML class, click the field in the existing table that will store that property. ▸



Name
Age

Type: String
Precision:
Scale:
Length: 10
☑ Allow Null Values



☑ Feature class already exists in database.

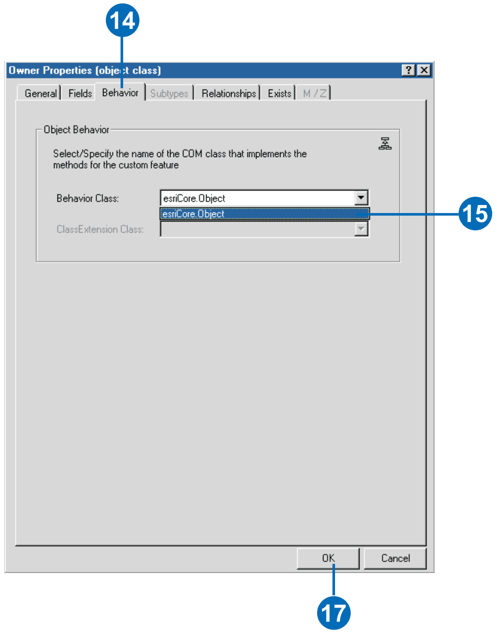
Feature class: GDB.OWNERS

Fields:

In Uml class	In existing object
OBJECTID	OBJECTID
Name	OWNER_NAME
Age	

<None>
<Add Field>
OWNER_NAME
DEED_DATE

13. If you chose to use ESRI-provided components for all of your classes, skip to step 17.
14. Click the Behavior tab.
15. Click the dropdown arrow to see a list of COM classes that are registered on your system. Click the COM class that implements the behavior for this table.
16. If this table has subtypes or relationship classes associated with it, examine their properties by clicking the Relationships and Subtypes tabs.
17. Click OK.
18. Repeat steps 2 through 17 for the rest of your tables.



Select/Specify the name of the COM class that implements the methods for the custom feature

Behavior Class: esriCore.Object

esriCore.Object

ClassExtension Class:

Setting properties for feature classes in a feature dataset

Once you have completed setting the properties for the object classes in your model, you must do the same for the feature classes in your model.

Tip

Spatial reference

When you select a feature dataset and assign it a spatial reference, all of the feature classes within that feature dataset share the same spatial reference.

The exception is the M-domain. Feature classes in the same feature dataset may have different M-domains.

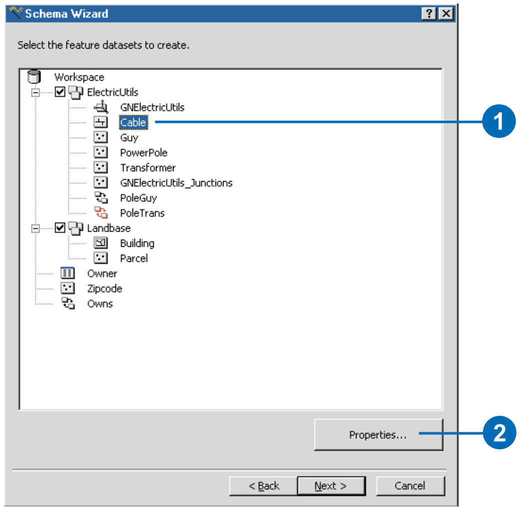
Standalone feature classes have their own spatial reference.

Tip

Selecting classes

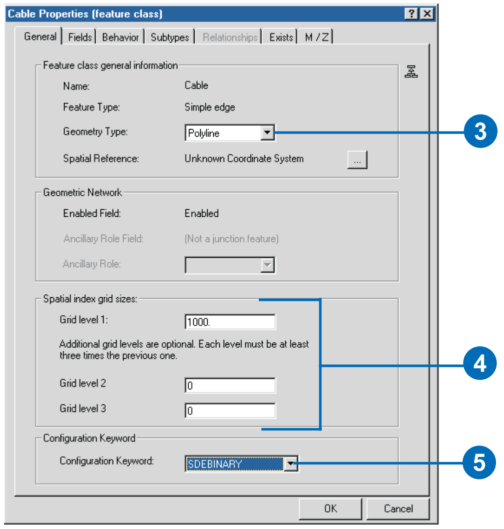
You can double-click the class to open the class properties dialog box.

1. Click the feature class whose settings you want to modify.
2. Click the Properties button to set the properties for this class.
3. Click the Geometry Type dropdown arrow and click the geometry type for this feature class.
4. Type the grid levels for the feature class. Personal geodatabase feature classes can have only one grid level.
5. If generating schema for an ArcSDE geodatabase, click the dropdown arrow and click a storage configuration keyword. If you do not select a keyword, DEFAULTS will be used.
6. Follow steps 5 through 18 of ‘Setting properties for object classes (tables)’ in this document.
7. Repeat steps 1 through 6 for the rest of the feature classes in your feature dataset.



Select the feature datasets to create.

```
Workspace
  ElectricUtils
    GNElectricUtils
    Cable
    Guy
    PowerPole
    Transformer
    GNElectricUtils_Junctions
    PoleGuy
    PoleTrans
  Landbase
    Building
    Parcel
  Owner
  Zipcode
  Owns
```



Feature class general information

Name:	Cable
Feature Type:	Simple edge
Geometry Type:	Polyline
Spatial Reference:	Unknown Coordinate System

Geometric Network

Enabled Field:	Enabled
Ancillary Role Field:	(Not a junction feature)
Ancillary Role:	

Spatial index grid sizes:

Grid level 1:	1000.
Grid level 2	0
Grid level 3	0

Additional grid levels are optional. Each level must be at least three times the previous one.

Configuration Keyword

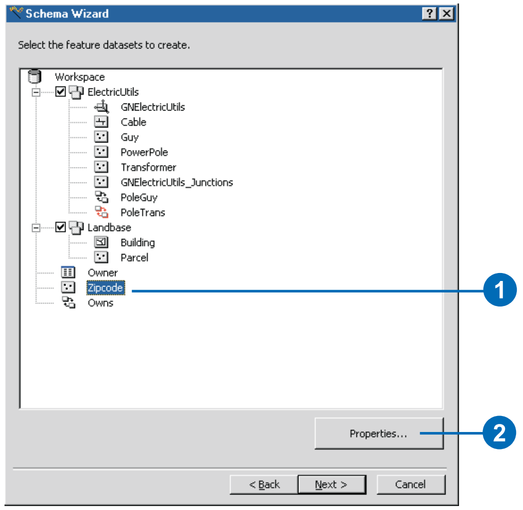
Configuration Keyword:	SDEBINARY

See Also

For a detailed explanation of spatial references and how to set them for objects in a geodatabase, see Building a Geodatabase.

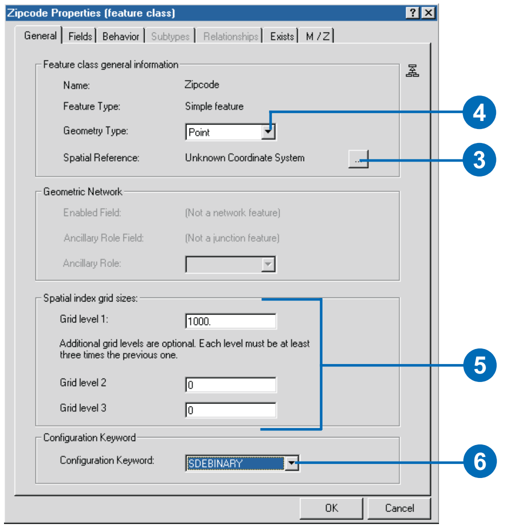
Setting properties for standalone feature classes

1. Click the feature class whose settings you want to modify.
2. Click the Properties button to set the properties for this class.
3. Click the button next to Spatial Reference to set the spatial reference for the feature class.
4. Click the Geometry Type dropdown arrow and click the geometry type for this feature class.
5. Type the grid levels for the feature class. Personal geodatabase feature classes can have only one grid level.
6. If generating schema for an ArcSDE geodatabase, click the Configuration Keyword dropdown arrow and click a storage configuration keyword. If you do not select a keyword, DEFAULTS will be used.
7. Follow steps 5 through 18 of 'Setting properties for object classes (tables)' in this document.
8. Repeat steps 1 through 6 for the rest of the standalone feature classes in your model.


Schema Wizard

Select the feature datasets to create.

- Workspace
  - ☑ ElectricUtils
    - GNElectricUtils
    - Cable
    - Guy
    - PowerPole
    - Transformer
    - GNElectricUtils_Junctions
    - PoleGuy
    - PoleTrans
  - ☑ Landbase
    - Building
    - Parcel
  - Owner
  - Zipcode
  - Owns


Zipcode Properties (feature class)

Feature class general information

Name:	Zipcode
Feature Type:	Simple feature
Geometry Type:	Point
Spatial Reference:	Unknown Coordinate System

Geometric Network

Enabled Field:	(Not a network feature)
Ancillary Role Field:	(Not a junction feature)
Ancillary Role:	

Spatial index grid sizes:

Grid level 1:	1000.

Additional grid levels are optional. Each level must be at least three times the previous one.

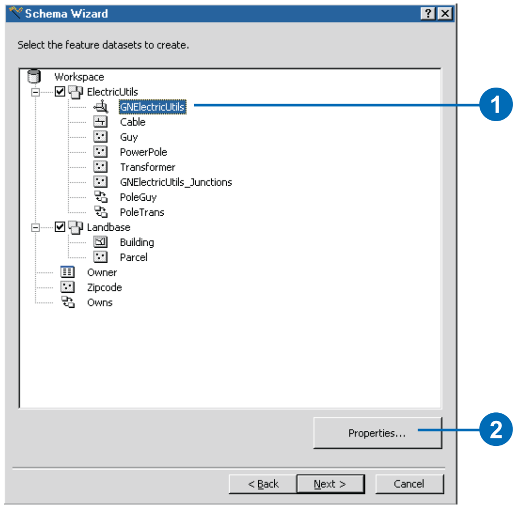
Grid level 2	0
Grid level 3	0

Configuration Keyword

Configuration Keyword:	SDEBINARY

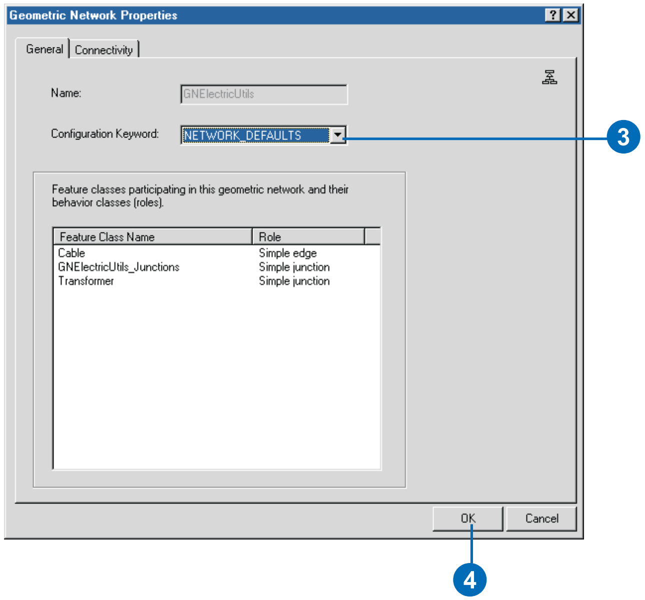
Setting properties for a geometric network

1. Click the geometric network whose settings you want to modify.
2. Click the Properties button to set the properties for this network.
3. If generating schema for an ArcSDE geodatabase, click the dropdown arrow and click a storage configuration keyword.
4. Click OK.


Schema Wizard

Select the feature datasets to create.

```
Workspace
  ElectricUtils
    GNElectricUtils
    Cable
    Guy
    PowerPole
    Transformer
    GNElectricUtils_Junctions
    PoleGuy
    PoleTrans
  Landbase
    Building
    Parcel
  Owner
  Zipcode
  Owns
```


Geometric Network Properties

General | Connectivity

Name: GNElectricUtils

Configuration Keyword: NETWORK_DEFAULTS

Feature classes participating in this geometric network and their behavior classes (roles).

Feature Class Name	Role
Cable	Simple edge
GNElectricUtils_Junctions	Simple junction
Transformer	Simple junction

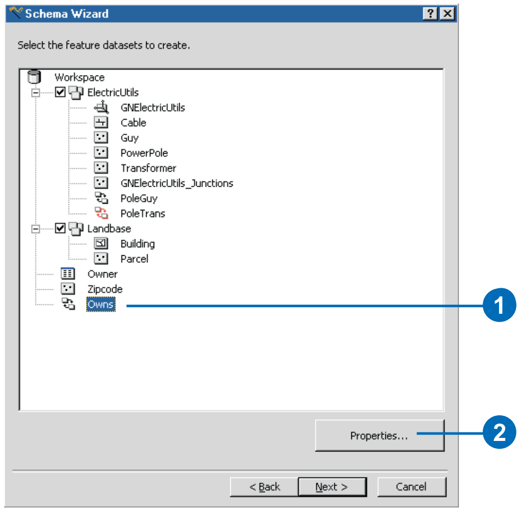
Setting properties for relationship classes

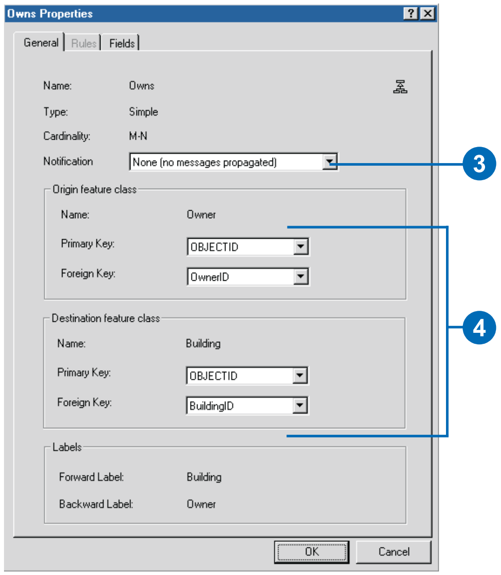
If you did not use tagged values for the properties of relationship classes in your UML model, you can set them in the Schema Creation Wizard.

See Also

For a detailed discussion of primary and foreign keys and relationship messaging, see Building a Geodatabase.

1. Click the relationship class for which you want to set the properties.
2. Click Properties.
3. Click the Notification dropdown arrow and click the notification direction for the relationship class.
4. Click the dropdown arrows and click the origin primary and foreign keys. ▸





Name:	Owns
Type:	Simple
Cardinality:	M-N
Notification	None (no messages propagated)

Origin feature class

Name:	Owner
Primary Key:	OBJECTID
Foreign Key:	OwnerID

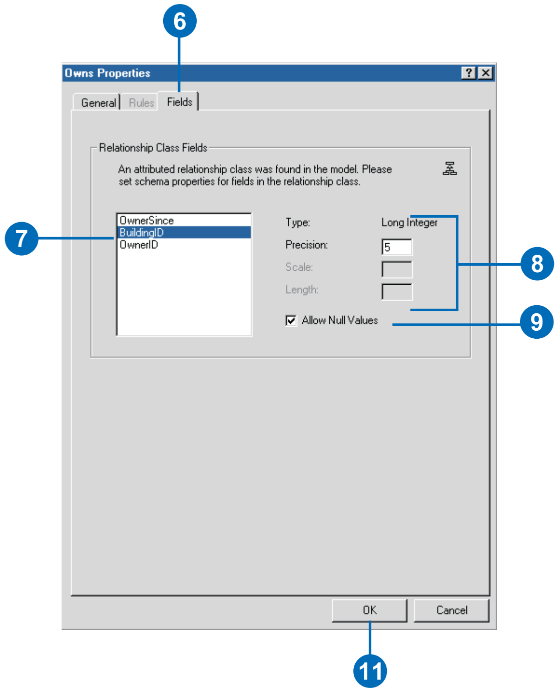
Destination feature class

Name:	Building
Primary Key:	OBJECTID
Foreign Key:	BuildingID

Labels

Forward Label:	Building
Backward Label:	Owner

5. If your relationship class is not modeled as a class with attributes, skip to step 11.
6. Click the Fields tab.
7. Click a field in the list of fields. If you used tagged values in your UML model to set the field properties, skip to step 11.
8. Type its precision if it is an integer field, its precision and scale if it is a float field, or its length if it is a text field.
9. Click Allow Null Values if you want null values to be permitted in this field.
10. Repeat steps 1 through 9 for each field in the relationship class.
11. Click OK.
12. Repeat steps 1 through 11 for the rest of your relationship classes.



An attributed relationship class was found in the model. Please set schema properties for fields in the relationship class.

Type: Long Integer

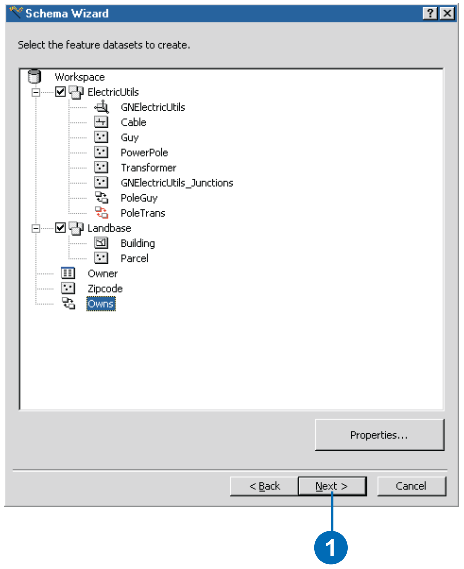
Precision: 5

☑ Allow Null Values

Creating the schema

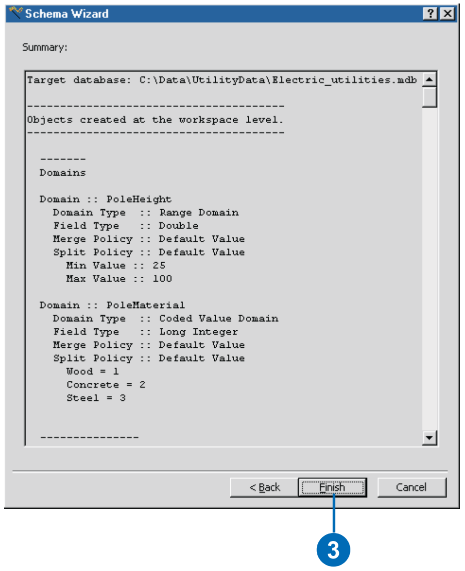
The last step of the wizard provides a summary of the elements that will be created. After clicking Finish, the creation of the schema will take place.

1. Click Next.
2. Review the options you specified in the Schema Wizard. If you want to change anything, click Back and change the appropriate parameters.
3. Click Finish to generate the schema in the geodatabase.



Select the feature datasets to create.

- Workspace
  - ☑ ElectricUtils
    - GNElectricUtils
    - Cable
    - Guy
    - PowerPole
    - Transformer
    - GNElectricUtils_Junctions
    - PoleGuy
    - PoleTrans
  - ☑ Landbase
    - Building
    - Parcel
  - Owner
  - Zipcode
  - Owns



Summary:

```
Target database: C:\Data\UtilityData\Electric_utilities.mdb

---------------------------------------
Objects created at the workspace level.
---------------------------------------

  -------
  Domains

  Domain :: PoleHeight
    Domain Type  :: Range Domain
    Field Type   :: Double
    Merge Policy :: Default Value
    Split Policy :: Default Value
      Min Value :: 25
      Max Value :: 100

  Domain :: PoleMaterial
    Domain Type  :: Coded Value Domain
    Field Type   :: Long Integer
    Merge Policy :: Default Value
    Split Policy :: Default Value
      Wood = 1
      Concrete = 2
      Steel = 3


  ---------------
```