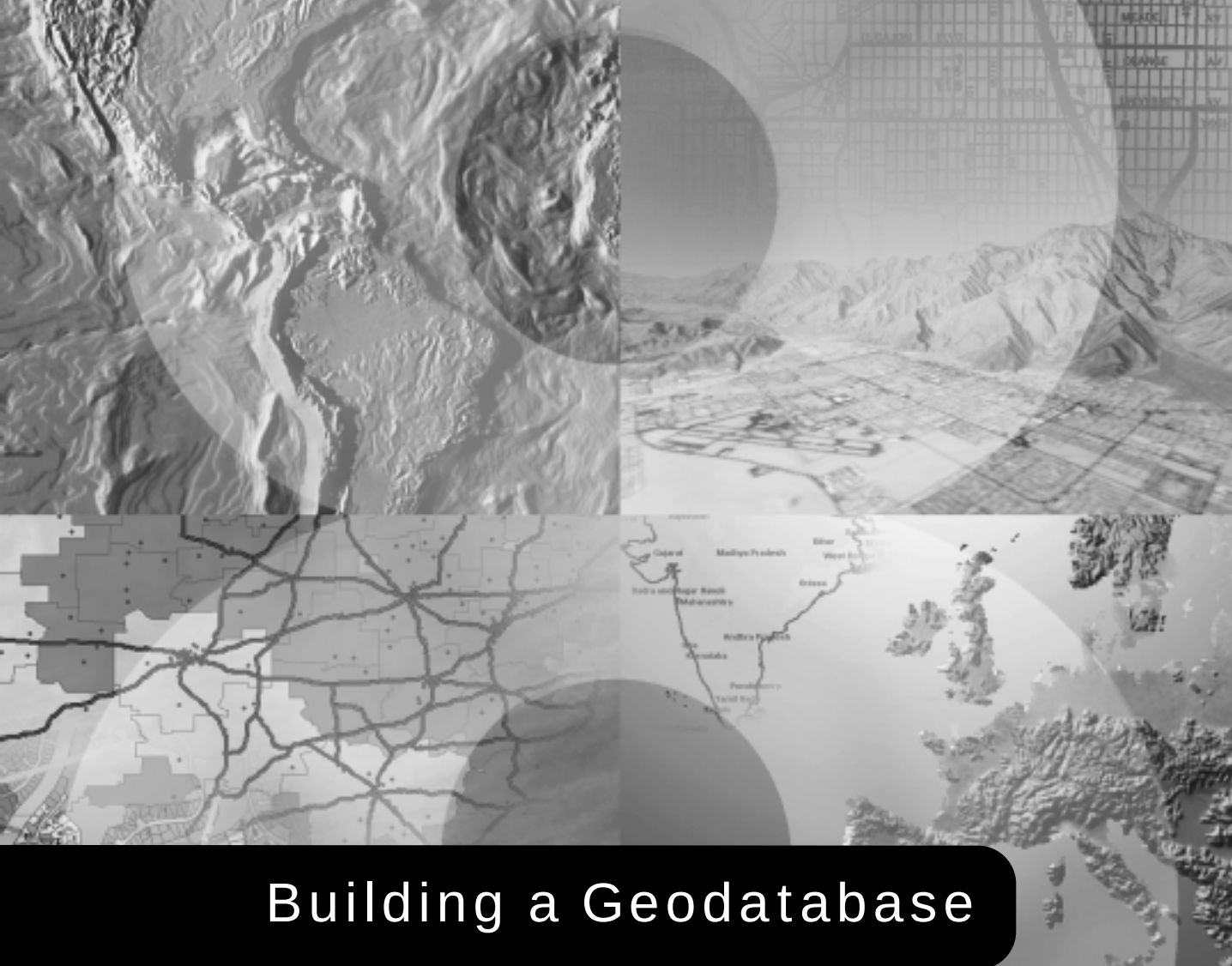




Building a Geodatabase

GIS by ESRI™

Copyright © 1999–2002 ESRI
All rights reserved.
Printed in the United States of America.

The information contained in this document is the exclusive property of ESRI. This work is protected under United States copyright law and other international copyright treaties and conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage or retrieval system, except as expressly permitted in writing by ESRI. All requests should be sent to Attention: Contracts Manager, ESRI, 380 New York Street, Redlands, CA 92373-8100, USA.

The information contained in this document is subject to change without notice.

DATA CREDITS

Graphical Editing Map: Wilson, North Carolina

Universal Data Editor Map, Editing in data view and layout view map: Greeley, Colorado

Context menus and shortcut keys map: P.F.R.A., Regina, Saskatchewan, Canada

Quick-Start Tutorial Data: Wilson, North Carolina; Greeley, Colorado

CONTRIBUTING WRITERS

Bob Booth, Scott Crosier, Jill Clark, Andy MacDonald

U.S. GOVERNMENT RESTRICTED/LIMITED RIGHTS

Any software, documentation, and/or data delivered hereunder is subject to the terms of the License Agreement. In no event shall the U.S. Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR §52.227-14 Alternates I, II, and III (JUN 1987); FAR §52.227-19 (JUN 1987) and/or FAR §12.211/12.212 (Commercial Technical Data/Computer Software); and DFARS §252.227-7015 (NOV 1995) (Technical Data) and/or DFARS §227.7202 (Computer Software), as applicable. Contractor/Manufacturer is ESRI, 380 New York Street, Redlands, CA 92373-8100, USA.

ESRI, the ESRI globe logo, ArcGIS, ArcMap, ArcCatalog, ArcInfo, ArcSDE, ArcToolbox, ArcEditor, ArcStorm, SDE, Spatial Database Engine, ArcView, StreetMap, ArcObjects, GIS by ESRI, the ArcGIS logo, and www.esri.com are trademarks, registered trademarks, or service marks of ESRI in the United States, the European Community, or certain other jurisdictions.

Other companies and products mentioned herein are trademarks or registered trademarks of their respective trademark owners.

Contents

1	Introduction	1
	Before you create your geodatabase	3
	Three ways to create a geodatabase	4
	Geodatabases and ArcCatalog	8
	Geodatabases and ArcMap	9
	The first step: creating a database	10
	Tips on learning how to build and edit geodatabases	15
2	Creating new items in a geodatabase	17
	Geodatabase items	18
	ArcGIS data types	23
	Upgrading a geodatabase	26
	Creating tables	27
	Creating feature datasets	31
	Creating feature classes	37
	Creating indexes	45
	Granting and revoking privileges	48
3	Migrating existing data into a geodatabase	49
	How data is converted	50
	Importing shapefiles	62
	Importing coverages	68
	Importing tables	75
	Importing a geodatabase feature class	77
	Importing a CAD feature class	80
	Importing rasters	81
	Copying geodatabase data	87
	Extracting data	89
	Using the Extract Data wizard	91
	Loading data into existing simple feature classes and tables	94
	Registering ArcSDE layers and tables with the geodatabase	98
	Analyzing geodatabase data	99
	Loading objects from other feature classes	100

4 Topology 107

- What is topology? 109
- Creating a topology 110
- Topology basics 112
- Topology and feature geometry 115
- Topologies and ArcCatalog 117
- Migrating data into a geodatabase to create topologies 118
- Creating a new topology 121
- Adding new feature classes to your topology 126
- Validating a topology 130
- Topology: defining the rules 131
- Planning for exceptions 135
- Refining topologies with subtypes 136
- Managing a topology 137
- Modifying a topology 138
- Summarizing topology errors 147
- Creating new polygons from lines 148
- Topology and versioned databases 150
- Topology and versioning 152
- Topology and disconnected editing 162

5 Subtypes and attribute domains 163

- What are subtypes and attribute domains? 164
- Working with attribute domain properties 168
- Browsing the attribute domains of a geodatabase 169
- Creating new attribute domains 171
- Modifying and deleting attribute domains 174
- Associating default values and domains with tables and feature classes 175
- Creating subtypes 176
- Modifying and deleting subtypes 179

6 Defining relationship classes 181

- What is a relationship class? 182
- Relationship classes in ArcCatalog and ArcMap 186
- Creating a simple relationship class 188
- Creating a composite relationship class 192
- Creating an attributed relationship class 195
- Creating relationship rules 197
- Managing relationship classes 199
- Exploring related objects in ArcMap 200
- Using related fields in ArcMap 203

7 Geometric networks 205

- What is a geometric network? 206
- Geometric networks and ArcCatalog 210
- Creating geometric networks 211
- Creating a new geometric network 216
- Building a geometric network from existing simple feature classes 220
- Adding new feature classes to your geometric network 228
- Network connectivity: defining the rules 231
- Establishing connectivity rules 232
- Managing a geometric network 234

8 Managing annotation 235

- Annotation in the geodatabase 236
- Annotation and ArcCatalog 239
- Creating annotation classes 240
- Converting labels to annotation 245
- Converting coverage annotation to geodatabase annotation 247

9 Dimensioning 249

- Dimensions in the geodatabase 250
- Dimensions and ArcCatalog 253
- Creating dimension feature classes 254
- Creating and managing dimension styles 259

10 Geocoding services 269

- Geocoding services 270
- Geocoding services in ArcCatalog and ArcMap 272
- Preparing reference data for a geocoding service 274
- Creating a geocoding service 279
- Maintaining geocoding indexes 282
- Working with geocoding indexes 284
- Preparing address data for geocoding 290

11 Working with a versioned geodatabase 293

- Integrating versioning with your organization's work flow 294
- Registering data as versioned 296
- Creating and administering versions in ArcCatalog 297
- Working with versions in ArcMap 304
- Editing and conflict resolution 307
- Editing a version 312
- Versioning scenarios 316

12 Disconnected editing 319

- Disconnected editing 320
- Checking out data from a geodatabase 338
- Customizing a check-out 340
- Checking in data to a geodatabase 343
- Managing check-outs 346

13 Building geodatabases with CASE tools 353

- What are CASE tools? 354
- Designing the object model in Microsoft Visio 365
- Creating UML packages and static structure diagrams 366
- Setting tagged values 367
- Creating feature datasets 369
- Creating feature classes 370
- Creating relationship classes 374
- Creating domains 379
- Creating subtypes 384
- Creating relationship rules 390
- Creating geometric networks 392
- Creating connectivity rules 394
- Extending classes with custom behavior 399
- Exporting a model to XMI 405
- Checking your model for errors 406
- Generating schema from an XMI or Microsoft Repository 408
- Setting feature dataset properties 413
- Setting properties for object classes or tables 414
- Setting properties for feature classes in a feature dataset 417
- Setting properties for relationship classes 420
- Creating the schema 422

Glossary 423

Index 445

Introduction

1

IN THIS CHAPTER

- Before you create your geodatabase
- Three ways to create a geodatabase
- Geodatabases and ArcCatalog
- Geodatabases and ArcMap
- The first step: creating a database
- Tips on learning how to build and edit geodatabases

The *geodatabase* supports a model of topologically integrated *feature classes*, similar to the *coverage* model. It also extends the coverage model with support for complex networks, topologies, *relationships* among feature classes, and other object-oriented *features*. The ESRI® ArcGIS™ applications (ArcMap™, ArcCatalog™, and ArcToolbox™) work with geodatabases as well as with coverages and *shapefiles*.

The ArcGIS geodatabase model is implemented on standard relational databases with the ArcSDE® application *server*. ArcSDE defines an open interface to *database* systems for our users. It allows ArcInfo™ or ArcEditor™ seats to manage geographic information on a variety of different database platforms including Oracle®, Microsoft® SQL Server™, IBM® DB2®, and Informix®.

The geodatabase model defines a generic model for geographic information. This generic model can be used to define and work with a wide variety of different user- or application-specific models. By defining and implementing a wide variety of *behavior* on a generic geographic model, ESRI provides a robust platform for the definition of a variety of user data models.

The geodatabase model supports an object-oriented vector data model. In this model, entities are represented as *objects* with properties, behavior, and relationships. Support for a variety of different geographic object types is built into the system. These object types include simple objects, geographic features (objects with location), network features (objects with geometric integration with other features), topologically related features, *annotation* features, and other more specialized feature types. The model

allows you to define relationships between objects and rules for maintaining referential and topological integrity between objects.

The main tools you will use to create and edit geodatabases are found in ArcCatalog and ArcMap. ArcCatalog has various tools for creating and modifying your geodatabase *schema*, while ArcMap has tools for analyzing and editing the contents of your geodatabase.

This book is one of three books designed to teach you how to make the most of geodatabases. In this book, principles of designing and implementing a geodatabase are covered. The second book, *Editing in ArcMap*, describes how to build and edit *data* in a geodatabase. The third book is a workbook that provides hands-on exercises that allow you to apply the concepts developed in the first two books.

This book, *Building a Geodatabase*, approaches the database from a geodatabase designer and implementer's perspective. It describes how to implement the special functionality available in geodatabases, including *subtypes*, relationships, default values, domains, *topology*, geometric networks, annotation, and dimensioning.

The second book approaches the geodatabase from the editor and data manager's perspective. It describes how to build and edit data within an existing geodatabase.

The third book contains tutorial exercises that first let you work with a completed geodatabase, reviews techniques for editing data in a geodatabase, then systematically shows you how to implement the geodatabase behavior you used

in the first section. The exercises are designed to correspond to material covered in the first two books.

Successfully implementing a multiuser geographic information system (GIS) with ArcInfo and ArcSDE starts with a good data model design and database tuning. How the data is stored in the database, the applications that access it, and the client and server hardware configurations are all key factors to a successful multiuser GIS system. Designing a geodatabase is a critical process that requires planning and revision until you reach a design that meets your requirements and performs well. Throughout this book, guidelines for good data modeling of each aspect of the geodatabase are discussed to help you implement a successful multiuser GIS system with ArcInfo, either with ArcSDE or with a *personal geodatabase*.

A critical part of a well-performing geodatabase is the tuning of the database management system (DBMS) in which it is stored. This tuning is not required for personal geodatabases; however, it is critical for ArcSDE geodatabases. For more information on tuning your database for ArcSDE and the geodatabase, see the *Configuration and Tuning Guide for <DBMS>* PDF file.

Once you have a design, you can create the geodatabase and its schema by loading existing shapefile and coverage data, creating new database *items* with ArcCatalog, using Unified Modeling Language (UML) and Computer-Aided Software Engineering (CASE) tools, or a combination of all three.

Before you create your geodatabase

One of the most important steps in creating an effective database is designing its schema, the structure and relationships of the data that it contains. The same is true for any geodatabase. When designing a geodatabase, you should consider questions such as:

- What kind of data will be stored in the database?
- In what *projection* do you want your data stored?
- Do you want to establish *rules* about how the data can be modified?
- How do you want to organize your *object classes* such as *tables*, feature classes, and subtypes of feature classes?
- Do you want to maintain relationships between objects of different types?
- Will your database contain geometric networks?
- Will your database contain topologically related features?
- Will your database store custom objects?

Once you have answered these and other questions, you are ready to begin creating your geodatabase design. You can use the data modeling guidelines in this book to help you design a geodatabase that meets your requirements and also performs well. This book will then guide you through the process of physically implementing your geodatabase design.

ESRI and a number of our users have been actively engaged in designing a series of GIS data models using topology and other capabilities in ArcGIS. These design efforts have resulted in a series of comprehensive design specifications for a number of thematic *layers*, including:

- Census and Administrative Units (applied to U.S. Census geography)
- Topographic basemaps for 1:24,000-scale maps
- Hydrography
- *Raster* imagery and elevation catalogs
- Streets and comprehensive address information
- Transportation (to support linear referencing, navigation, addressing, and cartography)
- Public Land Survey System (PLSS) (to support a national database of the legal survey fabric)
- Parcels (to support both U.S. and worldwide systems)
- Water facilities
- And numerous other efforts.

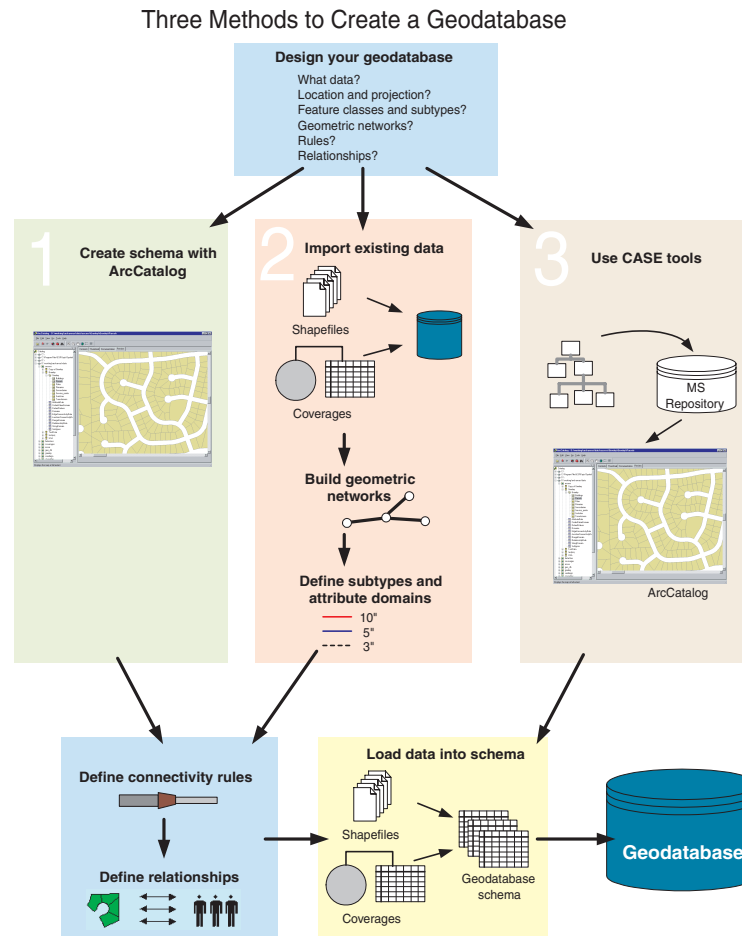
These data models were developed to provide a common design framework for key layers of geographic information, in order to promote openness and interoperability of GIS data. For more information on the current state of the data models, see <http://support.esri.com>.

Three ways to create a geodatabase

Once you have designed your geodatabase, you can employ any of three methods to create a new geodatabase. The method you choose will depend on what the source of your geodatabase data is, whether you will store custom objects in the geodatabase, or whether you intend to create a new geodatabase from scratch. In

practice, you will often use a combination of all or some of the methods outlined.

The three methods of creating a geodatabase are discussed briefly here. Subsequent chapters will outline how each task is performed.



The first step is always to design the geodatabase. This book and the book *Modeling Our World* are guides to help you design your geodatabase. Once this design is complete, you can proceed with the method that best suits your situation.

Design your geodatabase

What data?
Location and projection?
Feature classes and subtypes?
Geometric networks?
Rules?
Relationships?

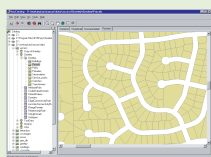
Creating a new geodatabase from scratch

In some cases, you may not yet have any data that you want to load into a geodatabase or the data you have to load only

accounts for part of your database design. In this case, you can use the tools provided in ArcCatalog to create the schema for *feature datasets*, feature classes, tables, *geometric networks*, topologies, and other items inside the database.

ArcCatalog provides a complete set of tools for designing and managing items you will store in the geodatabase.

1 Create schema with ArcCatalog



Migrating existing data into the geodatabase

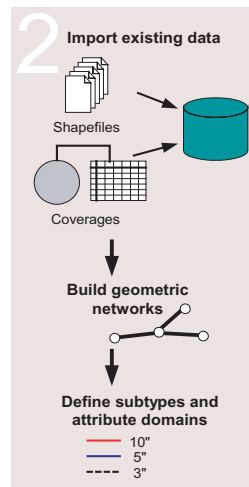
It is likely that you already have data in various formats—*shapefiles*, coverages, INFO™ tables, and dBASE® tables—that

you want to store in a geodatabase.

You may also have your data stored in other multiuser geographic information system data formats such as ArcStorm™, Map LIBRARIAN, and ArcSDE.

Through ArcCatalog, you can convert data stored in one of these formats to a geodatabase by importing it. A series of dialog boxes will guide you through the conversion process. Once you have become familiar with this process, more advanced batch data converters can be used to perform these operations more efficiently.

When converting data from one of these formats into the geodatabase, both the spatial and nonspatial



component of each object is translated. For example, when converting a shapefile to a feature class, both the shapes (geometry) and attributes are stored in the geodatabase. Attributes can be left out or renamed. Shapefiles of the same spatial *extent* can be imported into the same feature dataset. All or some of the feature classes from a coverage can be imported into a feature dataset. *Topology rules* can be created to regulate the spatial relationships between the features and feature classes stored in geodatabase feature datasets.

Converting ArcStorm and Map LIBRARIAN data is done using tools that are similar to those used for importing coverages. However, you must use ArcSDE for Coverages before ArcCatalog

or ArcToolbox can access and display ArcStorm and Map LIBRARIAN data.

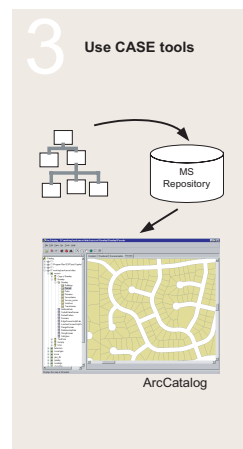
If you already have your data in a Spatial Database Engine™ (SDE®) 3.x database, you do not need to reload your data. ArcCatalog contains tools that allow you to register the existing data with the geodatabase. Once registered, you can also use ArcCatalog to reorganize that data into feature datasets.

ArcGIS and geodatabases do not support multiple feature types in a single feature class (for example, *points* and lines in the same feature class). If any of your SDE 3.x layers contain multiple-entity types, those must be reorganized into single feature type layers before you can view them in ArcInfo or register them with the geodatabase.

Annotation stored with SDE 3.x is read only in ArcGIS. If you want to use ArcMap to edit this annotation, you must convert it to geodatabase annotation. See the chapter ‘Managing annotation’ in this book for more information on converting SDE 3.x annotation to geodatabase annotation.

Once you have imported your data into the geodatabase, you can then use ArcCatalog to further define your geodatabase. ArcCatalog contains tools for building topologies and geometric networks and for establishing subtypes, *attribute domains*, and so on.

To learn how to move your existing data into the geodatabase, see the chapter on ‘Migrating existing data into a geodatabase’.



Building geodatabases with CASE tools

Computer-Aided Software Engineering consists of tools and techniques that automate the process of developing software systems and database design. You can use CASE tools to create new *custom objects* and generate a geodatabase schema from a UML diagram.

Object-oriented design tools can be used to create object models that represent the design of your custom objects. Based on these models, the CASE tools’ Code Generation *wizard*

will help you create a Component Object Model (COM) object that implements the behavior of the custom object and the database schema where these custom objects are created and managed.

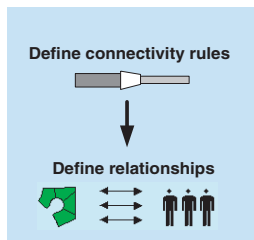
The steps for creating custom objects are:

1. Design the object model using UML in Visio® 2002 or Rational Rose®.
2. Export the model to XML Metadata Interchange (XMI).
3. Generate subcode and implement behavior.
4. Create a geodatabase schema for the custom object.

For details on steps 1 and 3, see *Modeling Our World* and the *Creating custom behavior with the UML* PDF file. Step 4 will be discussed in more detail in the chapter ‘Building geodatabases with CASE tools’.

Further refining the geodatabase

Whether you load data manually or use ArcCatalog to create the geodatabase schema, you can continue to define your geodatabase by establishing how objects in the database relate to one another.

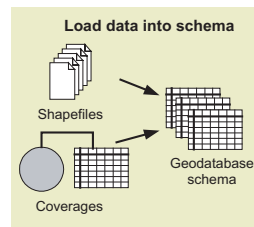


Using ArcCatalog, you can establish relationships between objects in different object classes, *connectivity rules* for objects participating in geometric networks, and topological rules for features in topologies. Some of these relationships and rules may be part of the schema that CASE tools generate, but often you will want to further refine what is generated by

CASE to meet your geodatabase design. Note that topologies cannot be designed with the existing CASE tools. You can continue to use the geodatabase management tools in ArcCatalog to refine or extend a mature database throughout its life.

Loading data into a geodatabase schema

Once you have generated your schema using one of the methods described, you will want to insert data into that schema. This is a different process than importing data. You can do this by editing the database in ArcMap to create new objects or loading objects from existing shapefiles, coverages, *computer-aided design (CAD) feature classes*, INFO tables, dBASE tables, ArcStorm, or Map LIBRARIAN.



Data creation and maintenance may involve managing *version* and topology information. ArcCatalog and ArcToolbox have wizards to help you with this—Simple Data Loader and Object Loader—that will be discussed in the chapter ‘Migrating existing data into a geodatabase’.

Geodatabases and ArcCatalog

ArcCatalog allows you to easily view and modify the contents of your geodatabase. ArcCatalog contains a full suite of tools to create and manage your geodatabase.

Accessing geodatabases in ArcCatalog

You can manage geographic data in a variety of formats in ArcCatalog. Some of the formats that you can manage directly include personal geodatabases, shapefiles, ArcInfo coverages, rasters, TINs, and tables.

In addition to managing data on your desktop or local network, you can manage remote ArcSDE geodatabases by creating a connection to the database. Database connections to remote geodatabases behave somewhat like personal geodatabases, with one important difference: when you delete a personal geodatabase, the database itself is deleted from the disk. When you delete a remote geodatabase connection, however, only the connection is deleted—the geodatabase and its data are unaffected.

Spatial database connections

Using data stored in a DBMS, such as Oracle, requires a database connection. There are two methods for connecting to a *spatial database* from ArcInfo. One method is to connect to an ArcSDE service that spawns a process on the server to broker the connection between ArcInfo and the database *instance*.

The second method is to use a *direct connection* to the database. In this case, ArcInfo connects directly to the database server. The functionality that is managed by the server process in the first connection method is transferred to the client, thus eliminating the middle tier. The direct connect method is a two-tiered rather than three-tiered architecture.

You can use the direct connect method to connect to your geodatabase if it is stored in Oracle8i™ or SQL Server. If connecting to SQL Server, you do not require any additional software to connect to the database. If direct connecting to Oracle8i, the Oracle client software needs to be installed on your machine and you need to provide an Oracle service name for your server.

For more information about direct connect, see the ArcSDE *Configuration and Tuning Guide for <RDBMS>* PDF file.

When you add a new connection to an ArcSDE geodatabase service or a direct database connection in ArcCatalog, it creates a connection file on disk. This file contains the information needed to establish a connection. The *username* and *password* can be included in the connection file and are encrypted for security.

You can set up connection files for your organization and distribute these such that end users will not require any information about the geodatabase server to which they are connecting.

Geodatabases and ArcMap

ArcMap allows you to edit the contents of your geodatabase. ArcMap contains a full suite of tools to edit *simple features*, geometric networks, and topologies in your geodatabase.

Editing geodatabases in ArcMap

You can edit geographic data in a variety of formats in ArcMap. ArcView® seats of ArcMap can be used to edit simple features in shapefiles and personal geodatabases. ArcInfo and ArcEditor seats have more extensive toolsets with special tools for creating and editing geographic data in topologies or geometric networks and performing spatial adjustment and *disconnected editing*.

Editing topologies

Editing a feature in a *map topology* or geodatabase topology with the topology edit tools automatically updates the geometry of the shared parts of all of the features. After you edit a topology, the geodatabase can discover if any of the edits violate the rules of the topology. If there are *errors*, ArcMap enables you to quickly see which features are involved and which rules have been violated and allows you to fix the error, mark it as an exception, or leave it as an error. Error reporting in ArcMap gives you a measure of how good your data is.

Editing geometric networks

Editing features in a geometric network allows you to maintain network *connectivity* of specified types of features and lets you move connected features without breaking the connections and add junctions to certain types of edges without splitting them. Geometric networks are useful for modeling connected linear features such as pipes, cables, and streams, and for modeling points that represent *nodes* in the network such as valves, switches, and stream confluences or gauging stations. The

features that participate in a geometric network become network feature classes, not simple feature classes.

Versions

ArcEditor and ArcInfo seats of ArcMap allow you to edit versioned geodatabases. Different versions of a geodatabase can be used to model complex “what if” scenarios, create different design alternatives, or manage the stages in a multistep planning, design, and construction *work flow* cycle without modifying or making multiple copies of your base data. Versioning also allows multiple editors to work on the same large continuous GIS *dataset* without the need to lock the data or divide it into tiles.

ArcMap lets you easily switch from one version to another to see how they differ.

When you are ready to merge changes from a version into another version, ArcMap can *reconcile* the versions. When multiple editors are working on an area, there is a chance for the same feature to be modified by more than one editor—in these cases, ArcMap helps you identify and resolve the conflicts.

Disconnected editing

ArcInfo and ArcEditor seats of ArcMap can be used to check out data from your geodatabase to use while disconnected from the network. This is especially useful for remote offices that need to work with a portion of the database without incurring network-related performance problems and for people who need to take a part of the geodatabase out into the field for editing or analysis on a field computer. Changes to the data made in the field can be checked back into the main geodatabase to allow them to be used by others.

The first step: creating a database

The first step in creating a geodatabase is to create the database itself using ArcCatalog.

There are two kinds of geodatabases: personal geodatabases and ArcSDE geodatabases. Creating a new personal geodatabase involves creating a new .mdb file on disk.

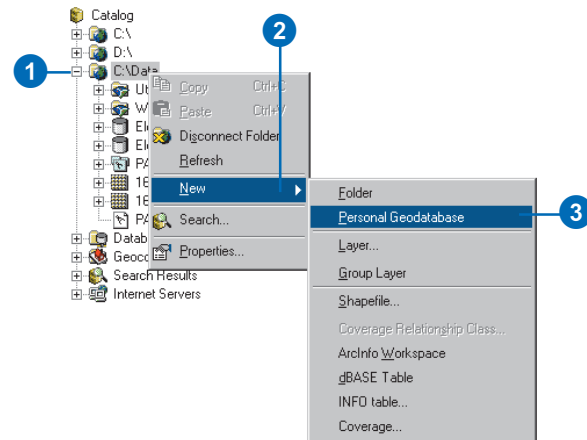
Before you can create data in an ArcSDE geodatabase, you must do some setup. Setting up the database for use as an ArcSDE geodatabase is described in *Managing ArcSDE Services* and in the *ArcSDE Installation Guide*, located on the ArcSDE CD-ROM. For direct connections only, please see the *ArcInfo Installation Guide* for setup instructions.

Several *versions* of an ArcSDE geodatabase can exist, although not every table or feature class in the geodatabase must be versioned. Feature editing in ArcMap requires a versioned feature class in a geodatabase.

New connections will automatically access the DEFAULT version of the database. To connect to an ►

Creating a new personal geodatabase

1. Right-click on the location in the ArcCatalog tree where you want to create the new personal geodatabase.
 2. Point to New.
 3. Click Personal Geodatabase.
- ArcCatalog creates a new personal geodatabase in the location you selected and sets its name to edit mode.
4. Type a new name for this personal geodatabase.
 5. Press Enter.



alternate version, you must provide your username and password along with the version name. If you do not specify the version, ArcCatalog connects to the DEFAULT version.

ArcSDE functionality is read-only for ArcView.

Tip

Testing the connection

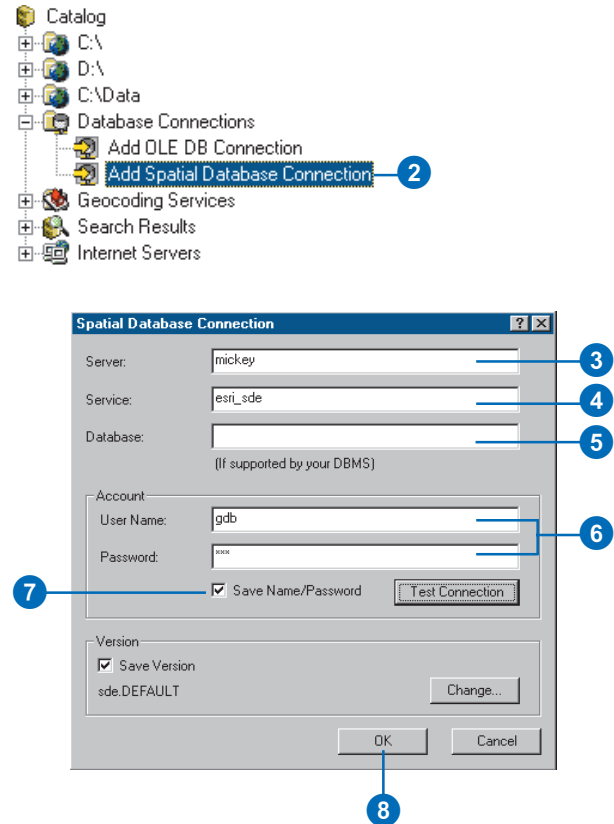
Clicking OK in the Spatial Database Connection dialog box does not actually connect to the database but creates the connection file on disk. To make sure that the connection parameters you entered are correct, you can click Test Connection.

See Also

For more information on how to use ArcCatalog to browse your file system, see Using ArcCatalog.

Adding a connection to an ArcSDE geodatabase service in ArcCatalog

1. Double-click Database Connections.
2. Double-click Add Spatial Database Connection.
3. Type either the name or the Internet Protocol (IP) Address of the server to which you want to connect.
4. Type either the name or the TCP/IP port number of the ArcSDE service to which you want to connect.
5. Type the name of the database to which you want to connect if your DBMS supports it; otherwise, skip to step 6.
6. Type the username and password with which you will connect to the ArcSDE geodatabase.
7. Check the check box to save the username and password in the connection file so that you can connect to the database without being prompted to log in.
8. Click OK.
9. Type a new name for the spatial database connection.
10. Press Enter.



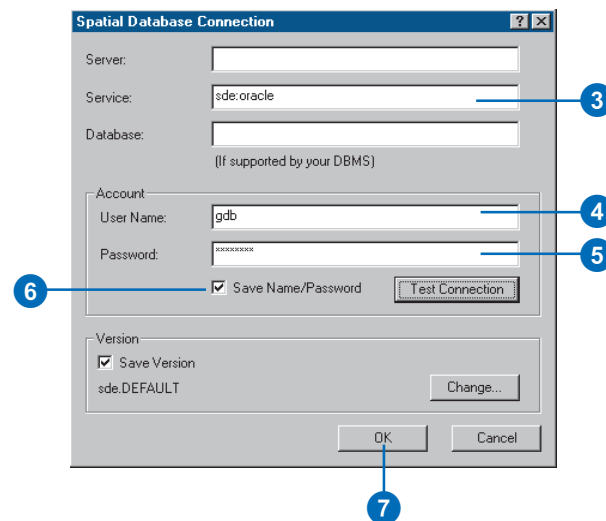
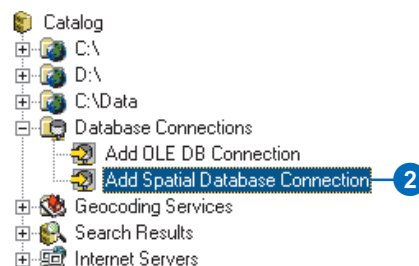
Tip

Oracle service name

You must create an Oracle service name on your client machine before you can create a direct connection to an Oracle database.

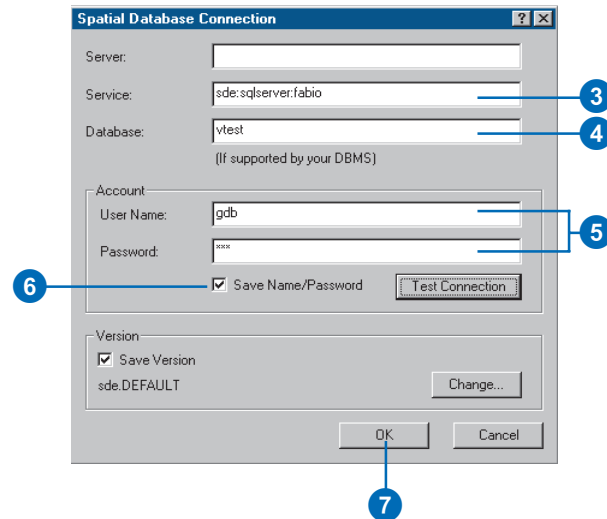
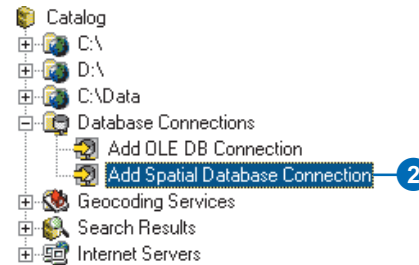
Adding a direct connection to an Oracle8i geodatabase in ArcCatalog

1. Double-click Database Connections.
2. Double-click Add Spatial Database Connection.
3. Type “sde:oracle” in the service box.
4. Type the username.
5. Type the password followed by “@<oracle service name>”.
6. Check the check box to save the username and password in the connection file so that you can connect to the database without being prompted to log in.
7. Click OK.
8. Type a new name for the spatial database connection.
9. Press Enter.



Adding a direct connection to a SQL Server geodatabase in ArcCatalog

1. Double-click Database Connections.
2. Double-click Add Spatial Database Connection.
3. Type "sde:sqlserver:<name or the IP Address of the server>" in the service box. In this example, the server name is fabio.
4. Type the name of the database you want to connect to.
5. Type the username and password.
6. Check the check box to save the username and password in the connection file so that you can connect to the database without being prompted to log in.
7. Click OK.
8. Type a new name for the spatial database connection.
9. Press Enter.

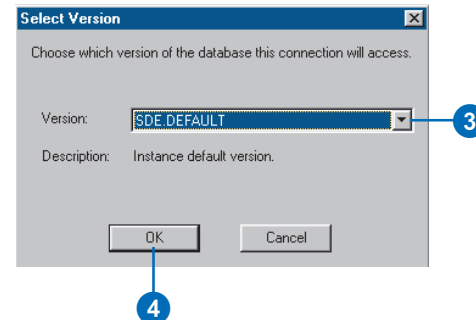
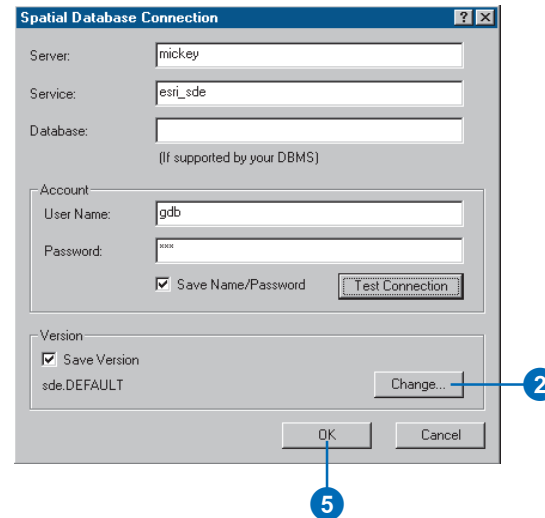


See Also

For more information on geodatabase versions, see the chapter 'Working with a versioned geodatabase' in this book.

Connecting to an alternative version of the database

1. Follow steps 1 through 7 for adding a connection to a spatial database geodatabase service or direct connect in ArcCatalog.
2. Click Change.
3. Click the Version dropdown arrow and click the version you want to access.
4. Click OK.
5. Click OK in the Spatial Database Connection dialog box.
6. Type a new name for the spatial database connection.
7. Press Enter.



Tips on learning how to build and edit geodatabases

If you're new to GIS, remember that you don't have to know everything about ArcCatalog, ArcMap, and geodatabases, or know how to extend the generic Geodatabase data model, to get immediate results. To learn how to create and edit geodatabases, see the *Geodatabase Workbook*. ArcGIS comes with the data used in the tutorials, so you can follow along step by step at your computer. You can also read the tutorial without using your computer.

Finding answers to questions

If you are like most people, your goal is to complete your tasks while investing a minimum amount of time and effort on learning how to use the software. You want intuitive, easy-to-use software that gives you immediate results without having to read pages of documentation. However, when you do have a question, you want to be able to find the answer quickly so that you can complete your task. That's what this book is all about—getting you the answers you need when you need them.

This book describes how to get your existing data into a geodatabase; how to create new items in your geodatabase; and then, once created, how to add a variety of behavior to that data and edit it. Although you can read this book from start to finish, you will likely use it more as a reference. When you want to know how to do a particular task, such as creating a geometric network or topology, just look it up in the table of contents or index.

What you will find is a concise, step-by-step description of how to complete tasks. Some chapters also include detailed information if you want to learn more about the concepts behind the tasks. Refer to the glossary if you come across any unfamiliar GIS terms or need to refresh your memory.

About this book

This book is designed to introduce how to build and edit a geodatabase. While this book does have some conceptual content about the different aspects of the geodatabase, it assumes that you already have a schema design that you are trying to implement. If you have not yet designed your schema or need more information on how to make the best schema design decisions, please take some time to read *Modeling Our World*, which you received with ArcGIS.

Getting help on your computer

In addition to this book, the ArcGIS online Help system is a valuable resource for learning how to use the software.

Contacting ESRI

If you need to contact ESRI for technical support, see the product registration and support card you received with ArcGIS or refer to 'Contacting Technical Support' in the 'Getting more help' section of the ArcGIS Desktop Help system. You can also visit ESRI on the Web at www.esri.com and support.esri.com for more information on the geodatabase and ArcGIS.

ESRI education solutions

ESRI provides educational opportunities related to geographic information science, GIS applications, and technology. You can choose among instructor-led courses, Web-based courses, and self-study workbooks to find educational solutions that fit your learning style. For more information, go to www.esri.com/education.

Creating new items in a geodatabase

2

IN THIS CHAPTER

- Geodatabase items
- ArcGIS data types
- Upgrading a geodatabase
- Creating tables
- Creating feature datasets
- Creating feature classes
- Creating indexes
- Granting and revoking privileges

The first step in creating any database is to design the tables it will contain. A good design will ensure that data retrieval is fast and efficient. *Modeling Our World* discusses the considerations to take into account when you build a *geodatabase*.

When your design is complete, you can start creating the database using ArcCatalog tools. You can create tables, feature datasets, and feature classes in a geodatabase. After adding data to tables and feature classes, you can build *indexes* for the appropriate fields to improve *query* performance. You can also grant and revoke privileges on your table, feature class, or feature datasets for another database user.

After creating feature classes, tables, and feature datasets, you can refer to further chapters in this book to create more advanced items such as *relationship classes*, topologies, and geometric networks.

ArcView can create simple feature classes and tables in a personal geodatabase. More advanced geodatabase functionality requires an ArcEditor or ArcInfo license.

Geodatabase items

Geodatabases organize geographic data into a hierarchy of data objects. These data objects are stored in feature classes, object classes, and feature datasets. An object class is a table in the geodatabase that stores nonspatial data. A feature class is a collection of features with the same type of geometry and the same attributes.

A feature dataset is a collection of feature classes that share the same *spatial reference*. Feature classes that store simple features can be organized either inside or outside a feature dataset. Simple feature classes that are outside a feature dataset are called standalone feature classes. Feature classes that store topological features must be contained within a feature dataset to ensure a common spatial reference.

ArcCatalog contains tools for creating object classes (tables), feature classes, and feature datasets. Once these items are created in the geodatabase, further items such as subtypes, relationship classes, geometric networks, and topologies can also be created. These geodatabase items are covered in subsequent chapters.

Spatial reference

When creating a new feature dataset or standalone feature class, you must specify its spatial reference. The spatial reference for a feature class describes its *coordinate system* (for example, geographic, UTM, and State Plane), *spatial domain*, and *precision*. The spatial domain is best described as the allowable *coordinate* range for x,y coordinates, m (measure) values, and z-values. The precision describes the number of system units per one unit of measure. A spatial reference with a precision of 1 will store integer values, while a precision of 1,000 will store three decimal places. Once the spatial reference for a feature dataset or standalone feature class has been set, only the coordinate system can be modified—the spatial domain is fixed.

All feature classes in a feature dataset share the same spatial reference. The spatial reference is an important part of geodatabase design because its spatial domain describes the maximum spatial extent to which the data can grow. You must be careful to choose an appropriate x, y, m, and z domain. For example, if you create a feature dataset with a minimum z-value of 0 and a precision of 1,000, none of the features in the feature dataset can have z-values that are less than 0, and all z-values will be stored to three decimal places. The same rule applies to x- and y-values. The exception to the rule is m domains; feature classes within the same feature dataset can have different m domains.

The spatial domain for a feature class or feature dataset cannot be changed. If the required x-, y-, m-, or z-value ranges for your database change, the data has to be reloaded into feature classes with a spatial reference that accommodates the new value range.

A collection of predefined geographic and projected coordinate systems is installed with ArcInfo. You can create custom coordinate systems, or you can import a coordinate system from an existing feature class, feature dataset, coverage, or shapefile. You can read more about spatial references and spatial domains in *Managing ArcSDE Services* and *Understanding Map Projections*.

Spatial index grid size

Each feature class has a spatial index that is automatically generated and maintained by the ArcInfo system. The spatial index is used to quickly locate features in a dataset that might match the criteria of a spatial search. The spatial index is calculated based on parameters that are provided when the feature class is created.

For most database management systems (DBMSs), the spatial index is a two-dimensional grid system that spans a feature class such as the locator grid you might find on a common road map.

For most data, only a single grid size is required. Because feature size is an important factor in determining an optimum grid size, data that contains features of very different sizes may require additional grid sizes so that larger features can be queried faster. Feature classes may have up to three grid sizes. Each grid size must be at least three times the previous grid size.

If the spatial index of the feature class is a grid, then it can be changed at any point in its lifetime. For a more detailed discussion of spatial indexes and grid sizes, see the *ArcSDE Configuration and Tuning Guide for <DBMS> PDF file*.

Field properties

When you use ArcCatalog to create a new table or feature class, you can specify any number of *fields* to be included. You can also specify settings for fields such as the field type and the maximum size of the data that can be stored in it.

Each field type has special properties. All fields have *property* default values, domains, *aliases*, and allow nulls. The field alias property will be discussed in the next section. You can set the allow nulls property to No if you do not want the field to store *null values*. If you set the allow nulls property to Yes, then the field will allow null values.

Use the default value property if you want the field to be automatically populated with a default value when a new feature or object is created with the ArcMap editing tools. You can set a domain, which is a valid set or range of values that can be stored in the field by using the domain property. Default values and domains are discussed in detail in the ‘Subtypes and attribute domains’ chapter.

The exceptions are fields of type ObjectID, binary large object (BLOB), and Geometry. ObjectID, BLOB, and Geometry type fields do not have a default value or domain property. Alias is the only

property of an ObjectID field you can modify, while BLOB and Geometry fields have special properties you can modify.

The properties of a Geometry field describe the kind of features that can be stored in a feature class, the size of the spatial index, and the spatial reference for the features.

Field precision and scale

The precision and scale of a field describe the maximum size and precision of data that can be stored in it. The precision describes the number of digits that can be stored in the field, while the scale describes the number of decimal places for float and *double* fields. When creating a new field in a geodatabase feature class or table, you can specify the field’s type, precision, and scale. When the field is actually created in the database, the field type may be changed based on the precision and scale values you specify.

Use the following guidelines for choosing the correct field type for a given precision and scale:

- When you create a float, double, or integer field and specify 0 for precision and scale, the geodatabase will attempt to create a binary type field if the underlying database supports it. Personal geodatabases support only binary type fields, and precision and scale are ignored.
- When you create float and double fields and specify a precision and scale, if your precision is greater than 6, use a double; otherwise, use a float. If you create a double field and specify a precision of 6 or less, a float field is created in the database. If you create a float field and specify a precision greater than 6, a double field is created.
- If you specify a scale of 0 and a precision of 10 or less, you should be creating integer fields. When creating integer fields,

your precision should be 10 or less or your field may be created as double.

Required fields

All tables and feature classes have a set of required fields that are necessary to record the state of any particular object in the table or feature class. These required fields are automatically created when you create a new feature class or table and they cannot be deleted. Required fields may also have required properties such as their domain property. You cannot modify the required property of a required field.

For example, in a simple feature class, OBJECTID and Shape are required fields. They do have properties, such as their aliases and geometry type, that you can modify, but these fields cannot be deleted.

Some types of feature classes have a number of required fields; for more information, see the chapters ‘Managing annotation’, ‘Geometric networks’, and ‘Topology’ in this book.

Field, table, and feature class aliases

The names of feature classes and tables in a geodatabase are the same as the names of the physical tables in the DBMS in which they are stored. When storing data in a DBMS, often the names for tables and fields are cryptic, and you require a detailed data dictionary to keep track of what data each table stores and what each field in those tables represents. For example, your database may have a feature class called “Pole” that has a field called “HGT”. Without consulting your data dictionary, you may have a difficult time determining that Pole stores utility poles and HGT has values for pole heights.

The geodatabase provides the ability to create aliases for fields, tables, and feature classes. An alias is an alternative name to refer to those objects. Unlike true names, aliases do not have to adhere to the limitations of the database, so they can contain special characters such as spaces. In the above example, you may set the alias for the Pole table to “Utility Poles” and the alias for the HGT field to “Height”. In ArcMap, when using data with aliases, the alias name is automatically used for feature classes, tables, and fields. However, in ArcCatalog these objects are always represented by their true names. You can view the alias for feature classes, tables, and fields by examining their properties.

Aliases can be specified when creating a feature class or table and can be modified at any time. Similarly, when creating new fields, the alias is set as a property of that field and can be modified at any time.

Tracking properties of geometry

Often when working with spatial data, you may want to query your data based on properties of the geometry. For example, you may want to query the water mains feature class for all mains that are more than 50 feet in length. Doing this by examining each geometry and calculating its length can be a slow process, especially if there is a large number of water mains in the feature class. To make this more efficient, the feature class has special fields that track this kind of information about the geometry of your features.

Line feature classes have a field that tracks the features’ length, while *polygon* feature classes have fields that track both the features’ area and perimeter. When changes are made to the geometry, the values in these fields are automatically updated. These fields behave like other fields, except that you cannot delete them, assign default values and attribute domains to them, or assign values to them while editing in ArcMap.

When you create a new feature class in a personal geodatabase, these fields will not be displayed in the fields panel of the wizard. However, when you open the properties dialog box for a polygon feature class, you will see fields named Shape_Area and Shape_Length that store the area and perimeter of the geometry. When you open the properties dialog box for a line feature class, you will see a field named Shape_Length that stores the length of the geometry. Point and *multipoint feature* classes do not have either field.

When you create a new feature class in an ArcSDE geodatabase, these fields will be called SHAPE.AREA and SHAPE.LEN, respectively.

In ArcMap, these fields behave as any other field in the identify window, the layer properties dialog box, the attribute table dialog box, and the query builder. For more information on these aspects of ArcMap, see *Using ArcMap*.

Feature datasets

Feature datasets exist in the geodatabase to define a scope for a particular spatial reference. All feature classes that participate in topological relationships with one another (for example, a geometric network or a topology) must have the same spatial reference. Feature datasets are a way to group feature classes with the same spatial reference so that they can participate in topological relationships with each other.

Feature datasets also have a natural organizational quality, much like a folder on a file system. Since for many GIS applications the majority of the data for a particular database has the same spatial reference, it is possible to use feature datasets as organizational containers.

Bear in mind that topologically related feature classes must reside in the same feature dataset. If you are interested in organizing feature classes purely by category, you can arrange layer files

into logical groups within folders at a shared location, without regard to the organization of the feature classes in the geodatabase.

Topologies

Many vector datasets have features that could share boundaries or corners. If you create a topology in the dataset, you can set up rules defining how features share their geometry. Editing a boundary or *vertex* shared by two or more features updates the shape of each of those features. Topology rules can govern the relationships between features within a single feature class, or between features in two different feature classes. For example, moving a slope boundary in one feature class could update two slope-class polygons and also update the boundary of a forest stand in another feature class.

Topologically associated feature classes in a geodatabase are stored in a feature's dataset with a *topology*. The feature classes remain simple feature classes and the topology manages the rules that control how features can be spatially related. You can use the topological editing tools in ArcMap to maintain the topological associations of features.

To learn more about topologies, see the chapter 'Topology' in this book.

For more information about topology in general, see *Modeling Our World*.

Geometric networks

Some vector datasets, particularly those used to model communications, material or energy flow, or transportation networks, need to support connectivity *tracing* and network connectivity rules. Geometric networks allow you to turn simple point and line features into network edge and junction features

that can be used for network analysis. Connectivity rules of geometric networks let you control what types of network features may be connected together when editing the network. Geometric networks, like topologies, must be created from a set of feature classes in the same feature dataset.

To learn more about geometric networks, see the ‘Geometric networks’ chapter in this book.

For more information about geometric networks in general, see *Modeling Our World*.

Relationship classes

Relationship classes define relationships between objects in the geodatabase. These relationships can be simple one-to-one relationships, such as you might create between a feature and a *row* in a table, or more complex one-to-many (or many-to-many) relationships between features and table rows. Some relationships specify that a given feature, row, or table is not only related to another, but that creating, editing, or deleting one will have a specified effect on the other. These are called composite relationships and they can be used to ensure that the links between objects in the database are maintained and up to date. Deleting a feature, such as a power pole, can trigger the deletion of other features, such as a transformer mounted on the pole, or the maintenance *records* in a related table.

To learn more about relationship classes, see the ‘Defining relationship classes’ chapter in this book.

Schema locking

In multiuser databases, more than one user may be reading and editing the same data at the same time. To be able to work with data in a geodatabase for applications such as ArcMap, the

application must assume that the schema for that data is fixed while it is working with it.

For example, when you add a feature class from a geodatabase to your map, its schema cannot be altered by you or another user. Once you have removed the feature class from your map and no other users are querying or editing that feature class, its schema can be altered.

ArcMap, ArcCatalog, and other applications written with the ArcGIS COM components will automatically acquire a shared lock when editing or querying the contents of a geodatabase feature class or table. Any number of shared locks can be acquired on a single feature class or table at one time. When using ArcCatalog to modify schema—add a field, alter rules, and so on—the application will attempt to acquire an exclusive lock on the data being altered.

An exclusive lock can only be acquired if no other locks—shared or exclusive—are already on the data. If there are already other locks on the feature class or table, ArcCatalog will not be able to establish its exclusive lock and the schema will not be editable. Once an exclusive lock has been acquired, no shared locks can be applied, so the data will not be accessible in ArcMap or ArcCatalog by another user.

Exclusive locks can only be acquired by the owner of the feature class or table being modified and, therefore, only the owner can ever modify the schema of an item in a geodatabase. Some of the items that exist in a geodatabase—which are discussed in further chapters—such as geometric networks, relationship classes, topologies, and so on, have special schema-locking behaviors. Each of these is discussed in its own respective chapter.

Schema locking in personal geodatabases works in much the same way except the locks are databasewide. Once an exclusive or shared lock is acquired on an item in a personal geodatabase, that lock applies to all items in the geodatabase.

ArcGIS data types

When creating tables, you will need to *select* a data type for each field in your table. The available types include a variety of number types, text, date, or binary large object (BLOB). Choosing the correct data type allows you to correctly store the data and will facilitate your analysis, data management, and business needs.

Numeric data types

Numeric fields can be stored as one of four numeric *data types*. These include short integers; long integers; *single-precision* floating point numbers, often referred to as floats; and double-precision floating point numbers, commonly called doubles. Each of these numeric data types varies in the size and method of storing a numeric value.

In numeric data storage, it is important to understand the difference between decimal and binary numbers. The majority of people are accustomed to decimal numbers, a series of digits between 0 and 9 with negative or positive values and the possible placement of a decimal point. On the other hand, computers store numbers as binary numbers. A binary number is simply a series of 0s and 1s. In the different numeric data types, these 0s and 1s represent different coded values, including the positive or negative nature of the number, the actual digits involved, and the placement of a decimal point. Understanding this type of number storage will help you make the correct decision in choosing numeric data types.

The most basic numeric data type is the short integer. This type of numeric value is stored as a series of 16 0s or 1s, commonly referred to as 16 bits. Eight bits are referred to as a byte, thus a short integer takes up two bytes of data. One bit states if the number is positive or negative and the remaining 15 translate to a numeric value with five significant digits. The actual numeric value for a short integer is approximately between -32,000 and

+32,000. A long integer is a four-byte number. Again, one bit stores the positive or negative nature of the number while the remaining bits translate to a numeric value with 10 significant digits. The actual range for a long integer is approximately between -2 billion and +2 billion. Both short and long integers can store only real numbers. That is to say that you cannot have fractions, or numbers, to the right of the decimal place. To store data with decimal values, you will need to use either a float or a double.

A float and double are both binary number types that store the positive or negative nature of the number, a series of significant digits and a coded value to define the placement of a decimal point. This is referred to as the exponent value. Floats and doubles are coded in a format similar to scientific notation. For example, if you wanted to represent the number -3,125 in scientific notation, you would say -3.125×10^3 or $-3.125E^3$. The binary code would break this number apart and assign one bit to state that it is a negative number; another series of bits would define the significant digits 3125; another bit would indicate whether the exponent value is positive or negative; and the final series of bits would define the exponent value of 3. A float is a four-byte number and can store up to seven significant digits, producing an approximate range of values between $-3.4E^{-38}$ to $-1.2E^{-38}$ for negative numbers and from $3.4E^{-38}$ to $1.2E^{-38}$ for positive numbers. A double is an eight-byte number and can store up to 15 significant digits, producing an approximate range of values between $-2.2E^{-308}$ to $-1.8E^{-308}$ for negative numbers and $2.2E^{-308}$ to $1.8E^{-308}$ for positive numbers.

It is important to note, however, that floats and doubles are approximate numbers. This is due to two factors. First, the number of significant digits is a limiting factor. For example, you could not express the number 1,234,567.8 as a float because this number contains more than the permissible seven digits. In order to store the number as a float, it will be rounded to 1,234,568, a

number containing the permissible seven digits. This number could easily be expressed as a double, as it contains less than the permissible 15 significant digits. There are also some limitations to numbers a binary value can represent. One analogy that can be made would be in expressing fractions versus decimals. The fraction 1/3 represents a particular value. However, if you try to express this number as a decimal, the number will need to be rounded at some point. It could be expressed as 0.3333333, however, this is still an approximation of the actual value. Just as fractions cannot always be expressed as decimals, some numbers cannot be exactly expressed in binary code, and these numbers are replaced by approximate values. One example of such a number is 0.1. This number cannot be expressed as a binary number. However, the number 0.099999 can be expressed in binary. Thus 0.1 would be replaced with an approximate value of 0.099999.

In choosing the numeric data type, there are two things to consider. First, it is always best to use the smallest byte size data type needed. This will not only minimize the amount of storage required for your geodatabase but will also improve the performance. You should also consider the need for exact numbers versus approximate numbers. For example, if you need to express a fractional number and seven significant digits will suffice, use a float. However, if the number must be more precise, choose a double. If the field values will not include fractional numbers, choose either a short or long integer.

Text fields

A text field represents a series of alphanumeric symbols. This can include street names, attribute properties, or other textual descriptions. An alternative to using repeating textual attributes is to establish a coded value. A textual description would be coded with a numeric value. For example, you might code road types with numeric values assigning a 1 to paved improved

Name	Specific range, length, or format	Size (Bytes)	Applications
Short integer	-32,768 to 32,767	2	numeric values without fractional values within specific range; coded values
Long integer	-2,147,483,648 to 2,147,483,647	4	numeric values without fractional values within specific range
Single-precision floating point number (Float)	approx. $-3.4E^{-38}$ to $1.2E^{38}$	4	numeric values with fractional values within specific range
Double-precision floating point number (Double)	approx. $-2.2E^{-308}$ to $1.8E^{308}$	8	numeric values with fractional values within specific range
Text	up to 64,000 characters	varies	names or other textual qualities
Date	mm/dd/yyyy hh:mm:ss AM/PM	8	date and/or time
BLOB	varies	varies	images or other multimedia

roads, a 2 to gravel roads, and so on. This has the advantage of using less storage space in the geodatabase, however, the coded values must be understood by the data user. If you define your coded values in a coded value domain in the geodatabase and associate the domain with the integer field storing your codes, the geodatabase will display the textual description when the table is viewed in ArcMap or ArcCatalog. For more information on coded value domains, see the chapter ‘Subtypes and attribute domains’ in this book.

Date fields

The date data type can store dates, times, or date and times. The default format in which the information is presented is mm/dd/yyyy hh:mm:ss and a specification of AM or PM. When you enter date fields in the table, they will be converted to this format.

BLOB fields

A BLOB, or binary large object, is simply some data stored in the geodatabase as a long sequence of binary numbers. Items such

as images, multimedia, or bits of code can be stored in this type of field.

Data types outside of ArcGIS

The data types explained in this section include all the data types available when creating a table using ArcMap or ArcCatalog and stored in a personal geodatabase. However, you might choose to store your tables in another DBMS such as Oracle or dBASE. When this is done, the data types between ArcGIS and your DBMS might not match directly. The types are matched to the closest data type available in the DBMS. This process is referred to as data type mapping. In this process, it is possible that the values will be stored in the DBMS as a different type, applying different criteria to the data attribute. To learn more about the data type mapping process with your database management system, see the *Configuration and Tuning Guide for <DBMS>* PDF file.

Upgrading a geodatabase

Geodatabases built using previous versions of ArcGIS do not support some of the newer functions of ArcGIS.

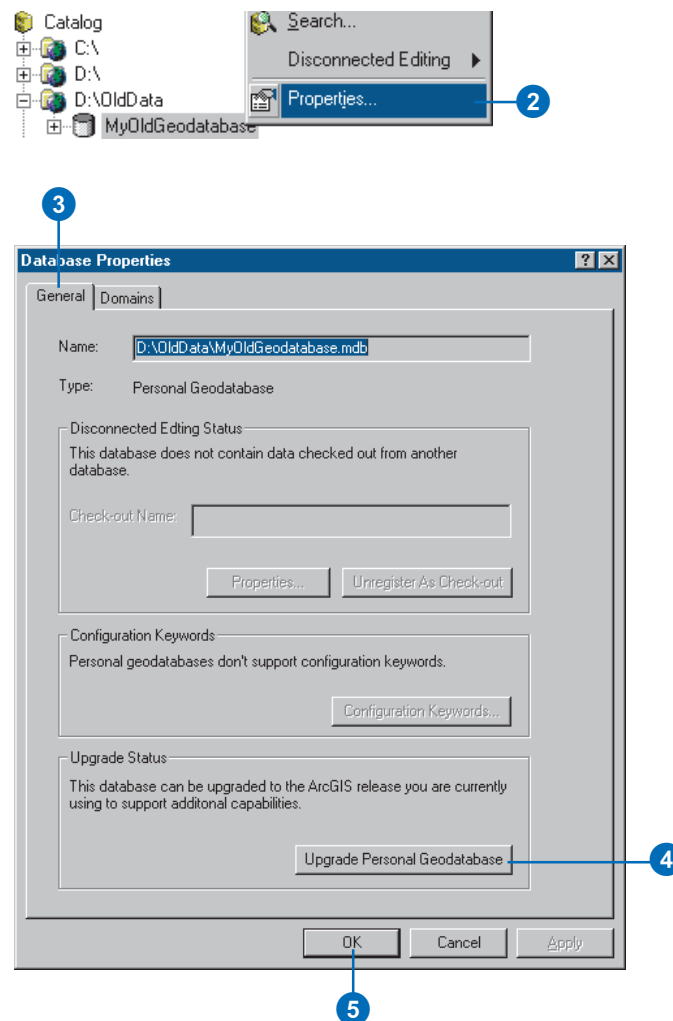
If your geodatabase was developed using a previous version of ArcGIS, you may wish to upgrade your geodatabase.

Tip

Creating a backup copy of your geodatabase

Bear in mind that once a geodatabase is upgraded, previous versions of ArcGIS can view, but not edit, the geodatabase. For this reason, you may wish to make a copy of the geodatabase and upgrade the copy, thus leaving you with both an original and an upgraded geodatabase.

1. Start ArcCatalog.
2. Right-click the geodatabase you want to upgrade, and click Properties.
3. Click the General tab.
4. Click Upgrade Personal Geodatabase.
5. Click OK.



Creating tables

You can create tables in a geodatabase with an easy-to-use table designer. If you accept the designer's defaults, you will create a table that uses simple row objects to represent each row in the table.

When defining a table's fields, be aware that each database has its own rules defining what names and characters are permitted. The designer checks the names you type using a set of common rules, but each database is slightly different. If you want more control over a field's data types or structure, create the table directly in the database.

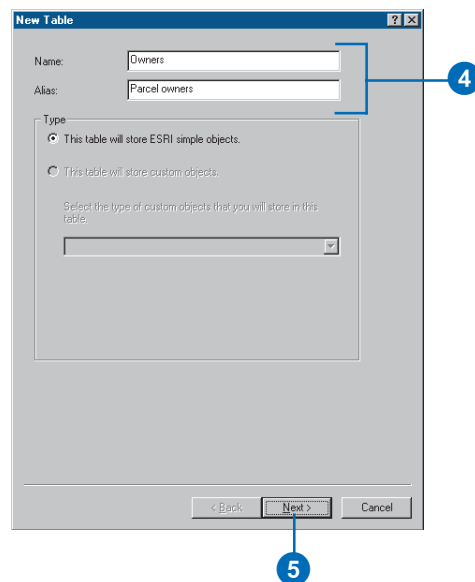
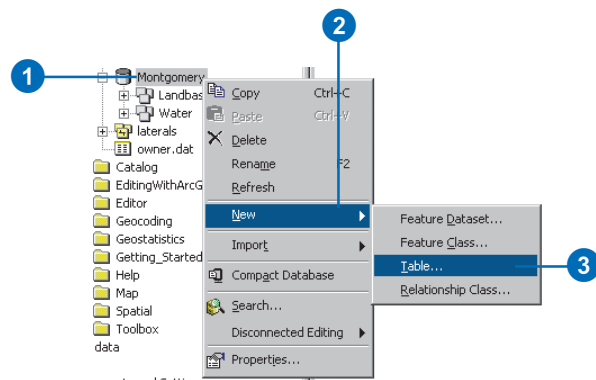
If you have custom row objects registered on your system, you can choose to create a table in which to store these objects. Custom objects usually have a set of required fields that are necessary to record the state of any particular object in the table. ►

See Also

For details about using configuration keywords with ArcSDE, see the ArcSDE Configuration and Tuning Guide for <DBMS> PDF file.

Creating a table to store simple objects

1. Right-click the database in the ArcCatalog tree in which you want to create a new table.
2. Point to New.
3. Click Table.
4. Type a name for the table. To create an alias for this table, type the alias.
5. Click Next. ►



These required fields are automatically prepopulated in the table wizard. You will be able to see these fields on the fields panel, but you will not be able to delete or modify them. You will only be able to add additional fields.

In the simple row table, the OBJECTID field is an example of a required field.

Tip

Using another table as a template

When creating a new table, you can use another table as a template. Click **Import**, navigate to the table whose field definitions you want to copy, then click **OK**. Now you can edit the field names and their data types.

Tip

Deleting a field

If you have entered a field that you do not want to include in the new table, select it by clicking its tab in the grid, then press **Delete**.

If your geodatabase does not use ArcSDE, skip to step 8.

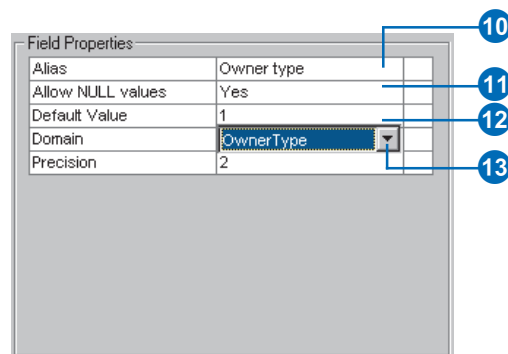
6. Click **Use configuration keyword** and type the keyword you want to use if you want to create the table using a custom storage keyword.
7. Click **Next**.
8. Click the next blank row in the **Field Name** column and type a name to add a field to the table.
9. Click in the **Data Type** column next to the new field's name and click its data type.

Tip

OBJECTID field

All simple tables in the geodatabase require an ObjectID type field. The ObjectID field uniquely identifies each object stored in the table in the database. The default ObjectID field will not be able to be deleted in this wizard.

10. Click the field next to Alias and type the alias for this field to create an alias for this field.
11. Click the field next to Allow NULL values, click the dropdown arrow, and click No to prevent nulls from being stored in this field.
12. Click the field next to Default Value and type the value to associate a default value with this field.
13. Click the field next to Domain, click the dropdown arrow to see a list of the domains that apply to this field type, and click the domain to associate a domain with this field.
14. Either click the property in the dropdown list or type the property to set other properties specific to the type of field.
15. Repeat steps 8 through 14 until all the table's fields have been defined.
16. Click Finish.



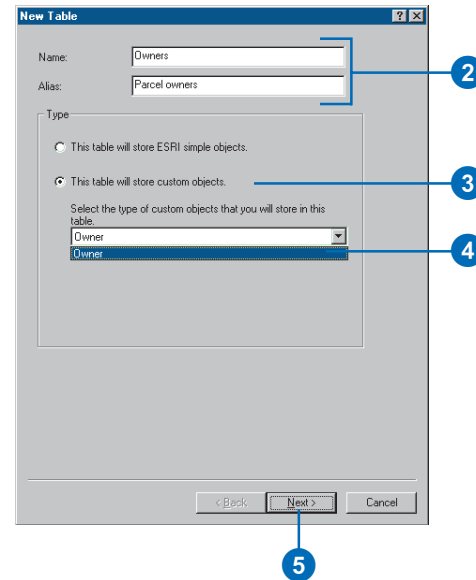
Tip

Custom objects

If you do not have any custom objects registered on your system, the option to specify one when creating a table will be unavailable.

Creating a table that stores custom objects

1. Follow steps 1 through 3 for 'Creating a table to store simple objects'.
2. Type a name for the table. To create an alias for this table, type the alias.
3. Click the second option to store custom objects in the table.
4. Click the dropdown arrow to see a list of available custom row objects and click the object you want to store.
5. Click Next.
6. Follow steps 6 through 16 for 'Creating a table to store simple objects'.



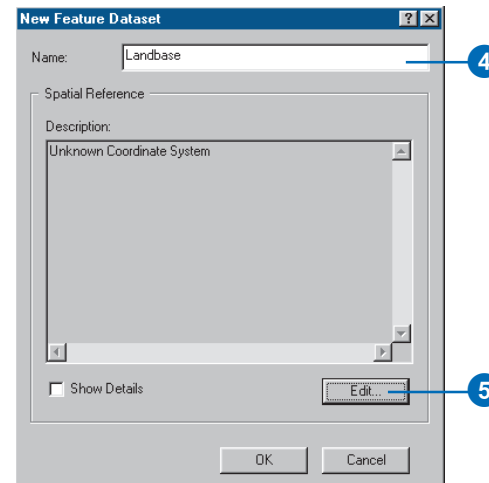
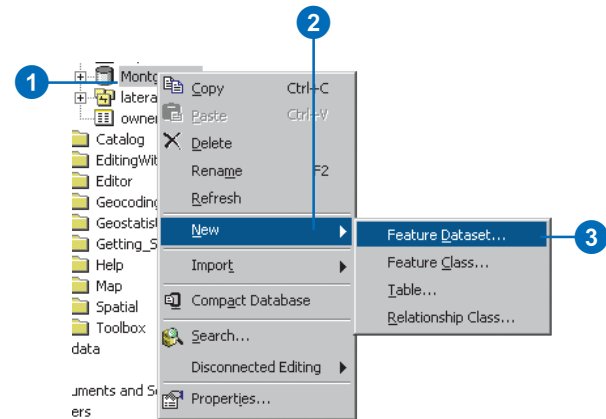
Creating feature datasets

When creating a new feature dataset, you must define its spatial reference. This includes its coordinate system—either geographic or a specific projection—and the coordinate domains—the minimum x-, y-, z-, and m-values and their precision. All feature classes in the dataset must use the same coordinate system, and each coordinate of every feature in all feature classes must fall within the coordinate domains. The exception to the rule is m domains; feature classes in the same feature dataset can have different m domains.

When defining the coordinate system, you can choose a predefined coordinate system, use an existing feature dataset or standalone feature class as a template, or define a custom geographic or projected coordinate system.

Creating a feature dataset with a predefined coordinate system

1. Right-click the database in the ArcCatalog tree in which you want to create a new feature dataset.
2. Point to New.
3. Click Feature Dataset.
4. Type a name for the feature dataset.
5. Click Edit to define the feature dataset's spatial reference. ►

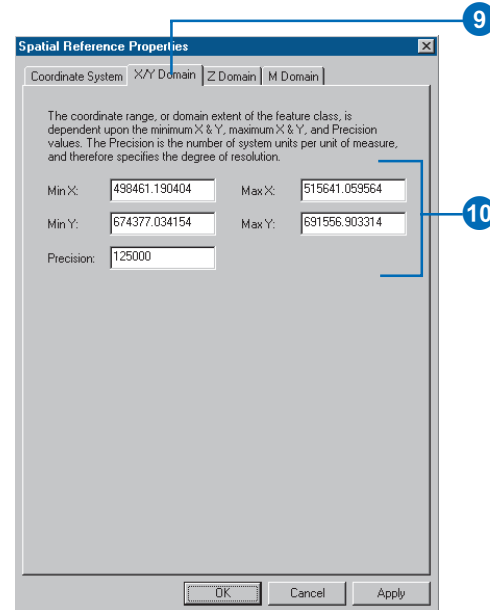
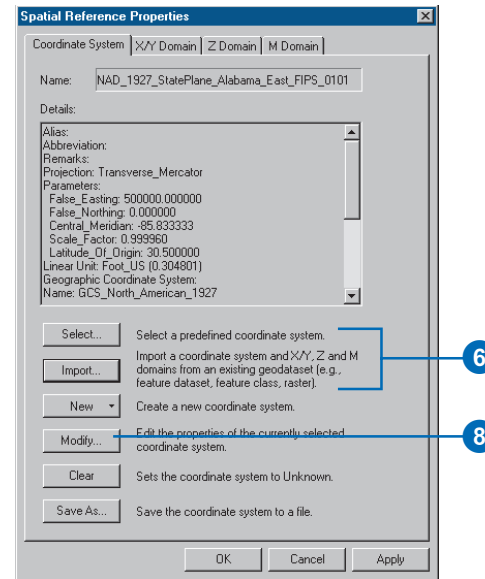


Tip

Saving the coordinate system

You can click *Save As* to save the coordinate system as a .prj file.

- Click Select or Import to set the feature dataset's spatial reference.
- Navigate to the spatial reference you want to use or navigate to the feature class or feature dataset whose spatial reference you want to use as a template.
- Click Modify if you want to change any parameters in the coordinate system you have chosen. Edit the coordinate system's parameters and click OK.
- Click the X/Y Domain tab.
- Type the minimum x, minimum y, maximum x, and maximum y coordinate values for the dataset and type the required precision for the coordinate values. ►

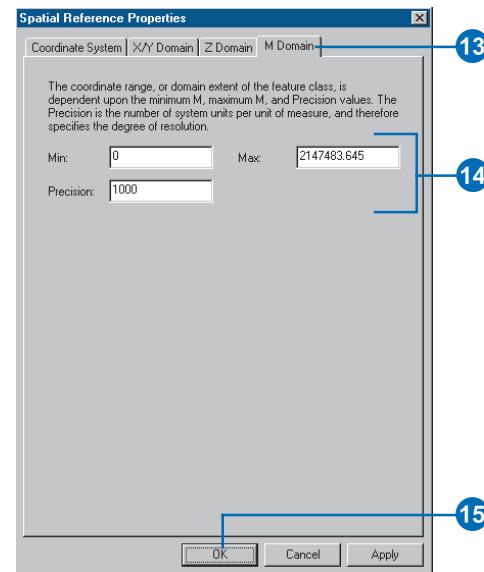
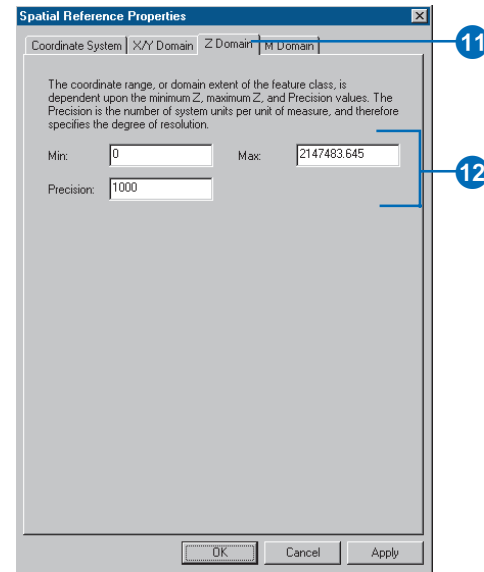


Tip

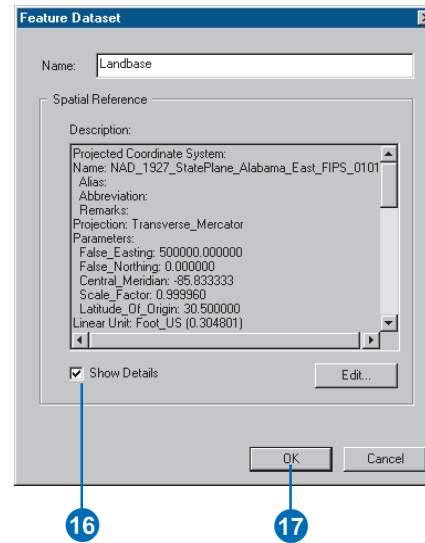
Precision

Since the size of the spatial domain is dependent on the value of precision, when the precision is changed, the maximum m- or z-value will change to fit within the size of the spatial extent. Similarly, when the maximum m- or z-value is changed, the precision will also change to fit the domain extent.

11. Click the Z Domain tab.
12. Type the minimum z-value and maximum z-value for the dataset and type the precision required for the z coordinates if any feature class in the feature dataset will have z-values.
13. Click the M Domain tab.
14. Type the minimum m-value and maximum m-value for the dataset and type the precision required for the m values if any feature class in the feature dataset will have measures.
15. Click OK. ►



16. Check Show Details to see the details of your new dataset's spatial reference.
17. Click OK.



Tip

Editing predefined parameters

You can easily create variations of a predefined coordinate system. For example, choose a predefined datum from the dropdown list; the text boxes now contain the selected datum's parameters and their contents are read-only. Now choose <custom> from the datum dropdown list. The contents of the text boxes do not change but you can now edit their values. Type a name for your datum in place of <custom>.

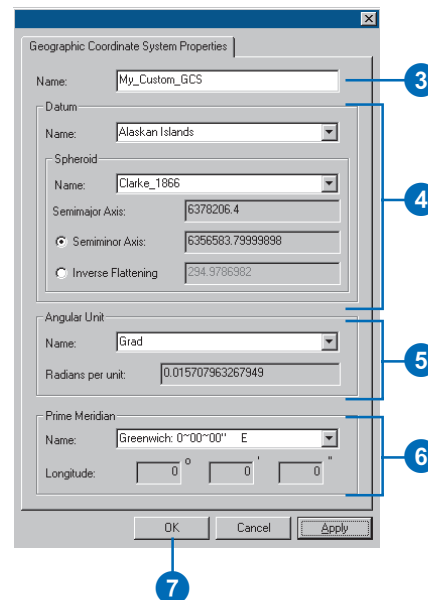
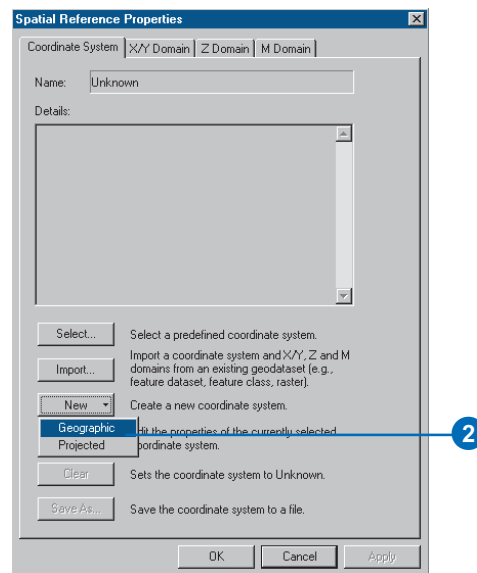
Tip

Saving the coordinate system

You can click Save As to save the coordinate system as a .prj file.

Defining new geographic coordinate systems

1. Follow steps 1 through 5 for 'Creating a feature dataset with a predefined coordinate system'.
2. Click New and click Geographic.
3. Type a name for the coordinate system.
4. Type the parameters for a custom datum or choose a predefined datum from the dropdown list.
5. Type the angular unit or choose a predefined angular unit from the dropdown list.
6. Type the degrees, minutes, and seconds defining the prime meridian's longitude or choose a predefined prime meridian from the dropdown list.
7. Click OK.
8. Follow steps 9 through 16 for 'Creating a feature dataset with a predefined coordinate system'.

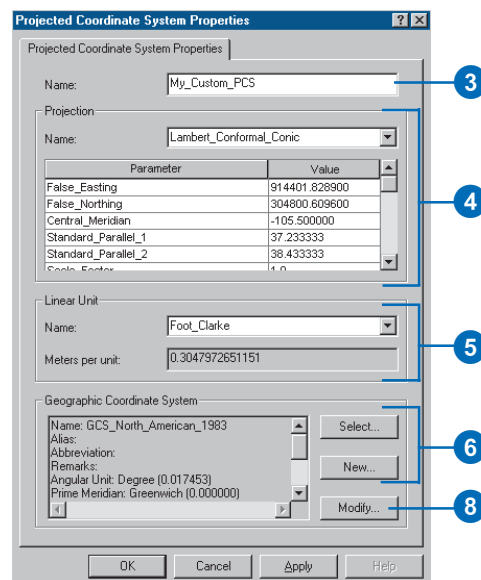
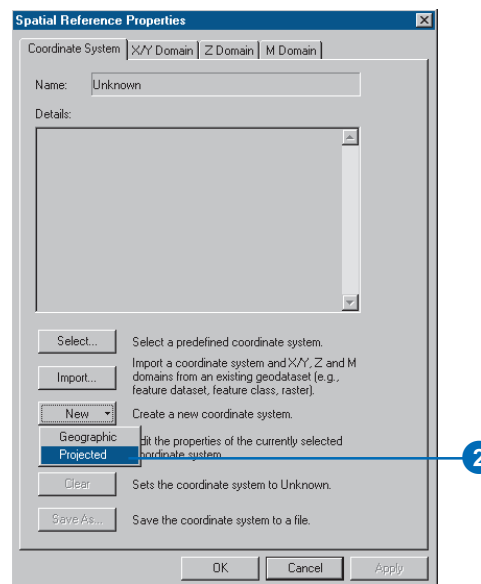


See Also

For information about which parameters are appropriate for which projection, see Understanding Map Projections.

Defining new projected coordinate systems

1. Follow steps 1 through 5 for 'Creating a feature dataset with a predefined coordinate system'.
2. Click New and click Projected.
3. Type a name for this coordinate system.
4. Choose a projection from the dropdown list and type the appropriate parameter values for that projection.
5. Type the linear unit or choose a predefined linear unit from the dropdown list.
6. Click Select or New to set the geographic coordinate system.
7. Navigate to the geographic coordinate system or navigate to the feature class or feature dataset whose geographic coordinate system you want to use as a template.
8. Click Modify if you want to change any parameters in the geographic coordinate system you have selected.
9. Click OK.
10. Follow steps 9 through 16 for 'Creating a feature dataset with a predefined coordinate system'.



Creating feature classes

You can create feature classes in a geodatabase with ArcCatalog. If you accept the wizard's defaults, you will create a feature class that uses simple feature objects—points, lines, or polygons—to represent its features. You can also create features with custom behavior.

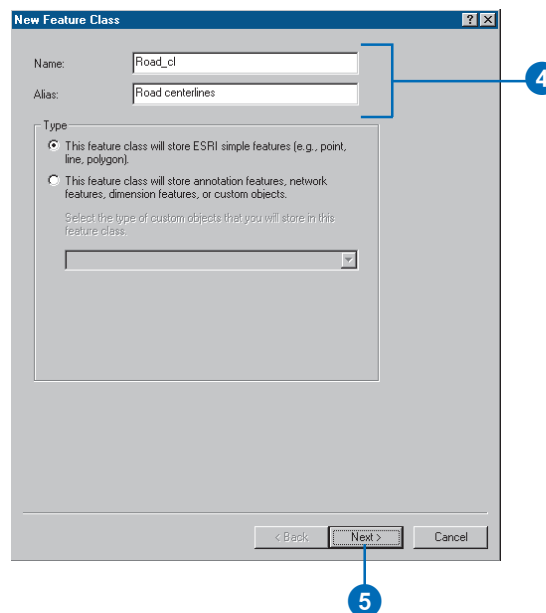
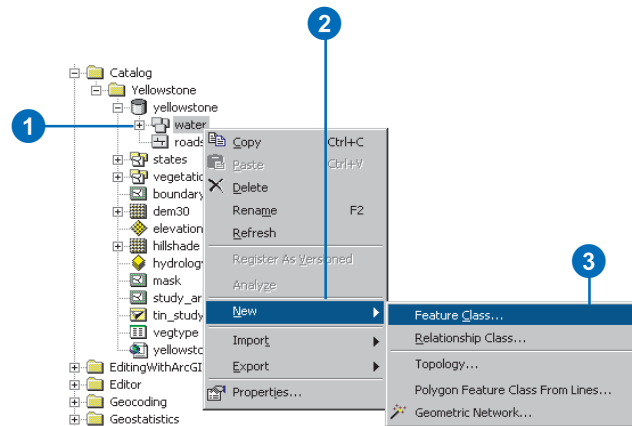
When creating a feature class, you must define the geometry field's properties such as its spatial index and the geometry type. When creating a standalone feature class, you must define its spatial reference. All feature classes in a feature dataset must use the same spatial reference, which was defined when the feature dataset was created. The exceptions to the rule are m domains; feature classes in the same feature dataset can have different m domains. ►

See Also

For details about using configuration keywords with ArcSDE, see the ArcSDE Configuration and Tuning Guide for <DBMS> PDF file.

Creating a feature class in a feature dataset

1. Right-click the feature dataset in the ArcCatalog tree in which you want to create a new feature class.
2. Point to New.
3. Click Feature Class.
4. Type a name for the feature class. To create an alias for this feature class, type the alias.
5. Click Next. ►



If you have custom features registered on your system, you can choose to create a feature class in which to store these objects. Custom features usually have a set of required fields that are necessary to record the state of any particular feature in the feature class.

These required fields are automatically prepopulated in the feature class wizard. On the Fields panel, you will be able to see these fields, but you will not be able to delete or modify them. You will only be able to add additional fields.

In the simple feature class, the OBJECTID and Shape fields are examples of required fields.

Tip

Using another feature class as a template

When creating a new feature class, you can use another feature class as a template. Click **Import**, navigate to the feature class whose field definitions you want to copy, and click **OK**. Now you can edit the field names and their data types.

If your geodatabase does not use ArcSDE, skip to step 8.

6. Click **Use configuration keyword** and type the keyword you want to use if you want to create the table using a custom storage keyword.
7. Click **Next**.
8. Click the next blank row in the **Field Name** column and type a name to add a field to the feature class.
9. Click in the **Data Type** column next to the new field's name and click its data type.

New Feature Class

Specify the database storage configuration.

☒ **Default**
This option uses the default storage parameters for the new table/feature class.

☐ **Use configuration keyword**
This option allows you to specify a configuration keyword which references the database storage parameters for the new table/feature class.

< Back **Next >** Cancel

New Feature Class

Field Name	Data Type
OBJECTID	Object ID
SHAPE	Geometry
Prefix	Text
Name	Text
Type	Long Integer
	Short Integer
	Long Integer
	Short Integer
	Text
	Date
	Geometry

Click any field to see its properties.

Field Properties

Alias	
Allow NULL values	Yes
Default Value	
Domain	
Precision	0

Import...

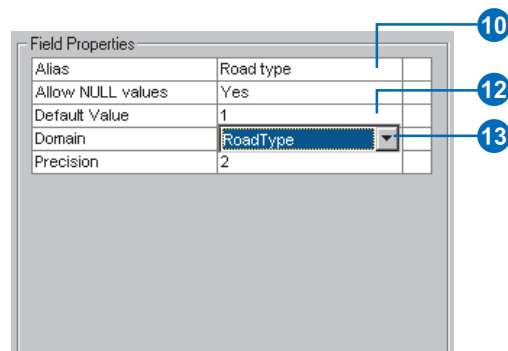
< Back **Finish** Cancel

Tip

OBJECTID and Shape fields

All simple feature classes in the geodatabase require an ObjectID and geometry type fields. The default ObjectID and geometry fields will not be deletable in this wizard.

10. Click the field next to Alias and type the alias for this field to create an alias for this field.
11. Click the field next to Allow NULL values, click the dropdown arrow, and click No to prevent nulls from being stored in this field.
12. Click the field next to Default Value and type the value to associate a default value with this field.
13. Click the field next to Domain, click the dropdown arrow to see a list of the domains that apply to this field type, and click the domain to associate a domain with this field.
14. Either click the property in the dropdown list or type property to set other properties specific to the type of field.
15. Repeat steps 8 through 14 until all the table's fields have been defined. ►



Tip

The Spatial Reference button

When adding a feature class to a feature dataset, this button lets you review the feature dataset's spatial reference parameters; however, you can't change them. The exceptions to the rule are m domains; feature classes in the same feature dataset can have different m domains.

16. Click the name of the geometry field in the Field Name column.
17. Click the field next to Alias and type the alias to create an alias for the geometry field.
18. Click the field next to Allow NULL values, click the dropdown arrow, and click No to prevent null shapes from being stored.
19. Click the field next to Geometry Type, click the dropdown arrow, and click the type of features you want to store in this feature class.
20. Click the fields next to the grid size you want to specify and type the grid value to set the spatial index grid parameters for the feature class.
21. Click the field next to Contains Z values, click the dropdown arrow, and click Yes if you want the shapes in this feature class to store z-values.
22. Click the field next to Contains M values, click the dropdown arrow, and click Yes if you want the shapes in this feature class to store m-values.
23. Click Finish.

The screenshot shows the 'Field Properties' dialog box for a geometry field. The 'Geometry Type' is set to 'Line'. The 'Spatial Reference' is 'NAD_1927_StatePlane_Al'. Numbered callouts point to specific fields: 17 points to the 'SHAPE' field name; 19 points to the 'Line' dropdown in the 'Geometry Type' row; 20 points to the 'Grid 1' and 'Grid 2' rows; 21 points to the 'Contains Z values' dropdown; and 22 points to the 'Contains M values' dropdown.

Field Properties	
Alias	SHAPE
Allow NULL values	Yes
Geometry Type	Line
Avg Num Points	0
Grid 1	1000
Grid 2	0
Grid 3	0
Contains Z values	No
Contains M values	No
Default Shape field	Yes
Spatial Reference	NAD_1927_StatePlane_Al

Tip

Saving the coordinate system

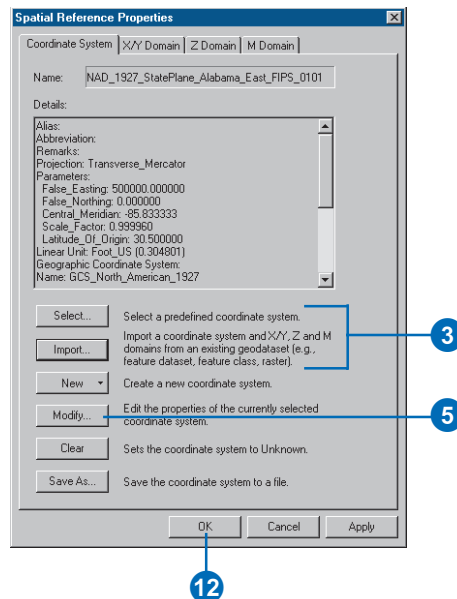
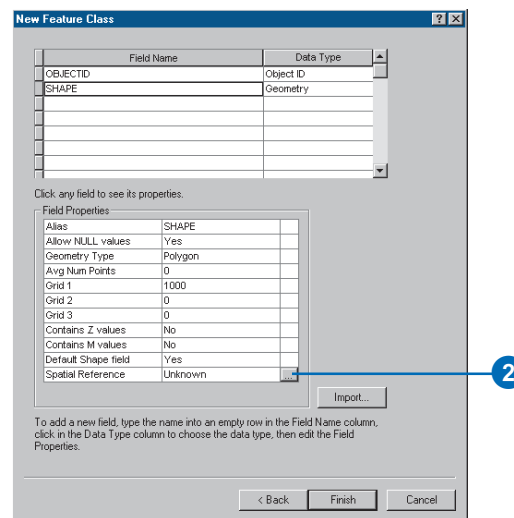
You can click **Save As** to save the coordinate system as a .prj file.

See Also

For examples of how to define new geographic and projected coordinate systems, see 'Creating feature datasets' in this chapter.

Creating a standalone feature class

1. Follow steps 1 through 22 for 'Creating a feature class in a feature dataset'.
2. Click the Spatial Reference Properties button to define the feature class's coordinate system.
3. Click Select or Import to set the feature dataset's spatial reference.
4. Navigate to the spatial reference you want to use or navigate to the feature class or feature dataset whose spatial reference you want to use as a template. Click Add.
5. Click Modify if you want to change any parameters in the coordinate system you have chosen. Edit the coordinate system's parameters and click OK. ►



Tip

Using another feature class as a template for the spatial reference only

Click **Import** to populate the *Spatial Reference Properties* dialog box with information from another feature class. You can then customize the template's spatial reference.

Tip

Specifying a custom coordinate system

To modify a predefined (or a template's) coordinate system or to define a custom coordinate system from scratch, click **Custom** on the *Coordinate System* dialog box.

6. Click the **X/Y Domain** tab.
7. Type the minimum x, minimum y, maximum x, and maximum y coordinate values for the dataset and type the required precision for the coordinate values.
8. Click the **Z Domain** tab, if present. If your feature class does not store z-values, skip to step 10.
9. Type the minimum z-value and maximum z-value for the dataset, then type the precision required for the z coordinates. ►

The screenshot shows the **Spatial Reference Properties** dialog box with the **X/Y Domain** tab selected. The dialog box has a title bar with a close button. Below the title bar are four tabs: **Coordinate System**, **X/Y Domain**, **Z Domain**, and **M Domain**. The **X/Y Domain** tab is active. The main area contains a text box with the following text: "The coordinate range, or domain extent of the feature class, is dependent upon the minimum X & Y, maximum X & Y, and Precision values. The Precision is the number of system units per unit of measure, and therefore specifies the degree of resolution." Below this text are four input fields: **Min X:** 498461.190404, **Max X:** 515641.059564, **Min Y:** 674377.034154, and **Max Y:** 691556.903314. Below these is a **Precision:** field with the value 125000. At the bottom are three buttons: **OK**, **Cancel**, and **Apply**. A blue bracket on the right side of the dialog box groups the **Min X**, **Max X**, **Min Y**, and **Max Y** fields, with a blue circle containing the number 7 next to it. Another blue circle containing the number 6 is next to the **X/Y Domain** tab.

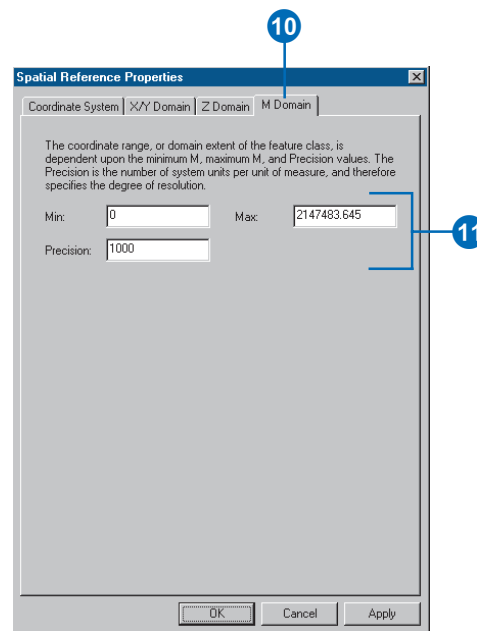
The screenshot shows the **Spatial Reference Properties** dialog box with the **Z Domain** tab selected. The dialog box has a title bar with a close button. Below the title bar are four tabs: **Coordinate System**, **X/Y Domain**, **Z Domain**, and **M Domain**. The **Z Domain** tab is active. The main area contains a text box with the following text: "The coordinate range, or domain extent of the feature class, is dependent upon the minimum Z, maximum Z, and Precision values. The Precision is the number of system units per unit of measure, and therefore specifies the degree of resolution." Below this text are three input fields: **Min:** 0, **Max:** 2147483.645, and **Precision:** 1000. At the bottom are three buttons: **OK**, **Cancel**, and **Apply**. A blue bracket on the right side of the dialog box groups the **Min**, **Max**, and **Precision** fields, with a blue circle containing the number 9 next to it. Another blue circle containing the number 8 is next to the **Z Domain** tab.

Tip

Precision

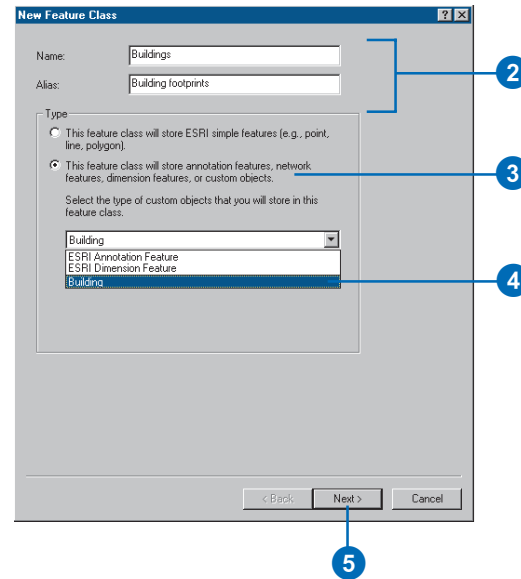
Since the size of the spatial domain is dependent on the value of precision, when the precision is changed, the maximum z-value will change to fit within the size of the spatial extent. Similarly, when the maximum z-value is changed, the precision will change to fit the domain extent.

10. Click the M Domain tab, if present. If your feature class does not store m-values, skip to step 12.
11. Type the minimum m-value and maximum m-value for the dataset, then type the precision required for the m-values.
12. Click OK.



Creating a feature class that stores custom features

1. Follow steps 1 through 3 for 'Creating a feature class in a feature dataset'.
2. Type a name for the feature class. To create an alias for this feature class, type the alias.
3. Click the second option to store custom objects in the feature class.
4. Click the dropdown arrow to see a list of available custom features, then click the feature you want to store.
5. Click Next.
6. Follow steps 6 through 23 for 'Creating a feature class in a feature dataset'.



Creating indexes

Once you have data in a table or feature class, you may want to create attribute indexes to make your queries faster. Spatial indexes increase the selection speed of graphical queries on spatial features. An attribute index is an alternate path used by the DBMS to retrieve a record from a table. It is much faster to first look up the index and go to the appropriate record than to start at the first record and search through the entire table.

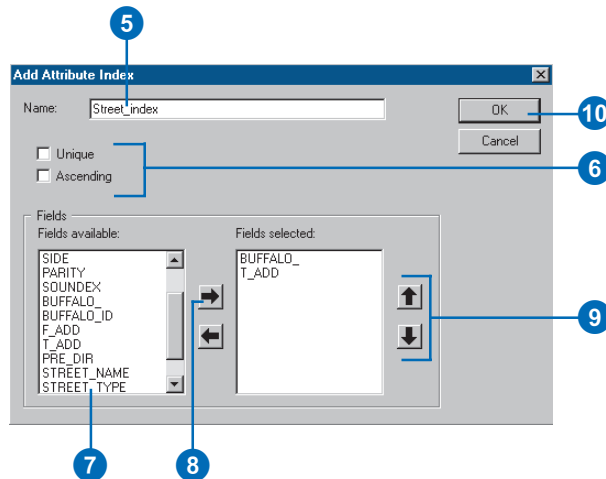
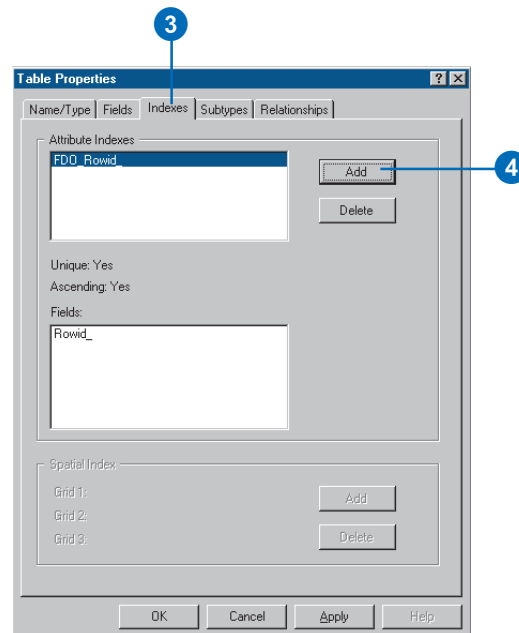
Attribute indexes can be created for single or multiple fields on the feature class and table property pages. Once an index has been added, it can be deleted and added again at any point in the lifetime of the feature class or table.

You can use the same property page to delete a spatial index from and add a spatial index to your feature class.

You can modify the spatial index for an ArcSDE feature class by deleting the index and reading it. You cannot access the features stored in an ArcSDE feature class if it doesn't have a spatial index. ►

Creating a new attribute index

1. Right-click the table or feature class in the ArcCatalog tree for which you want to create an index.
2. Click Properties.
3. Click the Indexes tab.
4. Click Add.
5. Type the name for the new index.
6. Check the Unique check box if your field values are unique. Check the Ascending check box to create an ascending index. Data in an ascending index is returned in ascending order.
7. Click the field or fields for which you want to build this index.
8. Click the arrow button to move the fields to the Fields selected list.
9. Use the up and down arrows to change the order of the fields in the index.
10. Click OK. ►



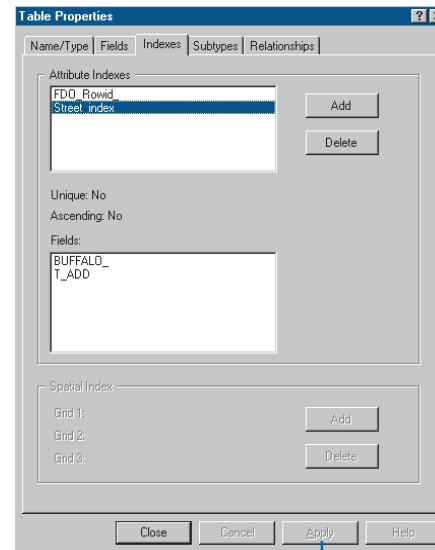
Building a new spatial index for an ArcSDE feature class is a server-intensive operation—it should not be done on very large feature classes when a large number of users are logged in to the server.

Tip

Deleting an index

You can delete an index by clicking it in the Attribute Indexes list and clicking Delete.

11. Click Apply to build the index and close the property page.



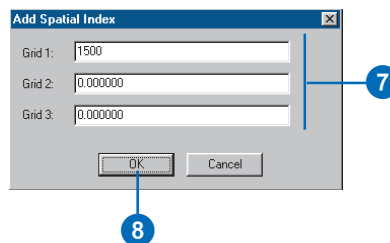
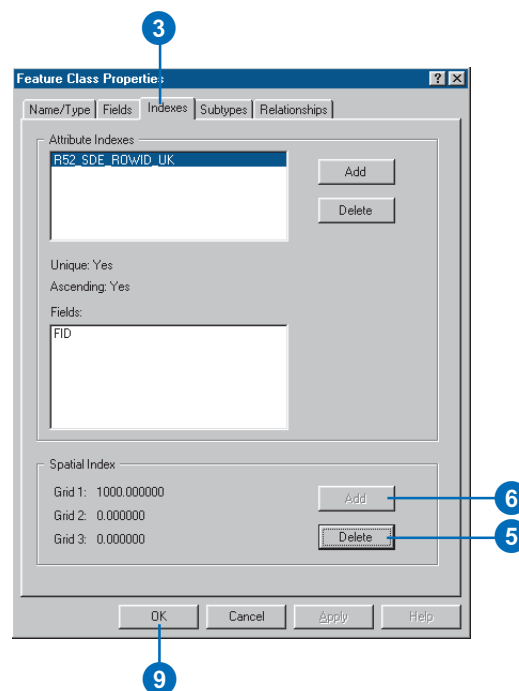
Tip

ArcSDE spatial indexes

For more information on what it means to have an ArcSDE feature class with no spatial index, see Managing ArcSDE Services.

Modifying the spatial index

1. Right-click the feature class in the ArcCatalog tree whose spatial index you want to modify.
2. Click Properties.
3. Click the Indexes tab.
4. Delete the spatial index first if there is one. If there is no spatial index, skip to step 6.
5. Click Delete.
6. Click Add.
7. Type new index parameters if you do not want to use the ones already in the settings for this feature class.
8. Click OK.
9. Click OK to build the spatial index and close the property page.



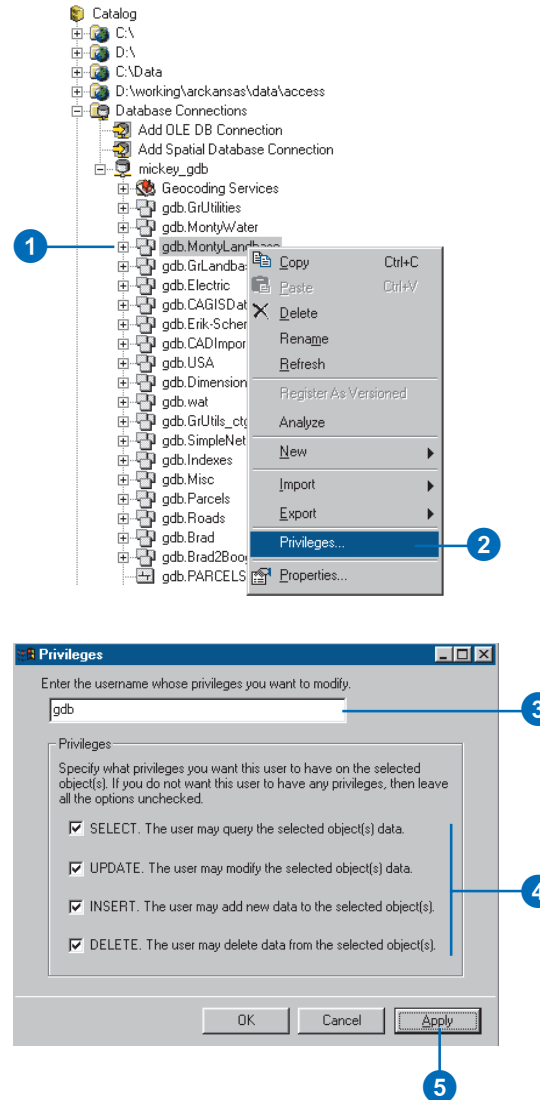
Granting and revoking privileges

If you want to let other database users view and/or modify the contents of any items, you must grant them the privilege to do so. The same tool for granting privileges can also be used to revoke privileges from a particular database user.

You have several options when granting privileges. You can specify that a user has no privileges. You can grant “select” privileges, meaning that they can view, but not modify, the contents of an item. Alternatively, you can grant a user full privileges (Select, Update, Insert, and Delete) to both view and modify the contents of an item.

Granting or revoking privileges on a feature dataset causes all of its contents to have the same privilege changes. When you add new items to a feature dataset or build a geometric network (see the chapter ‘Geometric networks’ in this book), you will need to grant privileges on the feature dataset again.

1. Right-click the item or items in the ArcCatalog tree for which you want to grant privileges.
2. Click Privileges.
3. Type the name of the user whose privileges you want to change.
4. Click the privileges you want them to have. UPDATE, INSERT, and DELETE will only be active if SELECT is clicked, and these work as a unit. If you leave all options unchecked, the user will have all access privileges revoked.
5. Click Apply to change the privileges.



Migrating existing data into a geodatabase

3

IN THIS CHAPTER

- How data is converted
- Importing shapefiles, coverages, and tables
- Importing a geodatabase feature class
- Importing a CAD feature class
- Importing rasters
- Copying geodatabase data
- Using the Extract Data wizard
- Loading data into existing simple feature classes and tables
- Registering ArcSDE layers and tables with the geodatabase
- Analyzing geodatabase data
- Loading objects from other feature classes

With ArcCatalog, you can import and load shapefiles, coverages, *computer-aided design (CAD)* data, and INFO and dBASE tables into a geodatabase. ArcCatalog also lets you move data between personal geodatabases and ArcSDE (formerly Spatial Database Engine [SDE]) geodatabases. All of these data import tools, data loading tools, and wizards are accessible from ArcCatalog and ArcToolbox.

If your existing data is stored in a format other than one listed above or those discussed in this chapter (such as TIGER® files), ArcToolbox has the tools you need to convert that data into a format that can be imported or loaded into the geodatabase.

You can create and load data into a personal geodatabase using ArcView. More advanced geodatabase functionality applies to ArcEditor or ArcInfo licenses.

How data is converted

ArcCatalog and ArcToolbox each contain tools to import spatial data from coverages, shapefiles, and CAD feature classes into a geodatabase. They also contain tools for importing nonspatial data from INFO and dBASE tables. Tables, coverages, and shapefiles can also be imported in other ways. However, it is recommended that you use these tools or a custom application written with the ArcInfo Component Object Model (COM) components to import data into a geodatabase for the following reason.

When a shapefile, coverage, or CAD feature class is imported using an ArcCatalog or ArcToolbox tool, a geodatabase feature class is created to store the features. This feature class stores both the geometry and attributes from the input data. The feature class is automatically registered with the geodatabase system tables so that it can participate in relationships and geometric networks, have *validation rules*, and so on. Similarly, when a table is imported, a table is created in the geodatabase and automatically registered with the geodatabase system tables. Coverages, shapefiles, and CAD feature classes are imported into ESRI simple feature classes. INFO and dBASE tables are imported into ESRI simple row tables.

Any table, shapefile, coverage, or CAD feature class that is imported by some other mechanism will not be registered with the geodatabase system tables and will therefore not be a true geodatabase feature class or table. The ArcInfo system has tools to register these feature classes and tables with the geodatabase; these tools will be discussed later in this chapter.

There are some basic rules about loading data into a geodatabase that are discussed later in this chapter. Briefly, these rules are:

- Try to do all your data loading before versioning your data.
- Try to load all your data before building geometric networks.
- If you load lots of data into a versioned geodatabase, run *compress* to reduce the size of the delta tables.

Spatial reference

When you import shapefiles, coverages, and CAD feature classes, you create a new standalone feature class, a new feature dataset and feature class, or a feature class in an existing feature dataset. In the first two cases, you have to define a spatial reference.

If your coverage, shapefile, or CAD feature class has a defined projection, the import tools will automatically create the new feature class with the same projection, unless you specify otherwise. If you do choose to import the data into a different projection than the source data is in, the features will automatically be projected.

Because all feature classes in a feature dataset share the same spatial reference, you must only specify a new spatial reference when loading the first feature class or when creating the feature dataset in ArcCatalog. For more information about spatial references, see the chapter ‘Creating new items in a geodatabase’ in this book.

Spatial index grid size

Each of the data importing tools suggests a grid size based on the spatial reference, average feature size, and number of features in the input shapefile or coverage. The suggested grid size is only an approximation. You can determine the optimum grid size for any feature class by examining the data and the needs of the application querying it.

Data mapping

When you convert data from one format to another, a question arises about how data of a specific type maps from one format to the other. Importing data into the geodatabase is no exception. In this case, how both the spatial type and the attribute field type map from one data type to another must be considered. For each geodatabase field type, the field itself is persistently stored in the database differently depending on which DBMS stores the geodatabase.

All feature class and attribute item types in coverages are mapped to geometry and field types in the geodatabase. More than one feature class type in a coverage will map to the same geometry type in the geodatabase. For example, points, *tics*, and nodes all map to geometry type point. Coverage feature classes often will contain additional fields that are not needed in the geodatabase. These fields can be removed while loading the data into the geodatabase. Table 1 illustrates the mapping of feature class type to geodatabase geometry type between coverages and the geodatabase.

Coverage and INFO table items are mapped based on a combination of their type and their width. For example, an item of type I can map to a short integer, long integer, or double, depending on its width. Table 2 summarizes the item to field type mapping between coverages and the geodatabase.

Table 1: Coverage feature class to geodatabase geometry type mapping

Coverage feature class	Geodatabase geometry
point	point
arc	line (polyline)
polygon	polygon
node	point
tic	point
region	polygon
route	line (polyline) with measures
annotation	annotation*

**Annotation in the geodatabase is not a geometry type but is implemented as a feature type. For more information on annotation feature classes in the geodatabase, see the chapter 'Managing annotation' in this book.*

Table 2: Coverage, INFO item to geodatabase field mapping

Item type	Item width	Geodatabase field type
B	4	long integer
C	1–320	text
D	8	date
F	4	float
F	8	double
I	1–4	short integer
I	5–9	long integer
I	10–16	double
N	1–9	float
N	10–16	double

Like coverages, all feature and attribute types in shapefiles are mapped to geodatabase geometry and attribute types. Shapefile feature types are similar to the different geometry types stored in a geodatabase, so their mapping is more straightforward. This is illustrated in Table 3.

Table 3: Shapefile to geodatabase geometry type mapping

Shapefile	Geodatabase geometry
point	point
point M	point with measures
point Z	point with Zs
polyline	line (polyline)
polyline M	line (polyline) with measures
polyline Z	line (polyline) with Zs
polygon	polygon
polygon M	polygon with measures
polygon Z	polygon with Zs
multipoint	multipoint
multipoint M	multipoint with measures
multipoint Z	multipoint with Zs
multipatch	multipatch

Each different shapefile and dBASE field type maps to a single geodatabase type independent of field size. The exception is the Number type field, which will map to a long integer if its number of decimals is zero and to a double if its number of decimals is greater than zero. The shapefile and dBASE field type to geodatabase field type mapping is summarized in Table 4.

Table 4: Shapefile, dBASE field to geodatabase field mapping

Field type	Field width	Geodatabase field type
date	-	date
string	1–254	text
boolean	-	short integer
number	1–4 (decimals = 0)	short integer
number	5–9 (decimals = 0)	long integer
number	10–19 (decimals = 0)	double
float	1–13	float
float	14–19	double
number	1–8 (decimals > 0)	float
number	9–19 (decimals > 0)	double

When converting *CAD feature classes* from AutoCAD® DWG, MicroStation® DGN, and Drawing Interchange File (DXF) formats to geodatabase feature classes, the geometric features found in the CAD drawing are mapped to geodatabase geometry. This mapping is summarized in Table 5.

Table 5: CAD to geodatabase geometry type mapping

CAD feature class	Geodatabase geometry
point	point
polyline	line (polyline) with Zs
polygon	polygon with Zs

The properties that are inherent to CAD features are preserved in the output feature class’s attribute table. These attributes include entity type, layer, color, and linetype as well as complex information such as tag data, block attributes, and database linkage values. The CAD field type to geodatabase field type mapping is summarized in Table 6.

Table 6: CAD field to geodatabase field mapping

Field type	Geodatabase field type
string	text
integer	long integer
double	double

Importing a geodatabase feature class

You can use the Feature class to Geodatabase tool in ArcCatalog and ArcToolbox to import feature classes from one geodatabase to another or to import features from one feature class into a new feature class in the same geodatabase. This tool creates simple feature classes only and does not preserve object identity.

Therefore, when you use this tool to import a custom or network feature class, a new simple feature class is created and the geometry and attributes of each feature are imported. If the input feature class has any subtypes, default values, relationships (each described in other chapters of this book), and so on, they are not imported along with the features.

Analyzing geodatabase data

If your geodatabase is stored in a DBMS, such as Oracle, SQL Server, DB2, or Informix, the Analyze command in ArcCatalog can be used to update the DBMS statistics on your datasets. The Analyze command updates the statistics of business tables, feature tables, and delta tables and the statistics on those tables' indexes.

The type of DBMS that your geodatabase is stored in will dictate which statistics the Analyze command updates. For more information on your DBMS, see the ArcSDE online documentation. For more information on when you should run the Analyze command on your data and which tables should be analyzed, see the chapter 'Working with a versioned geodatabase' in this book.

Copying and moving geodatabase data

In addition to the Feature class to Geodatabase tool, ArcCatalog contains tools to directly move and copy data between geodatabases while preserving object identity, subtypes, relationships, network connectivity, and so on. With this method of copying data, you can copy entire feature datasets or individual feature classes between geodatabases.

When the data is copied, the copy has all the behavior of the original data; any attribute domains referenced in the original geodatabase are copied along with the feature class or table. If the feature class or table participates in a relationship class, then that relationship class, along with the feature class or table it is *related* to through that relationship class, is also copied. As an example, if you copy a feature class with feature-linked annotation, the feature-linked annotation class is automatically copied with the feature class.

If you are copying a feature class into an existing feature dataset, either in the same geodatabase or in another geodatabase, then the spatial reference of the feature class and the feature dataset must match. If the spatial references do not match, you will not be able to copy the data.

You can move feature classes and relationship classes in and out of, or between, feature datasets in the same geodatabase by dragging and dropping them in ArcCatalog. When moving a feature class into a feature dataset, the feature class and the feature dataset must have the same spatial reference.

If you copy or move a network feature class, all the feature classes that participate in the network, and the geometric network itself, are also copied or moved with the feature class.

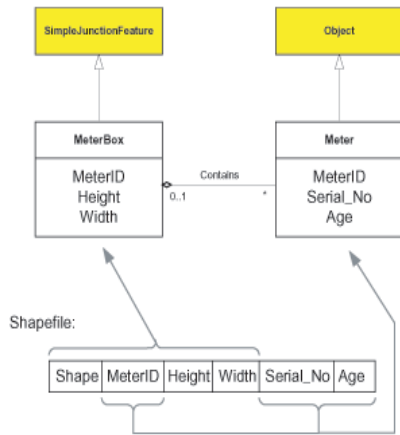
Coverage annotation

In the geodatabase, annotation is not a geometry type but a feature type. Coverage and shapefile features are imported into feature classes that store ESRI simple features with geometry types of point, line, or polygon; annotation is stored in feature classes that store annotation features.

You cannot import a coverage annotation feature class into the geodatabase using one of the data loading tools or wizards. A different process exists for converting coverage annotation to the geodatabase. This process is described in the chapter 'Managing annotation' in this book.

Data conversion versus data loading

The tools and wizards for importing coverages, shapefiles, and INFO and dBASE tables into the geodatabase require that each shapefile and coverage feature class be loaded into a new feature class and each INFO and dBASE table be loaded into a new table.



This diagram shows how you may use the Object Loader or Simple Data Loader to populate a geodatabase schema from a shapefile with different schema.

The feature class or table cannot exist before you begin the import process.

Because an existing feature class or table can be in any number of states, a separate data-loading tool is required to load data from a shapefile, coverage feature class, INFO table, or dBASE table into an existing geodatabase feature class or table.

A feature class can be in one of two states:

- Nonversioned simple data
- Versioned simple, network, or custom object data

In the case of nonversioned simple data, an *edit session* is not required to insert new features or rows into the table or feature class. Once loaded, all data is visible in all versions of the database. This data-loading operation is performed with the Simple Data Loader wizard in ArcCatalog. For information on how

to use this wizard, see ‘Loading data into existing simple feature classes and tables’ in this chapter.

In case of versioned simple, network, or custom objects, an edit session is required to insert new records into the table or feature class to ensure that the network connectivity and version information is managed correctly. This data-loading operation is performed with the Object Loader wizard in ArcMap. For more information on the Object Loader, see *Editing in ArcMap*.

As an example of when you might use the Simple Data Loader or the Object Loader, consider the following:

You have generated your schema using the CASE tool Schema Generation wizard (see the chapter ‘Building geodatabases with CASE tools’ in this book), and you have a simple junction feature class called MeterBox and a table called Meter. MeterBox and Meter participate in a one-to-many relationship class (see the chapter ‘Defining relationship classes’ in this book). MeterBox has the attributes MeterID, Height, and Width. Meter has the attributes Serial_No and Age and the embedded foreign key MeterID, which relates the meter to its meter box.

In your shapefile database, you have maintained your meter boxes and meters in a single shapefile that has the attributes MeterID, Height, Width, Serial_No, and Age. You can use the Object Loader to take the data in that shapefile and split it between the MeterBox feature class and the Meter table while maintaining the relationships between the meter and its meter box.

Use the Object Loader to load the shapefile into the MeterBox feature class, matching the MeterID, Height, and Width fields from the shapefile with those in the feature class. Repeat the process, loading the shapefile into the table (only the attributes will be loaded), matching Serial_No, Age, and MeterID. Since the objects in MeterBox are related to objects in Meter by the

embedded foreign key MeterID, the relationships will be maintained during the data-loading process.

Importing ArcStorm and Map LIBRARIAN data

ArcMap and ArcCatalog display and query ArcStorm and Map LIBRARIAN data that is served by ArcSDE for Coverages. ArcSDE for Coverages layers are treated in the same way as ArcSDE 8 layers in that they can be displayed and queried, but they can't be edited. To import ArcStorm and Map LIBRARIAN data into a geodatabase, use the Feature Class to Geodatabase tool described in 'Importing a geodatabase feature class' later in this chapter.

Loading data into an existing schema

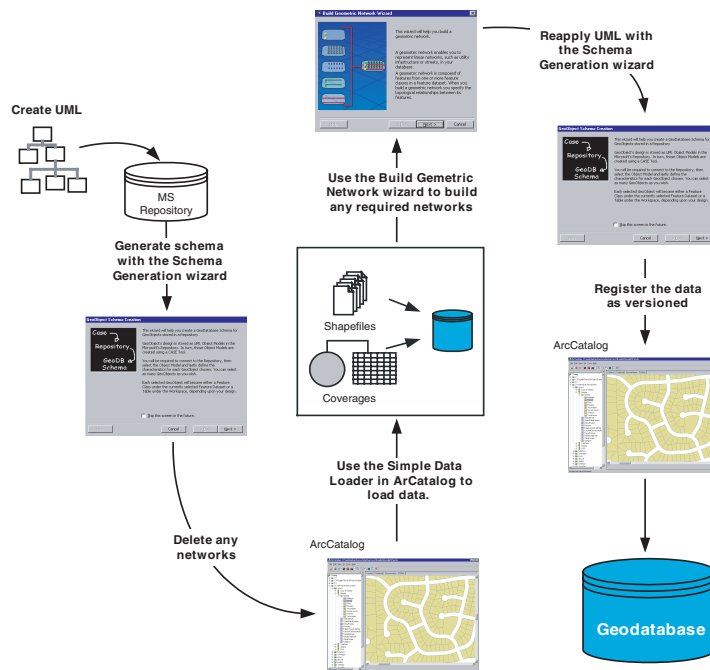
One way to create your geodatabase data model is to use Unified Modeling Language (UML) CASE tools to translate a UML design into an empty geodatabase schema. Another way is to copy the schema of an existing geodatabase. In either case you create an empty geodatabase schema—a set of feature classes, tables, relationship classes, geometric networks, and rules (for more information see the chapter 'Building geodatabases with CASE tools' in this book). If you copy another geodatabase schema you may also have empty topologies. You may want to start directly editing that schema to build your database, or you may want to populate that schema with existing data. When loading data into a geodatabase schema, you must consider how your database's performance will be impacted, 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 strategies and performance considerations of each are outlined below.

The steps presented here incorporate a work flow in which UML and CASE tools were used to generate the schema of your geodatabase, but the same general work flow can be followed if you have arrived at an empty geodatabase schema in another way. There are two strategies you can use to load your data:

Strategy 1: Using the Simple Data Loader:

1. Use the Schema Generation wizard to create the empty geodatabase schema from your UML model in your database.
2. Delete any networks that were created. This will also delete any associated connectivity rules and class extensions.



Loading data into an existing geodatabase schema: Strategy 1

3. Load all of your data into your database using the Simple Data Loader in ArcCatalog.
4. Build any required networks using the Build Geometric Network wizard in ArcCatalog or ArcToolbox.
5. Use the Schema Generation wizard to reapply the UML to the existing data to re-create the network connectivity rules and to assign any class extensions.
6. Create and *validate* any topologies.
7. Register your data as versioned.

This strategy has a number of advantages. Without a network, your data will load much faster. Since the data is not versioned, all of the data will be loaded directly into the base tables and you won't be required to compress your database. If your data model includes geometric networks, deleting the network in step 2 will automatically delete all connectivity rules associated with that network, and all of its participant feature classes will revert to simple feature classes. By reapplying the UML model after the network is built, your connectivity rules are reapplied and any class extensions described by the model are also reassociated with their corresponding classes.

Loading your data before creating topologies will eliminate the overhead of creating *dirty areas* for each new feature that you insert into a participating feature class. If the topology is created after the data is loaded, then a single dirty area spanning all of the features is created, which can then be validated as described in the 'Topology' chapter of this book.

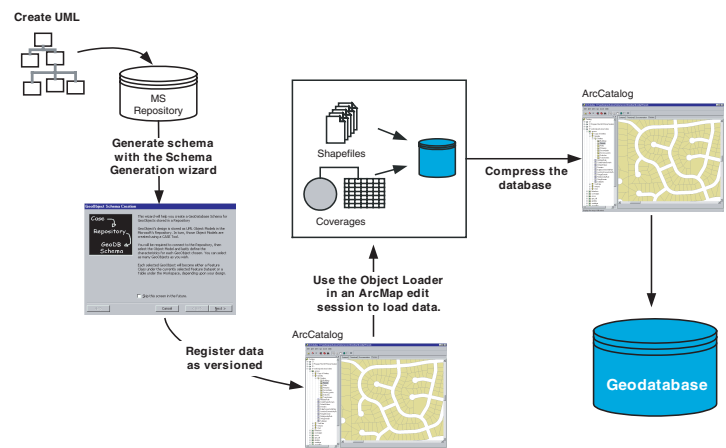
This method's only limitation has to do with custom objects that have custom object creation behavior. Using this strategy, custom creation behavior will not be executed. In this case, you may want to do a combination of the first and second method (see below): load all noncustom features, build networks, apply your model from which to create the custom object classes, then

version your data and use the Object Loader to populate the custom classes.

To learn more about geometric networks and connectivity rules, see the chapter 'Geometric networks' in this book. To learn more about versioning, see the chapter 'Working with a versioned geodatabase' in this book. To learn more about class extensions, see *Exploring ArcObjects*.

Strategy 2: Using the Object Loader:

1. Use the Schema Generation wizard to create the empty geodatabase schema in your database.
2. Use the Simple Data Loader in ArcCatalog to load your existing data into your simple feature classes and tables.
3. Create and validate any topologies.
4. Register your data as versioned.



Loading data into an existing geodatabase schema: Strategy 2

5. Use the Object Loader in ArcMap to load your existing data into your network feature classes. This step automatically builds network topology within an edit session.
6. Run compress to compress the database (you should always run compress after large data loads into versioned classes).
7. Use the Analyze command in ArcCatalog to update the database statistics for each feature class into which you loaded data.

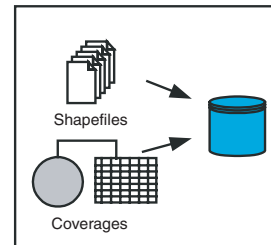
This strategy has a number of disadvantages. First, loading data into network feature classes is a slow process—up to several seconds per feature—that can make using this method impractical for loading a large number of network features. Since it is versioned, all of the data loaded will be in the delta tables, not the base tables for the feature classes. If you use this method to load your data, then once it is loaded you should run compress on your database to push all the records from the delta tables to the base tables. Having your data in the base tables will result in better query speed than having large amounts of data in your delta tables. For more details on base tables, delta tables, and compressing your database to improve performance, see the chapter ‘Working with a versioned geodatabase’ in this book.

Appending data to a geodatabase

Once your database is built and your data is loaded and in production, you may want to append additional data to your existing database. This data must fall within the X/Y domain of the dataset spatial reference established in the creation of the dataset. If you don’t have any network data in your geodatabase, the process is straightforward:

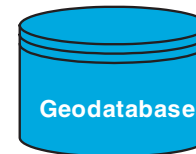
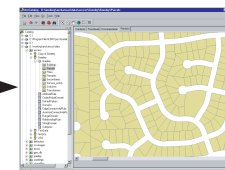
1. Use the Object Loader in ArcMap to load the new data into your feature classes.

Use the Object Loader in an ArcMap edit session to load data.



Compress the database

ArcCatalog



Appending data to a geodatabase using the Object Loader

2. Run compress to compress the database (you should always run compress after large data loads into versioned classes).
3. Use the Analyze command in ArcCatalog to update the database statistics for each feature class into which you loaded data.

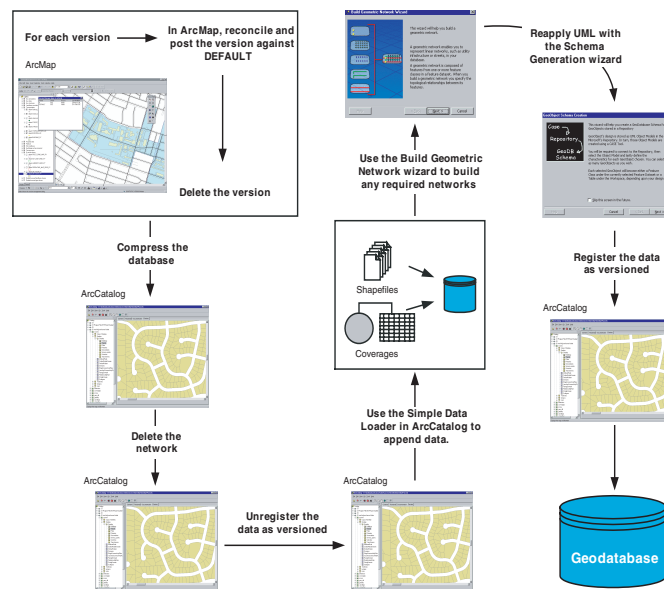
All object behavior is executed such as creating linked annotation for feature classes that have a linked annotation class.

If you are appending data to network feature classes, then the above method will work. However, since the entire network cannot be cached when using the Object Loader to load data into network feature classes, it will be a very slow process—up to several seconds per feature, depending on the number of feature classes in the network. This method may be slow, but it executes any necessary object behavior.

This method will work fine for simple features and features in a topology. If you are appending a large amount of data into a network (more than a few thousand features), then this method may be too time-consuming to be practical. When working with networks, there is an alternate, faster method to appending data.

This method involves unregistering the data as versioned and dropping the network for the duration of the data-loading operation. The fact that the data must be unregistered as versioned is important. When unregistering data as versioned, all edits that have been made on the data that are not in the base table will be lost. The following outlines the steps to take for this method to ensure no data will be lost:

1. Reconcile and *post* each outstanding version in the database against the DEFAULT version. After posting, delete each version.
2. Run compress to compress the database.
3. Unregister the data as versioned. Note: If you have not completed steps 1 and 2 before unregistering your data as versioned, then you will lose any edits that those versions contain.
4. Delete the geometric network.
5. Use the Simple Data Loader in ArcCatalog to load the new data to your existing feature classes.
6. Rebuild the geometric network using the Build Geometric Network wizard in ArcCatalog or ArcToolbox.
7. If you created your geodatabase schema using CASE tools, use the Schema wizard to reapply the UML to the existing data in order to re-create the network connectivity rules and to assign any class extensions. If you are not using CASE tools, then you will need to use ArcCatalog to re-create your connectivity rules.



Appending data to a geodatabase using the Simple Data Loader

8. Register your data as versioned and continue with production. Registering the data as versioned automatically updates the database statistics for the feature classes.

There are a number of limitations to this method that may make it necessary for you to use the first method:

- You cannot use this method if your network has any complex junction features with connection points and custom topology since the process of batch rebuilding the network will not re-create the custom topology.
- The process of rebuilding the network will reconnect all network features you may have disconnected from the network.

- If any of your network feature classes have feature-linked annotation, you cannot use the Simple Data Loader to load features into it. In this case, you must use the Object Loader so that all annotation features will be created along with the new network features.
- This method is incompatible with some work flows. If you have outstanding versions that cannot be reconciled and posted to DEFAULT, then you cannot use this method. Such versions include outstanding design versions that are not complete, are not ready for posting, or are historical versions. If this is the case, you need to use the Object Loader and append your data as part of an edit session.

To learn more about geometric networks, complex junctions, enabled and disabled network features, and connectivity rules, see the chapter ‘Geometric networks’ in this book. To learn more about reconcile, post, compress, and versioning, see the chapter ‘Working with a versioned geodatabase’ in this book.

ArcSDE 8 layers and tables and the geodatabase

In the ArcInfo system, ArcSDE 8 layers and tables are treated as geodatabase feature classes and tables; they can be displayed, queried, and edited. However, these layers and tables participate in other functionality such as validation rules and relationship classes.

ArcSDE 8 layers and tables as defined here were created with applications other than those that use the ArcInfo COM components to manage the database. This includes ArcSDE layers that were created with the ArcSDE administration tools and existing SDE layers from an SDE 3 database. All of these layers and tables can be displayed and queried in ArcInfo 8, and if versioned, they can also be edited.

In order for ArcSDE layers and tables to participate in the more advanced functions of the geodatabase, they must be registered

as feature classes and object classes (tables) with the geodatabase system tables. Once they are registered with the geodatabase, they can have subtypes, default values, and validation rules and can participate in relationship classes. You can drag and drop these feature classes into feature datasets that have the same spatial reference. ArcCatalog contains tools to register an ArcSDE layer or table with the geodatabase; these tools are discussed further in this chapter.

Importing rasters

Raster data can be loaded into a geodatabase using the Raster to Geodatabase tool or Raster to Geodatabase wizard in ArcCatalog and ArcToolbox. When raster data is stored in a geodatabase, it is converted from its original format into a new format for storage in a DBMS. In the DBMS, the raster is stored as many small binary large objects (BLOBs). Reducing the amount of data transferred between the client and the server optimizes performance. Storing multiple resolutions of the raster, cutting the raster into tiles that become the BLOBs, and spatially indexing the raster cause this. Each time the raster is queried, only the tiles necessary to satisfy the query are returned instead of the whole dataset. This makes it possible to store large seamless raster datasets (tens to hundreds of gigabytes) and serve them quickly to a client for display.

There are several storage parameters that you can use to tune your raster data: *pyramids*, tile size, and data compression.

Pyramids are reduced resolution representations of your dataset that are used to improve performance. Pyramids can speed up the display of raster data because the server returns only the data at a specified resolution that is required for the display. Pyramids are created by resampling the original data. The resample methods instruct the server how to resample the data to build the pyramids. Nearest neighbor should be used for nominal data or raster datasets with color maps such as land use data, scanned maps,

and pseudocolor images. Bilinear or cubic should be used for continuous data such as satellite imagery or aerial photography.

The tile size controls the number of pixels you want to store in each BLOB and, therefore, the size of the BLOB. This is specified as a number of pixels in X and Y. A good guideline is to use a tile size that is one quarter the size of the data you plan to serve. For example, if you plan to serve data to a client whose display window will normally be 500x500 screen pixels, use a tile size of 250x250 pixels.

Data Compression compresses the blocks of data before storing them in the geodatabase. The compression used is LZ77, which is a lossless compression, meaning the values of cells in your raster dataset will not be changed. The amount of compression will depend on the data. The fewer values and more homogeneous the data, the higher the compression ratio.

The primary benefit of compressing your data is that it requires less storage space. An added benefit may be improved performance of up to 20 percent because you are transferring smaller packets of data from the server.

A set of geographically continuous raster datasets can be loaded into the geodatabase and managed as individual rasters, a raster catalog, or one seamless raster. A raster catalog is a collection of individual rasters in the geodatabase. You can work with a raster catalog such as a single raster. The server manages the tiling of the individual rasters. Alternatively, you can mosaic your rasters into one seamless raster. A single raster will always perform better than many rasters, so to optimize performance it is best to mosaic spatially continuous raster datasets.

For more discussion of raster data, see *Modeling Our World*.

Importing shapefiles

A wizard lets you easily import shapefiles into new or existing feature datasets in the database or into the database as standalone feature classes. The wizard calculates default values, which can be used to import the shapefile. However, you can also customize those parameters. After using the wizard a few times, try using the Shapefile to Geodatabase tool in ArcCatalog, which lets you import several shapefiles into the database at once.

The wizard and tools for importing shapefiles into a geodatabase are also accessible from ArcToolbox.

See Also

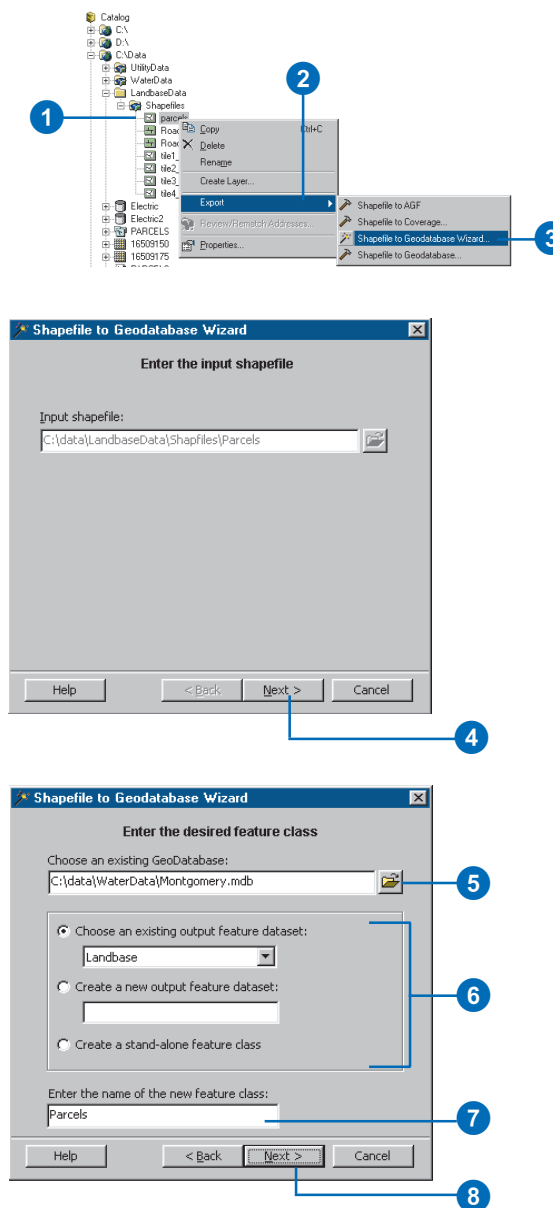
For more information on using tools to perform operations on multiple shapefiles, see Using ArcToolbox.

Importing shapefiles using default values

1. Right-click the shapefile in the ArcCatalog tree that you want to import into your geodatabase.
2. Point to Export.
3. Click Shapefile to Geodatabase Wizard.
4. Click Next.
5. Navigate to the database or database connection into which you want to import the shapefile or type its path.
6. Click the first option and click the feature dataset's name to import the shapefile into an existing feature dataset in the database.

Click the second option and type the new feature dataset's name to import the shapefile as a standalone feature class.

7. Type a name for the new feature class.
8. Click Next.



Tip

Using the Import menu

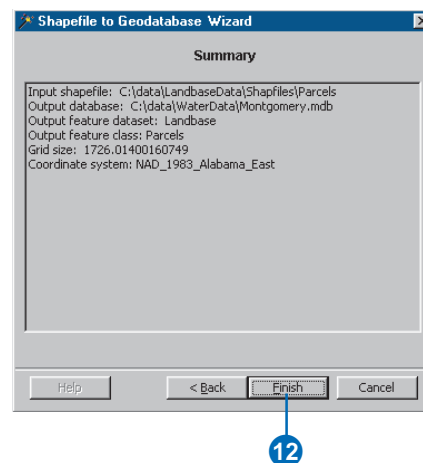
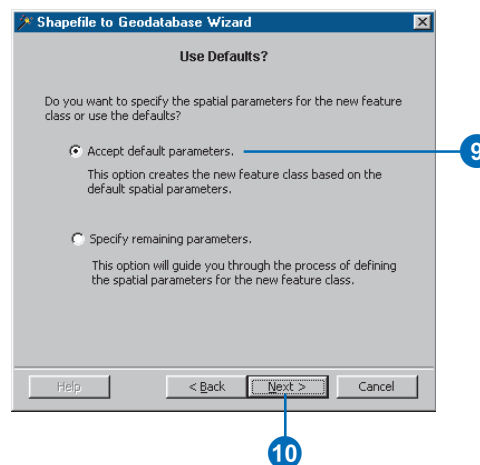
Shapefiles can also be imported into geodatabases by clicking the geodatabase and using the Import menu. In this case, the destination database is prepopulated, and you must browse for—or type the name of—the shapefile.

Tip

Batch importing

If you select multiple shapefiles from the contents view of ArcCatalog and click Import/Shapefile to Geodatabase, the tool will automatically be set in batch mode with all of the input shapefiles prepopulated.

9. Click the first option to accept the default parameters.
10. Click Next.
11. Review the options you specified for your data import operation. If you want to change something, you can go back through the wizard by clicking the Back button.
12. Click Finish to import the shapefile into the database when satisfied with your options.

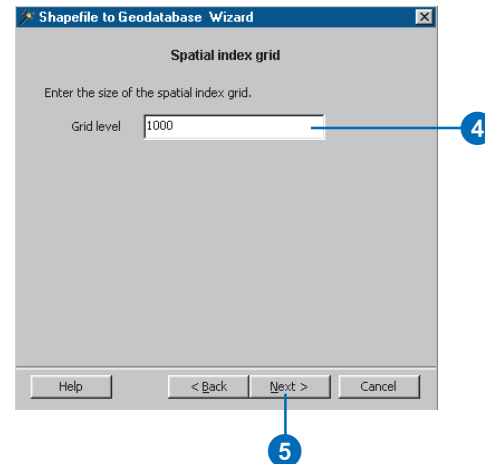


See Also

Managing ArcSDE Services
*explains in detail the spatial index
grid and how its values should be
calculated.*

Importing shapefiles using custom values

1. Follow steps 1 through 8 for 'Importing shapefiles using default values'.
2. Click the second option to import the shapefile defining custom parameters.
3. Click Next.
4. Type custom spatial index grid values if you do not want to use the defaults. (Only one index grid is used in personal geodatabases, while ArcSDE geodatabases use up to three.)
5. Click Next. ►



Tip

Corrected field names

Changes to the field names are proposed on invalid field names. For example, when a field name contains an invalid character, such as a hyphen, the hyphen is replaced by an underscore in the corrected field name. An error message indicating why the original name was corrected appears in the Original Error column.

Tip

Undoing field name edits

You can click Revert to change the corrected field names back to their original values, as automatically corrected by the Import Wizard.

Tip

Changing the spatial reference

All the feature classes in a feature dataset must use the same spatial reference. Therefore, once a feature dataset has been created, you cannot change its spatial reference. The spatial reference panel will only appear when you are importing the shapefile into a new feature dataset or into a standalone feature class.

6. Review the names in the Corrected Fields column. Click a name and type a new one if you do not want to use the default.

7. Double-click in the Delete field column, then click Yes if you do not want to include one of the original fields in the new feature class.

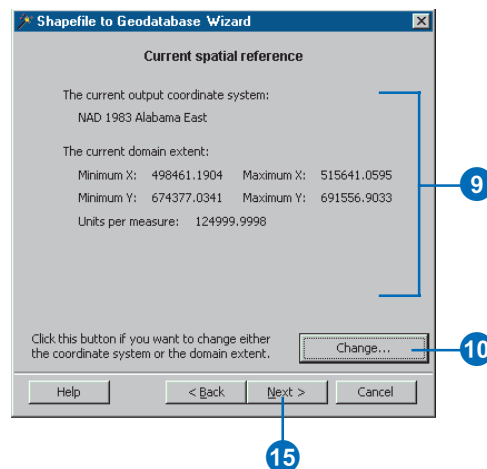
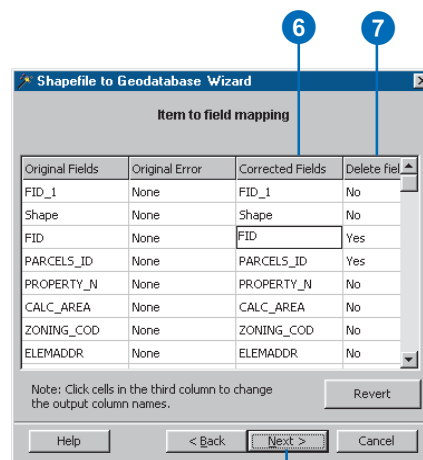
8. Click Next.

If you are importing the shapefile into an existing feature dataset, skip to step 16.

9. Review the summary of the coordinate system that will be used.

10. Click Change if you want to modify any of the shapefile's coordinate system parameters.

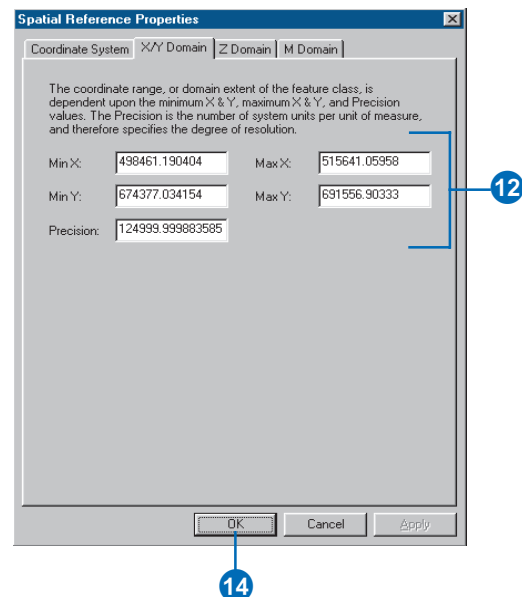
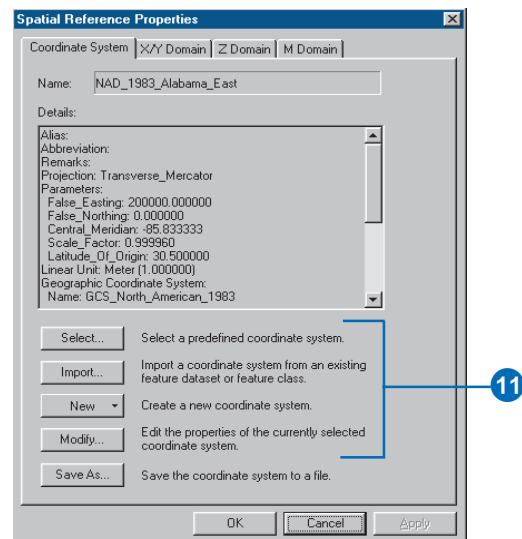
Otherwise, skip to step 15. ►



See Also

For details on how to create a new coordinate system and for important information about spatial reference and how it affects your data, see the chapter 'Creating new items in a geodatabase' in this book.

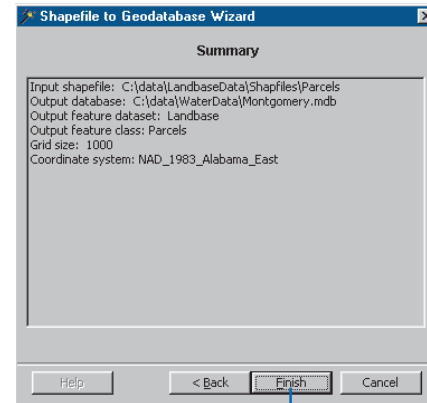
11. Click one of the buttons to change the default coordinate system by one of the following methods: selecting a preexisting one; importing a coordinate system from a shapefile, coverage, or feature class; defining a new one; or modifying the default coordinate system's parameters.
12. Click the X/Y Domain tab and modify the default parameters.
13. Repeat step 12 with the Z Domain and M Domain tabs, if present.
14. Click OK. ►



15. Click Next.
16. Review the summary of the parameters used to import the shapefile.

To change a parameter, navigate back to the appropriate panel by clicking Back.

17. Click Finish.



Importing coverages

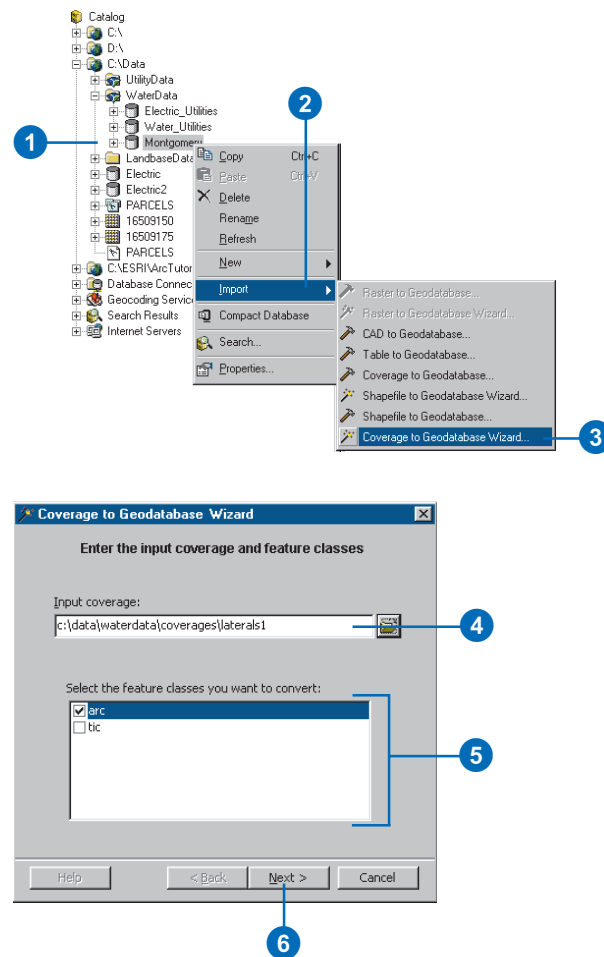
You can use a wizard to import a coverage's feature classes into new or existing feature datasets in the database or into the database as standalone feature classes. You can choose which feature classes you want to import. The wizard calculates default parameters, which can be used to import the feature classes; you can customize those parameters for each feature class. When you choose to customize the parameters, you will cycle through a series of wizard panels, once for each feature class.

After using the wizards a few times, try using the Coverage to Geodatabase tool in ArcCatalog, which lets you import several coverages into the database at once.

You can also access the wizards and tools for importing coverages into a geodatabase from ArcToolbox.

Importing coverages using default values

1. Right-click the database or feature dataset in the ArcCatalog tree into which you want to import the coverage.
2. Point to Import.
3. Click Coverage to Geodatabase Wizard.
4. Navigate to the coverage you want to import into the database or type its path.
5. Check the boxes next to the feature classes you want to import.
6. Click Next. ►



Tip

Feature class names

When importing the coverage with the default settings, the loader will automatically assign names to the output feature classes. If you want to specify names, you must load specifying the importing parameters.

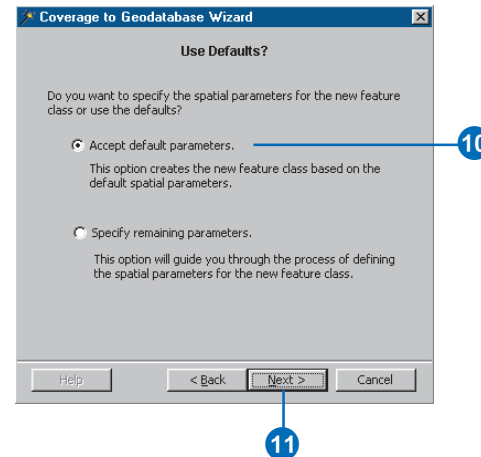
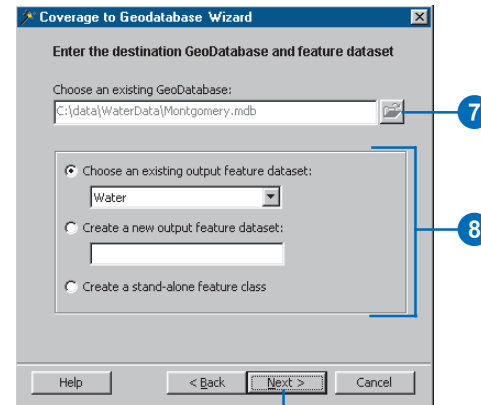
7. The geodatabase will be prepopulated with the one clicked in the ArcCatalog tree. Had you evoked this wizard by clicking export on the coverage, then you would have to browse for, or type in, the database.

8. If you right-click on a feature dataset, then the dataset will be prepopulated. If not, click the first option to choose an existing feature dataset and choose the feature dataset's name from the list.

Click the second option to create a new feature dataset and type the new feature dataset's name.

Click the third option to create a standalone feature class.

9. Click Next.
10. Click the first option to import the feature classes using the default parameters.
11. Click Next. ►



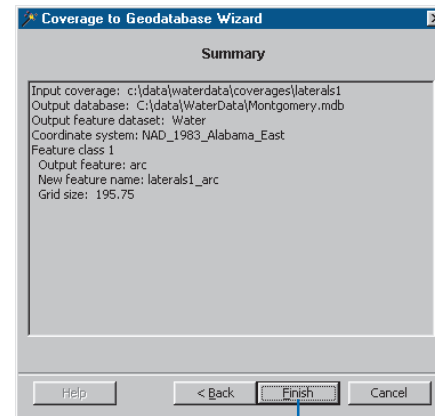
Tip

Importing multiple feature classes

If you have chosen to import multiple coverage feature classes into a geodatabase, each will be imported into a separate feature class.

A progress bar will be displayed for each coverage feature class imported.

12. Review the options you specified for your data import operation. If you want to change something, you can go back through the wizard by clicking Back.
13. Click Finish to import the coverage into the database when satisfied with your options.



Tip

Changing the spatial reference

All feature classes in a feature dataset must use the same spatial reference. Therefore, once a feature dataset has been created, you cannot change its spatial reference. The spatial reference panel will only appear when you are loading the coverage into a new feature dataset or into a standalone feature class.

See Also

For important information about spatial reference and how it affects your data, see the chapter 'Creating new items in a geodatabase' in this book.

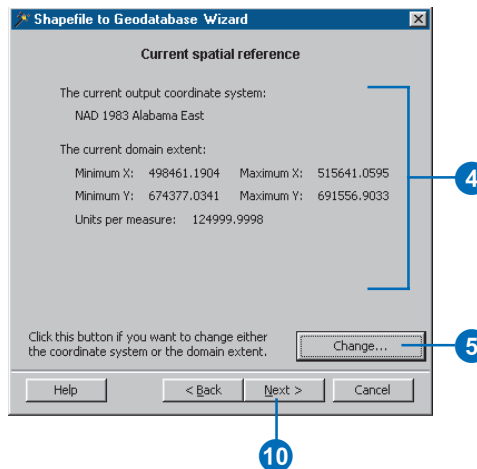
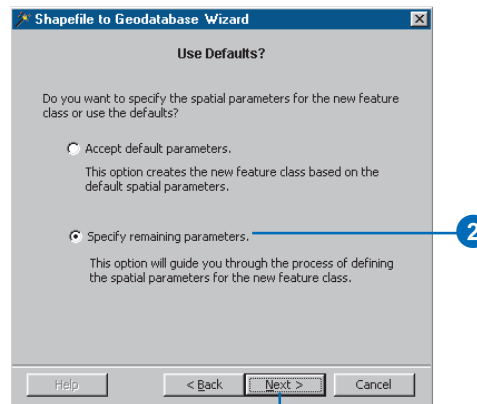
Importing coverages using custom values

1. Follow steps 1 through 9 for 'Importing coverages using default values'.
2. Click the second option to import the coverage defining custom values.
3. Click Next.

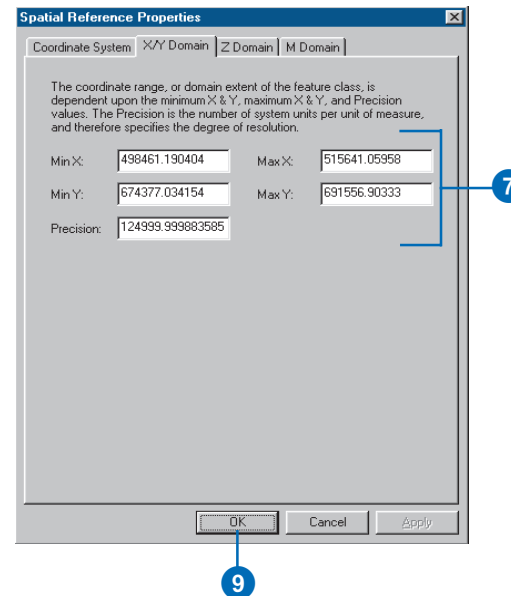
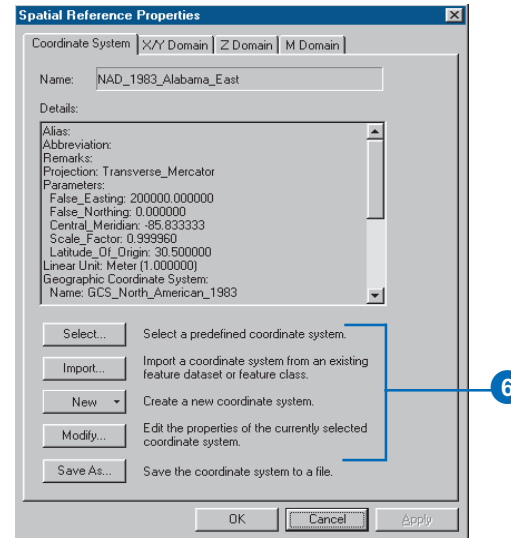
If you are importing the coverage into an existing feature dataset or into a standalone feature class, skip to step 11.

4. Review the summary of the coordinate system that will be used.
5. Click Change if you want to modify any of the coverage's coordinate system parameters.

Otherwise, skip to step 10. ►



6. Click one of the buttons to change the default coordinate system by selecting a preexisting one; importing a coordinate system from a shapefile, coverage, or feature class; defining a new one; or modifying the default coordinate system's parameters.
7. Click the X/Y Domain tab and modify the default parameters.
8. Repeat step 7 with the Z Domain and M Domain tabs, if present.
9. Click OK.
10. Click Next. ►



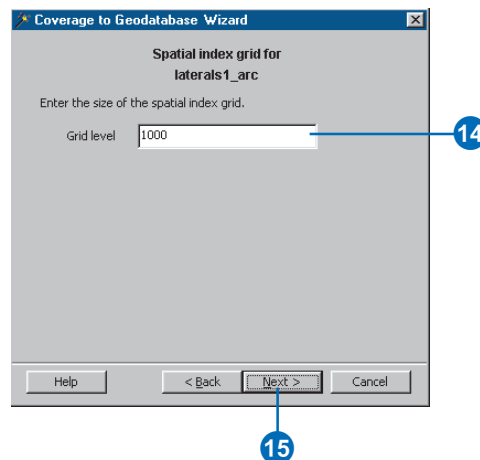
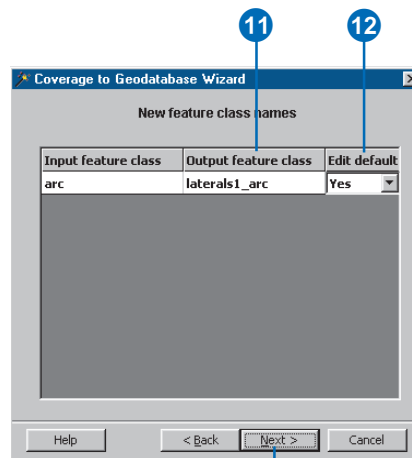
Tip

Naming a feature class

Each feature class in the database must have a unique name.

11. Review the names in the Output feature class column. Click a name, then type a new one if you do not want to use the default.
12. Double-click in the Edit defaults column and click Yes if you want to define custom parameters for loading a feature class.
13. Click Next.

If you are not defining custom parameters for any feature classes, skip to step 21.
14. Type custom spatial index grid values if you do not want to use the defaults. (Only one index grid is used in personal geodatabases, while ArcSDE geodatabases use up to three.)
15. Click Next. ►



Tip

Corrected field names

When importing the coverage with the default settings, the loader will automatically assign names to the output feature classes. If you want to specify names, you must load specifying the importing parameters.

Tip

Undoing field name edits

You can click *Revert* to change the corrected field names back to their original values as automatically corrected by the Import wizard.

16. Review the names in the Corrected Fields column. Click a name and type a new one if you do not want to use the default.

17. Double-click in the Delete field column. Click Yes if you do not want to include one of the original fields in the new feature class.

18. Click Next.

If you are importing the coverage into a feature dataset, skip to step 20.

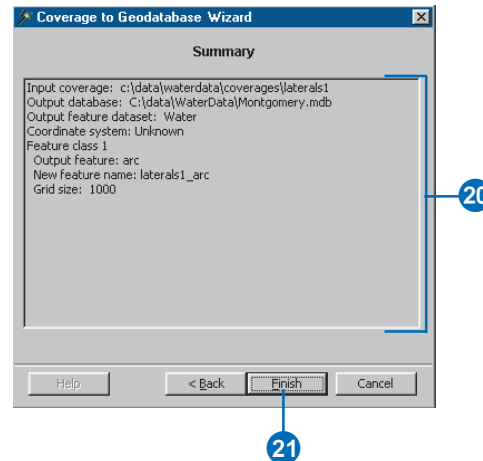
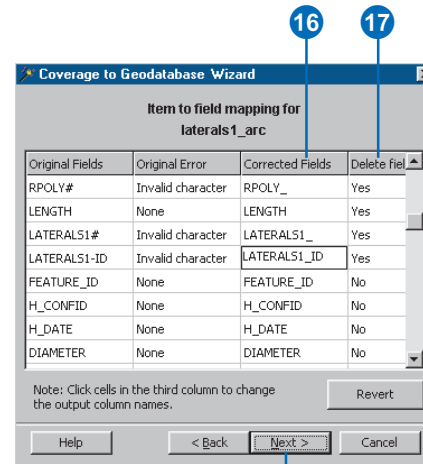
19. Review the summary of the coordinate system that will be used to create the standalone feature class. To modify the coordinate system, follow steps 5 through 10 in this task.

Repeat steps 14 through 19 for each feature class whose parameters you chose to customize.

20. Review the summary of the parameters used to import the coverage's feature classes.

To change a parameter, navigate back to the appropriate panel by clicking *Back*.

21. Click Finish.



Importing tables

The Table to Geodatabase import tool lets you easily import your dBASE and INFO attribute tables into a geodatabase.

The tool will automatically correct any illegal or duplicate field names and will also allow you to specify how these corrections are made.

This same tool can also be used to import multiple INFO and dBASE tables in the database. For more information on using the batch capabilities of this tool, see *Using ArcToolbox*.

You can also access the tools for loading tables into a geodatabase from ArcToolbox.

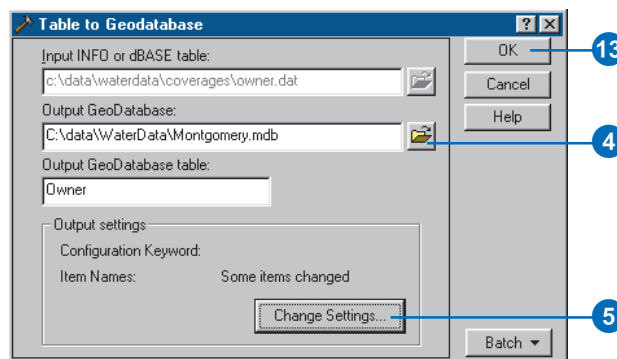
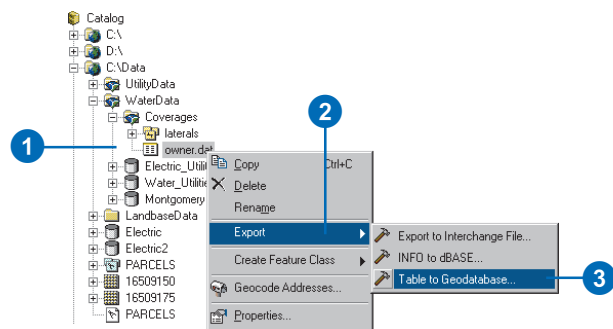
Tip

Batch loading

If you select multiple tables from the contents view of ArcCatalog and click Import to Geodatabase, the tool is automatically set in batch mode with all of the input tables prepopulated.

1. Right-click on the INFO or dBASE table in the ArcCatalog tree that you want to import.
2. Point to Export.
3. Click Table to Geodatabase.
4. Navigate to the geodatabase or ArcSDE geodatabase connection into which you want to import the table or type in its path.
5. Click Change Settings to manually modify illegal and duplicate field names or to specify a configuration keyword.

If you want the tool to autocorrect the field names and you want to use the default storage configuration, skip to step 9. ►

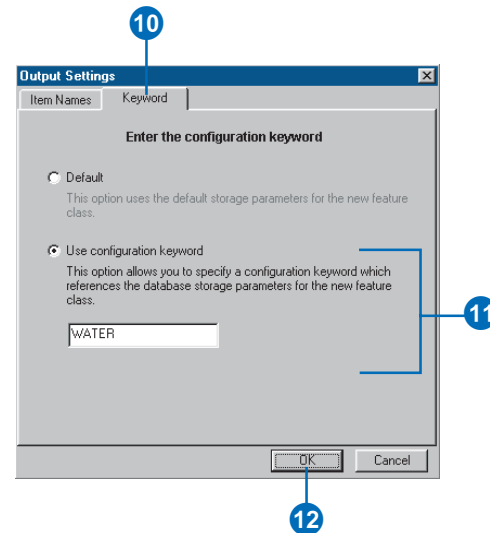
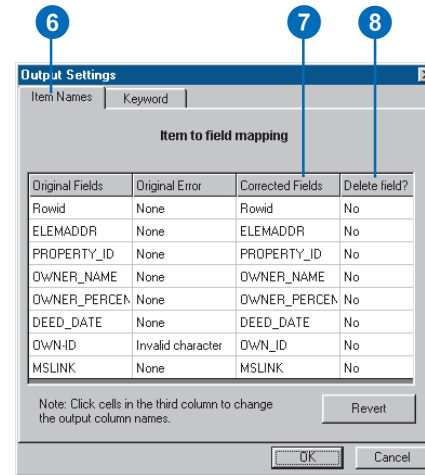


Tip

Undoing field name edits

You can click *Revert* to change the corrected field names back to their original values as automatically corrected by the Import tool.

6. Click Item Names to manually modify illegal and duplicate field names.
7. Review the names in the Corrected Fields column. Click a name and type a new one if you do not want to use the default.
8. Double-click in the Delete field column. Click Yes if you do not want to include one of the original fields in the new table.
9. Skip to step 12 if you are importing this feature class into a personal geodatabase.
10. Click the Keyword tab to specify a configuration keyword.
11. Click Use configuration keyword if you want to create the table using a custom storage keyword and type the keyword you want to use.
12. Click OK.
13. Click OK to import the table.



Importing a geodatabase feature class

The Geodatabase to Geodatabase data import tool lets you easily import a feature class from one geodatabase into another. You can also use this tool to import a feature class into a new feature class in the same geodatabase.

This process does not preserve object identity and always results in a simple feature class that contains no subtypes, default values, and so on (see the chapter ‘Defining relationship classes’ in this book).

Like the other import tools, you can change the spatial reference and modify the output fields and field names. These tools are accessible from either ArcCatalog or ArcToolbox.

Tip

Batch loading

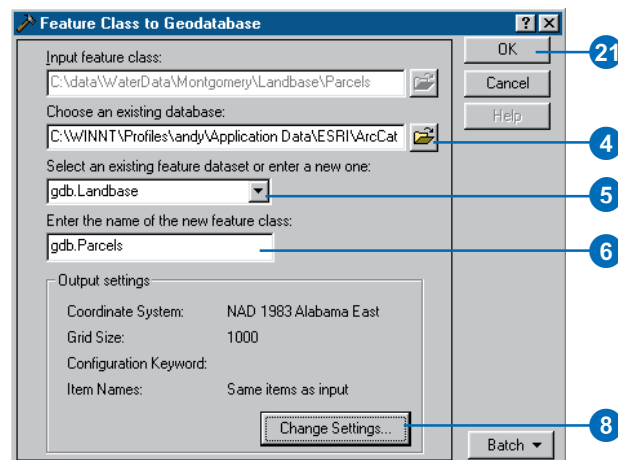
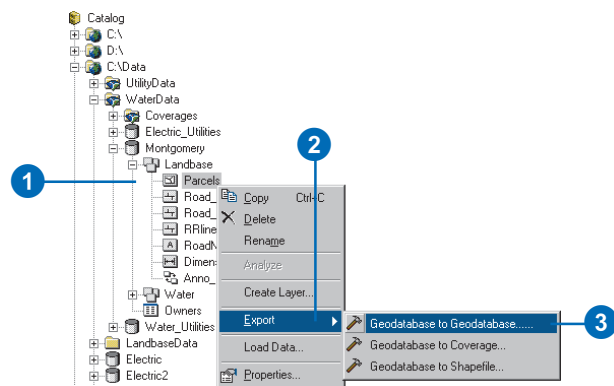
If you select multiple feature classes from the contents view of ArcCatalog and click on Import to Geodatabase, the tool will automatically be set in batch mode with all of the input feature classes prepopulated.

1. Right-click the feature class in the ArcCatalog tree that you want to import to a new geodatabase feature class.
2. Point to Export.
3. Click Geodatabase to Geodatabase.
4. Navigate to the geodatabase or ArcSDE geodatabase connection you want to import the feature class into or type its path.
5. Click the dropdown arrow to see a list of available feature datasets and click the feature dataset if you want to import the feature class into an existing feature dataset.

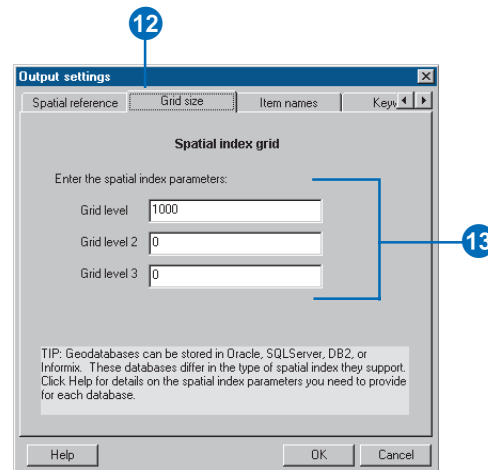
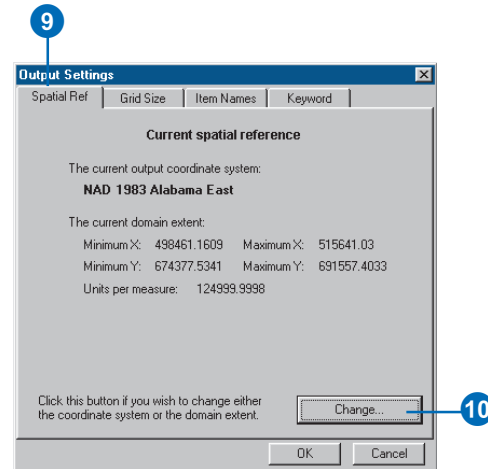
If you want to create a new feature dataset, type its name.

If you want to create a standalone feature class, then leave this blank.

6. Type the name for the new feature class.
7. Skip to step 21 if you want to load the data with all of the default settings.
8. Click Change Settings if you want to change the output spatial reference, grid size, or field names. ►



9. Click the Spatial Ref tab if you are importing this feature class into a standalone feature class or into a new feature dataset and want to change the output spatial reference.
10. Click Change. Otherwise, skip to step 13.
11. Follow steps 11 through 13 for 'Importing shapefiles using custom values' to change the spatial reference.
12. Click the Grid size tab to change the grid size.
13. Type custom spatial index grid values. (Only one index grid is used in personal geodatabases, while ArcSDE geodatabases use up to three.) ►

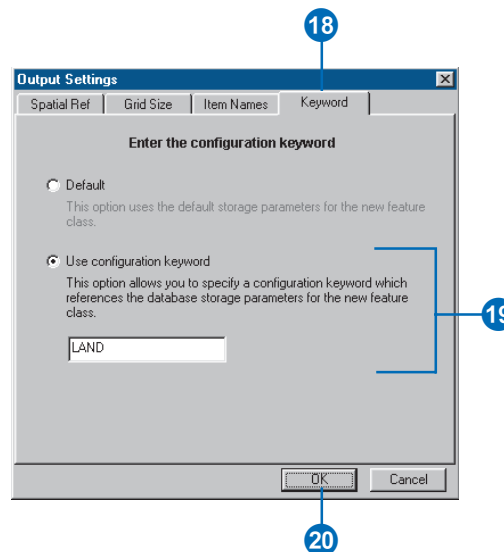
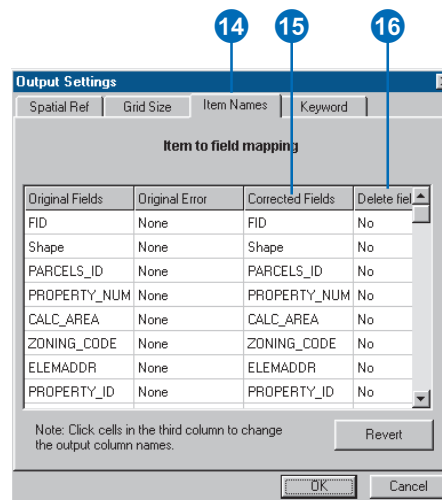


Tip

Undoing field name edits

You can click Revert to change the corrected field names back to their original values as automatically corrected by the Import tool.

14. Click the Item Names tab to manually modify illegal and duplicate field names.
15. Review the names in the Corrected Fields column. Click a name and type a new one if you do not want to use the default.
16. Double-click in the Delete field column, then click Yes if you do not want to include one of the original fields in the new feature class.
17. Skip to step 20 if you are importing this feature class into a personal geodatabase.
18. Click Keyword to specify a configuration keyword.
19. Click Use configuration keyword if you want to create the table using a custom storage keyword, then type the keyword you want to use.
20. Click OK.
21. Click OK to import the feature class.



Importing a CAD feature class

The CAD to Geodatabase data import tool lets you easily import a feature class from a CAD dataset to a geodatabase. You can use this tool to import CAD feature classes from AutoCAD DWG, MicroStation DGN, and Drawing Interchange File (DXF) formats.

Like the other import tools, you can change the spatial reference and modify the output fields and field names. These tools are accessible from either ArcCatalog or ArcToolbox.

Tip

Batch loading

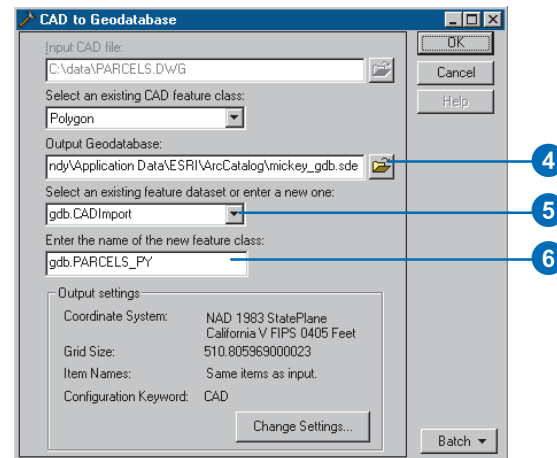
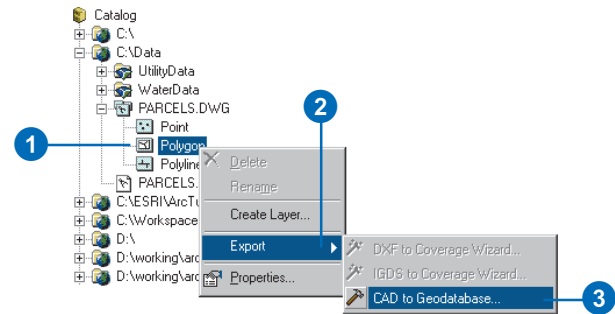
If you select multiple feature classes from the contents view of ArcCatalog and click on Import to Geodatabase, the tool will automatically be set in batch mode with all of the input feature classes prepopulated.

1. Right-click the CAD feature class in the ArcCatalog tree that you want to import as a new geodatabase feature class.
2. Point to Export.
3. Click CAD to Geodatabase.
4. Navigate to the geodatabase or ArcSDE geodatabase connection you want to import the feature class into or type its path.
5. Click the dropdown arrow to see a list of available feature datasets and click the feature dataset if you want to import the feature class into an existing feature dataset.

If you want to create a new feature dataset, type its name.

If you want to create a standalone feature class, then leave this blank.

6. Type the name for the new feature class.
7. Follow steps 7 to 21 for 'Importing a geodatabase feature class'.



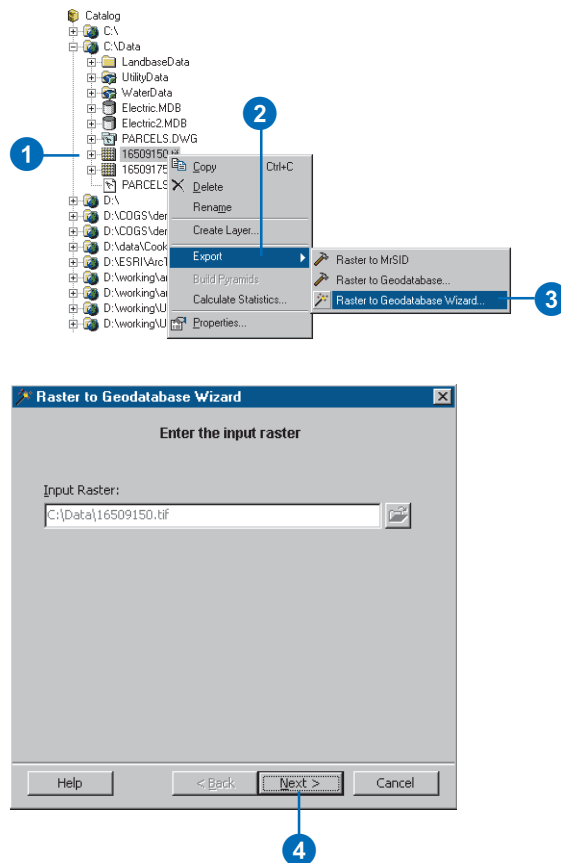
Importing rasters

Raster data can be loaded into a geodatabase using the Raster to Geodatabase tool or Raster to Geodatabase wizard in ArcCatalog and ArcToolbox.

There are several storage parameters that you can use to tune your raster data including pyramids, tile size, and data compression.

Importing rasters using default values

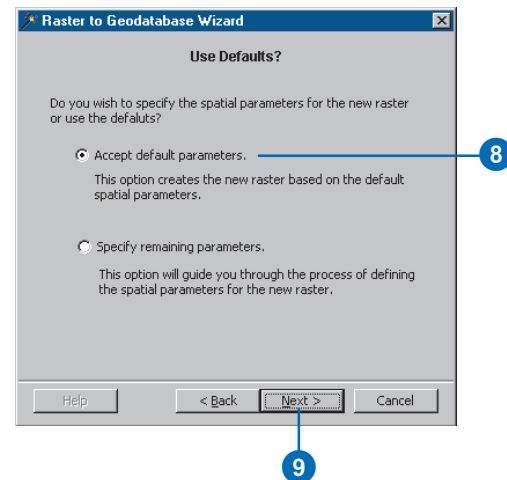
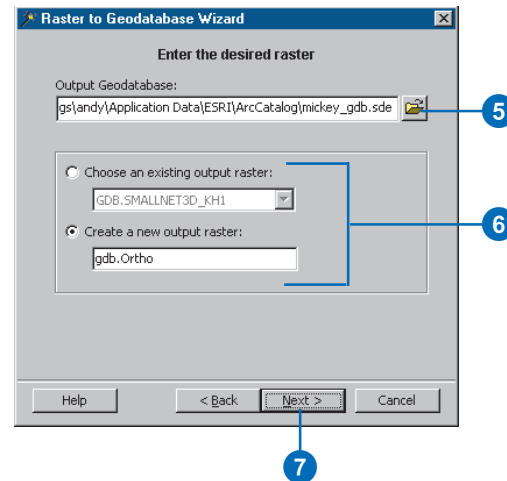
1. Right-click the raster in the ArcCatalog tree that you want to import into your geodatabase.
2. Point to Export.
3. Click Raster to Geodatabase Wizard. The wizard appears with the input raster field populated with the selected raster.
4. Click Next. ►



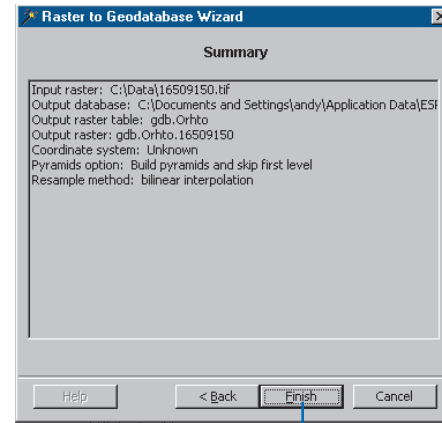
5. Navigate to the database connection into which you want to import the raster or type its path.
6. Click the first option to import and mosaic the raster into an existing raster in the database or to update an existing raster, then click the raster name from the dropdown list.

Or click the second option and type the new raster name to import the raster into a new raster.

7. Click Next.
8. Click the first option to accept the default parameters.
9. Click Next. ►

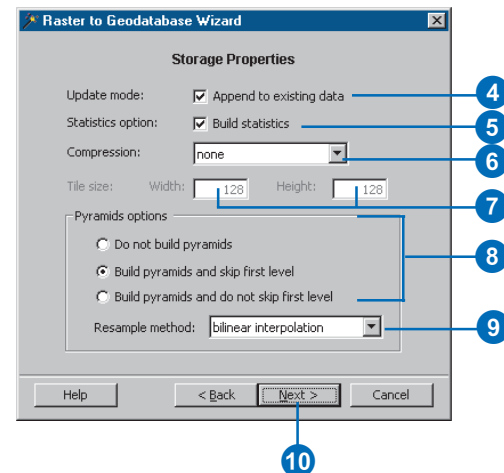
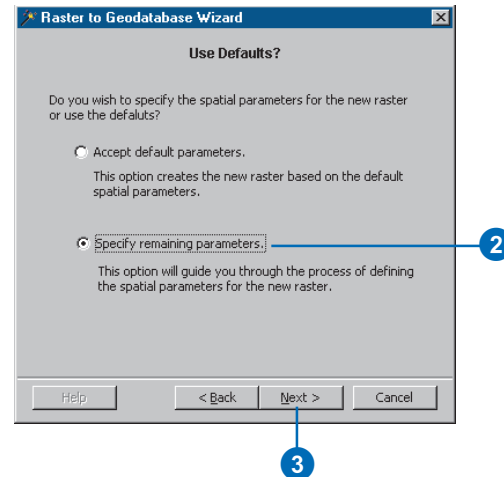


10. Review the options you specified for your data import operation. If you want to change something, you can go back through the wizard by clicking Back.
11. When satisfied with your options, click Finish to import the raster into the database.



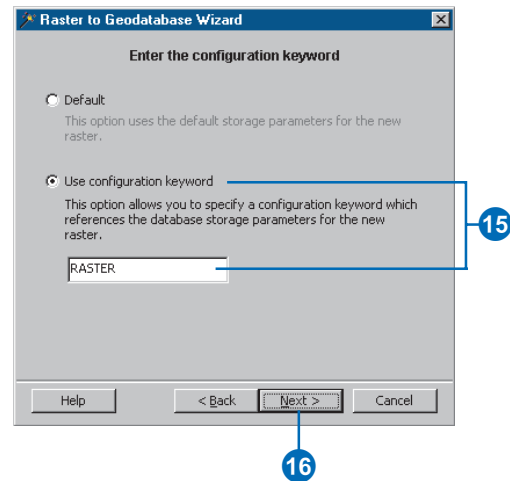
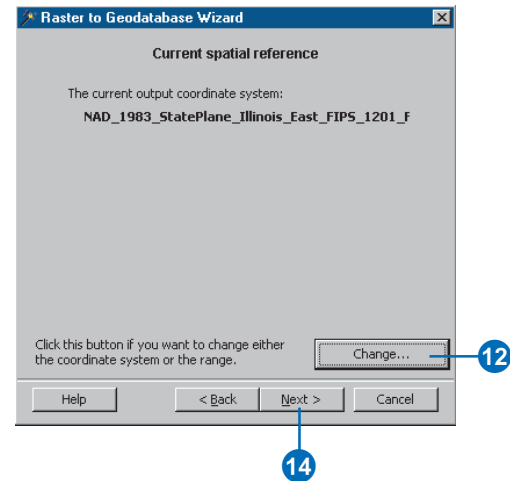
Defining custom values for raster to feature class conversion

1. Follow steps 1 through 8 for 'Importing rasters using default values'.
2. Click the second option to import the raster defining custom parameters.
3. Click Next.
4. Check the Append to existing data check box if you import to existing raster and want to mosaic the rasters.
5. Check the Build statistics check box if you want to build statistics for the raster in the database.
6. Click the Compression dropdown arrow and click none for no compression or click LZ77 to compress the raster.
7. Type the tile width and tile height if you want to change them.
8. Click the first option if you don't want to build pyramids after importing the raster into the database. Or click the second option to build pyramids and skip the first level for the raster imported into the database. Or click the third option to build pyramids and not skip the first level.



9. Click the resample method from the dropdown list if you choose build pyramids.
10. Click Next.
11. Review the summary of the coordinate system that will be used.
12. Skip to 15 if you are importing the raster into an existing table.

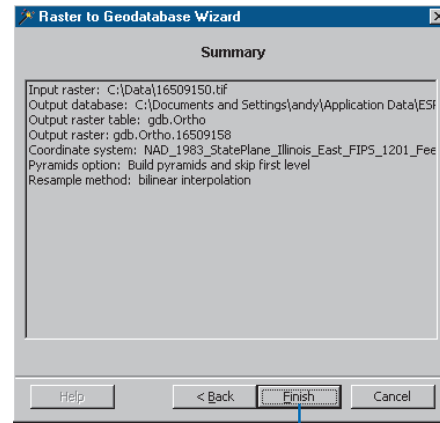
Click Change if you want to modify any of the raster's coordinate system parameters.
13. Follow steps 11 to 14 for 'Importing a shapefile with custom values'.
14. Click Next.
15. Click Use configuration keyword if you want to create the table using a custom storage keyword and type the keyword you want to use.
16. Click Next. ►



17. Review the summary of the parameters used to import the raster.

To change a parameter, navigate back to the appropriate panel by clicking Back.

18. Click Finish.



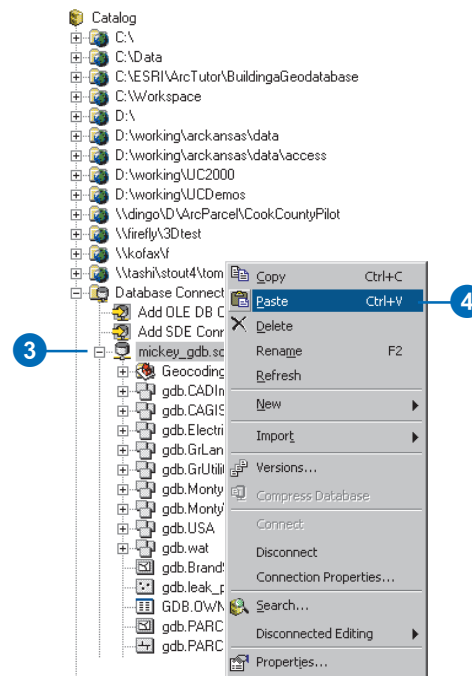
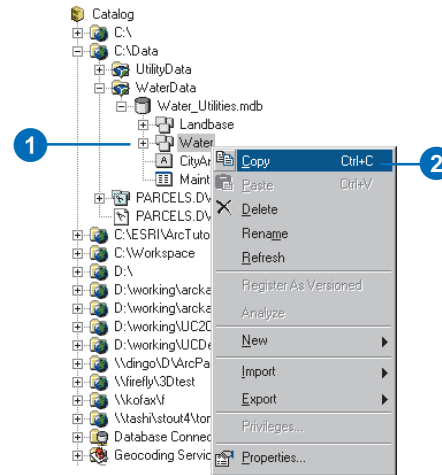
Copying geodatabase data

ArcCatalog contains tools that directly move and copy data between geodatabases while preserving object identity, subtypes, relationships, network connectivity, and so on. Using these tools, you can copy entire feature datasets or individual feature classes between geodatabases.

When the data is copied, the copy has all the behavior of the original and any attribute domains it referenced in the original geodatabase are also copied along with the feature class or table. If the feature class or table participates in a relationship class, then that relationship class, along with the feature class or table it is related to through that relationship class, is also copied. As an example, if you copy a feature class with feature-linked annotation, the feature-linked annotation class is automatically copied along with the feature class.

If you are copying a feature class into an existing feature dataset, either in the same geodatabase or in another geodatabase, then the ►

1. Right-click the dataset in the ArcCatalog tree that you want to copy. This can be a feature dataset, feature class, or table.
2. Click Copy.
3. Right-click the geodatabase to which you want to copy the data.
4. Click Paste. ►



spatial reference of the feature class and the feature dataset must match. If the spatial references do not match, you will not be able to copy the data.

You can move feature classes and relationship classes in and out of, or between, feature datasets in the same geodatabase by dragging and dropping them in ArcCatalog. When moving a feature class into a feature dataset, the feature class and the feature dataset must have the same spatial reference.

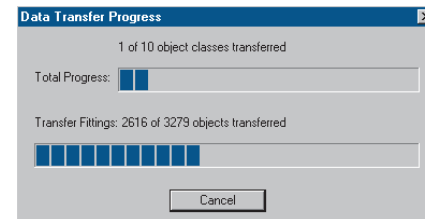
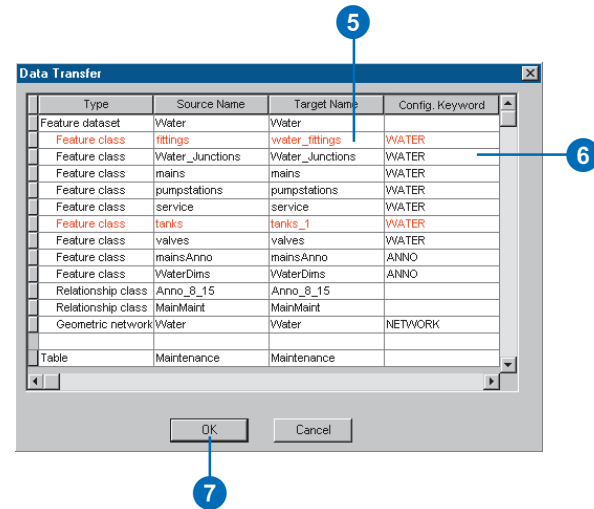
If you copy or move a network feature class, all the feature classes that participate in the network, and the geometric network itself, are also copied or moved along with the feature class.

When you Paste data into a geodatabase, a dialog box appears. It allows you to see exactly what data is being copied, resolve any name conflicts, and assign configuration keywords for each object being copied.

A dialog box appears that indicates what data is being copied. Any name conflicts are automatically resolved and highlighted in red.

5. Type over the target name to change any of the resolved names.
6. Type a configuration keyword under Config. Keyword for an object if you want to use a configuration keyword for any of the objects being copied.
7. Click OK.

A progress indicator will appear to indicate the progress of the data copy.



A progress indicator will appear to indicate the progress of the data copy.

Extracting data

Extracting data from a geodatabase is similar to copying and pasting data from one geodatabase to another but it also allows you to apply filters to the data you wish to extract, such as a selection set or definition query. This means you don't have to copy all the records for your selected datasets, only the records that you are interested in. Data may be extracted to/from either an ArcSDE or personal geodatabase.

ArcMap provides the framework for extracting data in ArcGIS. An ArcMap document should be configured in a manner to support each extract operation. For example, zoom in to the area of interest, select the required features, and apply the definition queries before extracting the data.

Once the extract operation has begun, the following default behavior will determine what data is extracted.

All layers and tables for the selected ArcSDE geodatabase present in the ArcMap document will be included in the extract operation.

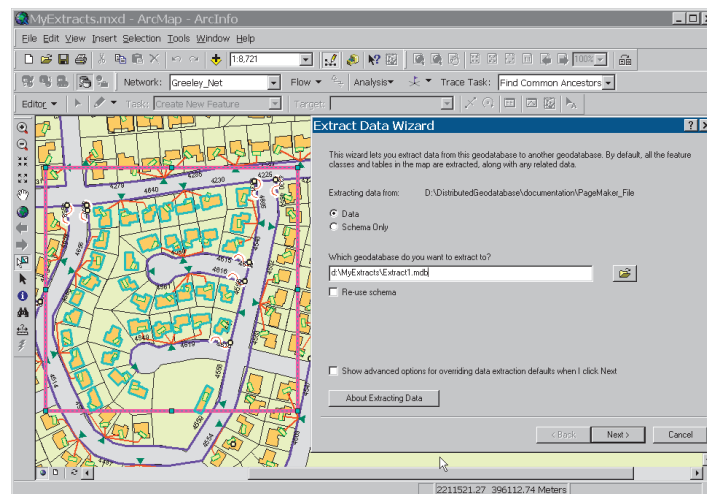
The following data filters will be applied:

- The current view extent of the ArcMap document or the boundary of a currently selected graphic will determine the spatial extent of the data to extract.
- Any selections that exist on those layers or tables will be honored.
- Any definition queries applied to those layers or tables will be honored.

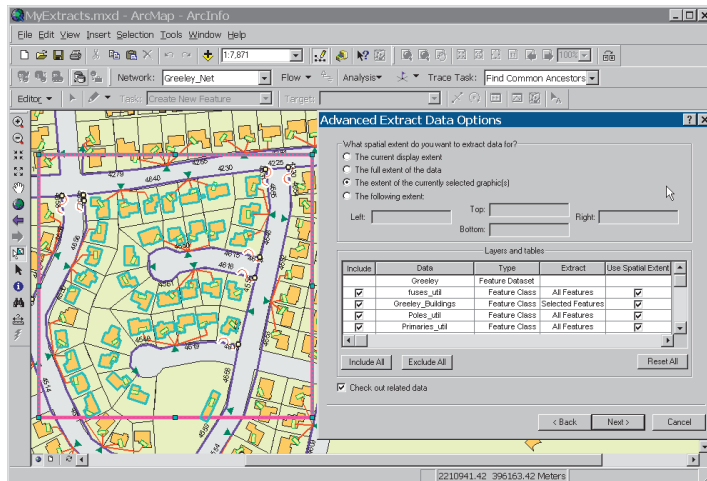
Note: The intersection of these filters will determine what data will be extracted.

- To maintain *data integrity*, any related objects will automatically be included in the extract (for example, annotation related to features) regardless of whether or not they are currently present in the ArcMap document.

- The list of data to extract will also be expanded to include dependent datasets; for example, all features classes in a geometric network, topology, or feature dataset will be included if just one feature class from the network, topology, or geometric network is selected for extraction.



These default options may be overridden as required through the settings of some advanced options to customize the extraction operation. For example, you can choose to use an existing selection set or definition query for a specific layer or table, or override this and get all the records instead. You can also choose



to override the spatial extent defaults and choose one of the other available spatial extents for each layer.

Extracting data is similar to checking out data—the same defaults, advanced options, and general processes for copying geometric networks, related data, and topologies apply to both operations. For more information on checking out data, see the ‘Disconnected editing’ chapter in this book. However, unlike checking out and checking in data, once data has been extracted it cannot be merged back into the source geodatabase at a later time.

Using the Extract Data wizard

The Extract Data wizard allows you to copy and paste data stored in one geodatabase to another geodatabase. Data can be copied in and out of both SDE and personal geodatabases. By default, all the feature classes and tables in the map are extracted along with any related data.

While the Extract Data wizard lets you copy and paste data from one geodatabase into another, any modifications that have been made on the extracted data cannot be merged back into the original geodatabase. The ability to check out, edit, and check data back in is described in the chapter 'Disconnected editing' in this book.

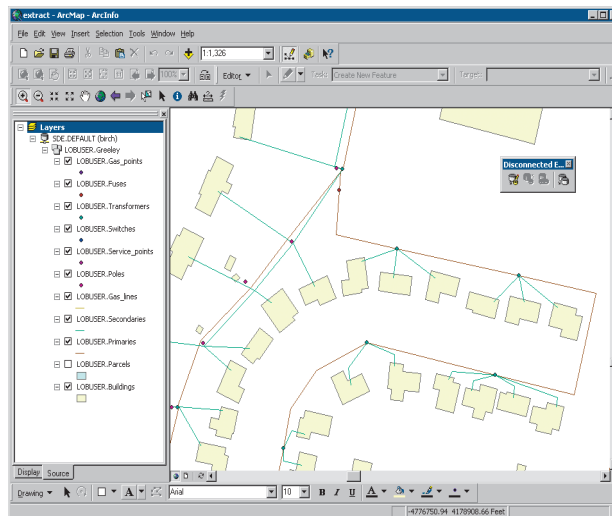
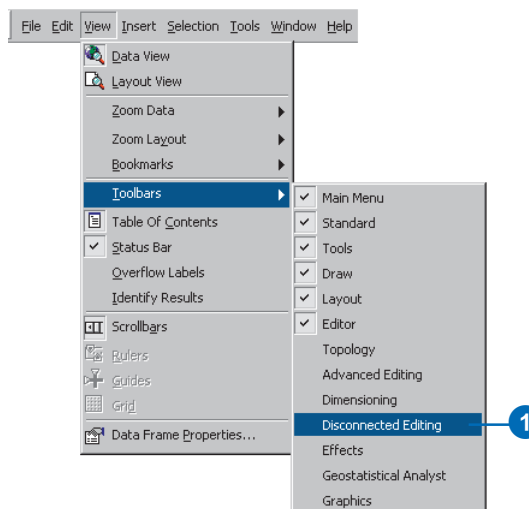
Tip

Copying only selected features

You can copy only selected features from a dataset when using the Extract Data wizard. Use the Select Features tool in ArcMap and select the desired features. The selected features will be the only features in that feature class that are copied.

1. Click View, click Toolbars, then select Disconnected Editing in ArcMap on the main menu.
The Disconnected Editing toolbar will now appear on your *map document*.
2. Adjust your map extent using the zoom functions in ArcMap to encompass the features you would like to copy into another geodatabase.
3. Click the Extract Data command on the Disconnected Editing toolbar.

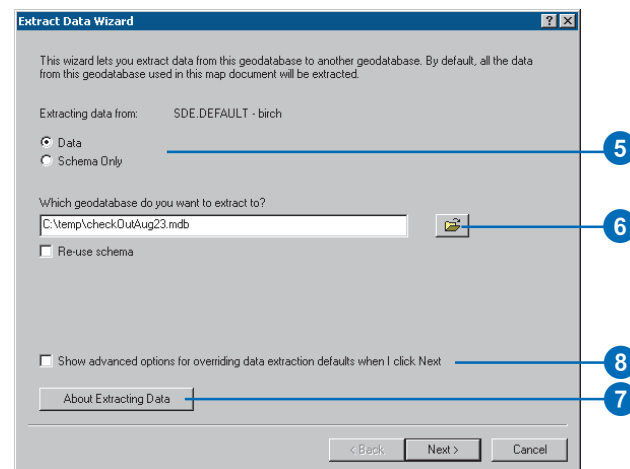
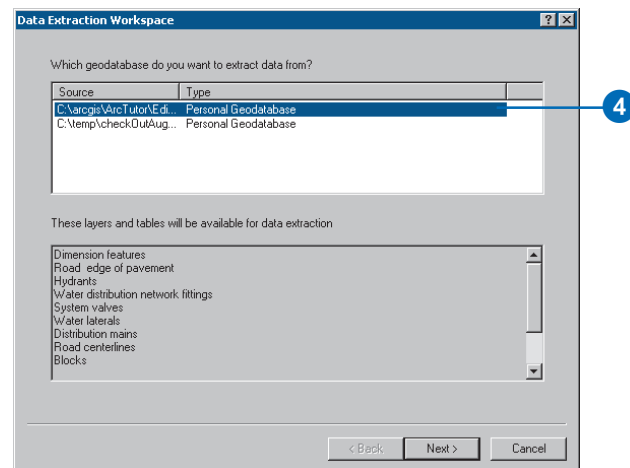
The Extract Data wizard will open. ►



See Also

For more information on geodatabase schemas, see the chapter 'Introduction' in this book.

4. Click the geodatabase that contains the data you wish to extract and click Next if data from more than one geodatabase is in the map extent.
5. Click the radio button to extract the data or the schema only.
6. Specify the destination geodatabase or type the path and name of a new geodatabase you wish to create.
7. Click About Extracting Data to read further details about the Extract Data wizard and a description of the default values.
8. Check the Show advanced options box and click Next to change the default values. To accept the default values, leave the check box empty, click Next, and proceed to step 11. ►



Tip

Using the Layers and Tables grid

The grid allows you to include or exclude specific layers and tables and to accept or override the current extent defaults. For example, if a layer has a selection set, you may use this selection or opt to ignore this and extract all the data for this layer or table.

Tip

Choosing a Post Data Extraction option

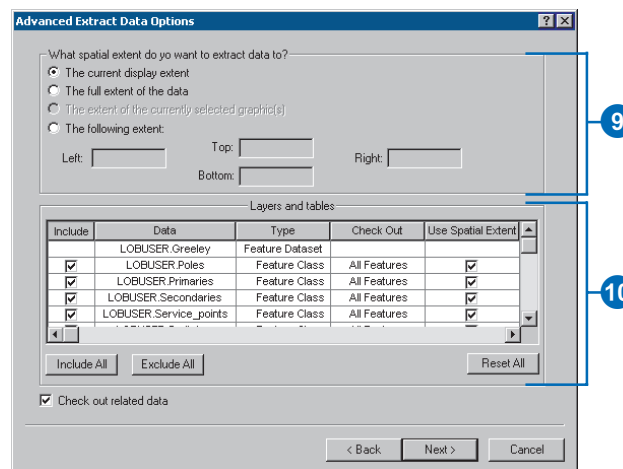
The Post Data Extraction Options dialog box gives three options. The first simply saves the data in the new geodatabase without linking the extracted data to any map document. The second option allows you to relink the current map document to the extracted data. The final option allows you to create a new map document using the extracted data.

9. Specify the spatial extent of the extracted data.
10. Modify the default settings of layers and tables as required.

You can exclude specific layers or tables, override selections, queries, and spatial extent settings.

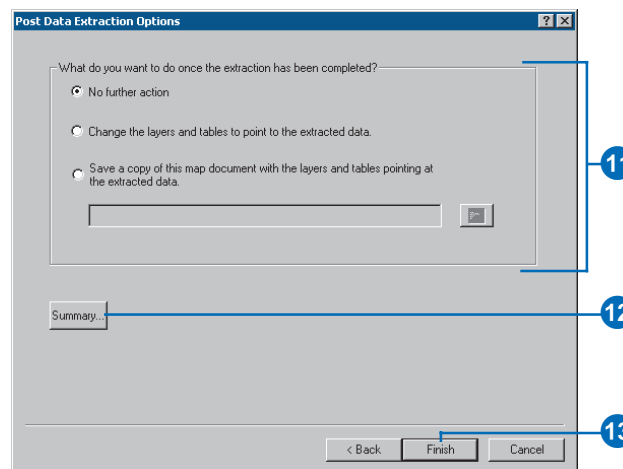
11. Select a post data extraction option.
12. Click the Summary button for an extract data summary. Click Back to make any desired changes.
13. Click Finish.

The data will be extracted to the designated geodatabase.



The Advanced Extract Data Options dialog box is shown. It has a title bar with a question mark and a close button. The main area is divided into two sections. The top section, labeled 'What spatial extent do you want to extract data to?', contains four radio buttons: 'The current display extent' (selected), 'The full extent of the data', 'The extent of the currently selected graphic(s)', and 'The following extent:'. Below these are input fields for 'Left', 'Top', 'Right', and 'Bottom'. The bottom section, labeled 'Layers and tables', contains a table with columns 'Include', 'Data', 'Type', 'Check Out', and 'Use Spatial Extent'. The table lists four items: 'LOBUSER.Greeley' (Feature Dataset), 'LOBUSER.Poles' (Feature Class), 'LOBUSER.Primaries' (Feature Class), and 'LOBUSER.Secondaries' (Feature Class). All 'Include' and 'Use Spatial Extent' checkboxes are checked. Below the table are buttons for 'Include All', 'Exclude All', and 'Reset All'. At the bottom of the dialog are buttons for '< Back', 'Next >', and 'Cancel'. A blue line with a circle containing the number 9 points to the spatial extent radio buttons. A blue line with a circle containing the number 10 points to the 'Layers and tables' table.

Include	Data	Type	Check Out	Use Spatial Extent
☒	LOBUSER.Greeley	Feature Dataset	All Features	☒
☒	LOBUSER.Poles	Feature Class	All Features	☒
☒	LOBUSER.Primaries	Feature Class	All Features	☒
☒	LOBUSER.Secondaries	Feature Class	All Features	☒



The Post Data Extraction Options dialog box is shown. It has a title bar with a question mark and a close button. The main area is divided into two sections. The top section, labeled 'What do you want to do once the extraction has been completed?', contains three radio buttons: 'No further action' (selected), 'Change the layers and tables to point to the extracted data.', and 'Save a copy of this map document with the layers and tables pointing at the extracted data.'. Below these is a text input field and a button with a right-pointing arrow. The bottom section contains a button labeled 'Summary...'. At the bottom of the dialog are buttons for '< Back', 'Finish', and 'Cancel'. A blue line with a circle containing the number 11 points to the radio buttons. A blue line with a circle containing the number 12 points to the 'Summary...' button. A blue line with a circle containing the number 13 points to the 'Finish' button.

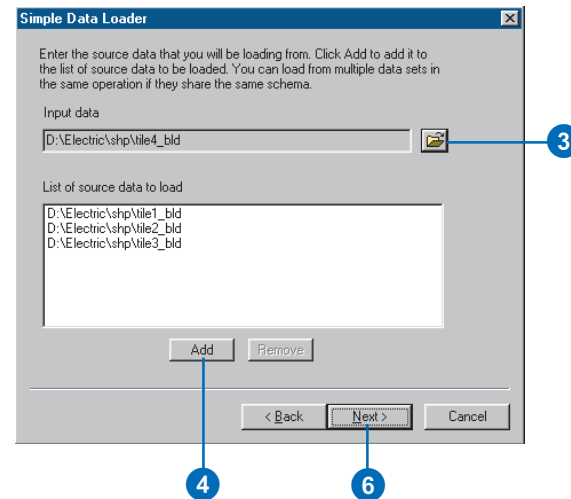
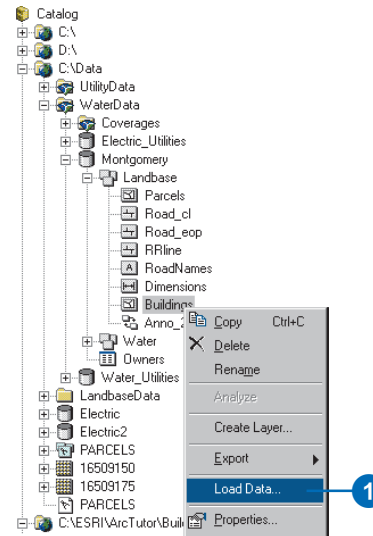
Loading data into existing simple feature classes and tables

The tools and wizards for importing coverages, shapefiles, and INFO and dBASE tables into the geodatabase require that each shapefile and coverage feature class be loaded into a new feature class and each INFO and dBASE table be loaded into a new table. The feature class or table cannot exist before you begin the import process.

Because an existing feature class or table can be in any number of states, a separate data-loading tool is required to load data from a shapefile, coverage feature class, INFO table, or dBASE table into an existing geodatabase feature class or table.

In the case of nonversioned simple data, an edit session is not required to insert new features or rows into the table or feature class. Once loaded, all data is visible in all versions of the database. This data loading operation is performed with the Simple Data Loader wizard in ArcCatalog.

1. Right-click the table or feature class in the ArcCatalog tree that you want to load data into and click Load Data.
2. Click Next on the introductory panel.
3. Browse to the input feature class or table.
4. Click Add to add the table of feature class to the list of source data.
5. Repeat steps 3 and 4 until you have specified all of the source data.
6. Click Next. ►



The wizard will allow you to specify a number of source tables and feature classes, provided their schema match. It also allows you to specify which fields in the input data are loaded into which fields of the target feature class or table.

The wizard also gives you the option to load all of the source data into a specific *subtype* of the target and lets you specify a query to limit the features loaded.

This wizard will only be available if the target stores simple, nonversioned data. For loading objects into nonsimple or versioned data using the Object Loader, see the chapter ‘Migrating existing data into a geodatabase’ in this book.

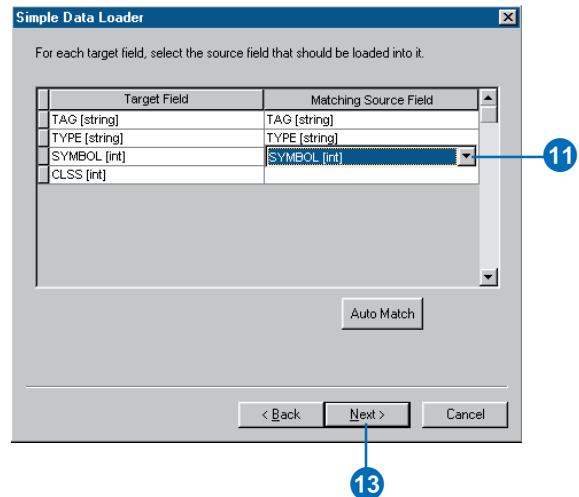
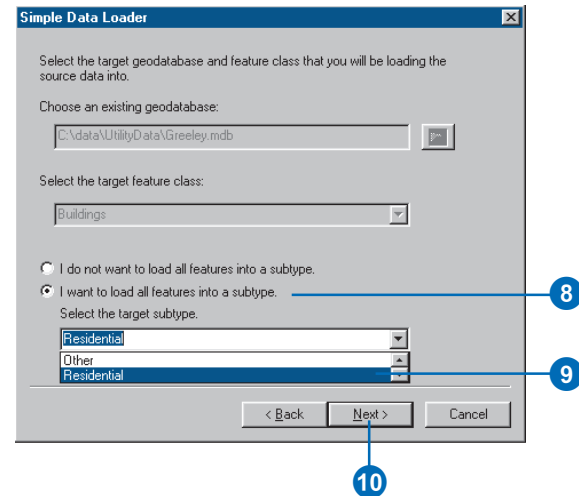
Tip

The subtype field

If you choose to load data into a specific subtype, you will not be able to match a field from the source data to the subtype field in the target data; the subtype field will be automatically populated.

7. Click the first option and skip to step 10 if you do not want to load data into a specific subtype of the target.
8. Click the second option if you want to load data into a specific subtype.
9. Click the dropdown arrow and click the subtype into which you want to load the source data.
10. Click Next.
11. Click the dropdown arrow in the Matching Source Field list and click the field from the source data that you want to match to the target field to match a source field with a field in the target feature class or table.

If you do not want data from a field in the source data to be loaded into the target data, leave the Matching Source Field blank.
12. Repeat step 11 until you have matched all the fields you want to load from your source data to the target fields.
13. Click Next. ►



Tip

Source data

When matching fields, you can browse the source data's field values to help you correctly match the source and target fields.

Tip

Relationships

If the feature class or table you want to load data into participates in a relationship class with messaging (such as a composite relationship class), the data is considered nonsimple and the Load Data command will be unavailable.

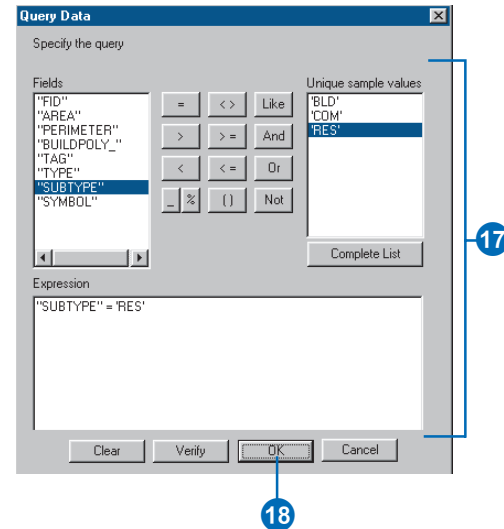
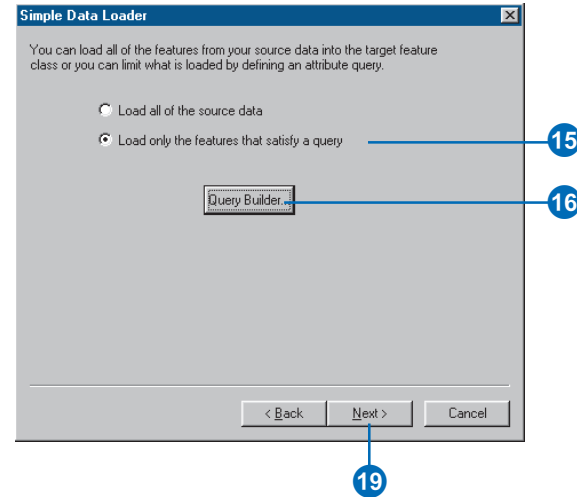
To load data into these feature classes and tables, you can either delete the relationship, or use the Object Loader.

To learn more about relationship classes and the Object Loader, see the chapter 'Defining relationship classes' in this book and Editing in ArcMap.

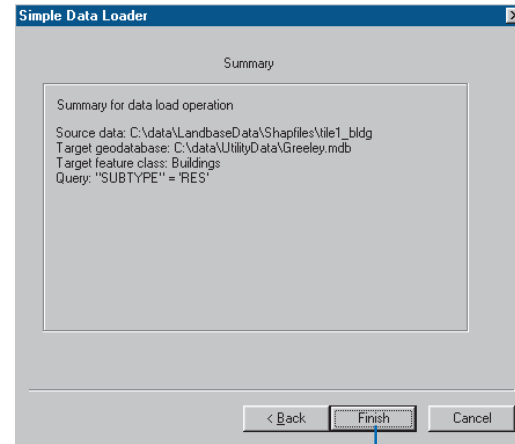
See Also

To learn more about using the Query Builder to query your data, see Using ArcMap.

14. Click the first option and skip to step 19 if you want to load all of the source data.
15. Click the second option if you want to limit the features from the source data loaded into the target using an attribute query.
16. Click Query Builder to open the query builder dialog box.
17. Use the query builder to create a query to limit the features or rows from the source data that are going to be loaded into the target.
18. Click OK.
19. Click Next. ►



20. Review the options you have specified for loading your data. If you want to change something, you can go back through the wizard by clicking Back.
21. Click Finish to load your data when satisfied with your options.



Registering ArcSDE layers and tables with the geodatabase

ArcSDE 8 layers and tables that were created with applications that were not written using the ArcInfo 8 COM components are not registered with the geodatabase system tables. These ArcSDE layers and tables still appear in the ArcCatalog tree as feature classes and tables; however, they cannot have subtypes, default values, or domains or participate in relationships and geometric networks.

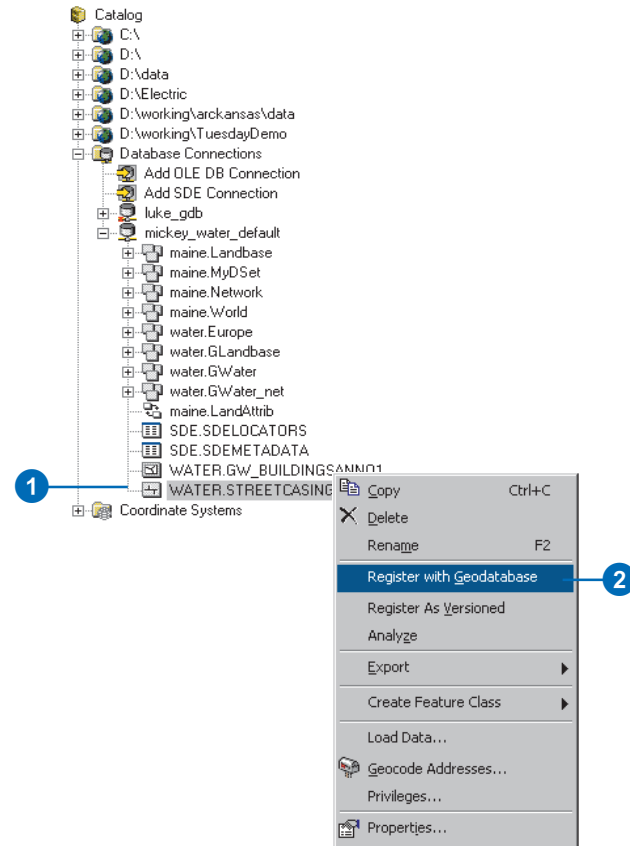
You can use ArcCatalog to register these ArcSDE layers and tables with the geodatabase. This operation will promote ArcSDE layers to geodatabase feature classes and ArcSDE tables to geodatabase object classes.

Tip

ObjectID column

If the ArcSDE layer or table is not already registered with the ArcSDE table registry, then this operation will register it and add an ObjectID field to the table. This field will be called OID for tables and FID for layers. If a field called OID or FID already exists on the table or layer, then another name is chosen.

1. Right-click the table or feature class in the ArcCatalog tree that you want to register with the geodatabase.
2. Click Register with Geodatabase.



Analyzing geodatabase data

If your geodatabase is stored in a DBMS, such as Oracle, SQL Server, DB2, or Informix, the Analyze command in ArcCatalog can be used to update the *RDBMS* statistics on your datasets. The Analyze command updates the statistics of business tables, feature tables, and delta tables along with the statistics on those tables' indexes.

Tip

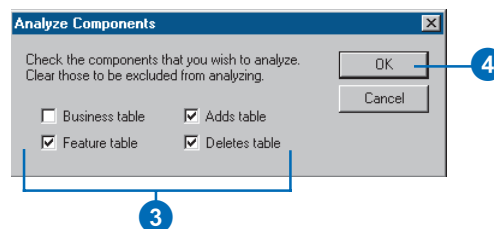
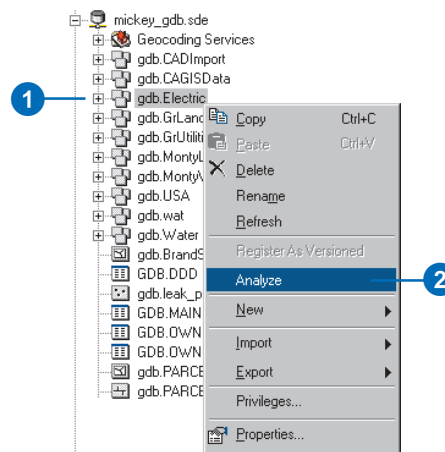
Analyzing feature datasets

When you analyze a feature dataset, all of the feature classes contained in that feature dataset are analyzed. If the feature dataset contains a geometric network, then the network tables are also analyzed.

See Also

For more information about which of your feature classes' tables should be analyzed and when, see the chapter 'Working with a versioned geodatabase' in this book.

1. Right-click the dataset in the ArcCatalog tree that you want to Analyze. This can be a feature dataset, feature class, or table.
2. Click Analyze.
3. Check those tables you want analyzed.
4. Click OK.



Loading objects from other feature classes

This chapter describes a number of tools for converting CAD, shapefile and coverage features, and INFO and dBASE tables to your geodatabase. These tools and wizards require that each shapefile and coverage feature class be loaded into a new feature class and each INFO and dBASE table be loaded into a new table. The feature class or table cannot exist before you begin the import process. The Simple Data Loader, as described in this chapter, can be used to load data into existing simple, nonversioned feature classes and tables.

Feature classes and tables that are versioned, or that store custom or network features, require an edit session to insert new records into the table or feature class. This ensures that the network connectivity and version information is managed correctly. The Object Loader wizard in ArcMap lets you do this. When the feature dataset that the data is being loaded to has a topology, the loaded data acts as raw data and is not involved with the topology. If you are adding the data to a feature class that already participates in a topology, the addition is considered an edit and validating the topology will need to be done independently of the Object Loader wizard.

The Load Objects command opens the Object Loader wizard. Before you use this tool, there are several things to consider. First, loading data into network feature classes is a slow process—several seconds per feature. This performance hit may make large data loads into network feature classes impractical.

If you use the Object Loader wizard to load data into versioned ArcSDE feature classes, once all of the data is loaded it will be in the delta tables, not the base tables for the feature classes. If you use this method, you should run Compress on your database once the data is loaded to push all the records from the delta tables to the base tables. Having your data in the base tables will result in better query speed than if you have large amounts of data in your delta tables. For more details on compressing your

database to improve performance, see the chapter ‘Working with a versioned geodatabase’ in this book.

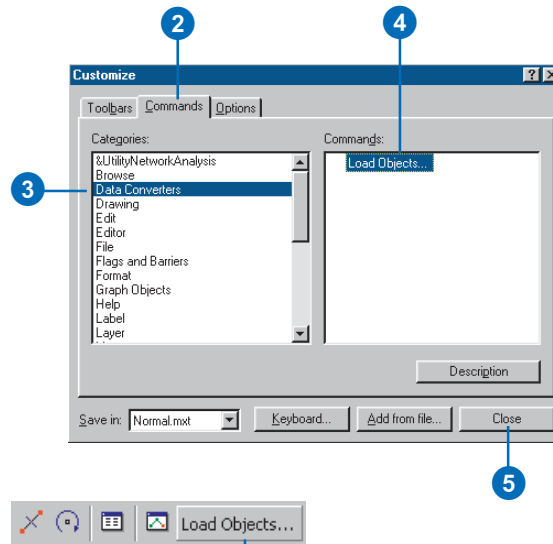
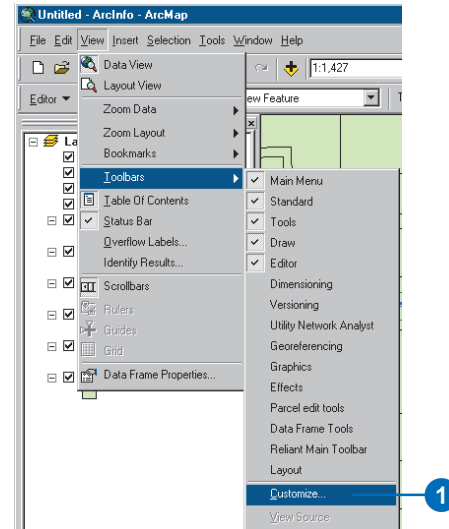
If you need to append large amounts of data to your database but want to avoid the performance hit associated with loading data into network feature classes or versioned feature classes, you should consider one of the advanced data appending strategies outlined in this chapter.

Adding the Load Objects command to ArcMap

1. Click View, point to Toolbars, and click Customize.
2. Click the Commands tab.
3. Click Data Converters.
4. Click and drag the Load Objects command from the Commands list and drop it on the *Editor toolbar*.

The command appears on the toolbar.

5. Click Close.



The command appears on the toolbar.

Tip

The subtype field

If you choose to load data into a specific subtype, you will not be able to match a field from the source data to the subtype field in the target data. The subtype field in this case is automatically populated.

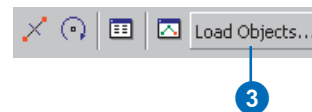
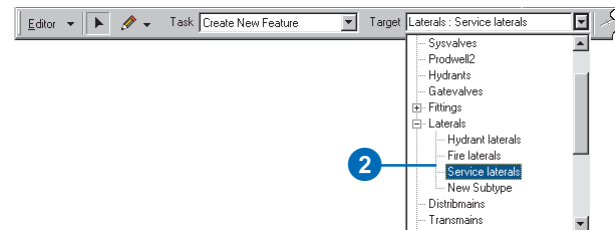
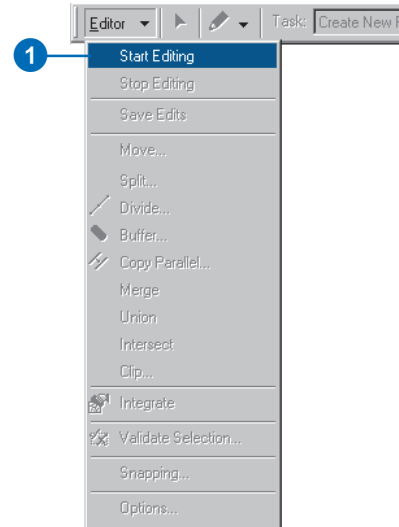
Tip

Versioned data

When you load data into a versioned feature class, the new features are only visible in the version you are working with.

Loading data with the Load Objects command

1. Add your data to ArcMap, click Editor, then click Start Editing.
2. Click the *Target layer* dropdown arrow and click the feature class or subtype into which you want to load data.
3. Click Load Objects. ►



Tip

Source data

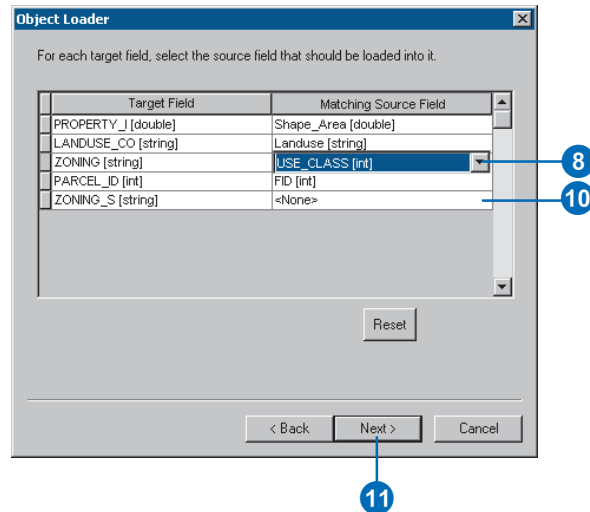
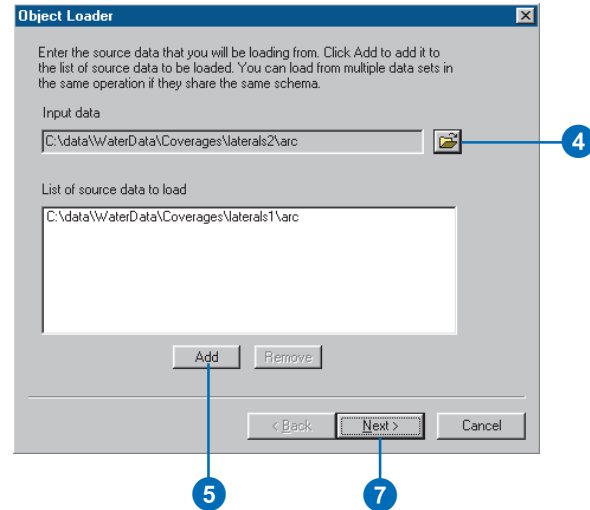
When matching fields, you can browse the source data's field values to help you correctly match the source and target fields.

Tip

Resetting the Matching Source Field

You can return all of the matching fields to <none> by clicking Reset under the table.

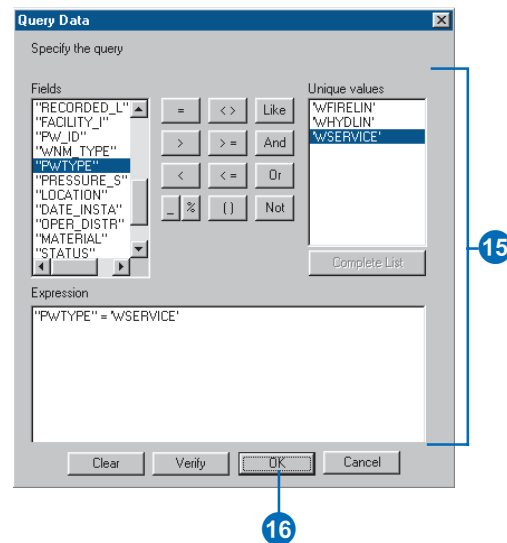
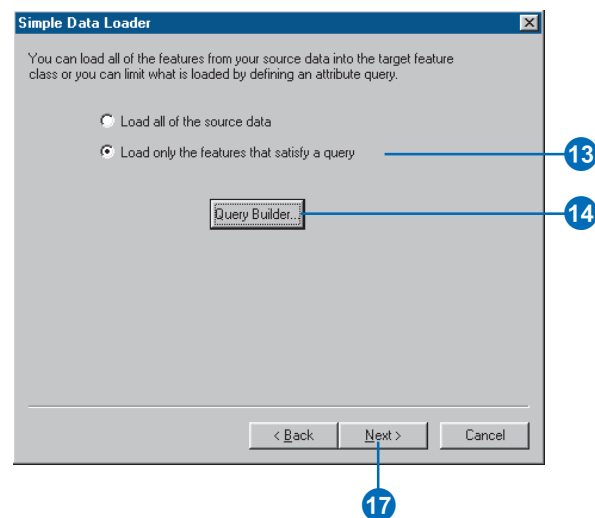
4. Browse to the source feature class.
5. Click Add to add it to the list of source data.
6. Repeat steps 4 and 5 until you have specified all of the source data.
7. Click Next.
8. Click the dropdown arrow in the Matching Source Field list and click the field from the source data you want to match to the target field to match a source field with a field in the target feature class or table.
9. Repeat step 8 until you have matched the fields you want loaded from your source data to the target fields.
10. Leave the Matching Source Field as <None> if you don't want data from a field in the source data to be loaded into the target data.
11. Click Next. ►



See Also

To learn more about using the *Query Builder* to query your data, see *Using ArcMap*.

12. Click the first option and skip to step 16 if you want to load all of the source data.
13. Click the second option if you want to limit the features from the source data to load into the target using an attribute *query*.
14. Click Query Builder to open the Query Data dialog box.
15. Create a query to limit the features or rows from the source data to be loaded into the target.
16. Click OK.
17. Click Next. ►



Tip

Network features

When loading data into an edge network feature class, the network connectivity is maintained, and default junctions are used as described earlier in this chapter.

Tip

Feature class validation

The option to validate the feature loaded applies only to the geodatabase validation rules and does not validate the topology. For more information on topologies, see the chapter 'Topology' in this book.

See Also

For more information on the ArcMap snapping environment, see Editing in ArcMap.

18. Click No if you don't want your features to be snapped to existing features in your edit session.

Click Yes if you want to use the current Editor *snapping environment* to snap the new features as they are loaded.

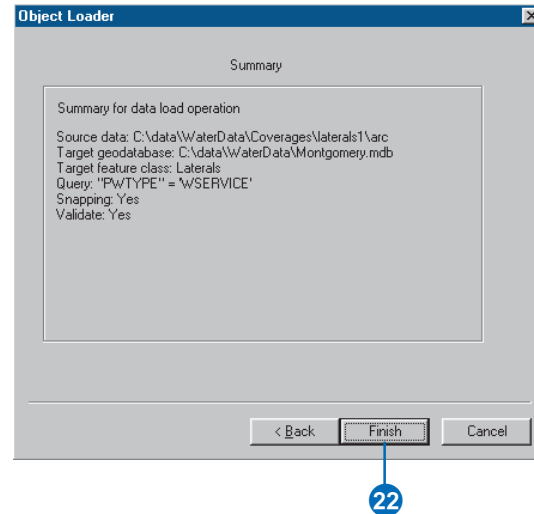
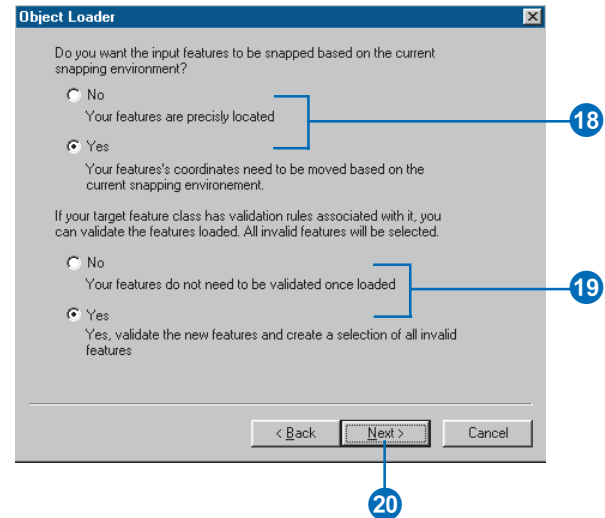
19. Click No if you don't want your new features to be validated after they are loaded.

Click Yes if the feature class or subtype into which you are loading data has rules associated with it and you want any new invalid feature to be selected after the loading process.

20. Click Next.

21. Review the options you have specified for loading your data. If you want to change something, go back through the wizard by clicking Back.

22. Click Finish to load your data when satisfied with your options.



Topology

4

IN THIS CHAPTER

- What is topology?
- Creating a topology
- Topology and feature geometry
- Adding feature classes to your topology
- Topology: defining the rules
- Refining topologies with subtypes
- Topology validation, errors, and exceptions
- Managing and modifying a topology
- Creating polygons from lines
- Topology and versioning
- Topology and disconnected editing

When you model geographic features, you may find that you want to model some features that have spatial relationships with other features around them. Countries might be modeled such that adjacent countries meet without gaps along a common border line but never overlap. States or provinces could be modeled such that they fall exclusively within one country. Streets may be modeled such that two streets always meet at an *intersection* and never share a *segment*. Bus stops may be modeled such that they must always occur on a street. These relationships are maintained in the geodatabase through an association called a topology.

In GIS technology, topology is the model used to describe how features share geometry, and it is also the mechanism for establishing and maintaining topological relationships between features. ArcGIS implements topology through a set of validation rules that define how features may share a geographic space and a set of editing tools that work with features that share geometry in an integrated fashion.

A topology is stored in a geodatabase as one or more relationships that define how the features in one or more feature classes share geometry. The features participating in a topology are still simple feature classes—rather than modifying the definition of the feature class, a topology serves as a description of how the features can be spatially related. ArcInfo and ArcEditor provide tools for creating, evaluating, and managing the quality of

those topological relationships. This chapter describes the physical geodatabase data model for topology, how you create and configure a topology, and the editing tools in ArcGIS that help you to maintain the quality of the *integrated features* in a topology. Another type of *topological association*, the geometric network, is discussed in the chapter ‘Geometric networks’ in this book.

You can create simple temporary topological relationships between features in ArcView. Creating or editing geodatabase topology requires an ArcEditor or ArcInfo license.

What is topology?

Topology has historically been viewed as a spatial data structure that is used primarily to ensure that the associated data forms a consistent and clean topological fabric. With advances in object-oriented GIS development, an alternative view of topology has evolved. The geodatabase supports an approach to modeling geography that integrates the behavior of different feature types and supports different types of key relationships. In this context, topology is a collection of rules and relationships that, coupled with a set of editing tools and techniques, enables the geodatabase to more accurately model geometric relationships found in the world.

Topology, implemented as feature behavior and rules, allows a more flexible set of geometric relationships to be modeled than topology implemented as a data structure. It also allows topological relationships to exist between more discrete types of features within a feature dataset. In this alternative view, topology may still be employed to ensure that the data forms a clean and consistent topological fabric, but also more broadly, it is used to ensure that the features obey the key geometric rules defined for their role in the database.

Why use topology?

Topology is used most fundamentally to ensure data quality and to allow your geodatabase to more realistically represent geographic features. A geodatabase provides a framework within which features can have behavior, such as subtypes, default values, attribute domains, validation rules, and structured relationships, to tables or other features. This behavior enables you to more accurately model the world and maintain referential integrity between objects in the geodatabase. Topology may be considered an extension of this framework for behavior that allows you to control the geometric relationships between features and to maintain their geometric integrity. Unlike other

feature behavior, topology rules are managed at the level of the topology and dataset, not for individual feature classes.

How do I work with topology?

Different people work with topology in different ways, depending on their role in an organization and its GIS design and management work flow.

Initially, creating a topology requires a geodatabase designer. A topology organizes the spatial relationships between features in a set of feature classes. The designer analyzes an organization's data modeling needs, identifies the key topological relationships required in the geodatabase, and defines the rules that will constrain different features' topological relationships.

Once the participating feature classes have been added to the topology and the rules defined, the topology is validated. Data quality managers use the topology tools to analyze, visualize, report, and where necessary, repair the spatial integrity of the database after it is initially created as well as after editing. Topology provides these users with a set of validation rules for the topologically related features. It also provides a set of editing tools that lets users find and fix integrity violations.

As the geodatabase is used and maintained, new features are added and existing features are modified. Data editors update features in the geodatabase and use the topology tools to construct and maintain relationships between features within the *constraints* imposed by the database designer. Depending on the work flow of the organization, the topology may be validated after each edit session or on a schedule.

Creating a topology

A topology consists of a set of rules structuring the relationship between a collection of features in one or more feature classes in a feature dataset.

To create a topology, you must specify which feature classes will participate in the topology and what rules will govern the interaction of features. All feature classes participating in a topology must be in the same feature dataset.

Because creating topological relationships involves *snapping* feature vertices together to make them *coincident*, a *cluster tolerance* must be specified for the topology. Vertices within the cluster tolerance may move slightly in the snapping process. The default cluster tolerance is the minimum possible cluster tolerance and is based on the precision defined for the dataset. The cluster tolerance should be very small, so only very close vertices are snapped together. A typical cluster tolerance is at least an order of magnitude smaller than the accuracy of your data. For example, if your features are accurate to 2 meters, your cluster tolerance should be no more than 0.2 meters.

Often, you will want to be able to control which feature classes are more likely to be moved in the *clustering* process. For example, when features in one feature class are known to have more reliable positions than another set of features, you may want the less reliable features to snap to the more reliable ones. *Ranks* are assigned to the feature classes in the topology to accommodate this common situation. Vertices of lower ranking features within the cluster tolerance will be snapped to nearby vertices of higher ranking features. Vertices of features of equal rank that lie within the cluster tolerance will be geometrically averaged together.

Building a topology

You may already have data from which you want to create a topology in your geodatabase. ArcCatalog contains tools to create a topology from that data.

The process of building a topology from existing data can be summarized in the following steps:

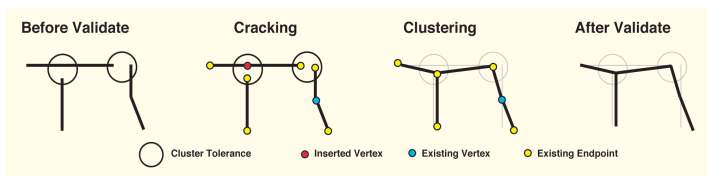
1. Use ArcCatalog or ArcToolbox tools to convert and load your data into a feature dataset in a geodatabase.
2. Use the ArcCatalog Topology wizard to build a topology from existing simple feature classes. In the wizard, you do steps 3–9.
3. Name the topology.
4. Set a *cluster tolerance* for the topology.
5. Choose the feature classes that will participate in the topology.
6. Choose the number of ranks to use in the topology.
7. Rank the feature classes in the topology.
8. Add topology rules to structure the spatial relationships between features.
9. Create the topology.
10. Use ArcCatalog or ArcMap to validate the topology.
11. Use the ArcMap Topology Error Inspector to identify topology errors.
12. Use ArcMap to fix topology errors or mark them as exceptions.

How topologies are built

Building topologies from existing features is a computationally intense operation that may take a considerable amount of time and system resources, depending on the number of input features.

If those features require snapping, the validating operation will spend most of its time in the clustering (feature snapping) phase. The validating process proceeds in the following sequence:

1. *Cracking* features
2. Clustering vertices



Validating topology passes through two stages: cracking and clustering. In the cracking stage, vertices are created along edges that fall within the cluster tolerance of an existing edge, vertex, or endpoint. In the clustering stage, the endpoints and vertices that fall within the cluster tolerance are snapped together.

When a vertex of one feature in the topology is within the cluster tolerance of an *edge* of any other feature in the topology, the topology engine creates a new vertex on the edge to allow the features to be geometrically integrated in the clustering process.

When clustering, or snapping, features during topology validating, it is important to understand how the geometry of

features is adjusted. All vertices of any feature in a feature class that participates in a topology can potentially be moved, if they fall within the cluster tolerance of another vertex. Vertices of higher ranking features will not move to lower ranking features, but vertices of equally ranked features will be geometrically averaged.

Schema locking

An exclusive lock is required on all of the input feature classes when building a topology. If any of the input feature classes has a shared lock, the topology will not be built.

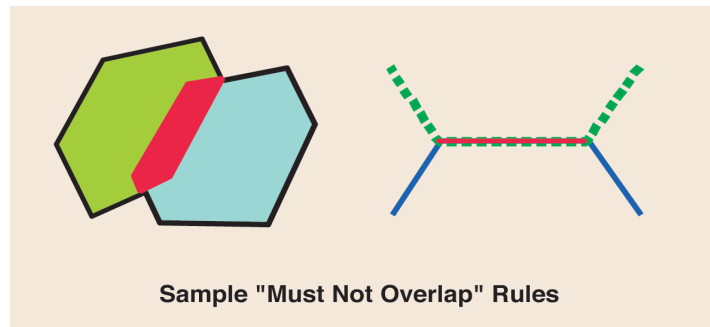
If any of the feature classes in a topology have a shared or exclusive lock, that lock is propagated to all of the other feature classes in the topology. For more information on exclusive locks and schema locking, see the ‘Creating new items in a geodatabase’ chapter in this book.

Topology basics

Topologies store several sets of parameters—rules, ranks, and a cluster tolerance. They also maintain internal feature layers that contain dirty areas, errors, and exceptions. These parameters and special features in a topology are discussed in more detail in the next few sections.

Rules

The rules you define for a topology control the allowable relationships of features within a feature class, between features in different feature classes, or between subtypes of features.

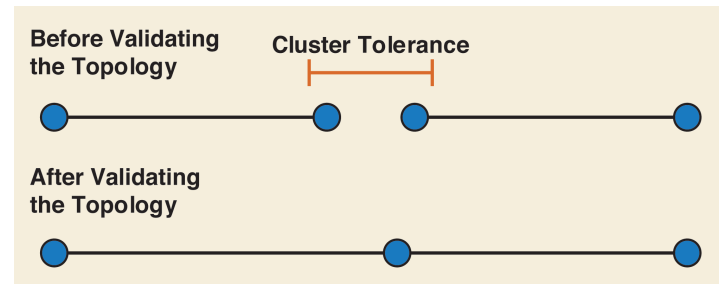


Example of a "Must not overlap" rule applied to polygons and lines. The red polygon and line mark the places where the rule is violated. These are stored in the topology as error features. Such rules can apply to features within the same feature class, to pairs of feature classes, or to subtypes of features.

The initial validation of the topology checks all of the features against all of the rules. This initial check can take some time, but subsequent checks are performed only on the areas that have been edited—the dirty areas.

Cluster tolerance

The cluster tolerance is the minimum distance between vertices of features that are not coincident. Vertices that fall within the cluster tolerance are defined as coincident and snapped together.

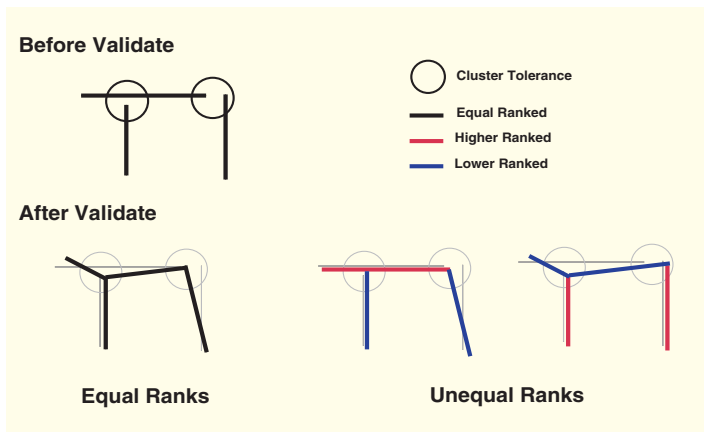


When you validate a topology, features within the cluster tolerance are snapped together.

The cluster tolerance is typically a very small actual distance, to minimize the movement of correctly placed features. The default cluster tolerance is the smallest cluster tolerance possible for a dataset and is calculated based on the dataset's precision and extent. The precision describes the number of system units per one unit of measure in the dataset; so it defines the smallest storable distance between coordinates in the dataset. A spatial reference with a precision of 1 will store integer values, while a precision of 1,000 will store three decimal places. The extent defines the maximum geographic extent that can be stored in the dataset.

Ranks

The ranks you specify for feature classes in the topology control the feature classes, which will be moved when snapping coincident vertices during the initial validation of the topology and during subsequent validations. When different feature classes have different levels of intrinsic reliability, such as when one was collected by survey or differential global positioning system (GPS) and another was digitized from less accurate source material or collected with uncorrected GPS, ranks can allow you to ensure that reliably placed vertices are not snapped to the location of less reliable ones. Lower ranked features' vertices will be snapped to the location of higher ranked vertices if they fall within the cluster tolerance. The location of equally ranked vertices are geometrically averaged when they are within the cluster tolerance.



When you validate a topology, the ranks of the feature classes in the topology control how features are snapped together. Lower ranking features snap to higher ranking features. Equally ranked features snap to the geometric average of their position.

Feature layers maintained by a topology

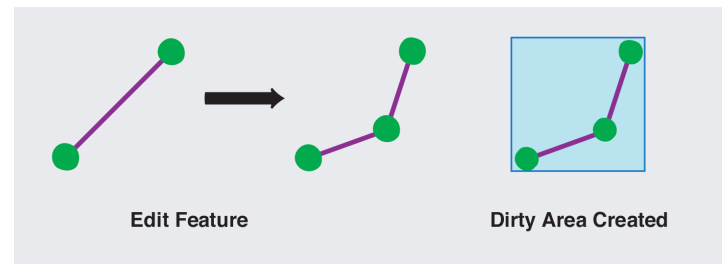
Instead of storing topological information with the feature classes, the topology discovers those relationships when the information is requested such as when you are editing using the shared geometry tool. In order to help you manage the process of creating and editing a logically consistent topology, the topology internally stores two additional types of feature classes: dirty areas and error features.

Dirty areas

Dirty areas let the topology efficiently track the places where topology rules may have been violated during editing. The dirty areas allow selected parts, rather than the whole extent of the topology, to be validated after editing.

Dirty areas are created when:

- A feature is created or deleted
- A feature's geometry is modified
- A feature's subtype is changed
- Versions are reconciled
- The topology properties are modified

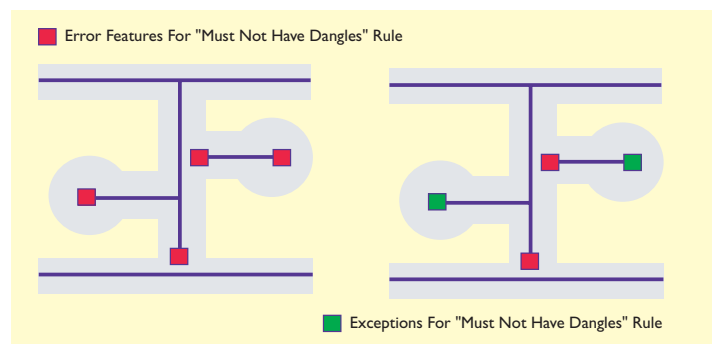


When you edit features in a topology, the topology creates a dirty area to mark the area that should be checked for violations of the topology rules.

Dirty areas are stored in the topology as a single feature, with each new dirty area united with the existing dirty area and each area that has been validated removed from the dirty area.

Errors and exceptions

Topologies also store error features, which record where topological errors were discovered during validation. Certain errors may be acceptable, in which case the error features can be marked as exceptions.



When you validate a topology, features that violate the rules are marked as error features. You can edit the features to fix the errors, or you can mark the errors as exceptions. In this example, the street line features cannot have dangles, which are endpoints that do not connect to other street features. Because cul-de-sac streets are a legitimate exception to this rule, they may be marked as exceptions in the topology. The remaining errors should be fixed by editing the street features.

ArcMap and ArcCatalog allow you to create a report of the total number of errors and exceptions for the feature classes in your topology. You can use the report of the number of error features

as a measure of the data quality of a topological dataset. In addition, the error inspector in ArcMap lets you select different types of errors and zoom to individual errors. You can correct topology errors by editing the features that violate the topology's rules. After you validate the edits, the error is deleted from the topology.

Topology review

Rules define the permissible relationships between features. Ranks control which features may be moved to other features when snapping the topology together in the initial validation and during subsequent validations of the topology. The cluster tolerance defines how close vertices must be to each other in order to be considered coincident and limits the distance features can move during validation. Dirty areas are areas that have been edited or affected by the addition, update, or deletion of features. Dirty areas allow the topology to limit the area that must be checked for topology errors during topology validation. Errors and exceptions are stored as features in the topology and allow you to render and manage the places where features do not obey the topology rules you specified.

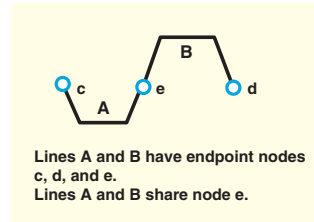
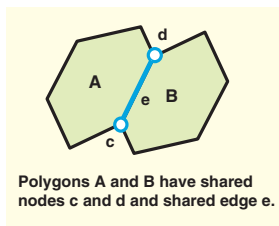
Topology and feature geometry

Geometries involved in a topology

The features participating in a topology belong to simple feature classes in the same dataset. Rather than modifying the definition of the feature class, a topology serves as a description of how the features in a feature dataset can be spatially related. Annotation, dimension, and geometric network features are not simple features and cannot participate in a topology. Feature classes outside of the topology's feature dataset cannot participate in the topology, and feature classes cannot participate in more than one topology at a time.

At the geometry level, topologies are about simple relationships such as coincidence, covering, and crossing between the geometric primitives that make up features. While all simple feature class geometries (point, line, polygon) may participate in topologies, internally, the types of geometry that are acted on when editing a topology are:

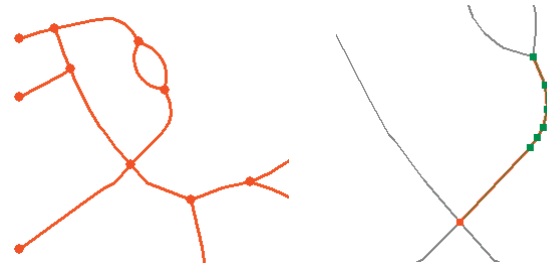
- Edges—Line segments that define lines or polygons.
- Nodes—Points at the end of an edge.
- *Pseudonodes*—A node connecting only two edges or a logical split defined in the topology cache while editing. Pseudonodes of the latter sort become a vertex after editing.



Ways of sharing geometry

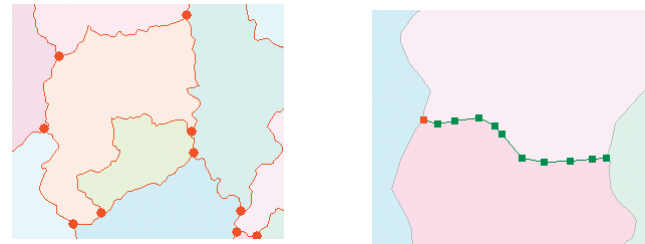
Features can share geometry within a topology in a number of ways:

- Line features can share endpoints (*arc-node topology*).



Line features can share edges and nodes, in red. Vertices define the shape of the edges, in green.

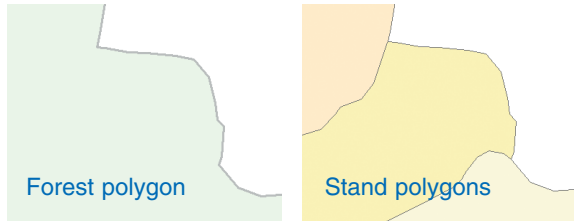
- Area features can share boundaries (polygon topology).



Polygon features share edges and nodes, in red. Vertices define the shape of the edges, in green.

- Line features can share segments with other line features (route topology).

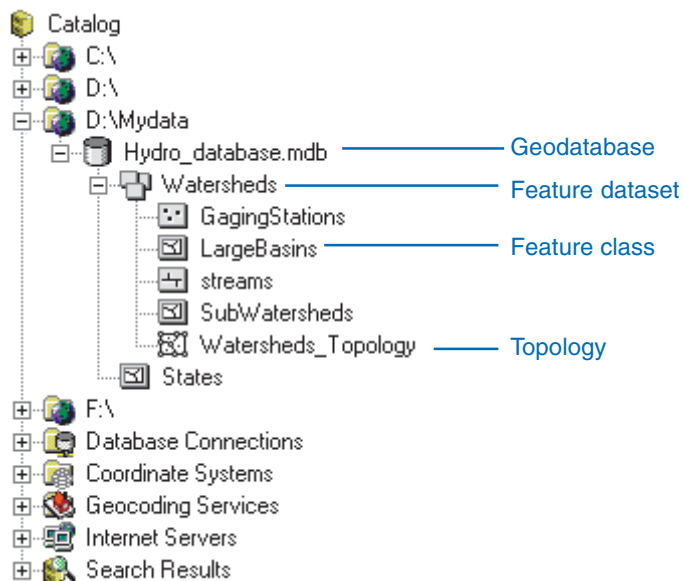
- Area features can be coincident with other area features (region topology).



- Line features can share endpoint vertices with other point features (node topology).
- Point features can be coincident with line features (point events).

Topologies and ArcCatalog

In ArcCatalog, you can view and manage topologies in geodatabases. Because all topologies must be inside a feature dataset, they appear in the ArcCatalog tree under their feature dataset.



It is not immediately evident in the ArcCatalog tree which feature classes in the dataset participate in the topology, which feature classes participate in which topology (if there is more than one topology in the dataset), and which participate in none. However, by examining the properties of a topology you can identify its constituent feature classes.

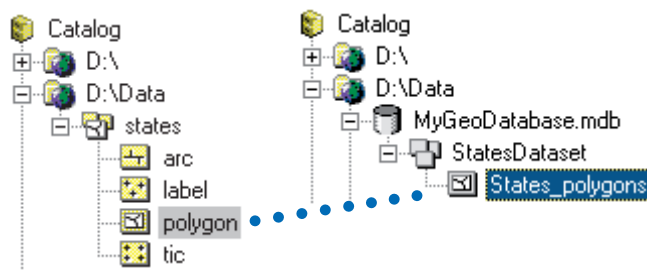
ArcCatalog also contains various tools to create, delete, and manage both topologies and the feature classes that participate in topologies. These tools are discussed in more detail later in this chapter.

Once you have created a topology using the Topology wizard in ArcCatalog, you can validate it in ArcMap or ArcCatalog. Validating in ArcCatalog is typically faster.

Migrating data into a geodatabase to create topologies

Before you create a topology, you should look at the data that you will be using to see what feature classes and topological rules you will need. In some cases, you will be migrating non-topological data, such as shapefiles, into a geodatabase and creating new topological relationships. In other cases, you will be moving data from the coverage topological data model. The coverage data model allows, and can be used to enforce, certain topological relationships. Some of these relationships may be useful for your database design and should be re-created using topology rules. Others may not be needed, in which case you may choose not to re-create them.

Coverages may also contain extra feature classes that are needed due to the coverage data model but that you no longer need in a geodatabase. You may choose not to import these feature classes.



Polygon coverage with polygon and supporting feature classes, compared to personal geodatabase polygon feature class. You do not typically need to import all of the coverage feature classes into your geodatabase.

Similarly, coverages maintain topological information in a number of attribute fields. Since geodatabase topology is not stored with the features, this information may be redundant. You may choose to drop some of these attributes when loading coverage data into a geodatabase.

OBJECTID	Shape*	AREA	PERIMETER	STATES	STATES_ID	Shape_Area	Shape_Length	STATE_NAME
1	Polygon	1736223600	270761.75	2	34	1736223017201597	2870761.89159487	Washington
2	Polygon	144819120	59693.63671875	3	38	144819034.0177	59693.6256835659	Washington
3	Polygon	504790592	165265.225458305	4	39	504790695.051825	165265.225458305	Washington

Annotations below the table:

- Columns managed by the geodatabase: points to OBJECTID, Shape*, and STATE_NAME.
- Columns managed by coverage topology: points to AREA, PERIMETER, and STATES.
- User assigned ID attribute: points to STATES_ID.
- Columns managed by the geodatabase: points to Shape_Area and Shape_Length.
- User assigned attribute: points to STATE_NAME.

Attribute table for a polygon feature class imported from a coverage. Polygon coverage feature classes have AREA, PERIMETER, and <COVER># fields (circled in red, above) that are managed by the ArcInfo coverage topology model. These are not updated by the geodatabase topology tools. You do not typically need to import these fields into your geodatabase. Certain attributes that are managed by the geodatabase are added to the attribute table during the import process. Shape stores the geometry, while Shape_Area and Shape_Length store these attributes of the geometry.

Migrating point features to a geodatabase

Point feature classes from shapefiles or coverages can migrate into point feature classes in a geodatabase. In a coverage, *label point* features may be related to an arc feature class to form polygons. The polygons' attributes are stored with the label points. In the geodatabase, the attributes of polygon features are stored with the polygons, so label points are unnecessary. You can use point features as an input feature class to supply attributes for new polygon features when creating polygons from line features. The attributes of the points are copied to the polygons; the points do not need to be stored with the polygon features in the geodatabase.

Geometry

Point features are not constructed from other feature classes in the coverage data model, so there are no supporting feature classes to keep or drop when importing coverage point features to a geodatabase. The Area and Perimeter items may be dropped, as they are used to manage polygon coverage topology.

Attributes

Node feature classes in an arc coverage may carry attributes and can be migrated to point feature classes in the geodatabase. The ARC# item in a node attribute table may be redundant, as it is primarily used to manage coverage arc–node topology.

Topology

In a geodatabase topology, point features can be topologically related to line features, the endpoints of line features, and polygon features. Points can be constrained to fall along lines, at the endpoints of lines, within polygons, or on the edges of polygons. See the ‘Topology: defining the rules’ section of this chapter for a more detailed discussion of the main topological relationships supported by the geodatabase for point features.

Migrating linear features to a geodatabase

Line features are represented by arcs in a coverage and by polylines in a shapefile. Arc and polyline feature classes migrate into line feature classes in a geodatabase.

Geometry

Arcs may be topologically related to nodes, polygons, or other arcs (routes) in the coverage data model. When migrating arc coverages into a geodatabase, you may want to import the node features as point features if you use the nodes to store attribute

values. If you do not store attributes with nodes, you can choose not to migrate the node features. When migrating route features, you can choose whether or not to migrate the supporting arc feature class to another line feature class.

Attributes

Coverage arc feature classes have a Length field that is superseded by the geodatabase’s Shape_length field. The geodatabase will not update this field, so it may be deleted when you load the feature class. Coverage arc feature classes also have FNODE#, TNODE#, LPOLY#, RPOLY#, and <cover name># fields that are not managed by geodatabase topology. You can drop these fields when you import the coverage.

Topology

Polyline features in shapefiles do not have topology rules or topological relationships to other features.

Coverage arc feature classes have built-in topology rules that specify that they must split at a node when they cross other arcs. Arcs have an additional topology rule that they must connect to more than one other arc at a node. Exceptions to this rule are called dangles—where an end of an arc fails to connect to another arc—and pseudonodes—where an arc connects to itself or one other arc.

In a geodatabase topology, line features can be topologically related to point features, other line features, and polygon features. Lines can be constrained to fall along other lines, to connect to other lines at their endpoints, or not to touch or intersect themselves or other lines. They can be constrained not to have dangles or pseudonodes. They can outline the edges of polygons or not touch polygons. See the ‘Topology: defining the rules’ section of this chapter for a more detailed discussion of the

main topological relationships supported by the geodatabase for line features.

Migrating area features to a geodatabase

Area features are represented by polygon feature classes in coverages and in shapefiles. They may also be represented by region feature classes in a coverage. Coverage polygon topology has certain inherent rules, requiring coverage polygon edges to be defined by the arcs' feature class, to completely tile the extent of the coverage, and not to overlap other polygons. Coverage polygons may also be topologically related to a label point feature class.

Shapefile polygons do not have any topology rules and are not topologically related to other feature classes.

Geometry

Coverage polygon feature classes are topologically related to two supporting feature classes, the label and arc feature classes. They may also be topologically related to other polygons to form regions, which are coverage area features that can overlap and which do not have to tile the whole extent of the coverage. Regions are somewhat like shapefile polygons. They migrate to polygon features in a geodatabase. You may choose not to migrate the supporting polygon feature classes if the regions store the attributes you use.

Attributes

Coverage polygon feature classes have Perimeter and Area fields that are superseded by the geodatabase's Shape_length and Shape_area fields. Perimeter and Area fields can be dropped when polygon coverages are migrated to the geodatabase. The

supporting arc and node feature classes can also be dropped as geodatabase polygons do not depend upon these feature classes.

Topology

In a geodatabase topology, polygon features can be topologically related to point features, line features, and other polygon features. Points can be constrained to fall inside or on the edges of polygons. Lines can be constrained to fall along the edges of polygons, within polygons, outside of or not touching polygons, or not crossing the edges of polygons. Polygons can be constrained to not overlap or to be allowed to overlap. Polygons from one feature class may tile within the polygons of another polygon feature class, may exactly cover the other feature class, or may not overlap polygons of another feature class. See the 'Topology: defining the rules' section of this chapter for a more detailed discussion of the main topological relationships supported by the geodatabase for polygon features.

Creating a new topology

Topologies are created inside feature datasets. When you create a topology, you must specify which feature classes in the feature dataset participate in the topology and define the topological rules that they must obey.

New feature classes can be added to a topology at any time. Only simple feature classes can participate in topologies.

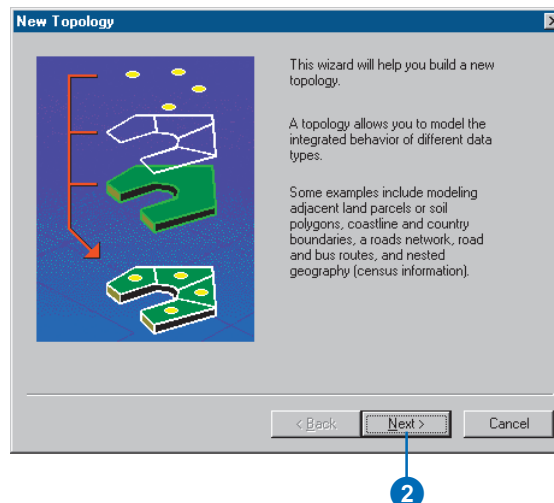
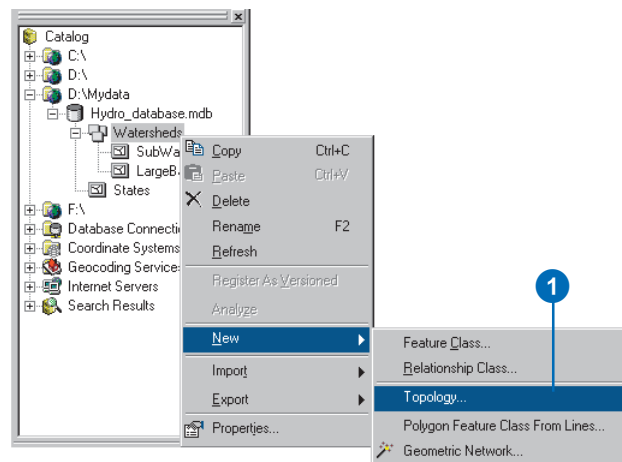
See Also

For more information on creating feature datasets and feature classes, see the chapter 'Creating new items in a geodatabase' in this book.

1. Right-click the feature dataset that will contain the topology, point to New, and click Topology.

The New Topology wizard starts.

2. Read the information on the first panel and click Next.



Tip

Cluster tolerance

The cluster tolerance defines the minimum distance between vertices in the topology. Vertices that fall within the cluster tolerance will be snapped together during the Validate Topology process. The default cluster tolerance is the minimum possible cluster tolerance, based on the precision and extent defined for the spatial reference of the dataset.

If you change the cluster tolerance, you should choose one that is an order of magnitude smaller than the accuracy of the most accurate feature class in your dataset. For example, if all of your features are accurate to 2 meters, you would set a cluster tolerance of 0.2 meters. If you have another feature class in the same dataset that is accurate to 0.25 meters, you should set the cluster tolerance to 0.025 meters, in order to avoid degrading the higher accuracy data. You will also want to assign a higher rank to the more accurate features in order to prevent them from being snapped to less accurate features. More information about ranking feature classes appears on the following page.

Tip

Feature classes not in the topology

Only feature classes in the same dataset can participate in a topology, but not all of the feature classes in a dataset are required to participate in the topology.

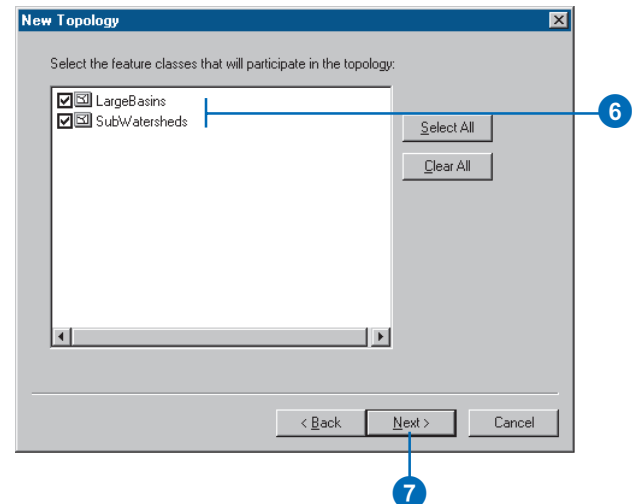
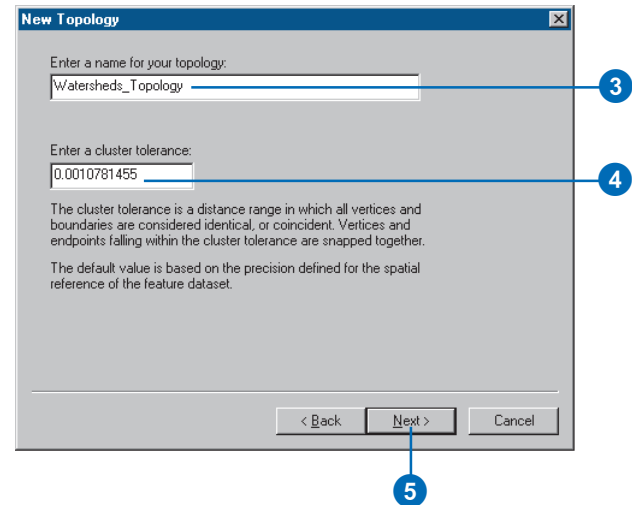
3. Type a name for the topology.
4. Type a cluster tolerance for the topology.

The default cluster tolerance is based on the precision of the dataset and is the minimum possible cluster tolerance.

5. Click Next.
6. Check the feature classes that will participate in the topology.

You will not see relationship classes, annotation classes, dimension classes, feature classes registered as versioned, or feature classes that participate in another topology or geometric network.

7. Click Next. ►



Tip

Feature class ranks

When a topology is validated, all of the vertices of each feature class are evaluated against the cluster tolerance. Vertices that are within the cluster tolerance of each other are snapped together. In order to avoid having vertices from a feature class collected with a high level of accuracy being snapped to vertices from a less accurate feature class, you assign each feature class a rank. Vertices from higher ranking feature classes will not be moved when snapping with vertices with lower ranked feature classes. The highest rank is 1, and you can assign up to 50 different ranks. The position of vertices belonging to feature classes of the same rank will be geometrically averaged when they fall within the cluster tolerance.

Tip

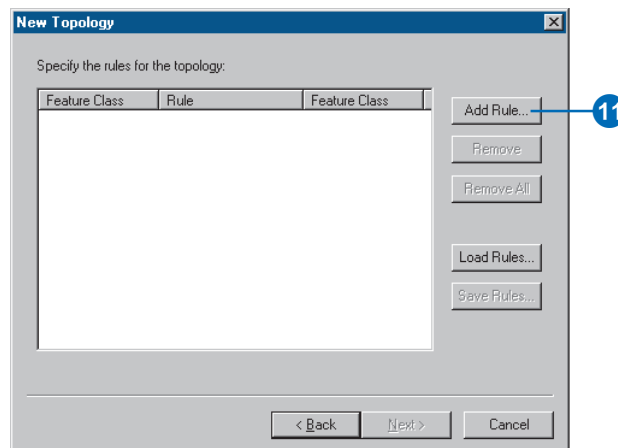
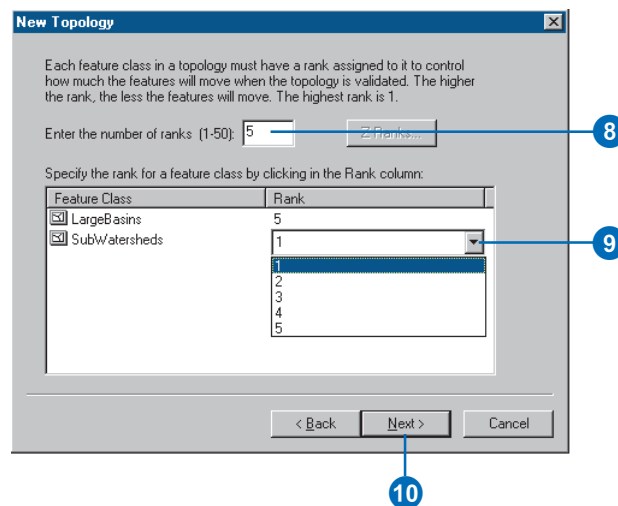
Z ranks

Feature classes that are z-aware (pointZ, polylineZ, polygonZ, multipatch, and so on) have z-values embedded in their geometry for each vertex. If there are z-aware feature classes in your topology, you can rank them so that the z-values of vertices collected with higher accuracy are not changed when snapping with the z-values of features of lower rank.

8. Type the number of ranks you want to allow in the topology.

Optionally, if your features have z-values embedded in their geometries, click the Z Rank button to set the Z ranks.

9. Click in the Rank column and assign each feature class a rank.
10. Click Next.
11. Click Add Rule. ►



12. Select a feature class that you want to participate in a topology rule.

13. Select a topology rule.

The rule description for the rule you picked appears in the Rule Description panel.

14. Select the second feature class if the rule relates the feature class to another feature class.

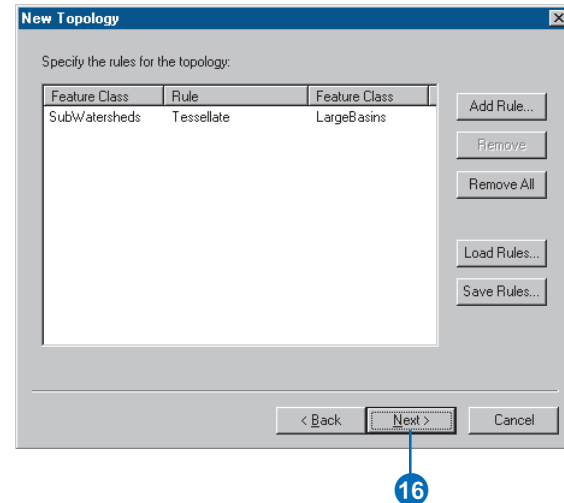
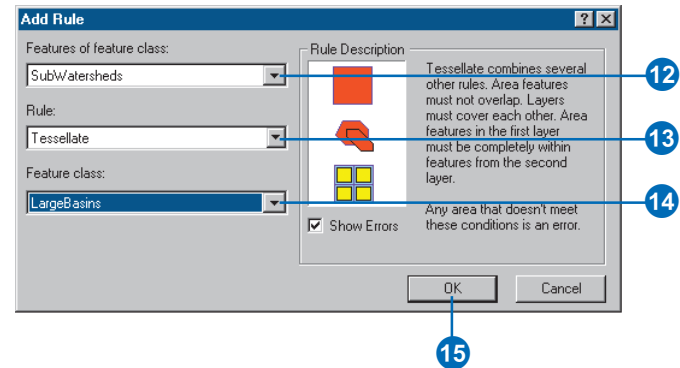
Optionally, click the Show Errors check box to hide or show which geometric relationships are considered errors for the rule.

15. Click OK.

Optionally, add additional topology rules, repeating steps 11 through 15.

Optionally, click Save rules. This will save the rules you've specified as a rule set. Saving a rule set is useful when you know you will be setting up another topology using the same or similar rules.

16. Click Next. ►



See Also

For details about using storage keywords with ArcSDE, see Managing ArcSDE Services.

Tip

Validating topology

The first time you create a topology, you will be asked to validate it. Validating a topology evaluates the rules and creates error features where the rules are violated. Validating the topology also starts the cracking and clustering process, which can take some time and is irreversible.

During cracking, vertices are created at the intersection of feature edges. During clustering, vertices that fall within the cluster tolerance are snapped together. The rank of a feature class determines whether or not its vertices will move when they fall within the cluster tolerance of vertices of another feature. Where vertices belong to features with the same rank—within the same feature class, for example—the clustering process geometrically averages the position of the vertices.

17. Click Yes, click the configuration keyword dropdown list, and click the keyword to use if your geodatabase is stored in an ArcSDE database and you have a configuration keyword for the topology storage. If not, skip to step 18.

18. Review the parameters and rules you've defined for the topology.

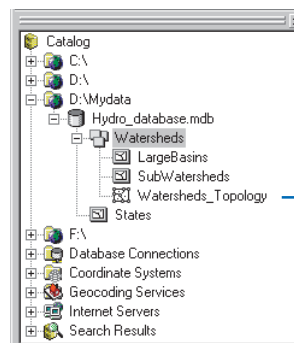
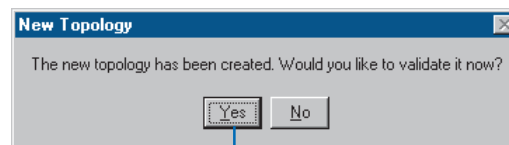
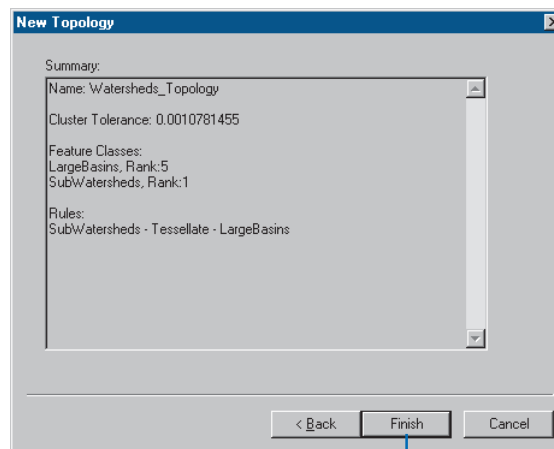
19. Click Finish.

The wizard begins creating the new topology and a progress bar appears. You can cancel the process by clicking Cancel.

Once the topology is created, you will be asked if you want to validate the topology.

20. Click Yes.

The topology is validated and appears in the feature dataset.



New topology added to dataset in ArcCatalog

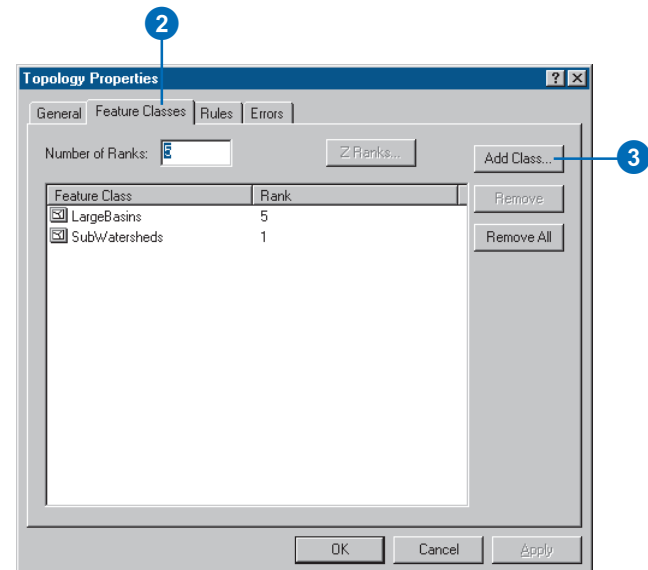
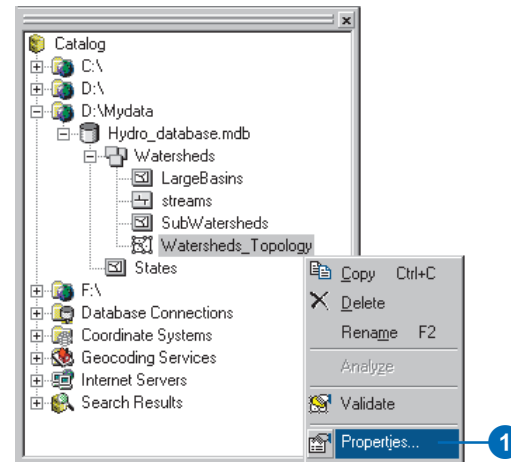
Adding new feature classes to your topology

At any time, you can add new feature classes to a topology. These new feature classes can be empty or may contain existing features. The new feature class must be in the same feature dataset as the topology. Versioned feature classes cannot be added to a topology.

When adding a feature class to a topology, you must specify the rules that govern the feature classes' spatial relationships. Adding new rules to a topology automatically makes the entire topology dirty, so when you finish adding rules you will need to revalidate the topology. The new features may create error conditions, depending on the rules that you add.

Adding a new feature class to a topology

1. Right-click the topology and click Properties.
2. Click the Feature Classes tab.
3. Click Add Class. ►

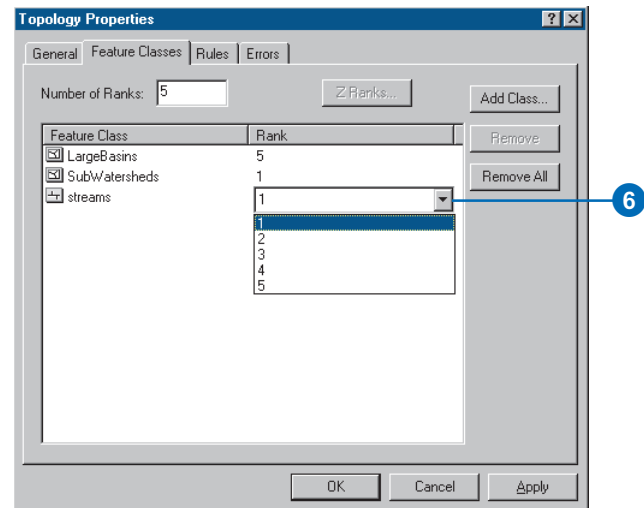
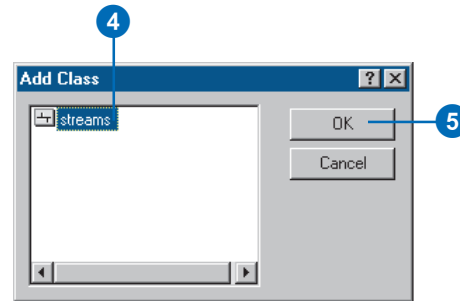


- Click the feature class that you want to add to the topology.

Only feature classes that are within the dataset, and that are not currently participating in a topology or geometric network, can be added.

You will not see relationship classes, annotation classes, dimension classes, feature classes registered as versioned, or feature classes that participate in another topology or geometric network.

- Click OK.
- Click in the Rank column and set a rank for the new feature class. ►

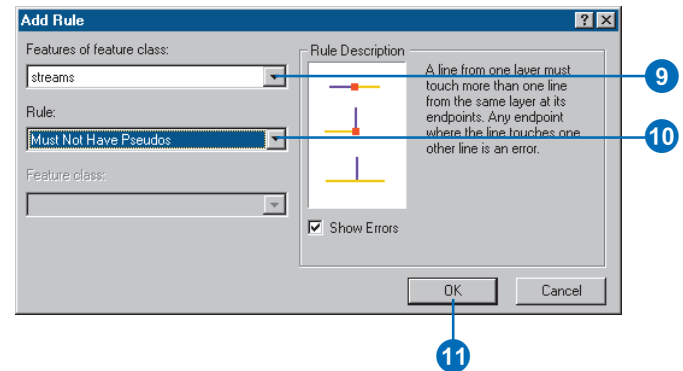
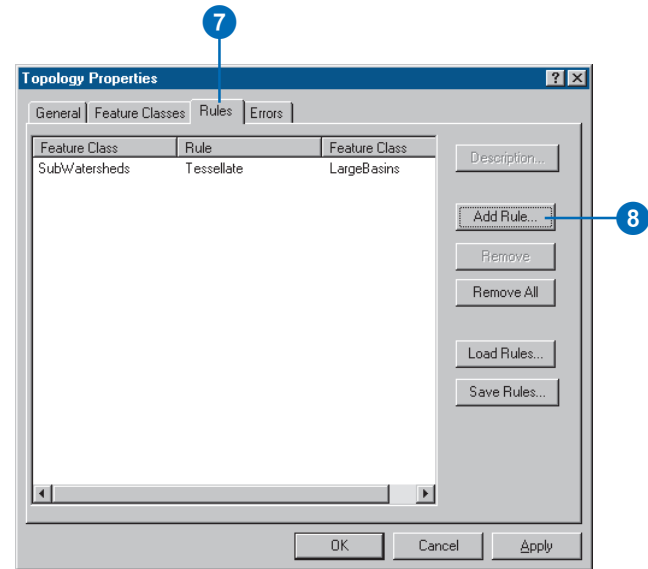


7. Click the Rules tab.
8. Click Add Rule.
9. Choose the feature class that will participate in the rule.
10. Choose the rule.

Optionally, choose the other feature class that will participate in the rule. Some rules only apply to the features in one feature class, while others apply to features in two different classes.

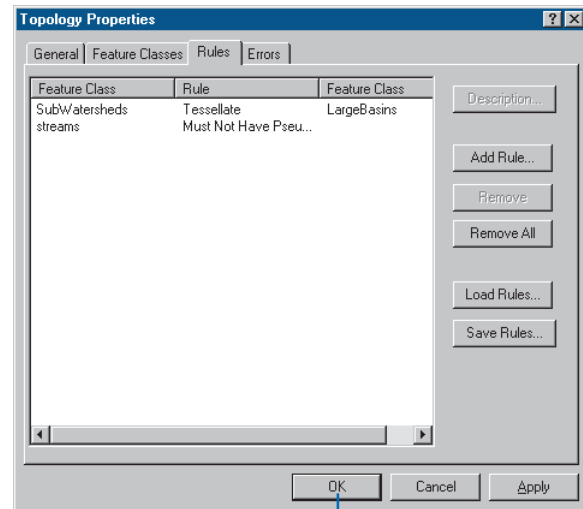
11. Click OK.

Optionally, repeat steps 8–10 to define other rules involving the new feature class. ►



12. Click OK.

The new feature class and rules have been added to the topology. The topology will need to be validated again.



12

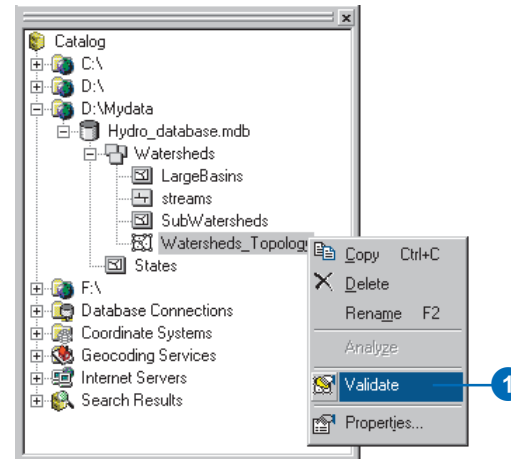
Validating a topology

When the rules or other properties of a topology are changed, the topology will need to be validated again. Validating the topology evaluates the features against the rules and finds any new errors related to new rules or feature classes. It will also remove errors related to rules or feature classes you've removed.

Validating the topology starts the *cracking* and *clustering* process, which can take some time and which is irreversible. During cracking, vertices are created at the intersection of feature edges. During clustering, vertices that fall within the cluster tolerance are snapped together. The rank of a feature class determines whether or not its vertices will move when they fall within the cluster tolerance of vertices of another feature. Where vertices belong to features with the same rank (within the same feature class, for example), the clustering process geometrically averages the position of the vertices.

Once a topology has been validated, it will typically only experience cracking and clustering again where new features are added, unless you change the cluster tolerance or add feature classes.

1. Right-click the topology and click Validate.



Topology: defining the rules

There are many topology rules that you can implement in your geodatabase depending on the spatial relationships that are most important for your organization to maintain. You should carefully plan the spatial relationships that you will enforce on your features.

Some topology rules govern the relationships of features within a given feature class, while others govern the relationships between features in two different feature classes. Topology rules can also be defined between subtypes of features in one or another feature class. This could be used, for example, to require street features to be connected to other street features at both ends, except in the case of streets belonging to the cul-de-sac or dead-end subtypes.

Some of the key topological rules that you can impose on your data are discussed in the following pages.

Polygon rules

Must Not Overlap

This rule requires that the interior of polygons in the feature class not overlap. The polygons can share edges or vertices. This rule is used when an area cannot belong to two or more polygons. It is useful for modeling administrative boundaries, such as ZIP Codes or voting districts, and mutually exclusive area classifications such as land cover or landform type.

Must Not Have Gaps

This rule requires that polygons not have voids within themselves or between adjacent polygons. Polygons can share edges, vertices, or interior areas. Polygons can also be completely disconnected. This rule is used when polygons or blocks of contiguous polygons should not have empty spaces within them. It is useful for modeling land ownership, as in a parcel fabric, where a given area is completely allotted to various polygons, but

where external areas (roadways, for example) are not modeled in the same feature class.

Must Not Overlap With

This rule requires that the interior of polygons in one feature class must not overlap with the interior of polygons in another feature class. Polygons of two feature classes can share edges or vertices or be completely disjointed. This rule is used when an area cannot belong to two separate feature classes. It is useful for combining two mutually exclusive systems of area classification, such as zoning and water body type, where areas defined within the zoning class cannot also be defined in the water body class and vice versa.

Must Be Covered By Feature Class Of

This rule requires that a polygon in one feature class must share all of its area with polygons in another feature class. An area in the first feature class that is not covered by polygons from the other feature class is an error. This rule is used when an area of one type, such as a state, should be completely covered by areas of another type, such as counties.

Must Cover Each Other

This rule requires that the polygons of one feature class must share all of their area with the polygons of another feature class. Polygons may share edges or vertices. Any area defined in either feature class that is not shared with the other is an error. This rule is used when two systems of classification are used for the same geographic area and any given point defined in one system must also be defined in the other. One such case occurs with nested hierarchical datasets, such as census blocks and block groups or small watersheds and large drainage basins. The rule can also be applied to nonhierarchically related polygon feature classes, such as soil type and slope class.

Must Be Covered By

This rule requires that polygons of one feature class must be contained within polygons of another feature class. Polygons may share edges or vertices. Any area defined in the contained feature class must be covered by an area in the covering feature class. This rule is used when area features of a given type must be located within features of another type. This rule is useful when modeling areas that are subsets of a larger surrounding area, such as management units within forests or blocks within block groups.

Tessellate

This is a group of rules that requires area features within two feature classes to meet three conditions:

- a. Areas in the same feature class must not overlap.
- b. Feature classes must cover each other.
- c. Area features from the first feature class must fall completely within area features of the second feature class.

This rule is used where features in one feature class are composed of subsidiary features in the other feature class. One example would be parks and management areas, where all management areas would be nonoverlapping, all parks would be composed of one or more management areas, and all of the area accounted for in parks would be allocated to a management area and vice versa.

Boundary Must Be Covered By

This rule requires that boundaries of polygon features must be covered by lines in another features class. This rule is used when area features need to have line features that mark the boundaries of the areas. This is usually when the areas have one set of attributes, and their boundaries have other attributes. For ex-

ample, parcels might be stored in the geodatabase along with their boundaries. Each parcel might be defined by one or more line features that store information about their length or the date surveyed, and every parcel should exactly match its boundaries.

Area Boundary Must Be Covered By Boundary Of

This rule requires that boundaries of polygon features in one feature class be covered by boundaries of polygon features in another feature class. This is useful when polygon features in one feature class, such as subdivisions, are composed of multiple polygons in another class, such as parcels, and the shared boundaries must be aligned.

Contains Point

This rule requires that a polygon in one feature class contain at least one point from another feature class. Points must be within the polygon, not on the boundary. This is useful when every polygon should have at least one associated point, such as when parcels must have an address point.

Line rules

Must Not Overlap

This rule requires that lines not overlap with lines in the same feature class. This rule is used where line segments should not be duplicated, for example, in a stream feature class. Lines can cross or intersect but cannot share segments.

Must Not Intersect

This rule requires that line features from the same feature class not cross or overlap each other. Lines can share endpoints. This rule is used for contour lines that should never cross each other or in cases where the intersection of lines should only occur at endpoints such as street segments and intersections.

Must Not Have Dangles

This rule requires that a line feature must touch lines from the same feature class at both endpoints. An endpoint that is not connected to another line is called a dangle. This rule is used when line features must form closed loops such as when they are defining the boundaries of polygon features. It may also be used in cases where lines typically connect to other lines as with streets. In this case, exceptions can be used where the rule is occasionally violated as with cul-de-sac or dead-end street segments.

Must Not Have Pseudo-nodes

This rule requires that a line connect to at least two other lines at each endpoint. Lines that connect to one other line (or to themselves) are said to have pseudo-nodes. This rule is used where line features must form closed loops, such as when they define the boundaries of polygons, or when line features logically must connect to two other line features at each end, as with segments in a stream network, with exceptions being marked for the originating ends of first-order streams.

Must Not Intersect Or Touch Interior

This rule requires that a line in one feature class must only touch other lines of the same feature class at endpoints. Any line segment in which features overlap or any intersection not at an endpoint is an error. This rule is useful where lines must only be connected at endpoints, such as in the case of lot lines, which must split (only connect to the endpoints of) back lot lines and which cannot overlap each other.

Must Not Overlap With

This rule requires that a line from one feature class not overlap with line features in another feature class. This rule is used when line features cannot share the same space. For example, roads

must not overlap with railroads, or depression subtype of contour lines cannot overlap with other contour lines.

Must Be Covered By Feature Class Of

This rule requires that lines from one feature class must be covered by the lines in another feature class. This is useful for modeling logically different but spatially coincident lines such as routes and streets. A bus route feature class must not depart from the streets defined in the street feature class.

Must Be Covered By Boundary Of

This rule requires that lines be covered by the boundaries of area features. This is useful for modeling lines, such as lot lines, that must coincide with the edge of polygon features, such as lots.

Endpoint Must Be Covered By

This rule requires that the endpoints of line features must be covered by point features in another feature class. This is useful for modeling cases where a fitting must connect two pipes or a street intersection must be found at the junction of two streets.

Must Not Self Overlap

This rule requires that line features not overlap themselves. They can cross or touch themselves, but must not have coincident segments. This rule is useful for such features as streets, where segments might touch, in a loop, but where the same street should not follow the same course twice.

Must Not Self Intersect

This rule requires that line features not cross or overlap themselves. This rule is useful for lines, such as contour lines, that cannot cross themselves.

Must Be Single Part

This rule requires that lines must have only one part. This rule is useful where line features, such as highways, may not have multiple parts.

Point rules

Must Be Covered By Boundary Of

This rule requires that points fall on the boundaries of area features. This is useful when the point features help support the boundary system, such as boundary markers, which must be found on the edges of certain areas.

Must Be Properly Inside Polygons

This rule requires that points fall within area features. This is useful when the point features are related to polygons, such as wells and well pads or address points and parcels.

Must Be Covered By Endpoint Of

This rule requires that points in one feature class must be covered by the endpoints of lines in another feature class. This rule is similar to the line rule, Endpoint Must Be Covered By, except that, in cases where the rule is violated, it is the point feature that is marked as an error rather than the line. Boundary corner markers might be constrained to be covered by the endpoints of boundary lines.

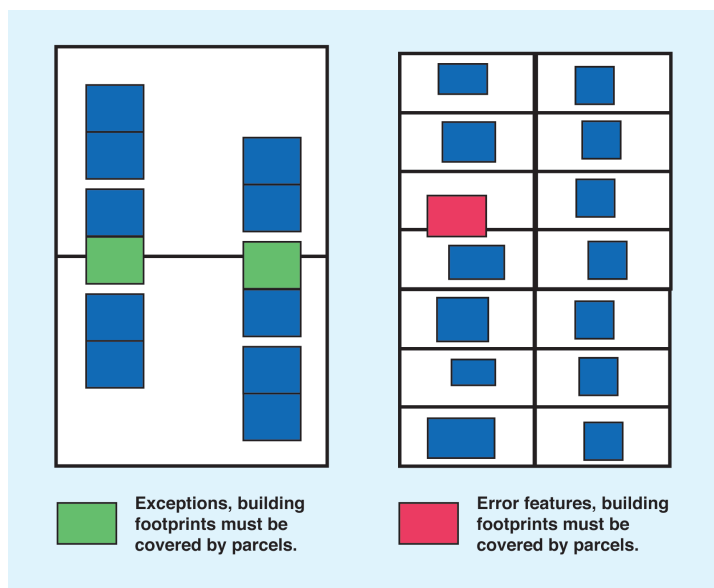
Must Be Covered By Line

This rule requires that points in one feature class must be covered by lines in another feature class. It does not constrain the covering portion of the line to be an endpoint. This rule is useful for points that fall along a set of lines, such as highway signs that fall along highways.

Planning for exceptions

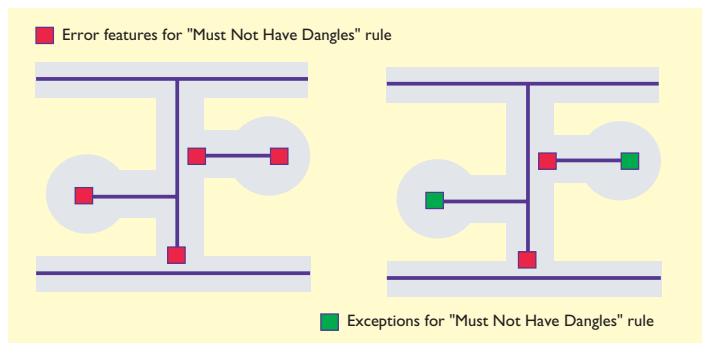
Topology rules may represent an ideal situation, but geodatabases are flexible enough to handle exceptions to the rules found in real-world data. Violations of topology rules are stored as errors in the topology but, where appropriate, you can mark them as exceptions. Exceptions are thereafter ignored when the error inspector searches for errors though you can return them to error status if you decide that they are actually errors and that the features should be modified to comply with the topology rules.

Here are a couple of cases where you might implement topology rules that you know will have exceptions. If you manage a database of parcels and you want to *digitize* a feature class of buildings, you might implement a topology rule requiring that parcels cover building features (i.e., that building features not cross parcel lines) as a quality control for the building digitizing effort. This rule might be true for 90 percent of the features in



your management area but it could be violated by some high-density housing and commercial buildings. You can designate the buildings that represent acceptable violations of the rule as exceptions.

Similarly, if you manage a street database for a city, you might create a rule that centerlines must not have dangles (i.e., that they connect at both ends to other centerlines). This rule would ensure that street segments are correctly snapped to other street segments when the streets are edited. However, some streets are cul-de-sacs, one end of which does not snap to other centerlines. These cases could be marked as exceptions and you would still be able to use the rule to find cases where streets were incorrectly digitized or edited.

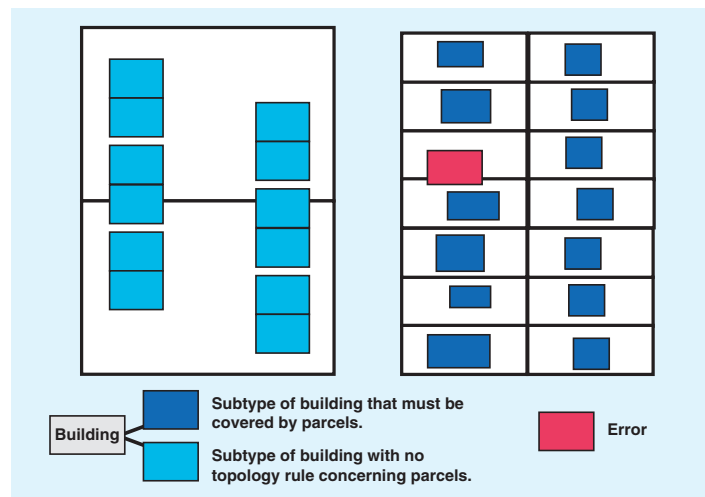


Refining topologies with subtypes

When you design a geodatabase, you should be aware of the option of creating topological relationships between subtypes of features classes. Subtypes allow you to model real-world objects more effectively by creating specialized default values and domains for specific subtypes of features. Subtypes also allow you to represent a variety of real-world objects in a given feature class instead of in several feature classes, which provides some performance benefits for the geodatabase.

Subtypes extend your design options for topology rules. In some cases, you will want a topology rule to apply to all features in a feature class, except for a certain type of feature. One way to handle this design requirement is to create the rule for the whole feature class and systematically mark the exceptions. Another is to use subtypes, rather than feature classes, in your topology rules. This allows you to create rules that apply only to specific subtypes.

Using the building footprint example from the previous page, you could solve the problem of a small percentage of buildings that can legitimately cross parcel boundaries by creating subtypes of buildings and only creating the Must Be Covered By parcels topology rule for the subtypes that cannot extend across a parcel.



Subtypes allow you finer control in setting up topology rules.

Managing a topology

You can manage topologies using ArcCatalog. Unlike most items that appear in ArcCatalog, the topology does not represent a single entity, such as a table, shapefile, or feature class. A topology is actually an association among several feature classes and is represented by several tables in the database. Managing a topology is different from managing other items in ArcCatalog.

Managing the topology itself

Some of the standard operations on the topology are handled the same way as other items in ArcCatalog. A topology can be renamed or deleted. Renaming the topology doesn't affect any of its member feature classes or the structure of the topology. Deleting a topology does not affect the participating feature classes; it merely removes the rules governing their spatial relationships.

You can delete a topology in two ways. The first is to delete the entire feature dataset that contains the topology. This action deletes from the geodatabase all of the participating feature classes, all of the topology rules, and any other objects stored inside that feature dataset. The second method is to simply delete the topology itself and leave the rest of the feature dataset intact.

You can also copy and paste a topology from one feature dataset to another. Copying a topology also copies the feature classes that participate in the topology.

Managing topologically related feature classes

Managing feature classes in a topology is more restricted than managing feature classes that do not participate in a topology. You must remove the feature class from the topology or delete

the topology before you can rename or delete a feature class that participates in a topology. You can still add fields, alter subtypes, and modify domains of fields for feature classes in a topology.

Schema locking

An exclusive lock is required to modify a topology's rules or to rename or delete a topology. An exclusive lock can only be acquired for a topology if the feature classes that participate in the topology can also be locked. Therefore, if another user has an exclusive or shared lock on any of the feature classes in a topology, then the properties of the topology cannot be edited.

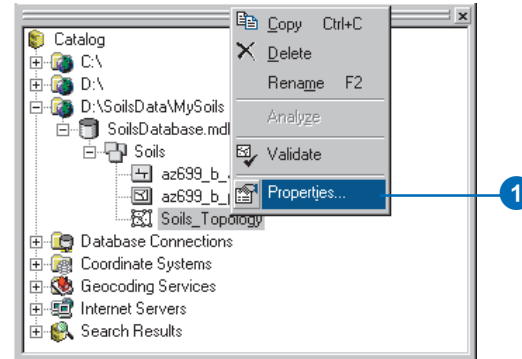
For more information on exclusive locks and schema locking, see the chapter 'Creating new items in a geodatabase' in this book.

Modifying a topology

You can change the properties of a topology that is not registered as versioned. In some cases, such as when renaming a topology, the change has no effect upon the state of the topology. In other cases, the change may require that the topology be validated again. Some changes, such as adding new feature classes or new rules or changing the cluster tolerance, may create new dirty areas, new error features, and necessitate that features undergo cracking and clustering.

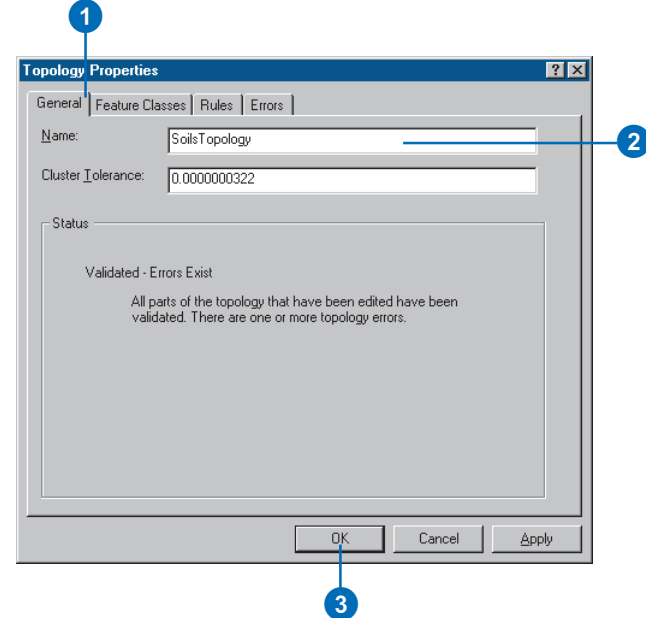
Getting the properties of a topology

1. Right-click the topology and click Properties.



Renaming a topology

1. Click the General tab on the Topology Properties dialog box.
2. Click in the Name text box and type a new name.
3. Click OK.



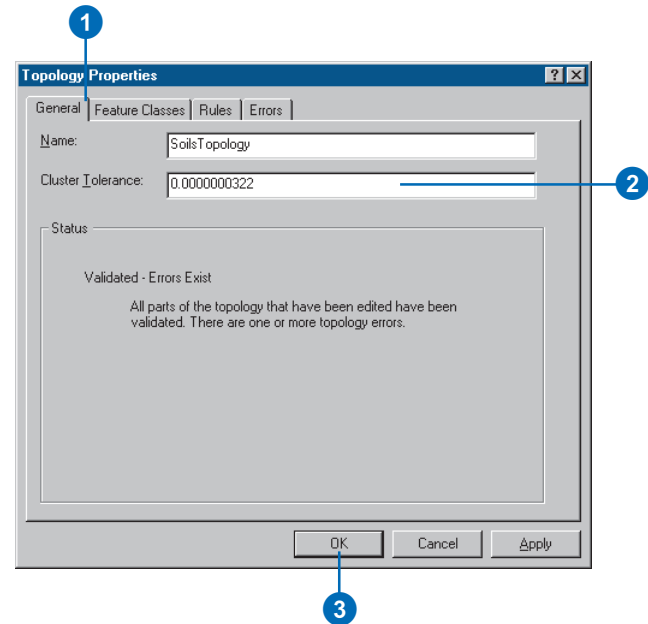
Tip

Changing the cluster tolerance

Changing the cluster tolerance of a topology will require the topology to be validated again. The larger the cluster tolerance, the greater the likelihood that features in your data will be moved from their current positions or have their shape changed.

Changing the cluster tolerance of a topology

1. Click the General tab on the Topology Properties dialog box.
2. Click in the Cluster Tolerance text box and type a new cluster tolerance.
3. Click OK.



Tip

Adding a feature class

Adding a feature class will require the topology to be validated again.

Before you validate the topology, you may want to assign topology rules to the new feature class.

Adding a feature class to a topology

1. Click the Feature Classes tab on the Topology Properties dialog box.

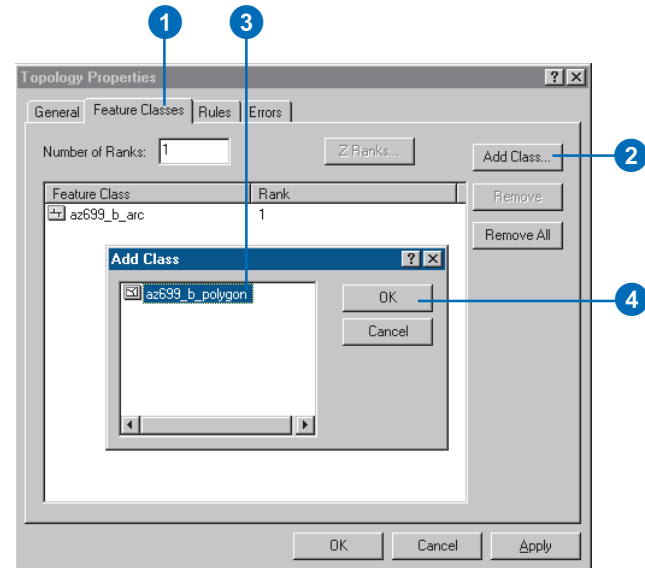
2. Click Add Class.

You see a list of the simple feature classes in the dataset that do not yet participate in a topology.

3. Click a class that you want to add.

4. Click OK.

You will need to add topology rules for this feature class.



Tip

Removing a feature class

Removing a feature class also removes all of the topology rules associated with that feature class.

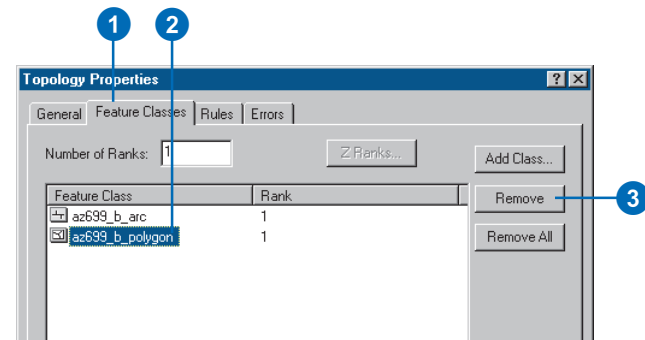
Removing a feature class will require the topology to be validated again.

Removing a feature class from a topology

1. Click the Feature Classes tab on the Topology Properties dialog box.

2. Click the feature class you want to remove.

3. Click Remove.



Tip

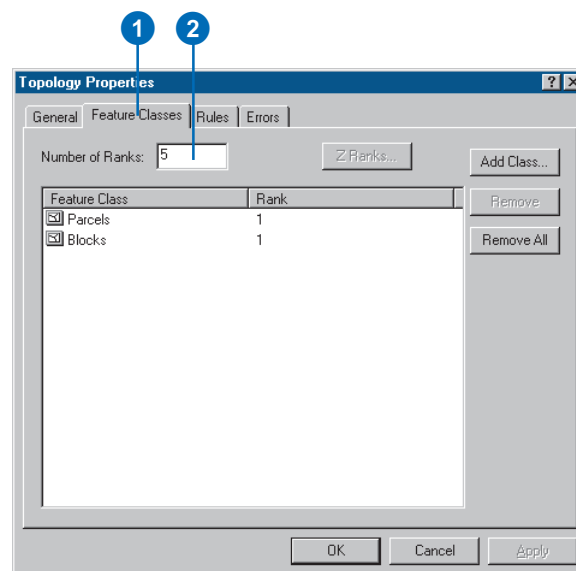
Changing the number of ranks

Changing the number of ranks will not require the topology to be validated again.

Changing the number of ranks in a topology

1. Click the Feature Classes tab on the Topology Properties dialog box.
2. Click the Number of Ranks textbox and type a number of ranks.

A topology can support up to 50 ranks to which feature classes may be assigned.



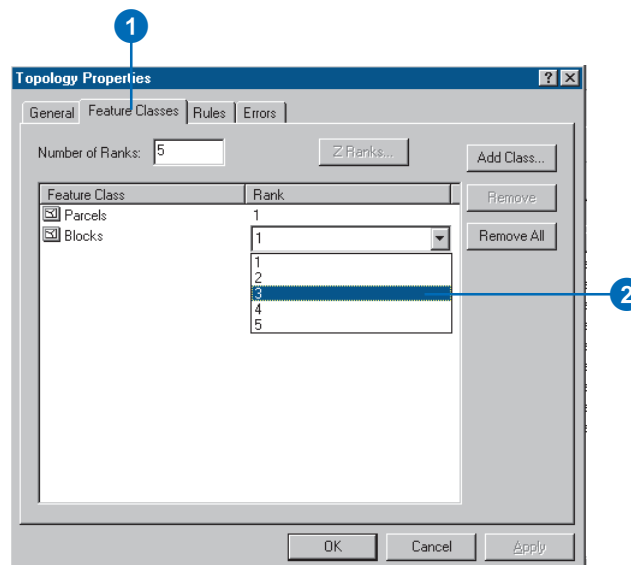
Tip

Changing the rank of a feature class

Changing the rank of a feature class will require the topology to be validated again.

Changing the rank of a feature class in a topology

1. Click the Feature Classes tab on the Topology Properties dialog box.
2. Click the current rank of the feature class.



Tip

Adding a rule

Adding a rule will require the topology to be validated again.

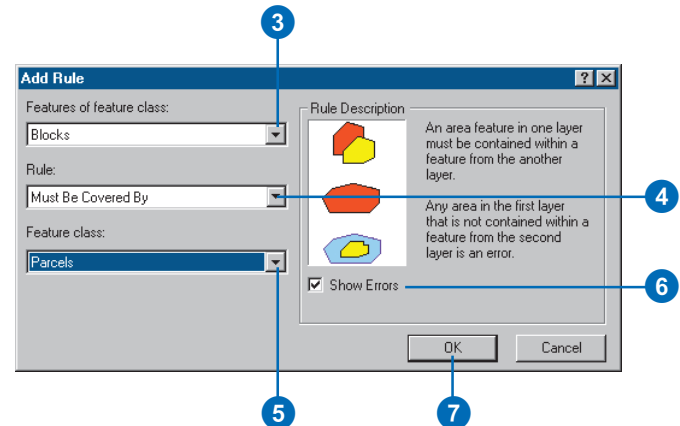
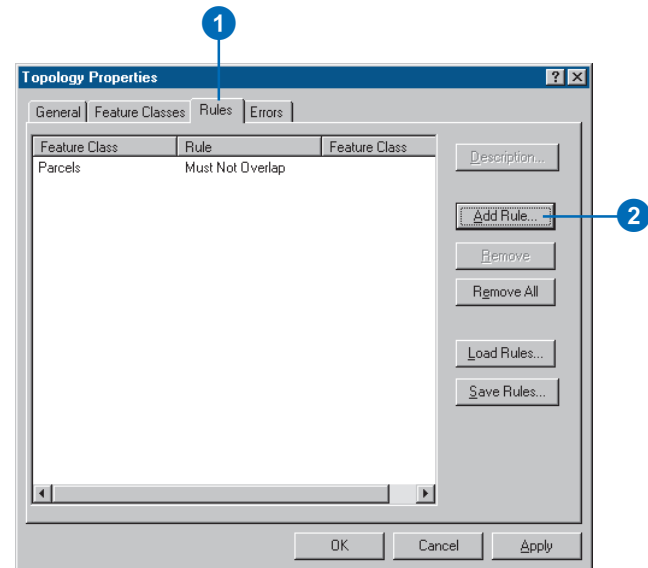
Tip

Adding feature classes before adding rules

You must add feature classes to the topology before you can specify rules.

Adding a rule to a topology

1. Click the Rules tab on the Topology Properties dialog box.
2. Click Add Rule.
3. Click the dropdown list to select the feature class or subtype that the rule will apply to.
4. Click the dropdown list to select the rule that you want to apply.
5. Click the dropdown list to select the feature class or subtype if the rule involves a topological relationship with another feature class.
6. Optionally, check and uncheck the Show Errors textbox to see simplified graphics of what constitutes a violation of this rule.
7. Click OK.



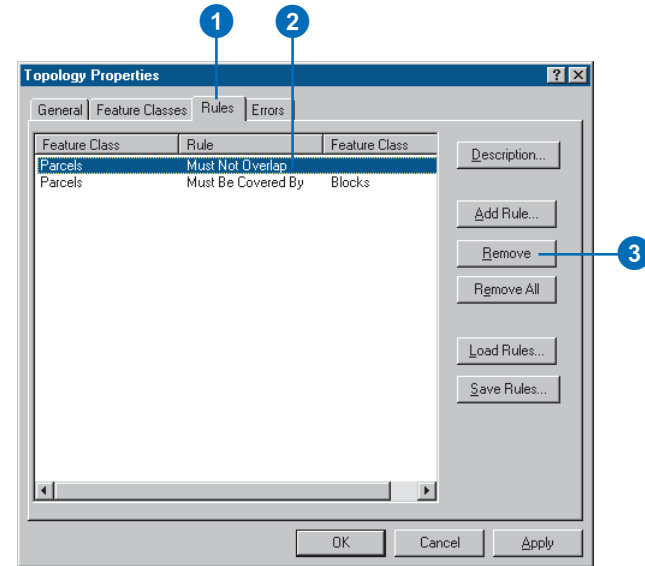
Tip

Removing a rule

Removing a rule will require the topology to be validated again.

Removing a rule from a topology

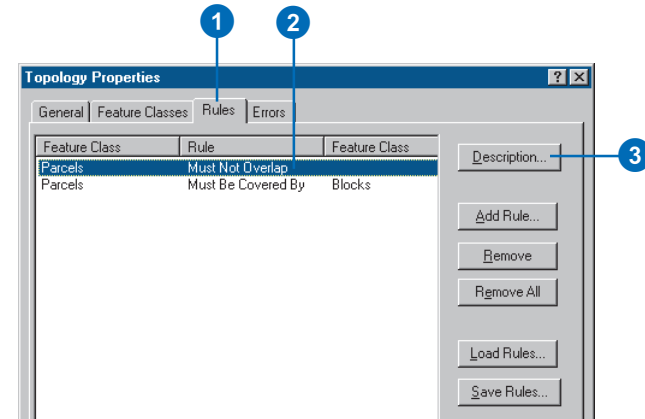
1. Click the Rules tab on the Topology Properties dialog box.
2. Click the rule that you want to remove.
3. Click Remove.



Getting the description of a topology rule

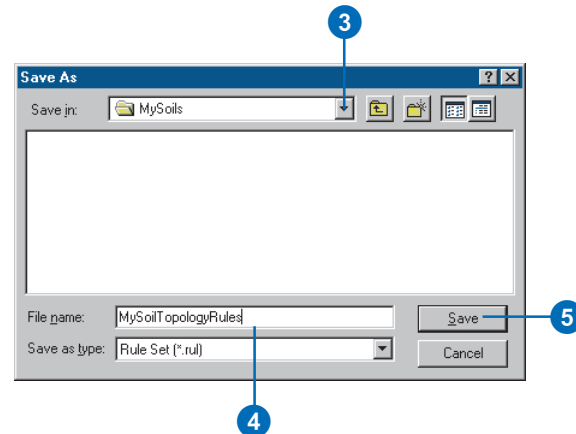
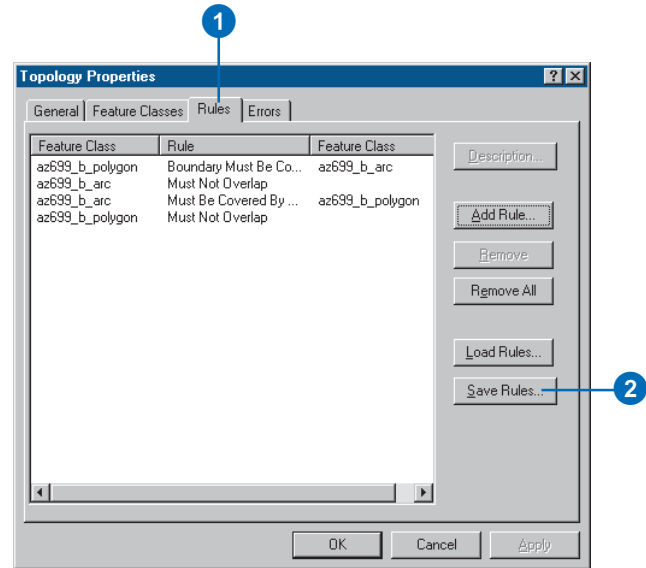
1. Click the Rules tab on the Topology Properties dialog box.
2. Click the rule that you want to see a description of.
3. Click Description.

You can also view the description of a topology rule for which errors have been detected using the Show Rule Description item in the Fix Topology Error context menu.



Saving rules for a topology into a Rule Set file

1. Click the Rules tab on the Topology Properties dialog box.
2. Click Save Rules.
3. Navigate to the place where you want to save the rules that you've defined for the topology.
4. Type a name for the Rule Set.
5. Click Save.



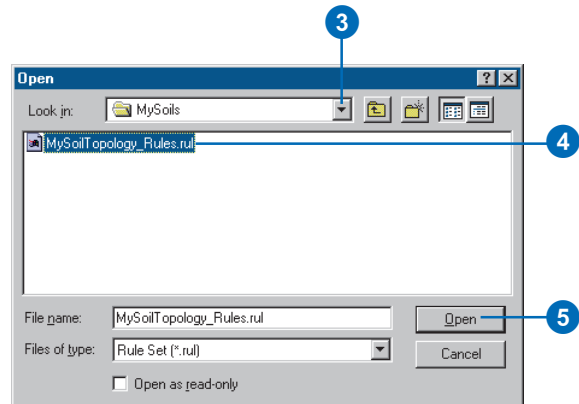
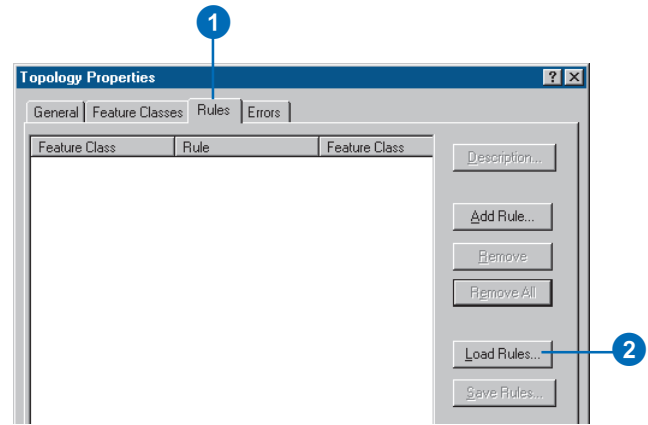
Tip

Loading rules

Loading a Rule Set removes all existing rules and replaces them with the rules specified in the Rule Set. Loading a Rule Set will require the topology to be validated again.

Loading rules for a topology from a Rule Set file

1. Click the Rules tab on the Topology Properties dialog box.
2. Click Load Rules.
3. Navigate to the place where the Rule Set that you want to load is saved.
4. Click the Rule Set.
5. Click Open. ►



Tip

Loading a rule set when some feature classes cannot be matched

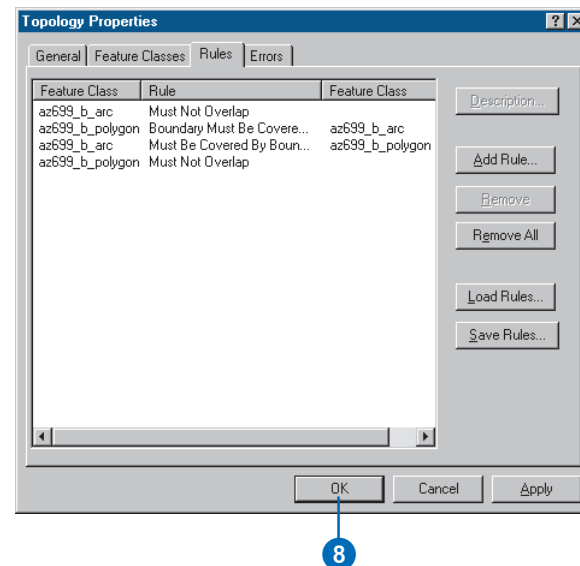
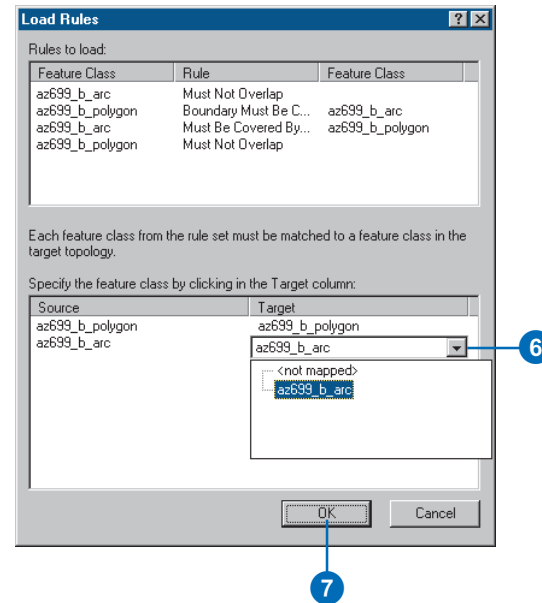
If there are feature classes specified in a rule set that cannot be matched to feature classes in the new topology, rules involving the unmatched feature classes will not be loaded.

The Load Rules feature class mapping dialog box appears.

If the rule set was created from a topology that had the same feature class names as the feature classes in the new topology you're defining, the feature classes named in the rule set should be correctly matched to the feature classes in the new topology.

If the names are different, you will need to match the feature classes mentioned in the rule set to their corresponding feature classes in the new topology.

6. Click in the Target column and click the feature class that it corresponds to in the new topology for each Source feature class that is not mapped.
7. Click OK.
8. Click OK on the Topology Properties dialog box.

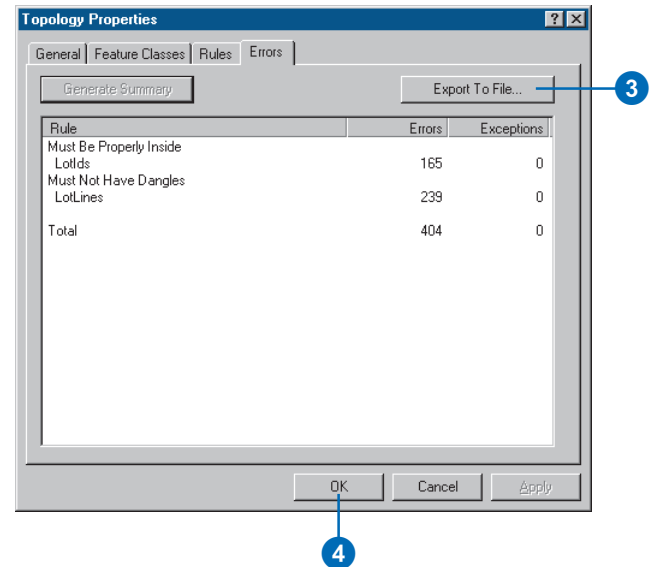
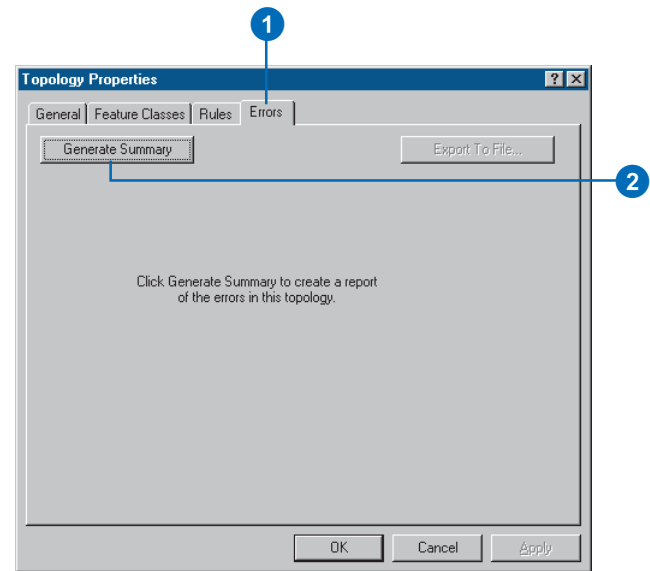


Summarizing topology errors

You can view a summary of the number of errors in a topology from the Topology Properties dialog box. The summary tells you how many errors and exceptions exist for each of the topology rules.

You can save the summary as a text file to create a record of the state of the topology at a given time. This can be a useful way to document and monitor progress on a large topology editing project.

1. Click the Errors tab on the Topology Properties dialog box.
2. Click Generate Summary.
3. Optionally, click Export To File if you want to save the contents of the summary to a text file.
4. Click OK.



Creating new polygons from lines

Sometimes you need to create polygon features from line feature data. For example, you might have digitized the boundaries of a set of features into a line feature class, or you may have only been able to obtain line features from a data provider.

The Polygon Feature Class From Lines tool lets you create new polygon features from line and polygon features in one or more feature classes within a given dataset in a geodatabase. You can specify a point feature class that will supply attributes for the new polygon features.

Tip

Make sure your linework forms closed areas

The Create Polygon feature class tool creates polygons from areas that are completely enclosed by lines or polygon edges. If there are gaps in the linework defining a polygon's boundary, a polygon will not be created. Very small gaps can be spanned by increasing the cluster tolerance.

1. Right-click the dataset, point to New, and click Polygon Feature Class From Lines in ArcCatalog.
2. Type a name for the new polygon feature class.
3. Optionally, change the cluster tolerance.

You can increase the cluster tolerance to span very small gaps in the linework, but a large cluster tolerance may create polygons that you do not want. It is generally best to clean up the linework before creating polygons.

4. Check the feature classes that you want to be considered when finding closed areas.

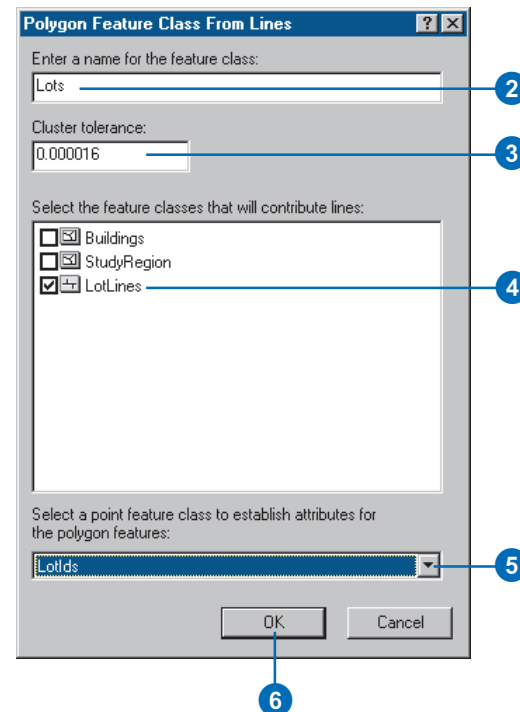
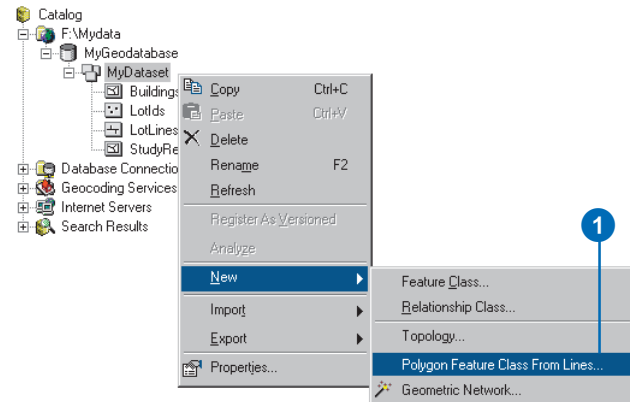
If you check more than one feature class, then areas that are enclosed by lines from any of the feature classes will become polygon features.

If you check a polygon feature class, the boundaries of the polygons will be considered as lines.

5. Optionally, select a point feature class that will be used to assign attributes to the new polygons.

Polygons that enclose a point will be given the attributes of the point.

6. Click OK. ►

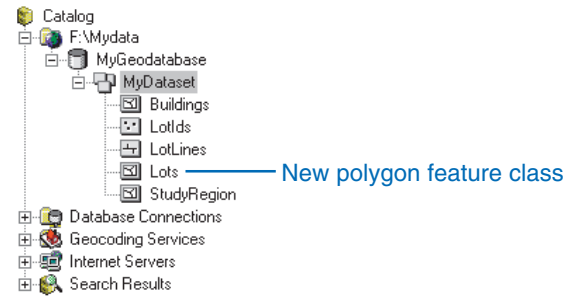


Tip

Creating polygons from lines in ArcMap

You can also use the Construct Features tool while editing in ArcMap to create new features in a target feature class from selected features. The Construct Features tool is on the Topology toolbar and works when you are editing data with map topology or a geodatabase topology.

The new feature class is created in the feature dataset.



Topology and versioned databases

You can use geodatabase topologies in a versioned geodatabase but you must have appropriate permission to edit the geodatabase.

You can only create a topology and edit the properties of a topology in a nonversioned dataset. After the topology has been created, you can register the dataset as versioned and work with the topology in any version.

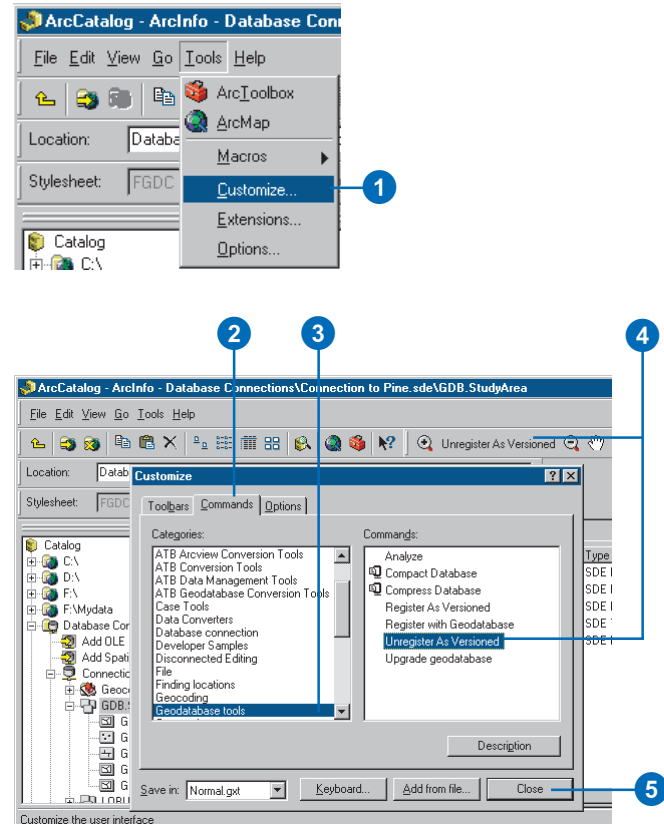
If you want to add a topology to a versioned dataset, you will first need to unregister the dataset as versioned. To unregister the dataset as versioned, you may need to add the Unregister As Versioned command to ArcCatalog.

For more information about how versioned data behaves with topology, see the section ‘Topology and versioning’ in this chapter.

Creating topology in a versioned database

If you have edits in existing versions, they will be lost when you unregister the data as versioned. To preserve the edits, you must compress the database before you unregister it as versioned.

1. Click Tools and click Customize.
2. Click the Commands tab.
3. Click Geodatabase tools.
4. Click and drag Unregister As Versioned onto a toolbar.
5. Click Close.
6. Reconcile and post each outstanding version in the database against the DEFAULT version. After posting, delete each version.
7. Run compress to compress the database. ►

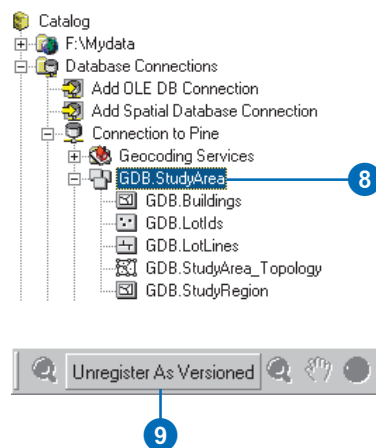


Tip

Removing a customization

After you've added a new command or tool to the interface and used it, you can remove it by opening the *Customize* dialog box and dragging the command or tool from the interface back onto the *Customize* dialog box.

8. Click the dataset.
9. Click Unregister As Versioned.
10. Unregister the data as versioned. Note: If you have not completed steps 6 and 7 before unregistering your data as versioned, then you will lose any edits that those versions contain. A warning dialog box will appear informing you that outstanding edits still remain in existing versions.
11. Create the topology.
12. Register the dataset as versioned.



Topology and versioning

It is helpful to understand how the versioning process works in geodatabases before planning your workflow. For more information on how versioning works in a geodatabase, see the chapter ‘Working with a versioned geodatabase’ in this book.

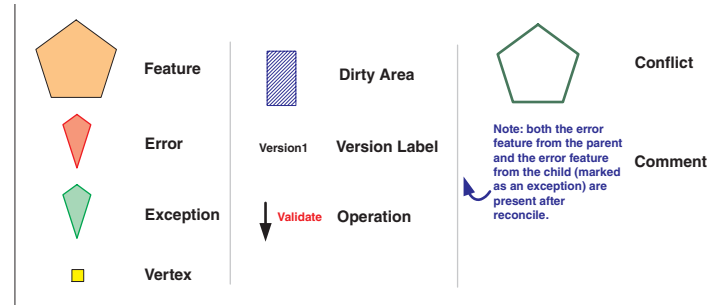
In a versioned geodatabase, feature classes that participate in a topology do not have any special version reconciliation or *conflict* detection and resolution behavior. However, the dirty areas, error features, and exceptions that are maintained by the topology itself do have special behavior in the version reconciliation and conflict detection process to ensure the integrity of the topology.

The nature of topological edits and the cracking and clustering modifications that the validate process makes to feature geometry may result in conflicts during version reconciliation.

You should consider the behavior of dirty areas, error features, and the types of conflicts that may result from topological edits when you are planning your work flow for managing versioned topological feature classes.

The following sections describe the results of a reconcile on dirty areas, errors, exceptions, and potential conflicts. In each case, the results are based on a reconcile in which the parent and child versions have both been edited since they were created. If the parent version is not edited before the child version is reconciled, the results of the reconcile will be the contents of the child version. In each example, Version2 is created as a child of Version1. Both versions are then edited in the manner described in the example, then Version2 is reconciled against Version1. The

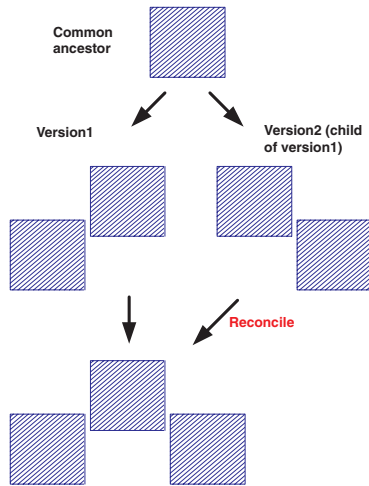
result of the reconcile of Version2 against Version1 is described here. For the illustrations in the following examples, use the following as a legend:



Dirty areas

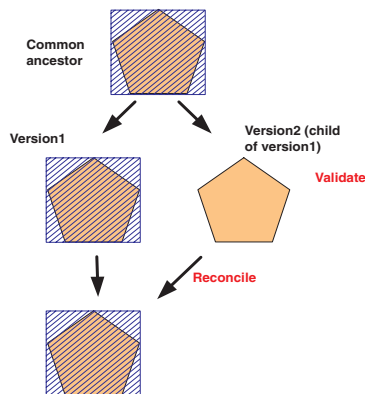
New topology errors may occur when edited parent and child versions are reconciled, even when the dirty areas within each version have been validated and are free of errors. In order to detect such topology errors, dirty areas are treated in a special way by the reconcile process. Results of the reconcile on dirty areas can be summarized as:

- Any dirty areas present in the parent or child version that did not exist in the common ancestor (that is, did not exist before the parent and child version were created) will remain dirty as a result of the reconcile:



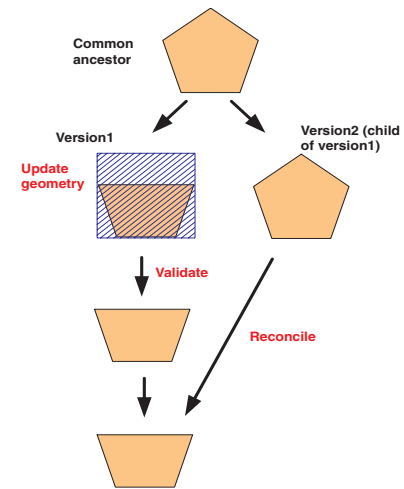
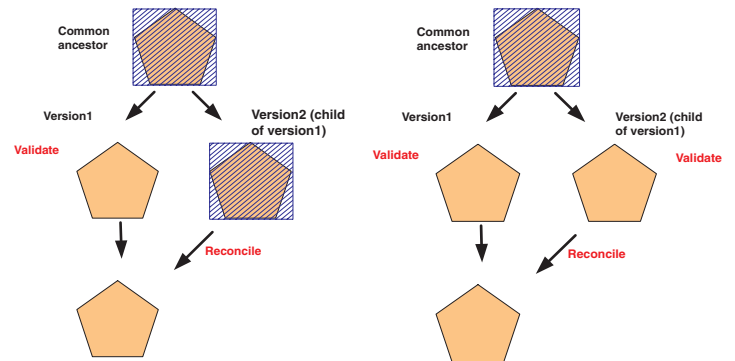
Any dirty areas created in either the parent or child version will remain dirty as a result of the reconcile.

- Any dirty area that was present in the common ancestor and validated in the child version will become dirty as a result of the reconcile:



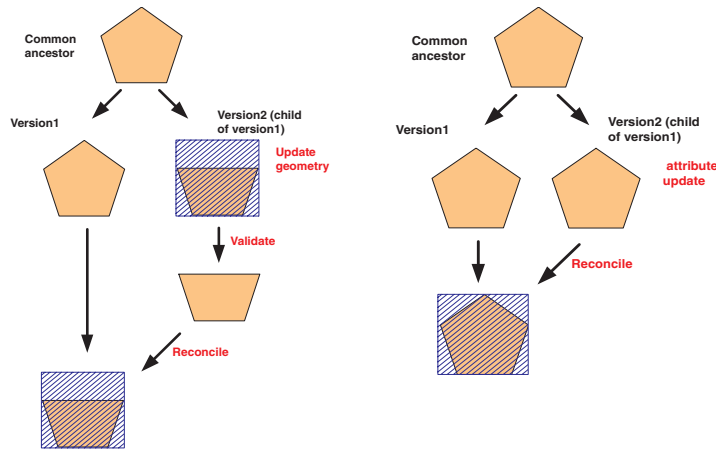
A dirty area in the common ancestor, validated in the child version, will become dirty as a result of the reconcile.

- Any dirty area that is validated in the parent version, whether it was present in the common ancestor or created in the parent version, will remain validated as a result of the reconcile:



Dirty areas validated in the parent version remain validated as a result of the reconcile.

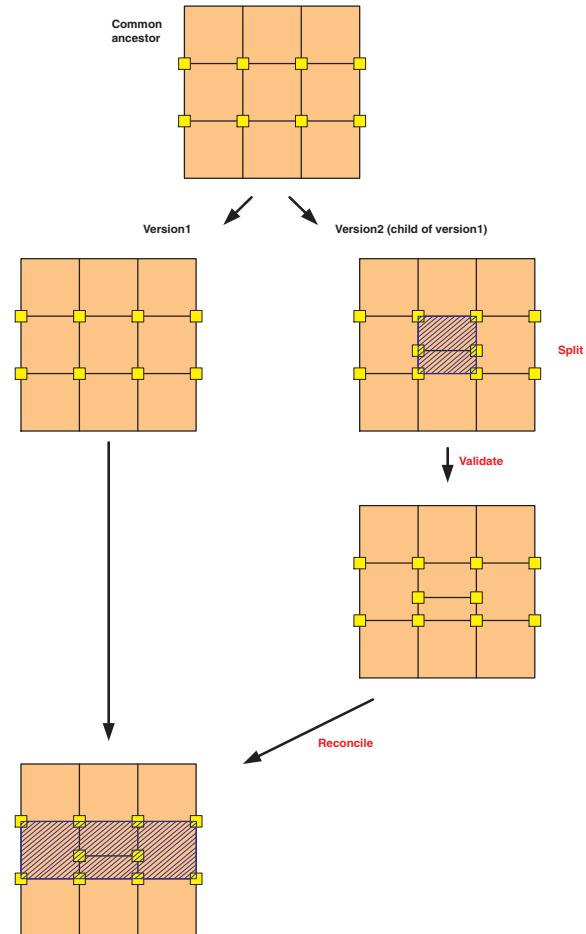
- Any edits made to topology features in the child version will result in a dirty area on the reconcile, even if the dirty area resulting from the edit is validated in the child version. This is also the case when the original edit did not result in a dirty area (e.g., an attribute update):



Edits made to topology features in the child version result in dirty areas after the reconcile.

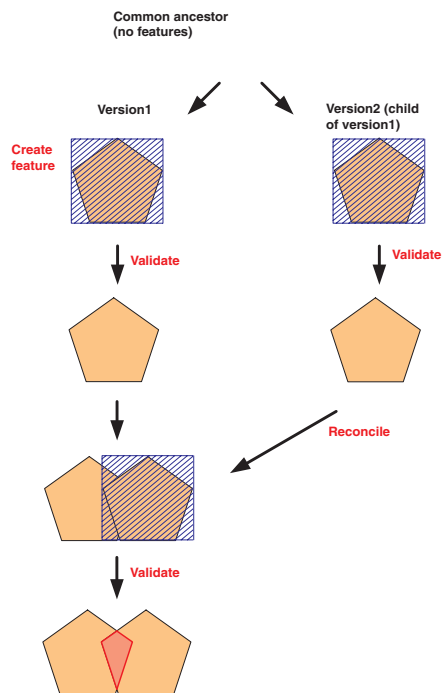
There are a number of scenarios where a reconcile can result in new dirty areas that did not exist in the parent or child, because of cracking and clustering during the validate process. In the following example, Version2 is created as a child of Version1. Both versions contain polygons that share edges in a topology that existed in the common ancestor. A polygon is split in the child version and the dirty area is validated. Splitting the polygon deletes the original feature and replaces it with two new ones. When the dirty area is validated, cracking and clustering introduces new vertices into the shared edges of the adjacent polygons. When the versions are reconciled, all of the features that

have been modified in the child version—the split polygons and the polygons modified by cracking and clustering—are covered by dirty areas:



Insertion of new vertices during cracking and clustering may result in additional dirty areas on the reconcile.

The following examples illustrate why this is necessary. After Version1 and Version2 are created, they can both be independently updated. In the following example, new features were created in each version and the resulting dirty areas were validated, resulting in no errors. On a reconcile, dirty areas must be re-created such that errors introduced by merging the changes from the two versions together can be discovered. In this example, features added in Version1 and Version2 overlap each other, which is a violation of the “no overlaps” rule:



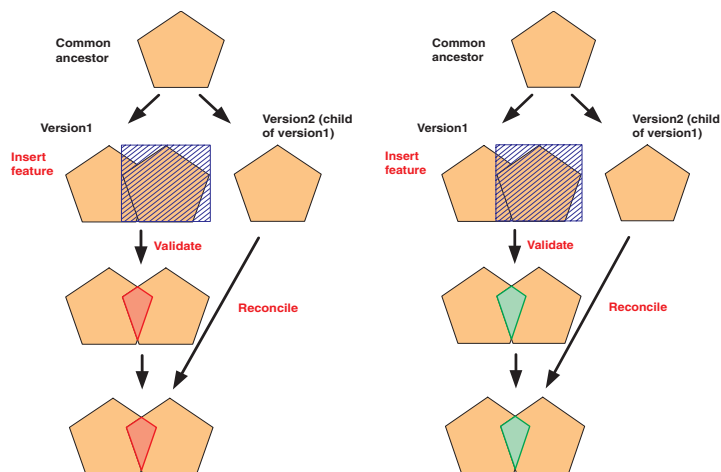
It is necessary to reactivate dirty areas from the child version after a reconcile to discover errors that may result from additional edits made in the parent version.

Errors and exceptions

Error features and error features marked as exceptions have special logic with respect to how they are treated during the version reconciliation process. Since errors and exceptions cannot be edited directly, the system will not report conflicts for them between two versions. Errors are only created by the topology validation process, and can be deleted by correcting the error using the topology error correction tools in ArcMap or by editing features and using the validation process. Error features can only be updated by marking an error as an exception or by marking an exception as an error.

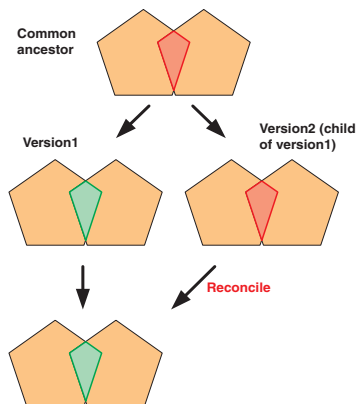
The results of a reconcile on errors and exceptions in the parent version can be summarized as:

- Any error created in the parent version, whether or not it is marked as an exception, will be brought into the child version as a result of the reconcile:



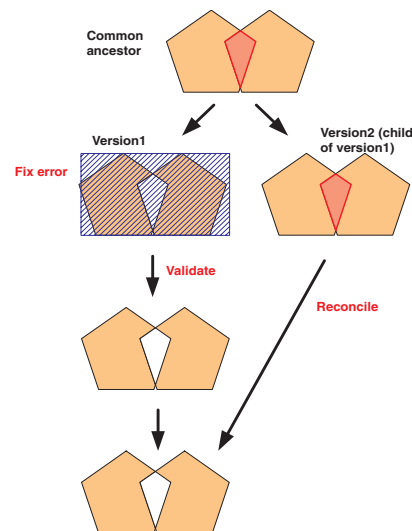
Errors and exceptions created in the parent version are brought into the child version as a result of the reconcile.

- Any error that existed in the common ancestor and is marked as an exception in the parent version will be marked as an exception after the reconcile:



Errors that existed in the common ancestor and marked as exceptions in the parent version remain exceptions after the reconcile.

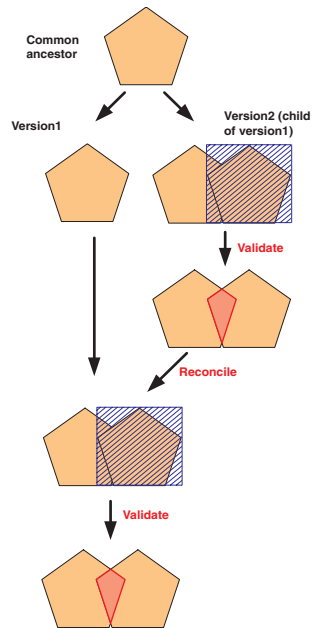
- Any error or exception that is deleted in the parent version—either by fixing the error or by the validation process—will be deleted from the child version as a result of the reconcile. This includes errors that were in the common ancestor and errors that were created in the parent version:



Errors and exceptions deleted in the parent version remain deleted in the child version as a result of the reconcile.

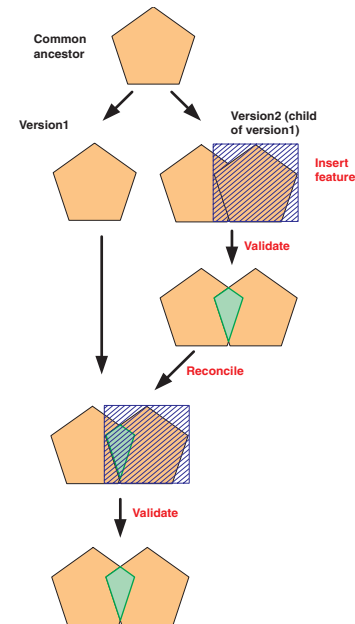
The results of the reconcile on errors and exceptions in the child version can be summarized as:

- Any error created in the child version will be deleted as a result of the reconcile, and by definition will be covered by a dirty area (see the section 'Dirty areas' in this chapter). The error can then be rediscovered by validating the dirty area:



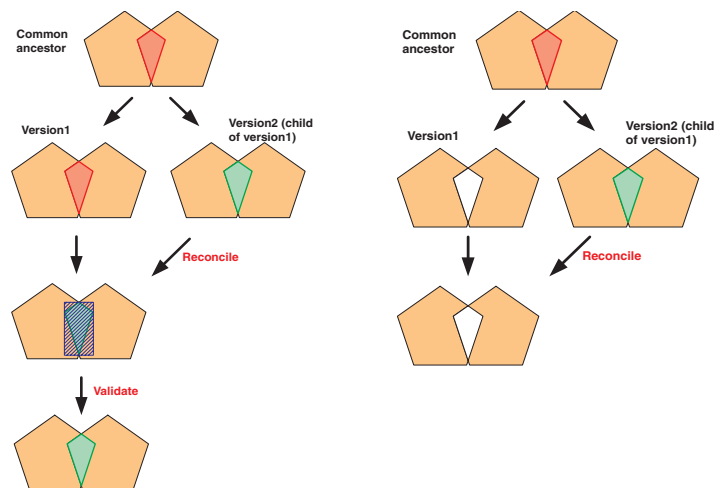
Errors created in the child version are deleted as a result of the reconcile.

- Any error created in the child version and marked as an exception will remain an exception as a result of the reconcile. By definition, it will be covered by a dirty area:



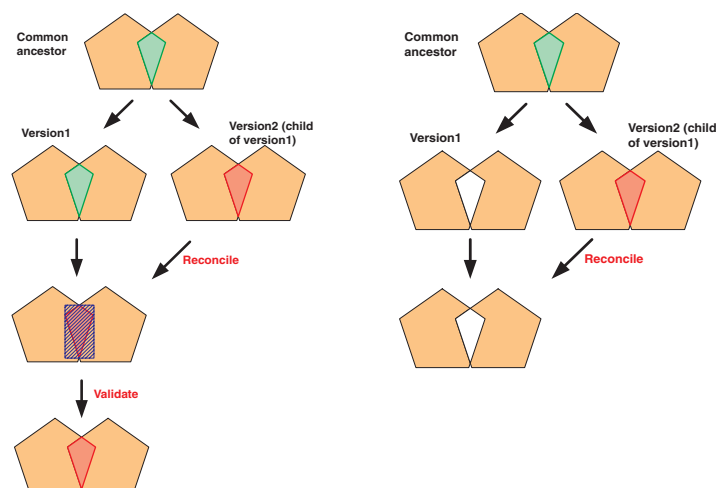
Errors created in the child version and marked as exceptions remain as a result of the reconcile and are always covered by a dirty area.

- An error that existed in the common ancestor and is marked as an exception in the child version will remain an exception as a result of the reconcile and will be covered by a dirty area. However, if the error was deleted in the parent version, then it will remain deleted in the child version:



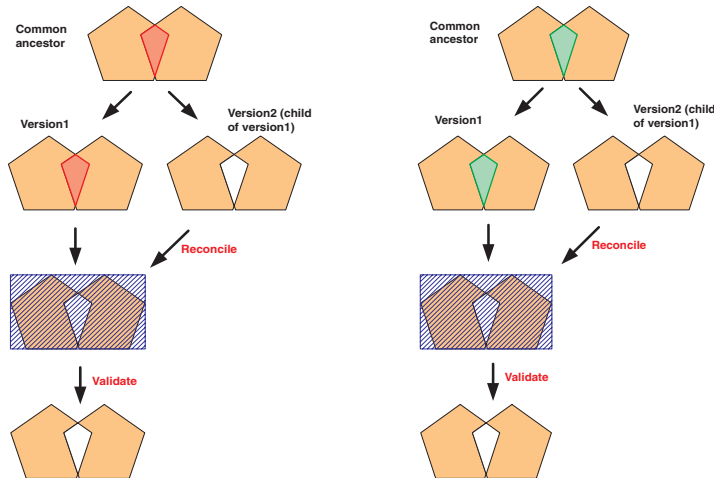
Errors that existed in the common ancestor and are marked as exceptions in the child version will remain as a result of the reconcile but will be covered by a dirty area. If the same error is deleted in the parent version, it will be deleted in the child version as a result of the reconcile.

- An exception that existed in the common ancestor and is marked as an error in the child version will remain an error as a result of the reconcile and will be covered by a dirty area. However, if the exception was deleted in the parent version, then it will remain deleted in the child version:



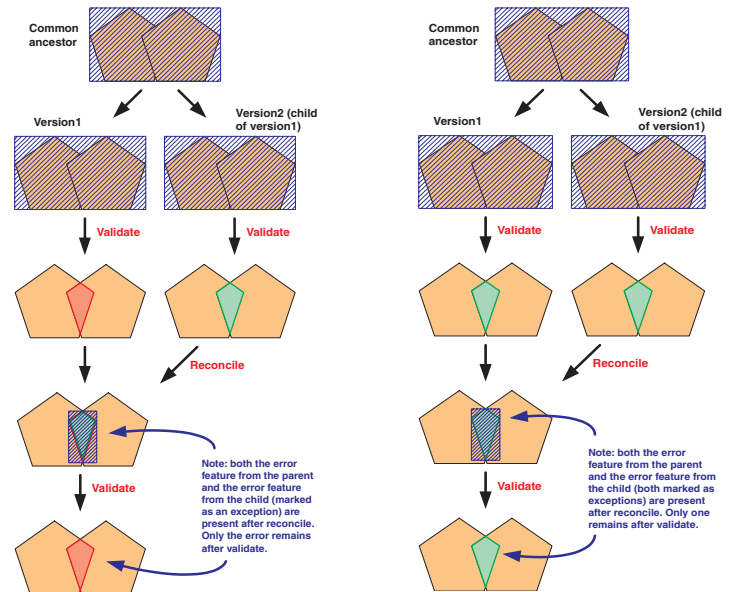
Exceptions that existed in the common ancestor and are marked as errors in the child version will remain as a result of the reconcile but will be covered by a dirty area. If the same exception is deleted in the parent version, it will be deleted in the child version as a result of the reconcile.

- An error or exception that existed in the common ancestor and is deleted in the child version will remain deleted as a result of the reconcile:



Errors and exceptions that existed in the common ancestor and are deleted in the child version will remain deleted as a result of the reconcile.

There are cases in which the same error can be created in both the parent and child versions by validating a dirty area that existed in the common ancestor. If this error is marked as an exception in either the parent or child version, the reconcile will result in duplicate error features. In these cases, the error features will be covered by a dirty area and will be reduced to a single error or exception when the dirty area is validated.

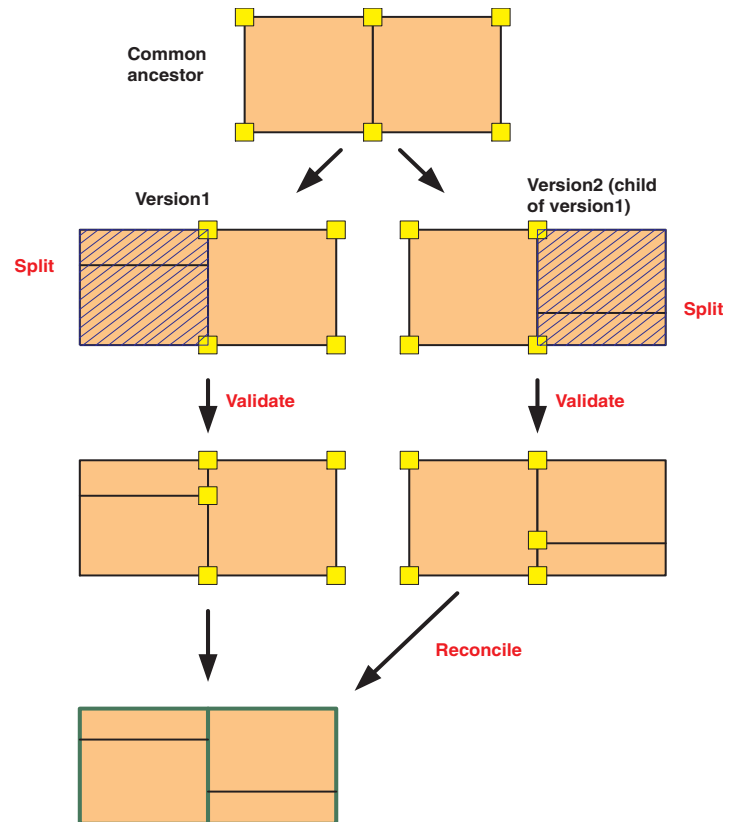


There are cases in which the same error can be discovered in both the parent and child. If the error is marked as an exception in the parent version, child version, or both, there will be duplicate errors as a result of the reconcile. Validating the resulting dirty area will eliminate the duplicate.

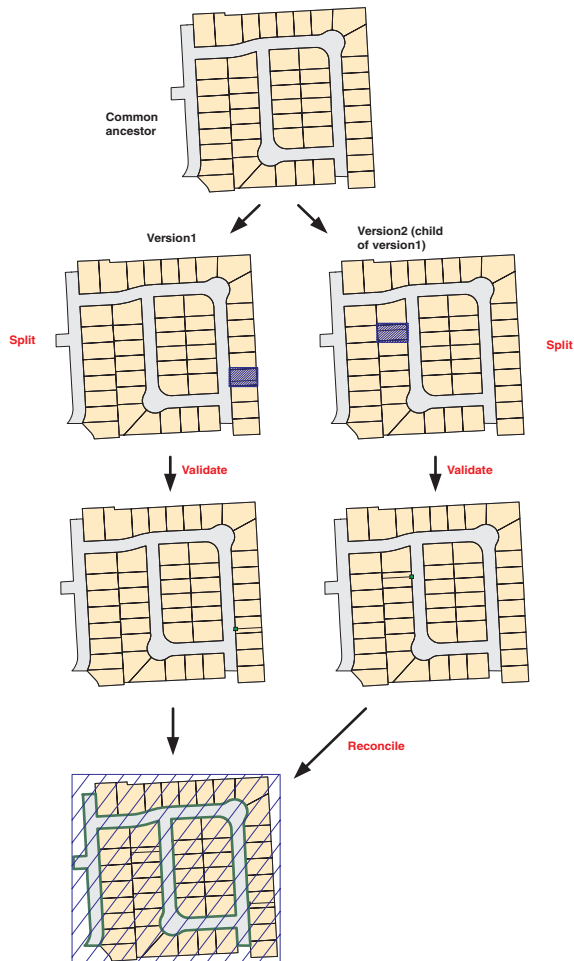
Conflicts and topology features

Features that participate in topology do not have any special behavior with respect to conflicts resulting from version reconciliation. If the same feature is edited in two separate versions and those versions are reconciled, they will be in conflict. The type of edits that are made to topologies, and the effect of clustering and cracking during the validation process, can result in a large number of conflicts. It is important to understand the potential for conflicts resulting from topological edits when you design your data model and your work flow.

The most common source of conflicts resulting from the validation process with topologies is the introduction of vertices due to the integration of features during the cracking and clustering phase of validation. The following examples illustrate how conflicts can result from the validate process.



Polygons that share edges in a topology in the parent version are inherited by the child version. One polygon is split in the parent version, an adjacent polygon is split in the child version, and the dirty areas are validated. Splitting the polygons deletes the original features and replaces each of them with two new ones. When the versions are reconciled, both original polygons are reported as update-delete conflicts. In other words, the feature that was deleted in the parent version was updated by the cracking and clustering process in the child version, and the feature that was deleted in the child version was updated by the cracking and clustering process in the parent version.



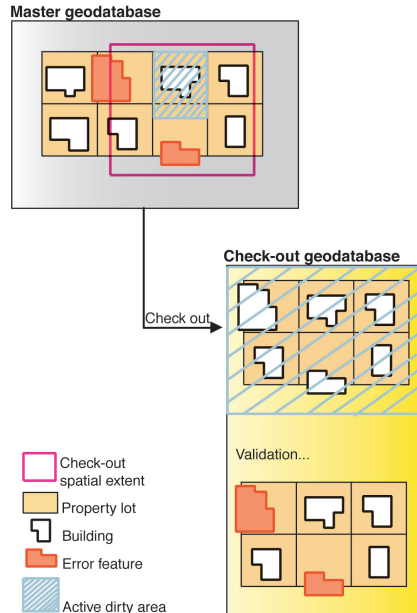
This illustrates another example where the introduction of vertices by the validate process can introduce conflicts in a land records database. In this case, the parcel polygons that are split share edges with a very large right-of-way polygon.

Conflicts like the ones described here can be avoided by structuring your work flow so that editors are working in different areas, or by using disconnected editing to control where users can and cannot edit. Additionally, data model design can help reduce the types of conflicts illustrated in the second example. These types of conflicts can be reduced by subdividing the right-of-way polygon into smaller chunks.

Topology and disconnected editing

As in the case of versioned databases, features in feature classes that participate in a topology do not have any special behavior for disconnected editing.

When a topology is checked out of a master geodatabase, all of the feature classes that participate in the topology are checked out as well. The entire extent of the checked out data is marked as dirty in the check-out geodatabase. An editor of the data in the check-out geodatabase can then validate the dirty area, make edits, and find and correct errors or mark errors as exceptions.



When the edited data is checked back into the master geodatabase, all of the feature classes are checked back in, and any differences between the checked-in data and the original version in the master are marked as dirty areas. These dirty areas

in the master geodatabase can then be validated to identify topology errors. Errors are not checked in to the master; they must be discovered by validating the checked-in area.

For more information about check-out geodatabases and disconnected editing, see the chapter 'Disconnected editing' in this book.



Subtypes and attribute domains

5

IN THIS CHAPTER

- **What are subtypes and attribute domains?**
- **Working with attribute domain properties**
- **Browsing the attribute domains of a geodatabase**
- **Creating new attribute domains**
- **Modifying and deleting attribute domains**
- **Associating default values and domains with tables and feature classes**
- **Creating subtypes**
- **Modifying and deleting subtypes**

When maintaining your geographic database, care must be taken to ensure that when you edit the data you do so in a manner that is consistent with the system you are modeling. The geodatabase together with the ArcMap Editor provides mechanisms to ensure that the data you store in your geodatabase is consistent with your data model.

The geodatabase has several data integrity and data management capabilities including validation rules, subtypes, relationship classes, geometric networks, and so on. Each of these capabilities and how you use them are covered throughout this book. This chapter describes the subtypes and the first class of validation rules—attribute domains.

Subtypes can be used to add finer control of spatial relationships in topologies, and in connectivity rules for geometric networks. See the chapter ‘Topology’ in this book for more information about using subtypes in topologies. See the chapter ‘Geometric networks’ in this book for more information about using subtypes in geometric networks.

ArcView can view feature classes that use advanced geodatabase functionality. To create or edit feature classes that take advantage of advanced geodatabase functionality, you need an ArcEditor or ArcInfo license.

What are subtypes and attribute domains?

The geodatabase stores objects. These objects may represent nonspatial real-world entities, such as manufacturers, or they may represent spatial objects such as pipes in a water network. Objects in the geodatabase are stored in feature classes (spatial) and tables (nonspatial).

The objects stored in a feature class or table are organized into subtypes and can have a set of validation rules associated with them. The ArcInfo system uses these validation rules to help you maintain a geodatabase that contains valid objects. This chapter outlines how to create subtypes for your feature classes and tables and how to establish attribute validation rules for the objects stored in them. These validation rules are distinct from the validation rules established in a topology.

Subtypes and validation rules

Tables and feature classes store objects of the same type—that is, that have the same behavior and attributes. For example, a feature class called *WaterMains* may store pressurized water mains. All of the water mains behave alike and have the attributes *ReferenceID*, *Depth*, *Material*, *GroundSurfaceType*, *Size*, and *PressureRating*.

You may be modeling a system in which water mains can only be made of cast iron, ductile iron, or copper. They can only be a certain size based on their type, and there are four possible ground surface types. When you create a new water main object using ArcMap, you may want these attributes to take on certain default values. Similarly, when ArcMap changes values for an attribute, you may want to make sure that only legal or valid values are inserted into the attributes for that lateral.

When an object in a feature class or table has valid values for all of its attributes, it is considered to be a valid object. If one of its attributes contains an invalid value, it is considered to be an invalid object. When designing your geodatabase, you can

specify what makes any particular object in a feature class or table a valid feature by establishing one or more validation rules.

In addition to topology rules, there are four broad classes of validation rules: attribute domains, connectivity rules, relationship rules, and custom rules. Connectivity rules are discussed further in the chapter ‘Geometric networks’ in this book; relationship rules are discussed further in the chapter ‘Defining relationship classes’ in this book; and custom rules are discussed further in *Exploring ArcObjects*. This chapter focuses on attribute domains.

Attribute domains are rules that describe the permissible values of a field type. Multiple feature classes and tables can share attribute domains stored in the database. This allows the water mains feature class to use the same domain for the ground surface type field as a feature class that stores water laterals.

Although all objects in a feature class or table must have the same behavior and attributes, not all objects must share the same attribute domains. For example, it may be true in a water network that only transmission water mains can have a pressure between 40 and 100 psi, while distribution water mains can have a pressure between 50 and 75 psi. You would use an attribute domain to enforce this restriction. To implement this kind of validation rule, you do not have to create separate feature classes for transmission and distribution water mains but you would want to distinguish these types of water mains from each other to establish a separate set of domains and default values. You can do this using subtypes.

Feature classes and tables can contain subtypes. An object’s subtype is determined by its subtype code value. The subtype code is stored in an integer field in the feature class or table. Each subtype can have its own set of default values and attribute domains for a given field, different connectivity rules, and different topology rules.

Geometry		Unique identifier		Attributes			Subtype code
Shape	ObjectID	ReferencID	GroundSurfaceType	Material	PressureRating	TypeCode	
	1	M1111-TM	1	CI	25	1	
	2	M1112-TM	1	CI	25	1	
	3	M1111-DM	3	CO	10	2	
	4	M1111-BM	2	CO	15	3	

Features in the geodatabase have behavior, geometry, a system-managed unique identifier, and attributes and can also have subtypes. Different subtypes can have different sets of valid values for their attributes.

When to use subtypes

An important geodatabase design issue arises when you must decide where it is appropriate to use subtypes and where additional feature classes are required. When you are trying to distinguish objects by their default values, attribute domains, connectivity rules, and relationship rules, it is recommended that you create separate subtypes for a single feature class or table.

You can also specify subtypes instead of feature classes in topology rules, which allows you to more precisely specify the spatial relationships that are appropriate for a given subtype.

When you want to distinguish objects based on different behaviors, attributes, access privileges, or whether the objects are multiversed (see the chapter ‘Working with a versioned geodatabase’ in this book), you must create additional feature classes.

Attribute domains

Attribute domains are used to constrain the values allowed in any particular attribute for a table, feature class, or subtype. Each feature class or table has a set of attribute domains that apply to

different attributes and/or subtypes. These attribute domains can be shared across feature classes and tables in a geodatabase.

There are two different types of attribute domains: range domains and coded value domains. Each domain has a name, a description, and a specific attribute type to which it can apply.

A range domain specifies a valid range of values for a numeric attribute. In the water mains example, you could have subtypes transmission, distribution, and bypass water mains. Distribution water mains can have a pressure between 50 and 75 psi. For a distribution water main object to be valid, its pressure value must be some value between 50 and 75 psi. A range domain specifies this range of values.

A coded value domain can apply to any type of attribute—text, numeric, date, and so on. Coded value domains specify a valid set of values for an attribute. The GroundSurfaceType field on the water mains feature class stores the type of material above the water main. Water mains may be buried under different types of surfaces: pavement, gravel, sand, or none (for exposed water mains). The coded value domain includes both the actual value that is stored in the database (for example, 1 for pavement) and a more user-friendly description of what that value actually means.

When editing your feature classes and tables, you can enforce these rules by validating individual or sets of objects. For details on editing objects with subtypes and validation rules, see *Editing in ArcMap*.

Attribute domains do not have a property that allows or disallows null values in an associated field. When a table or feature class is created in a geodatabase, each field has a property that indicates whether or not null values are permissible values. The database

itself will not permit null values to be inserted into columns that do not support them. Therefore, all domains treat null values as valid values.

Splitting and merging features

Often when editing data, a single feature is split into two features, or two separate features are combined, or merged, into a single feature. For example, in a land base database, a land parcel may be split into two separate land parcels due to rezoning. Similar zoning changes may require two adjacent parcels to be merged into a single parcel.

While the results of these types of edit operations on the feature's geometry are easily predictable, their effects on the attribute values are not. The behavior of an attribute's values when a feature is split is controlled by its *split policy*. When two features are merged, an attribute's value is controlled by its *merge policy*.

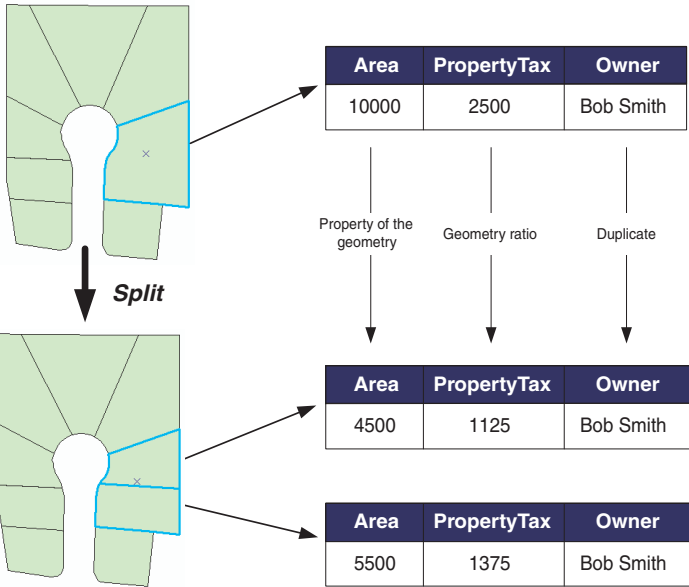
Each attribute domain has both a split policy and a merge policy. When a feature is split or merged, the ArcInfo system looks to these policies to determine what values the resulting feature(s) have for a particular attribute.

An attribute for any given table, feature class, or subtype can have one of three split policies that control the value of an attribute in the output object:

- Default value: the attribute of the two resulting features takes on the default value for the attribute of the given feature class or subtype.
- Duplicate: the attribute of the two resulting features takes on a copy of the original object's attribute value.
- Geometry ratio: the attribute of resulting features is a ratio of the original feature's value. The ratio is based on the ratio in

which the original geometry is divided. If the geometry is divided equally, each new feature's attribute gets half of the value of the original object's attribute. Geometry ratio policies only apply to domains for numeric field types.

In the parcel example, when a parcel is split, the Area attribute is automatically assigned as a property of the resulting geometry. The value for Owner is copied to the new objects (in this database, splitting a parcel does not affect its ownership). The PropertyTax is calculated based on the area, or size, of a parcel. To calculate the PropertyTax for each of the new objects, the split policy divides the PropertyTax of the original parcel proportionally among the new features according to their area.

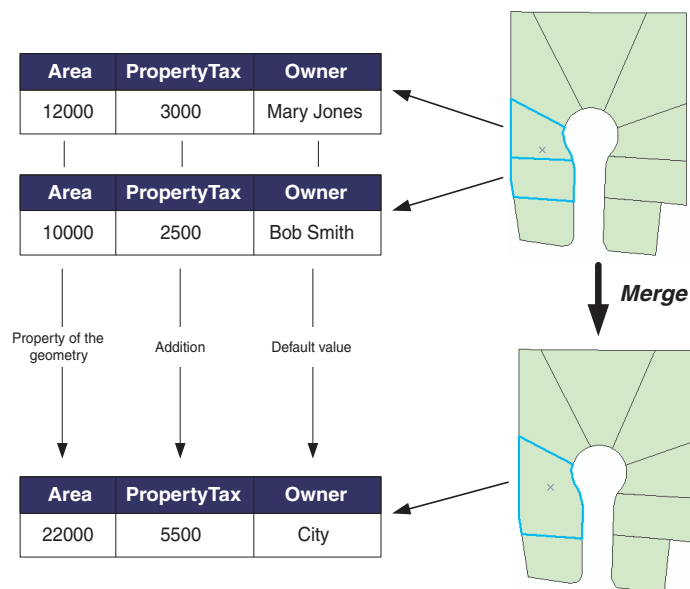


This example shows how split policies can be applied to attributes of a parcel object.

When two features are merged into a single feature, merge policies control the value of attributes in the new feature. An attribute for any given feature class or subtype can have one of three merge policies:

- **Default value:** the attribute of the resulting feature takes on the default value for the attribute of the given feature class or subtype. This is the only merge policy that applies to nonnumeric fields and coded value domains.
- **Sum values:** the attribute of the resulting feature takes on the sum of the values from the original features' attribute.
- **Geometry weighted:** the attribute of the resulting feature is the weighted average of the values of the attribute from the original features. This average is based on the original features' geometry.

In the parcel example, when two parcels are merged, the Area attribute is automatically assigned as a property of the resulting geometry. Owner is assigned its default value. As the PropertyTax for the merged feature is the sum of the original feature's PropertyTax, its merge policy is to sum the values.



This example shows how merge policies can be applied to attributes of a Parcel object.

This chapter shows you how to use ArcCatalog to create attribute domains for a geodatabase and how to create subtypes for a feature class or table. It then discusses how to create an attribute validation rule by associating an attribute domain to an attribute for a table, feature class, or subtype.

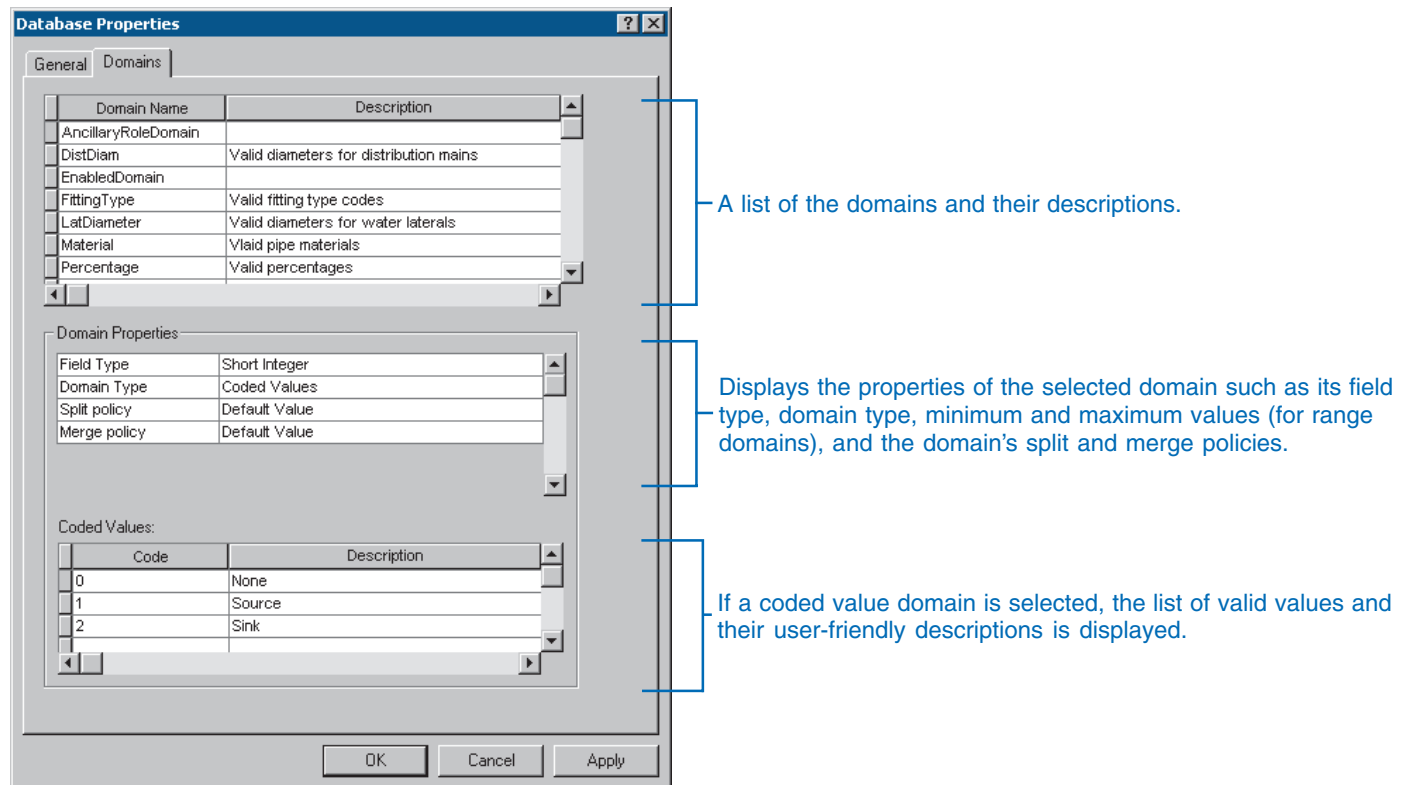
Schema locking

An exclusive lock is required on a feature class or table to modify its subtypes, default values, and attribute domains. For more information on exclusive locks and schema locking, see the chapter 'Creating new items in a geodatabase' in this book.

Working with attribute domain properties

The Domains tab of the Database Properties dialog box includes a list of all the domains that exist in a geodatabase. Each domain's name, description, properties, and valid set of values are displayed.

From this dialog box, you can also add, remove, and modify domains. An explanation of how to use this property page to manage your geodatabase's domains comes later in this chapter.



Browsing the attribute domains of a geodatabase

Attribute domains are stored geodatabase-wide. Once a user creates a new attribute domain, that user and all other users can view the properties of that domain and use the domain in a feature class or table.

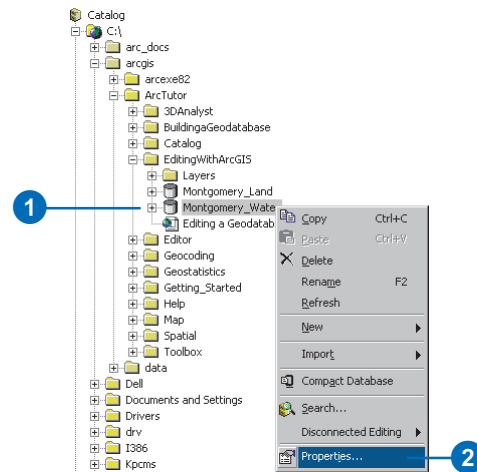
Attribute domains are managed using the Domains tab on the Database Properties dialog box. This dialog box can be displayed as part of the geodatabase's properties or from the feature class or table properties dialog box.

Attribute domains can be added, deleted, and modified with the Domain properties dialog box.

Browsing the domains of a personal geodatabase

1. Right-click the personal geodatabase, in the ArcCatalog tree, whose domains you want to browse.
2. Click Properties.

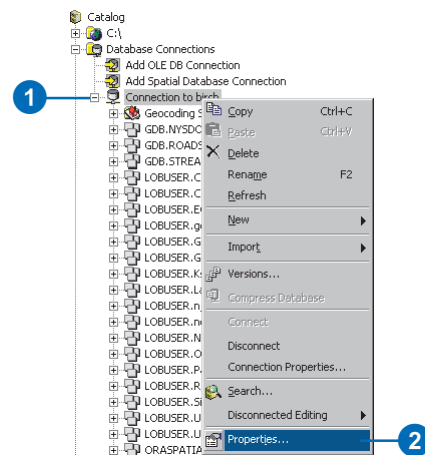
The Database Properties dialog box appears.



Browsing the domains of an ArcSDE geodatabase

1. Right-click the ArcSDE connection, in the ArcCatalog tree, for the geodatabase whose domains you want to browse.
2. Click Properties.

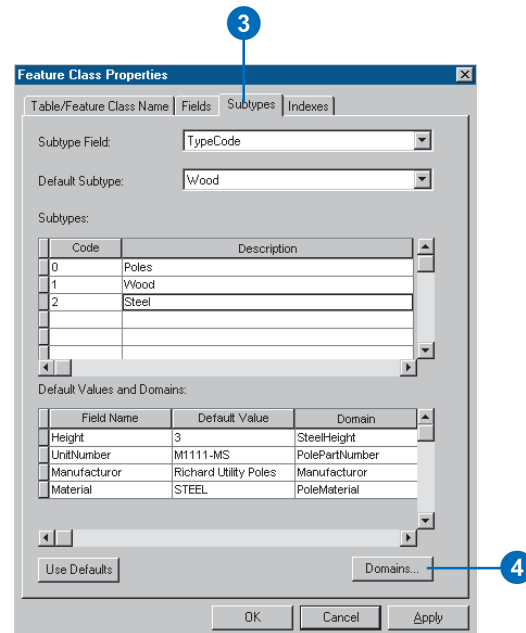
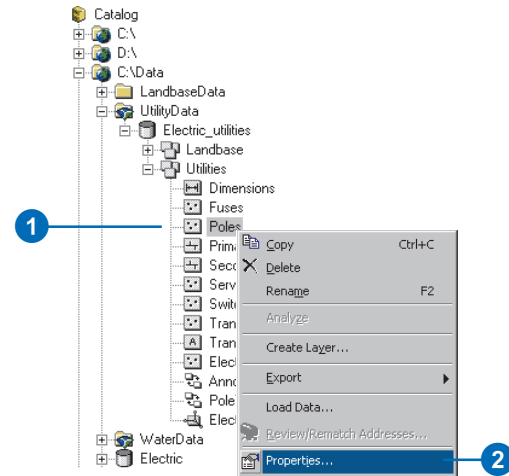
The Database Properties dialog box appears.



Browsing the attribute domains of a geodatabase from a feature class or table

1. Right-click a feature class or table in the ArcCatalog tree.
2. Click Properties.
3. Click the Subtypes tab.
4. Click Domains.

The Database Properties dialog box appears.



Creating new attribute domains

At any time in the life of a geodatabase, a new attribute domain can be created using the Domains tab on the Database Properties dialog box.

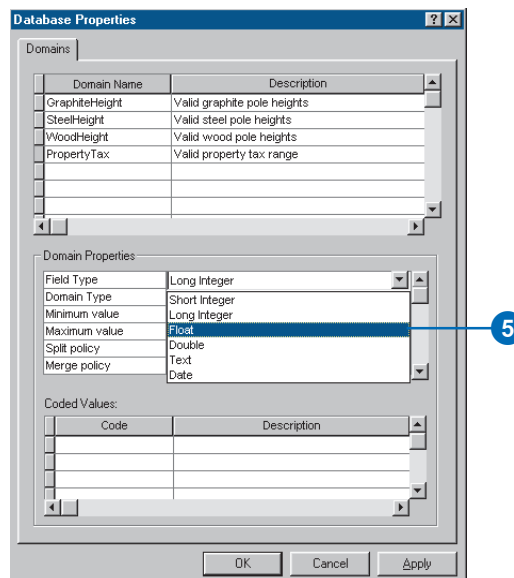
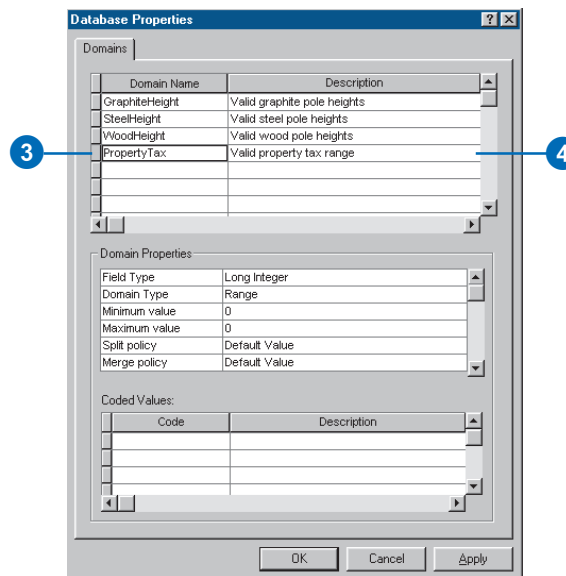
You can create new attribute range domains or coded value domains.

See Also

To learn how to apply an attribute domain to a field in a feature class or table, see 'Associating default values and domains with tables and feature classes' in this chapter.

Creating a new attribute range domain

1. Right-click the geodatabase in the ArcCatalog tree and click Properties.
2. Click the Domains tab.
3. Click the first empty field under Domain Name and type a name for the new domain.
4. Press the Tab key or click the new domain's Description field and type a description for the domain.
5. Click the field next to Field Type in Domain Properties, click the dropdown arrow, and click the type of attribute field to which this domain will be applied. ►



Tip

Range domains

Range domains can't be created for text fields. They can only be applied to numeric and date fields.

- Click the field next to Domain Type, click the dropdown arrow, and click Range from the list of domain types.
- Click the field with the minimum value for the range domain and type the minimum value. Do the same for the maximum value.
- Click the field next to Split policy, click the dropdown arrow, and click the split policy for the new domain. Do the same for the merge policy.
- Click Apply to create the new domain in the geodatabase or OK to create the domain and close the dialog box.

Database Properties

Domains

Domain Name	Description
GraphiteHeight	Valid graphite pole heights
SteelHeight	Valid steel pole heights
WoodHeight	Valid wood pole heights
PropertyTax	Valid property tax range

Domain Properties

Field Type: Float

Domain Type: Range

Minimum value: Range

Maximum value: Coded Values

Split policy: Default Value

Merge policy: Default Value

Coded Values:

Code	Description
------	-------------

OK Cancel Apply

Database Properties

Domains

Domain Name	Description
GraphiteHeight	Valid graphite pole heights
SteelHeight	Valid steel pole heights
WoodHeight	Valid wood pole heights
PropertyTax	Valid property tax range

Domain Properties

Field Type: Float

Domain Type: Range

Minimum value: 500

Maximum value: 1000

Split policy: Geometry Ratio

Merge policy: Default Value

Coded Values:

Code	Description
Default Value	
Sum Values	

OK Cancel Apply

Tip

Coded value descriptions

When you add a new value to a domain's coded value list, you must also add a more user-friendly description. When you edit an attribute value for a field that has this domain, the user-friendly values appear in the ArcMap Editor. Descriptions help you select the right value.

Tip

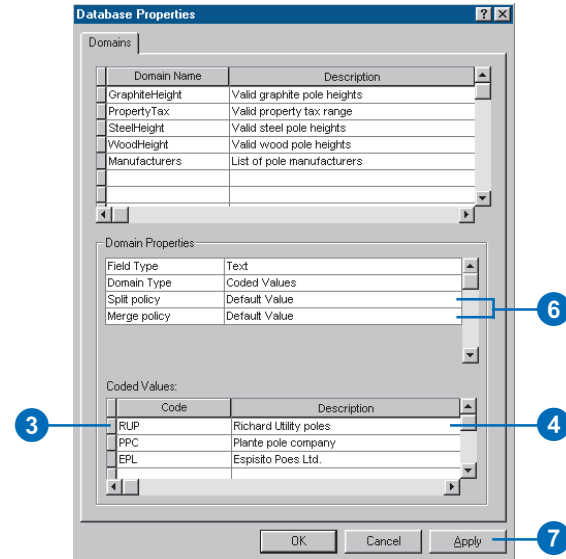
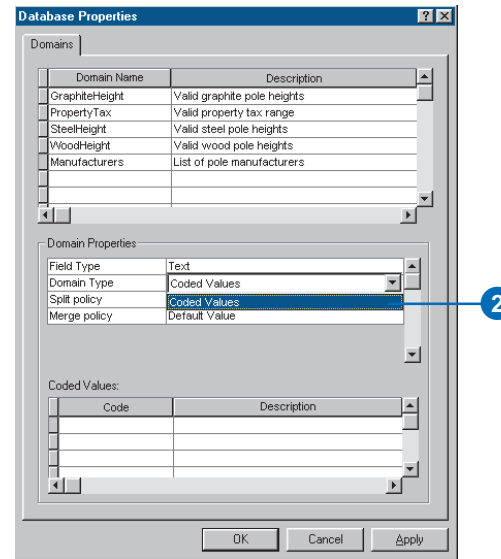
Coded value split/merge policies

Coded value domains support only default value and duplicate split policies.

Coded value domains support the default value merge policy only.

Creating a new coded value domain

1. Follow steps 1 through 4 for 'Creating a new attribute range domain'.
2. Click the field next to Domain Type, click the dropdown arrow, and click Coded Values from the list of domain types.
3. Click the first empty field under Coded Values and type the first valid code.
4. Press the Tab key or click the new coded value's Description field. Type a user-friendly description for this coded value.
5. Repeat steps 3 and 4 until all valid values and their descriptions have been typed.
6. Click the field next to Split policy, click the dropdown arrow, and click the split policy for the new domain. Do the same for the merge policy.
7. Click Apply to create the new domain in the geodatabase or OK to create the domain and close the dialog box.



Modifying and deleting attribute domains

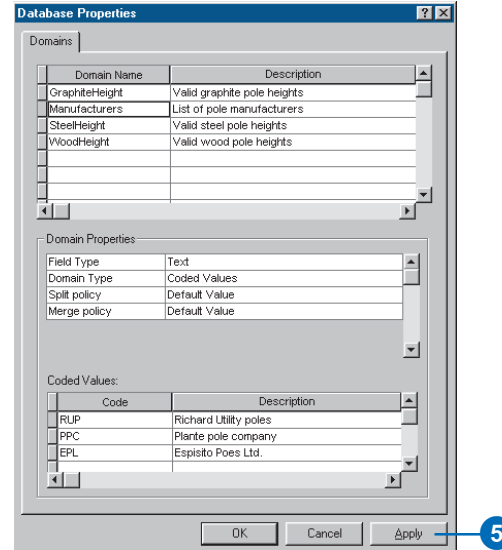
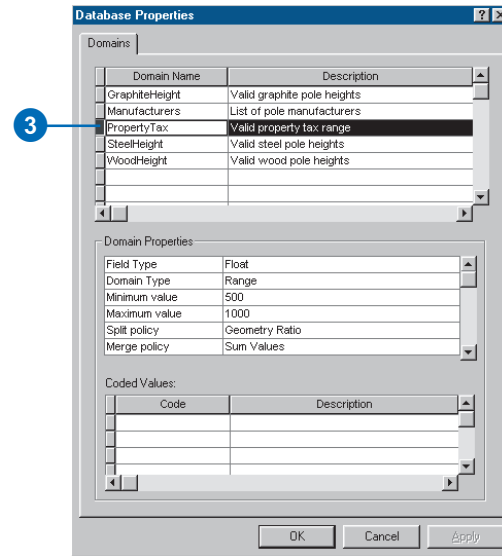
The Domains property page can be used to delete an attribute domain from the geodatabase or to modify an existing domain.

When a new domain is created, the owner of that domain—that is, the user who created it—is recorded. Only the owner of an attribute domain can delete or modify it.

As you will see later in this chapter, domains can be associated with particular fields for a feature class or table or for a subtype of a feature class or table. While a domain is being used by a table or feature class, it cannot be deleted or modified.

You can modify domains simply by selecting them on the domains properties dialog box and changing anything from their name to their type and valid values. This is done in the same manner as when you create a new domain.

1. Right-click the geodatabase in the ArcCatalog tree and click Properties.
2. Click the Domains tab.
3. Click the domain you want to delete by clicking the left-hand tab in the grid.
4. Press the Delete key.
5. Click Apply to delete the domain from the geodatabase or OK to delete the domain and close the dialog box.



Associating default values and domains with tables and feature classes

Once you have created an attribute domain, or domains, you can associate them and their default values with fields in a table or feature class. Once a domain is associated with a feature class or table, an attribute validation rule is created in the database.

The same attribute domain can be associated with multiple fields of the same table, feature class, or subtype and can be associated with multiple fields in multiple tables and feature classes.

Tip

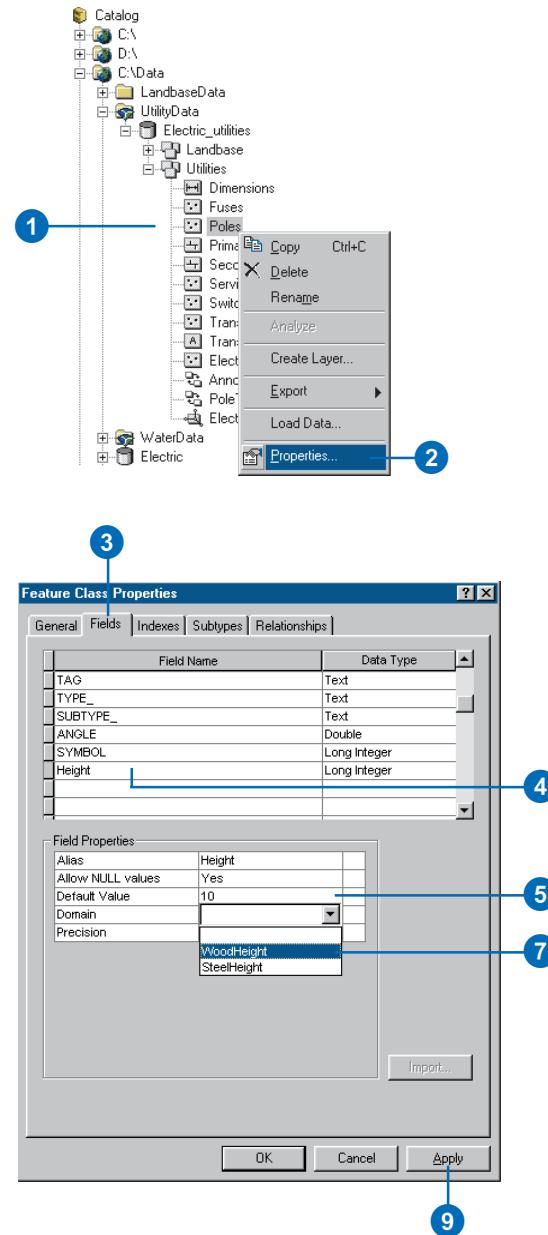
Subtypes

Not all of the objects in a table or feature class must have the same domains or default values applied to the same fields. To apply different domains and default values to the same field in a single table or feature class, you must create subtypes.

You'll learn how to create subtypes and associate domains and default values to a subtype's fields later in this chapter.

1. Right-click the table or feature class in the ArcCatalog tree with which you want to associate domains.
2. Click Properties.
3. Click the Fields tab.
4. Click the field for which you want to create a default value and associate a domain.
5. Click the field next to Default Value and type the default value.
6. Skip to step 9 if you don't want to associate a domain to the field.
7. Click the field next to Domain, click the dropdown arrow, and click the domain you want to associate with the field.
8. Repeat steps 4 through 7 until you have associated default values and domains for all fields that you want to have these properties.
9. Click Apply.

Only those domains that apply to the field type are displayed in the list.



Creating subtypes

You can use ArcCatalog to add subtypes and to set default values and attribute domains for the fields of each subtype.

You can manage subtypes using the properties dialog box for each table or feature class. You can define the subtype field, add new subtypes, and remove or modify existing subtypes.

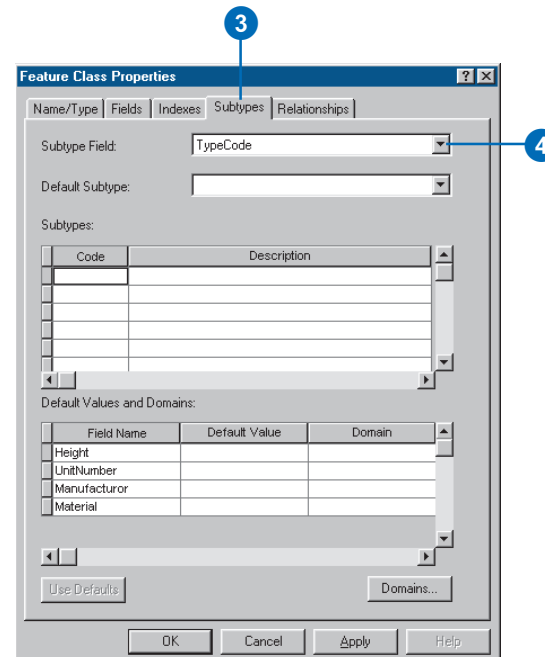
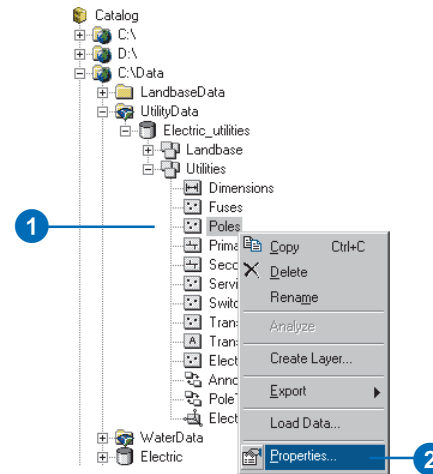
Tip

Subtype field

The subtype field must be a long integer or short integer field. If you have no subtype field selected, you will not be able to add subtypes.

Creating new subtypes for a feature class or table

1. Right-click the feature class or table in the ArcCatalog tree to which you want to add subtypes.
2. Click Properties.
3. Click the Subtypes tab.
4. Click the dropdown arrow and click the subtype field from the list of available long integer and short integer fields. ►



Tip

The default subtype

The default subtype serves two purposes. When you create a new subtype, click *Use Defaults* and the subtype will inherit all of the default values and domains for its fields from the default subtype. These can then be modified to meet the requirements for the new subtype. As you add additional subtypes, the default subtype can be changed at any time.

When you create a new feature in the feature class without specifying a subtype, it will automatically be assigned the default subtype.

- Click the first empty field under Code and type an integer value that will be the code for that subtype to add a new subtype.
 - Press the Tab key or click the Description field and type a description for the subtype.
 - Type a default value in the appropriate field in the table for each field.
 - Click the Domain field, click the dropdown arrow, and click the domain from the list of domains to associate an attribute domain with a field for the new subtype.
- Only those domains that apply to the field type are displayed in the list.
- Click the dropdown arrow and click it from the list of subtypes to set this subtype as the default subtype. ►

Feature Class Properties

Table/Feature Class Name | Fields | Subtypes | Indexes

Subtype Field: TypeCode

Default Subtype: Poles

Subtypes:

Code	Description
0	Poles
1	Wood

Default Values and Domains:

Field Name	Default Value	Domain
Height	3.0	WoodHeight
UnitNumber	M1111-Mw	PolePartNumber
Manufacturer	Richard Utility Poles	Manufacturer
Material	WOOD	Material

Use Defaults

OK Cancel Apply

Feature Class Properties

Table/Feature Class Name | Fields | Subtypes | Indexes

Subtype Field: TypeCode

Default Subtype: Poles

Subtypes:

Code	Description
0	Poles
1	Wood

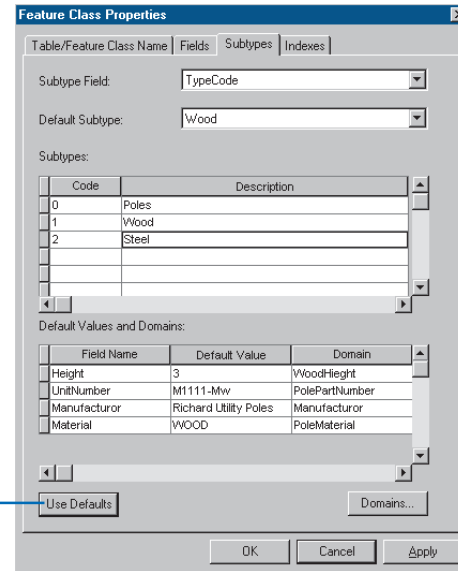
Default Values and Domains:

Field Name	Default Value	Domain
Height	3.0	WoodHeight
UnitNumber	M1111-Mw	PolePartNumber
Manufacturer	Richard Utility Poles	Manufacturer
Material	WOOD	PoleMaterial

Use Defaults Domains...

OK Cancel Apply

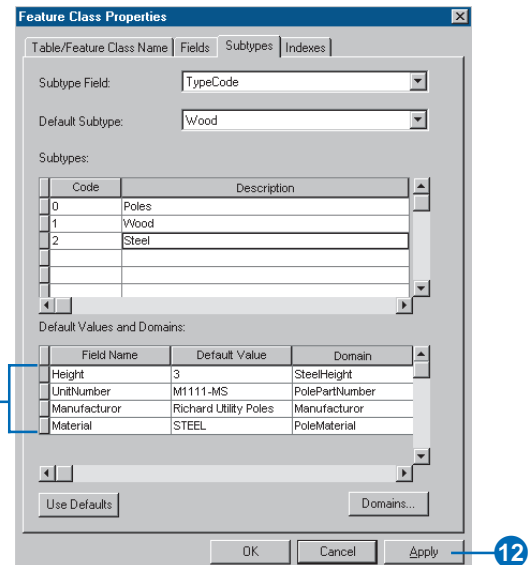
10. Repeat steps 5 through 8 to add additional subtypes. You can reset the default subtype at any time.
11. Click Use Defaults to have your new subtype take all of the default values and domains from the default subtype when adding a new subtype. You can then modify all or some of these.
12. Click Apply to create the new subtypes in the geodatabase or OK to create the subtypes and close the dialog box when you are finished creating your subtypes and have selected the default subtype.



The dialog box shows the 'Subtypes' tab. The 'Subtypes' table has three rows: 0 Poles, 1 Wood, and 2 Steel. The 'Default Values and Domains' table is as follows:

Field Name	Default Value	Domain
Height	3	WoodHeight
UnitNumber	M1111-Mw	PolePartNumber
Manufacturer	Richard Utility Poles	Manufacturer
Material	WOOD	PoleMaterial

A blue circle with the number 11 is next to the 'Use Defaults' button.



The dialog box shows the 'Subtypes' tab. The 'Subtypes' table has three rows: 0 Poles, 1 Wood, and 2 Steel. The 'Default Values and Domains' table is as follows:

Field Name	Default Value	Domain
Height	3	SteelHeight
UnitNumber	M1111-MS	PolePartNumber
Manufacturer	Richard Utility Poles	Manufacturer
Material	STEEL	PoleMaterial

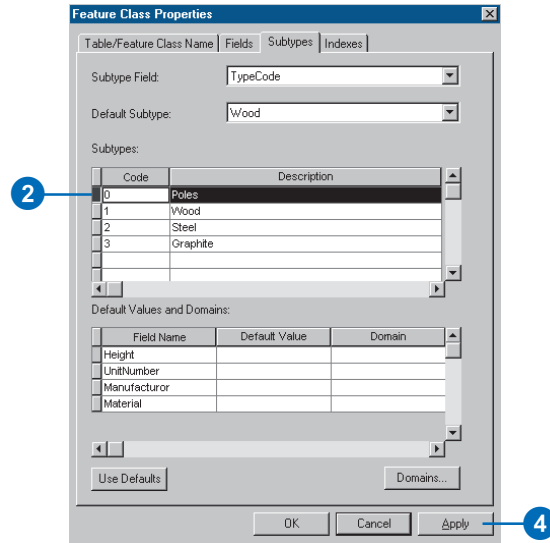
A blue circle with the number 12 is next to the 'Apply' button.

Modifying and deleting subtypes

Subtypes for a feature class or table can also be modified or deleted using the properties dialog box for the table or feature class. You can modify any aspect of a subtype including its description, its default values, and its domains. Modifying each aspect of a subtype is done the same way as creating a new subtype.

You cannot delete a subtype if it is currently referenced by a topology rule.

1. Follow steps 1 through 3 for 'Creating new subtypes for a feature class or table'.
2. Click the left-hand tab next to the subtype you want to delete.
3. Press the Delete key.
4. Click Apply to delete the subtype from the geodatabase or OK to delete the subtype and close the dialog box.



Defining relationship classes

6

IN THIS CHAPTER

- What is a relationship class?
- Relationship classes in ArcCatalog and ArcMap
- Creating a simple relationship class
- Creating a composite relationship class
- Creating an attributed relationship class
- Creating relationship rules
- Managing relationship classes
- Exploring related objects in ArcMap
- Using related fields in ArcMap

Objects in a real-world system often have particular associations with other objects in the database. These kinds of associations between objects in the geodatabase are called relationships. Relationships can exist between spatial objects (features in feature classes), nonspatial objects (rows in a table), or spatial and nonspatial objects. While spatial objects are stored in the geodatabase in feature classes and nonspatial objects are stored in tables, relationships are stored in relationship classes.

ArcCatalog contains tools to create, modify, and manage relationship classes in your geodatabase, while ArcMap provides tools to create, delete, and use relationships to find objects that are associated with other objects in the geodatabase. This chapter describes how to use ArcCatalog to manage these relationship classes and how to use relationships in ArcMap. *Editing in ArcMap* discusses how to create and delete relationships.

ArcView can view feature classes that use advanced geodatabase functionality. To create or edit feature classes that take advantage of advanced geodatabase functionality, you need an ArcEditor or ArcInfo license.

What is a relationship class?

Objects in a real-world system, such as an electrical network or a parcel database, often have particular associations with other objects in the database. For example, in an electrical network, poles support transformers. In a parcel database, a parcel will have one or many owners.

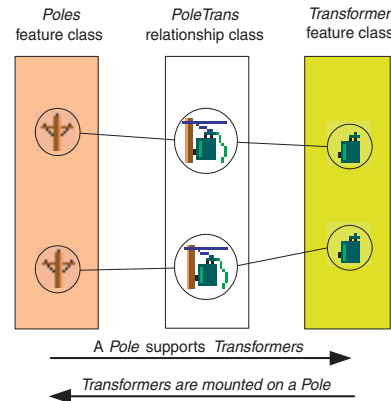
These kinds of associations between objects in the geodatabase are called relationships. Relationships can exist between spatial objects (features in feature classes), nonspatial objects (rows in a table), or spatial and nonspatial objects. While spatial objects are stored in the geodatabase in feature classes and nonspatial objects are stored in tables, relationships are stored in relationship classes.

To store relationships, such as between electric transformers and poles, you must create a relationship class. If, in your geodatabase, transformers also have relationships to transformer attribute objects, then a second relationship class is required to store those relationships.

The anatomy of a relationship

As with any association, relationships have particular characteristics. One obvious characteristic is the notion of cardinality. Cardinality describes how many objects of one type are related to an object of another type. In the transformer–pole example, a single pole may support more than one transformer, but a transformer can only be mounted on a single pole. The relationship between transformers and poles is one to many: one pole, which is an object in the origin class of the relationship, to many transformers, which is an object in the destination class of the relationship.

In general, relationships can have one-to-one, one-to-many, many-to-one, and many-to-many cardinalities. As you will see later in this chapter, certain types of relationships support certain

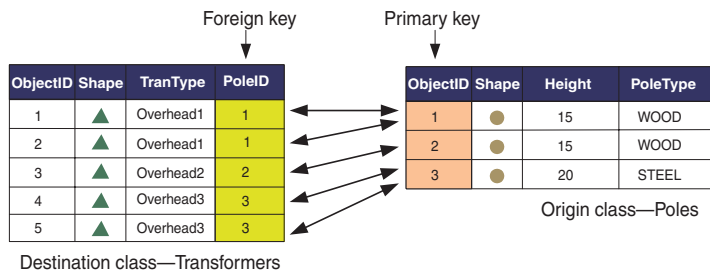


A relationship is an association between two or more objects in two feature classes or tables in a geodatabase. Relationships are stored in relationship classes.

cardinalities, and you can control cardinalities for any relationship class when you define relationship rules.

A relationship between two objects is maintained through attribute values for key fields. In the transformer–pole example, the unit number of the pole that supports the transformer may be included in the attributes of the transformer object. This is referred to as an embedded foreign key. It tells us what object in the pole feature class this particular transformer is related to.

Relationship classes can have attributes. Any relationship class that has attributes must be stored as a table in the database and have a pair of foreign keys, each referencing the origin and destination classes of the relationship class. In this case, each relationship is stored as a row in the relationship classes table.



A relationship between two objects is maintained through attribute values for key fields.

Similarly, any many-to-many relationship classes require a table in the database to store at least the foreign keys.

Relationship classes have path labels. Forward and backward path labels describe the relationship when navigating from one object to another. The forward path label describes the relationship navigated from the origin class to the destination class; the backward path label describes the relationship when navigating from the destination to the origin class. In the transformer–pole example, when navigating from the transformer to the pole, the relationship path label may be “is mounted on”. The same relationship, when navigated from the pole to the transformer, may have a path label of “supports”.

Relationship classes can also be used to propagate standard messages between related objects. Messaging is the mechanism that objects related to each other use to notify each other when they change. For example, in a relationship between poles and transformers, when the pole is deleted, it sends a message to its related transformer objects informing them it was deleted. As transformers can’t exist without a pole to support them, these transformer objects could respond to the message by deleting themselves.

The geodatabase supports two kinds of relationships: simple, or peer-to-peer, relationships and composite relationships. Each is discussed briefly below.

Simple relationships

Simple, or peer-to-peer, relationships are relationships between two or more objects in the database that exist independently of each other.

In a relationship between object A and object B, if object A is deleted from the database, object B continues to exist. For example, in a railroad network you may have railroad crossings that have one or more related signal lamps. However, a railroad crossing can exist without a signal lamp, and signal lamps exist on the railroad network where there are no railroad crossings.

Simple relationships can have one-to-one, one-to-many, or many-to-many cardinality.

Composite relationships

The geodatabase also supports the notion of a composite relationship, where the lifetime of one object controls the lifetime of its related objects. The pole–transformer relationship is an example of a composite relationship. Once a pole is deleted, a delete message is sent to its related transformers, which are deleted from the transformer’s feature class.

Composite relationships are always one-to-many but can be constrained to be one-to-one using relationship rules.

Attributed relationship classes

One-to-one and one-to-many relationship classes do not require a new table in the geodatabase to be created to store the relationships. However, many-to-many relationship classes do require a

new table in the database to store the foreign keys from the origin and destination classes to make the relationship. This table can also have other fields to store attributes of the relationship itself that are not attributes of either the origin or destination class.

For example, in a parcel database you may have a relationship class between parcels and owners, where owners “own” parcels and parcels are “owned by” owners. An attribute of that relationship may be the percentage of ownership.

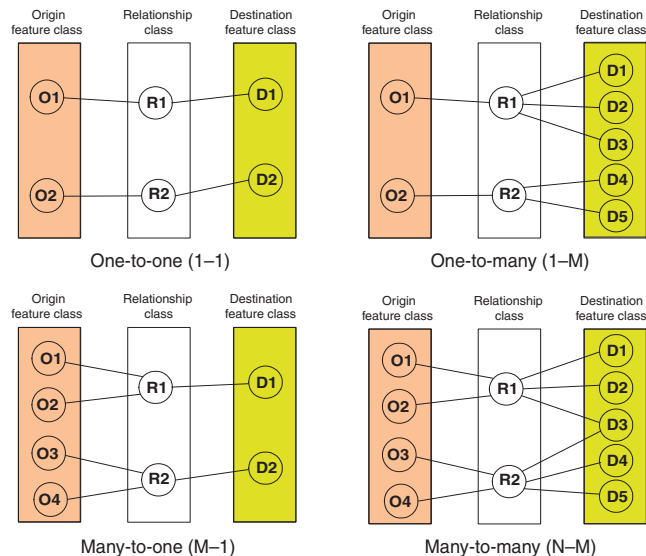
One-to-one and one-to-many relationship classes may also have attributes; in this case, a table would be created to store the relationships.

Relationship rules

Relationship classes can have an associated set of relationship rules. Relationship rules control which object *subtypes* from the origin class can be related to which object subtypes in the destination class. They can also be used to specify a valid cardinality range for all permissible subtype pairs.

For example, the subtype wood pole may be able to support from 0 to 3 transformers, whereas the subtype steel pole may support 0 to 5 transformers. In the first case, the cardinality range would be 0–3; in the second case, it would be 0–5.

You can establish a relationship between two or more objects in the geodatabase by using tools available in ArcMap. Once the relationship is established, use ArcMap tools to navigate it and symbolize features based on attributes in a related object. You



Relationships have cardinality. Cardinality describes how many objects of type A are associated with objects of type B. Relationships can have 1–1, 1–M, M–1, or N–M cardinality.

can find all objects that have a relationship with a particular object through any particular relationship class.

Performance considerations

When editing composite features, edits, such as move, rotate, and delete, also affect the related objects through the relationship class. There is a cost when navigating these relationships. The cost is minimized when indexes are maintained for the primary and foreign keys for the relationship class. When a new relationship class is created with ArcCatalog, the primary and foreign keys are automatically indexed if they do not already have indexes.

It is important to realize that when a feature class participates in a relationship class, that feature class utilizes messaging. When editing that feature class in ArcMap, the related class must be opened so it can respond to the message—either by moving, deleting itself, or implementing some *custom behavior*. If the related class is not already in the map you are working with, it will automatically be opened to respond to the message, then closed. Each time it responds to a message, it will need to be reopened.

In general, when working with a class in ArcMap, have all related classes also in the map. This way, the related classes are opened once when they are added to ArcMap. If they are not in the map, then each time you access related objects, the class must be opened.

With many ArcInfo coverage data models, the feature–attribute table contained as few items as possible, and many of the attributes for a feature class were contained in a related table. This can be done with geodatabase feature classes; however, navigating a relationship in the geodatabase is a more costly operation than navigating relates in INFO. In the INFO environment, it was common to store the *symbolology* for a feature in an external, related table called a lookup table. This can still be done in the geodatabase using relationship classes; however, for large data, symbolizing this way will be slow, even with indexes on the primary and foreign keys. Try to keep attributes for symbolization on the feature class’s table. For performance considerations, it is recommended that symbolology information be stored in the feature class itself.

Schema locking

An exclusive lock is required when modifying a relationship class’s relationship rules or when renaming or deleting a relationship class. An exclusive lock can only be acquired for a relationship class if the feature classes or tables that participate in the

relationship can also be locked. Therefore, if a user has an exclusive or shared lock on either the origin, destination, or both classes, the properties of the relationship class cannot be edited.

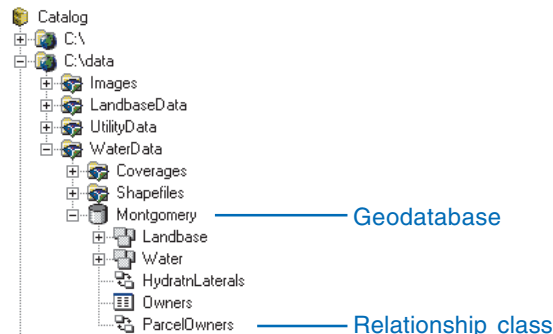
For more information on exclusive locks and schema locking, see the chapter ‘Creating new items in a geodatabase’ in this book.

Relationship classes in ArcCatalog and ArcMap

Relationship classes in ArcCatalog

In ArcCatalog, you can automatically work with a relationship class in a geodatabase. Relationship classes can exist both inside feature datasets and at the root level of the geodatabase.

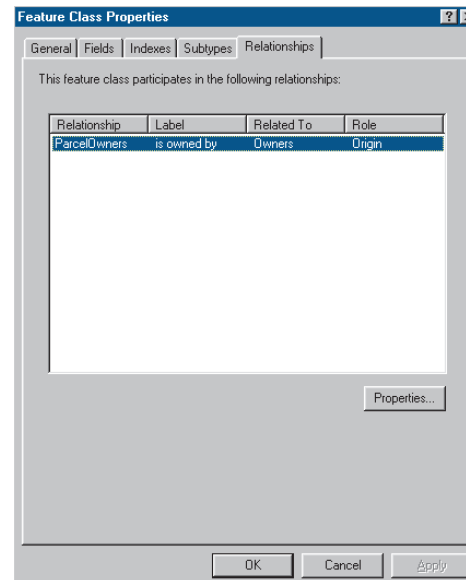
When you look at a particular relationship class in the ArcCatalog tree, it is not immediately evident how feature classes or tables are related. However, by examining the properties of both the feature class or table and the relationship class, you can achieve a clear picture of this.



Relationship classes appear in the ArcCatalog tree either at the geodatabase level or inside a feature dataset.

In the Feature Class Properties or Table Properties dialog box, the Relationships tab displays the relationship classes, if any, in which a feature class or table participates. For each relationship class, the Catalog tree displays the path label, the other feature class or table, and its role in the relationship. You can click Properties to view the properties for the selected relationship class.

The properties dialog box for each relationship class—whether opened from the Feature Class Properties or Table Properties



By clicking the Relationships tab on the properties dialog box for a feature class or table, you can view what relationship classes the feature class or table participates in along with what role the feature class or table plays within the relationship class. To get more details about the relationship class, click Properties.

dialog box tree—contains more detailed information about the relationship class. It also lets you establish relationship rules. The procedure for creating and modifying relationship rules is discussed later in this chapter.

ArcCatalog also contains various tools to create, delete, and manage relationship classes. The tools will also be discussed in more detail in this chapter.



The Relationship Class Properties dialog box, whether opened from the feature class or table property page, contains detailed information about the relationship class.

Relationship classes in ArcMap

Once you have established a relationship class between feature classes or tables, you can use these relationships in ArcMap. For example, when you identify a feature in your map, you can see all of the objects related to that feature. When working with tables, you can select one or more rows or features and open the related table to see the selected related objects.

You may want to use fields on a related table or feature class to symbolize or label your map. Once you have added a feature class to your map that participates in a relationship class, you can do this by establishing a join between the feature class and its related feature class or table. You can use these joined fields like you use other fields in your feature layer.

For more information on maps, feature layers, symbolizing, and labeling your features, see *Using ArcMap*.

There are also a number of tools in ArcMap for editing relationships and related objects. For example, when you select a feature, you can edit the properties of its related objects. You can also use ArcMap to add new relationships and delete existing relationships. For more information on editing relationships, see *Editing in ArcMap*.

Deciding between relationship classes and joins and relates

Geodatabase relationship classes are usually created to establish an enduring business process relationship between a feature class and another table or feature class. ArcMap joins and relates are useful in data building, data exploration, or analysis.

Relationship classes have many advantages over joins and relates. The relationship class is stored with the data in the geodatabase. This makes the relationship class accessible to anyone who uses the geodatabase. A relationship class allows greater interaction among related objects when editing the feature classes that participate in the relationship class. Relationship classes allow you to build behavior into the relationship. For example, the deletion or modification of one feature could delete or alter another related feature.

You may find, however, that a join or relate will perform the desired function. Joins and relates are independent of the geodatabase. This offers several advantages. First, a join or relate can be performed by a user without influencing the data in the geodatabase. Further, joins and relates are stored with the map document and are not geodatabase specific. They can establish relationships among data found in different geodatabases or data that is not in a geodatabase. For more information on joins and relates, see *Using ArcMap*.

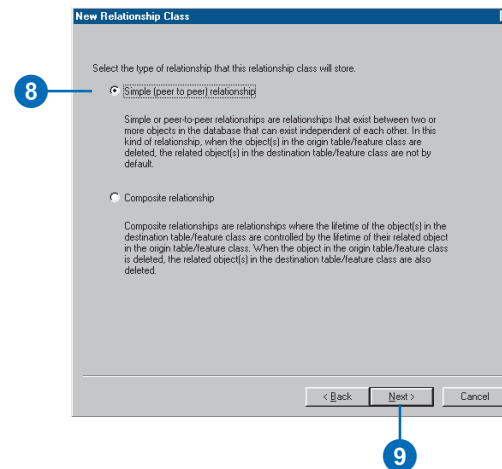
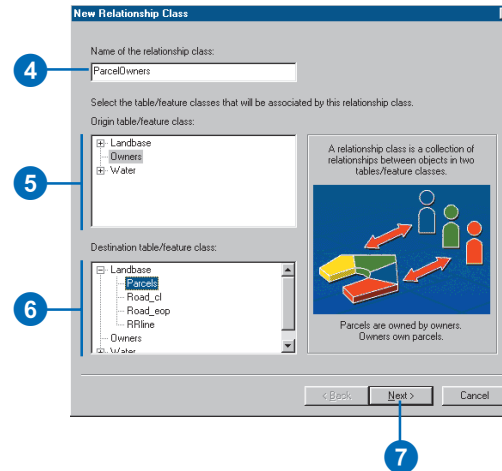
Creating a simple relationship class

You can create new relationship classes between any feature class or table within your geodatabase using tools in ArcCatalog. These tools can be used to create simple, composite, and attributed relationship classes.

Relationship classes appear in the Catalog tree, and you can inspect their properties as well as the relationships for any particular feature class.

The example in this task shows how to create a relationship class between a feature class that stores parcel objects and a table that stores owner objects. It is a simple, nonattributed relationship. In the database, a parcel can be owned by a single owner, and an owner can own a single parcel, so it is a one-to-one (1-1) relationship.

1. Right-click the geodatabase or feature dataset, in the ArcCatalog tree, in which you want to create the new relationship class.
2. Point to New.
3. Click Relationship Class.
4. Type the name for the new relationship class.
5. Click the origin table or feature class.
6. Click the destination table or feature class.
7. Click Next.
8. Click Simple (peer-to-peer) relationship.
9. Click Next. ►



Tip

M-N relationship classes

Many-to-many (M-N) relationship classes require the relationship class to have its own table in the database. You can optionally add attributes to this table or you can allow the ArcInfo system to manage the schema of the table for you.

Tip

Notification direction

By default, the notification direction for a simple relationship is None.

10. Type the forward and backward path labels.
11. Click the message notification direction.
12. Click Next.
13. Click the first cardinality option. In this example, an owner can own a single parcel and a parcel can be owned by a single owner, so this is a one-to-one (1-1) relationship.
14. Click Next. ►

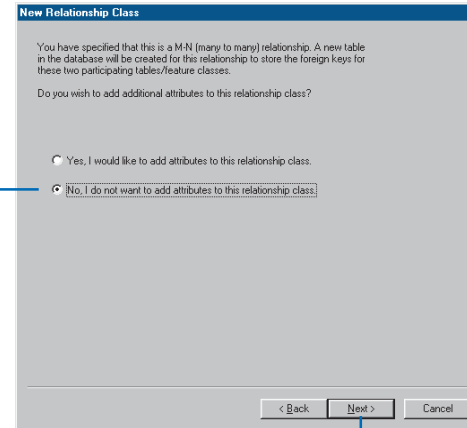
The screenshot shows the 'New Relationship Class' dialog box. It has two text input fields for labels: the first contains 'Owns' and the second contains 'is owned by'. Below these are three radio button options for message propagation direction: 'Forward (origin to destination)', 'Backward (destination to origin)', and 'Both'. The 'None (no messages propagated)' option is selected. At the bottom are '< Back', 'Next >', and 'Cancel' buttons. Blue callout circles with numbers 10, 11, and 12 point to the label fields, the direction options, and the 'Next >' button respectively.

The screenshot shows the 'New Relationship Class' dialog box at the cardinality selection step. It asks to 'Select the cardinality for this relationship class (origin - destination)'. There are three radio button options: '1 - 1 (one to one)', '1 - M (one to many)', and 'M - N (many to many)'. The '1 - 1 (one to one)' option is selected. A note states: 'Note: If this is a composite relationship class, then the cardinality must be 1-1 (one-to-one) or 1-M (one-to-many)'. To the right, a diagram illustrates a 1-1 relationship between 'Parcels Table/Feature Class' and 'Owners Table/Feature Class' using icons of a house and a person. At the bottom are '< Back', 'Next >', and 'Cancel' buttons. Blue callout circles with numbers 13 and 14 point to the '1 - 1 (one to one)' option and the 'Next >' button respectively.

See Also

In this first example, you are not adding attributes to your relationship class, although any relationship class can have attributes. For more information on how to create an attributed relationship class, see 'Creating an attributed relationship class' in this chapter.

15. Click No. The relationship class does not require attributes in this example.
16. Click Next.
17. Click the dropdown arrow to see a list of fields from the origin table or feature class. Click the primary key for this feature class or table.
18. Click the dropdown arrow to see a list of fields from the destination table or feature class. Only those fields that are the same type as selected in step 17 are displayed. Click the foreign key that refers to the primary key selected in step 17.
19. Click Next. ►



New Relationship Class

You have specified that this is a M-N (many to many) relationship. A new table in the database will be created for this relationship to store the foreign keys for these two participating tables/feature classes.

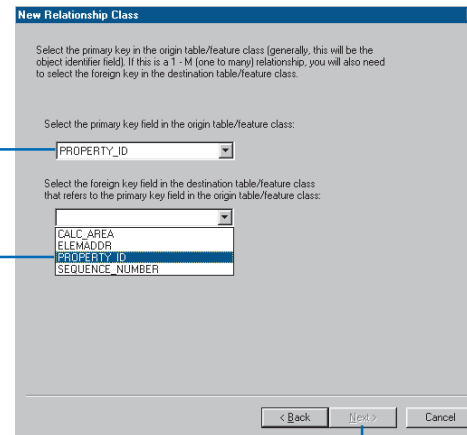
Do you wish to add additional attributes to this relationship class?

☐ Yes, I would like to add attributes to this relationship class.

☒ No, I do not want to add attributes to this relationship class.

< Back Next > Cancel

15



New Relationship Class

Select the primary key in the origin table/feature class (generally, this will be the object identifier field). If this is a 1 - M (one to many) relationship, you will also need to select the foreign key in the destination table/feature class.

Select the primary key field in the origin table/feature class:

PROPERTY_ID

Select the foreign key field in the destination table/feature class that refers to the primary key field in the origin table/feature class:

CALC_AREA
ELEMADDR
PROPERTY_ID
SEQUENCE_NUMBER

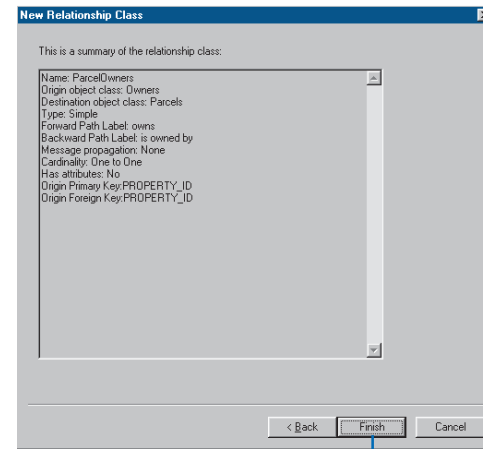
< Back Next > Cancel

17

18

19

20. Review the options you specified for your new relationship class. If you want to change something, you can go back through the wizard by clicking Back.
21. Click Finish to create the new relationship class when satisfied with your options.



Creating a composite relationship class

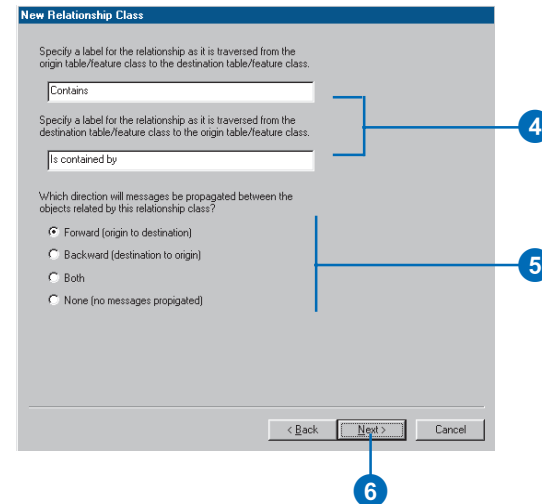
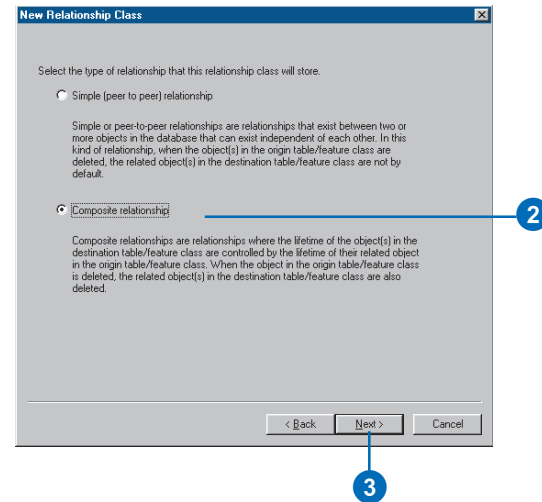
You can use a wizard to create a composite relationship class. The example in this subtask shows how to create a relationship class between a feature class that stores transformer banks and one that stores transformer units.

The existence of a transformer unit in the database is dependent on the presence of a transformer bank. This relationship class is a composite relationship with the transformer bank as the origin feature class.

The relationship will be nonattributed; composite relationships are by definition one-to-many (1-M) relationships.

Creating a composite relationship involves many of the same steps used in the task for creating a simple relationship. The steps outlined here reflect the differences between the two tasks including using different origin and destination classes.

1. Follow steps 1 through 7 for 'Creating a simple relationship class'.
2. Click Composite relationship.
3. Click Next.
4. Type the forward and backward path labels.
5. Click the message notification direction.
6. Click Next. ►

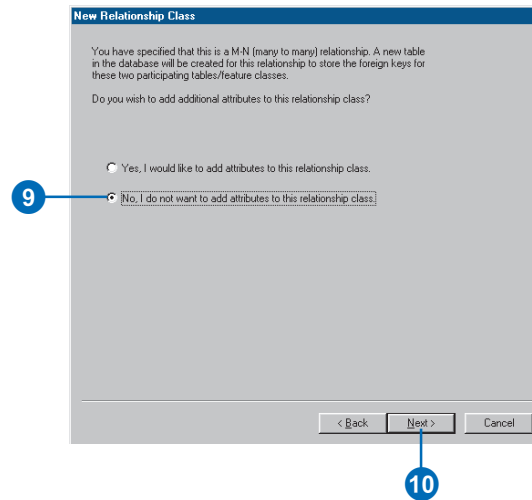
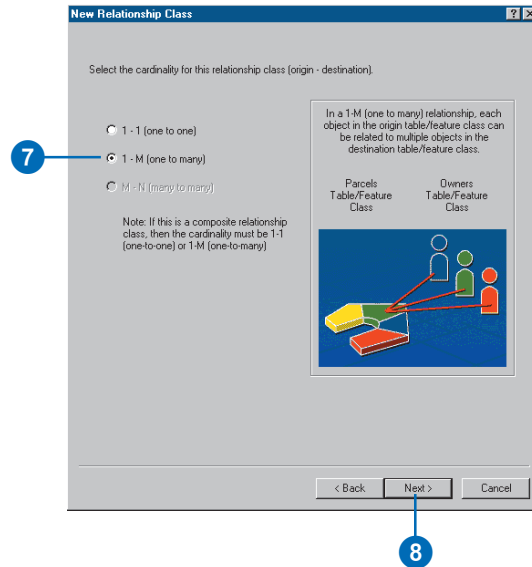


Tip

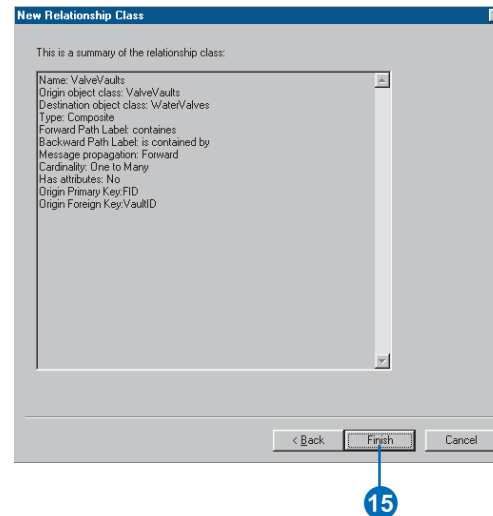
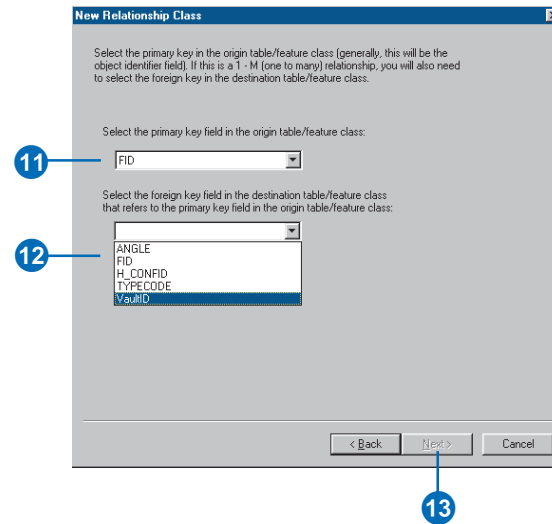
One-to-many relationships

When creating a one-to-many relationship, whether simple or composite, the one side must be the origin class. The many side must always be the destination class.

7. Click the second cardinality option. A composite relationship is, by definition, a 1-M or 1-1 relationship.
8. Click Next.
9. Click No. The relationship class does not require attributes in this example.
10. Click Next. ►



11. Click the dropdown arrow to see a list of fields from the origin table or feature class. Click the primary key for this feature class or table.
12. Click the dropdown arrow to see a list of fields in the destination table or feature class. Only those fields that are the same type as selected in step 11 are displayed. Click the foreign key that refers to the primary key selected in step 11.
13. Click Next.
14. Review the options you specified for your new relationship class. If you want to change something, you can go back through the wizard by clicking Back.
15. Click Finish to create the new relationship class when satisfied with your options.



Creating an attributed relationship class

Any relationship class—whether simple or composite, of any particular cardinality—can have attributes. Relationship classes with attributes are stored in a table in the database. This table contains at least the foreign key to the origin feature class or table and the foreign key to the destination feature class or table.

An attributed relationship can also contain any other attribute. The example in this subtask shows how to create a simple relationship between a feature class that stores water laterals and a feature class that stores hydrants.

Water lateral objects have their own attributes, and hydrant objects have their own attributes. The relationship class in this example describes which water laterals feed which hydrants. Because you want to store some kind of information about that relationship, such as the type of riser connecting the two, you can store this information as attributes in the relationship class.

1. Follow steps 1 through 14 for 'Creating a simple relationship class' or steps 1 through 8 for 'Creating a composite relationship class'.
2. Click the first option to add attributes to the relationship class.
3. Click Next.
4. Click the next row in the Field Name column and type a name to add a field.
5. Click in the Data Type field next to the new field's name, then click its data type.
6. Set the new field's properties in the property sheet.
7. Repeat steps 4 through 6 until all the relationship class's fields have been defined.
8. Click Next. ►

New Relationship Class

This relationship does not require a new table to be created in the database. If you wish to include attributes with this relationship, then a table will be created to store them.

Do you wish to add attributes to this relationship class?

☒ Yes, I would like to add attributes to this relationship class.

☐ No, I do not want to add attributes to this relationship class.

< Back Next > Cancel

New Relationship Class

Field Name	Data Type
RISMODEL	Text
RISMAT	Text

Click any field to see its properties.

Field Properties

Alias	Riser material
Allow NULL values	Yes
Default Value	
Domain	
Length	255

Import...

To add a new field, type the name into an empty row in the Field Name column, click in the Data Type column to choose the data type, then edit the Field Properties.

< Back Next > Cancel

Tip

Relationship table foreign keys

In an attributed relationship, the relationship table must have fields that act as foreign keys to the origin and destination feature classes or tables.

These foreign keys relate to the primary keys on the origin and destination feature class or table primary keys.

9. Click the dropdown arrow to see a list of fields from the origin table or feature class. Click the primary key for this feature class or table.
10. Type the name of the foreign key field for the origin table or feature class.
11. Click the dropdown arrow to see a list of fields from the destination table or feature class. Click the primary key for this feature class or table.
12. Type the name of the foreign key field for the destination table or feature class.
13. Click Next.
14. Review the options you specified for your new relationship class. If you want to change something, you can go back through the wizard by clicking Back.
15. Click Finish to create the new relationship class when satisfied with your options.

The 'New Relationship Class' dialog box is shown at Step 1. It has a title bar 'New Relationship Class' and a close button. The main text says: 'Select the primary keys in the origin and destination tables/feature classes (generally, these will be the object identifier fields). You also need to supply the names of the foreign keys in the relationship table that refer to the primary keys in the origin and destination tables/feature classes.'

There are two main sections: 'Origin Table/Feature Class' and 'Destination Table/Feature Class'.

In the 'Origin Table/Feature Class' section:
- A dropdown arrow (labeled 9) is next to the text 'Select the primary key field in the origin table/feature class:'.
- The dropdown menu shows 'FID' (labeled 10).
- Below it, a text box contains 'LateralID' (labeled 10).

In the 'Destination Table/Feature Class' section:
- A dropdown arrow (labeled 11) is next to the text 'Select the primary key field in the destination table/feature class:'.
- The dropdown menu shows 'FID' (labeled 11).
- Below it, a text box contains 'HydrantID' (labeled 12).

At the bottom, there are three buttons: '< Back' (disabled), 'Next >' (labeled 13), and 'Cancel'.

The 'New Relationship Class' dialog box is shown at Step 2. It has a title bar 'New Relationship Class' and a close button. The main text says: 'This is a summary of the relationship class:'.

Below the text is a list box containing the following information:
Name: HydrantLateralis
Origin object class: Hydrants
Destination object class: Laterals
Type: Simple
Forward Path Label: feeds
Backward Path Label: is fed by
Message propagation: None
Cardinality: One to One
Has attributes: Yes
Origin Primary Key: FID
Origin Foreign Key: LateralID
Destination Primary Key: FID
Destination Foreign Key: HydrantID

At the bottom, there are three buttons: '< Back' (disabled), 'Finish' (labeled 15), and 'Cancel'.

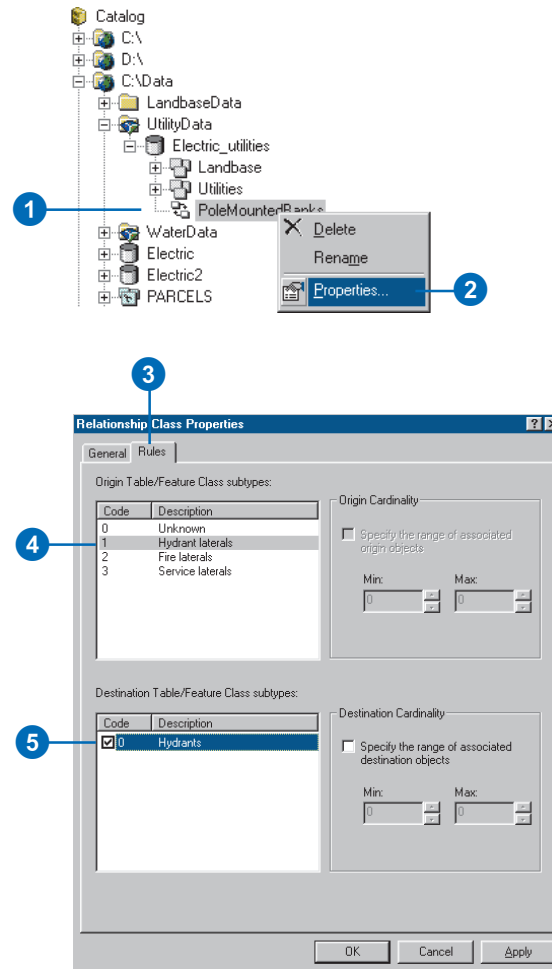
Creating relationship rules

Relationship rules let you restrict the type of objects in the origin feature class or table that can be related to a certain kind of object in the destination feature class or table.

Relationship classes are created with general cardinalities such as one to many and many to many. In a real system, however, relationship cardinalities are more specific.

In this task, a relationship rule is being created between the hydrant laterals subtype on the water laterals feature class and the hydrants feature class. This rule says that it is valid for hydrants to be fed by hydrant laterals. Using the cardinality properties, you can specify exactly how many hydrants can be related to each hydrant lateral. In this example, it is invalid for a hydrant lateral not to feed a hydrant, and it is also invalid for a hydrant lateral to feed more than one hydrant. Therefore, the minimum and maximum cardinality will be 1.

1. Right-click the relationship class in the Catalog tree.
2. Click Properties.
3. Click the Rules tab.
4. Click the subtype that you want to associate with a relationship rule if your origin class has subtypes. If the origin class has no subtypes, the relationship rule will apply to all features.
5. Check the subtype that you want to make relatable to the selected subtype in the origin class if the destination class has subtypes. If the destination class has no subtypes, the relationship rule will apply to all features. ►



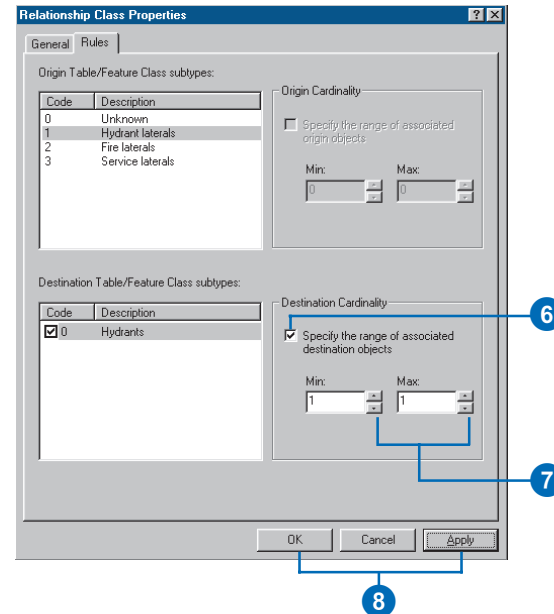
Tip

Relationship rules

Once a relationship rule is added to a relationship class, that rule becomes the only valid relationship that can exist. To make other relationship combinations and cardinalities valid, you must add additional relationship rules.

If one or both sides of the relationship class is a many, you can limit the specific range of cardinality. In this example, the origin side of the relationship is a 1, so you cannot modify its range. However, the destination side is a many, so you can modify its range.

6. Check the check box to specify the range of destination objects per related origin objects.
7. Click the up and down arrows to increase or decrease the minimum and maximum number of related destination objects.
8. Repeat steps 4 through 7 until you have specified all of the relationship rules for this relationship class. Click OK or Apply to create the rules in the database.



Managing relationship classes

Once created, a relationship class cannot be modified. However, you can add, delete, and modify its rules.

Relationship classes can be deleted and renamed using ArcCatalog. Relationship classes are deleted and renamed in the same manner as any other object in the database.

Tip

Deleting the origin or destination relationship class

When you delete a feature class or table in ArcCatalog, if that feature class or table participates in a relationship class, the relationship class is also deleted.

Tip

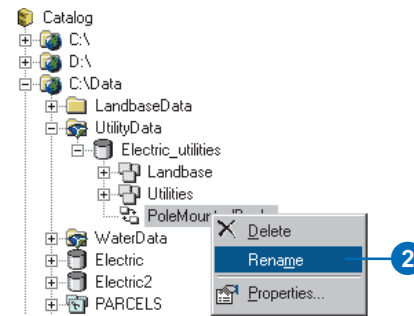
Registering as versioned

If you register either the origin or destination class as versioned in ArcCatalog, then both the relationship class and the class that it is related to are also registered as versioned.

To learn more about versioning, see the chapter 'Working with a versioned geodatabase' in this book.

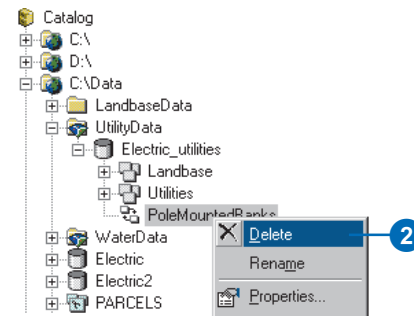
Renaming a relationship class

1. Right-click the relationship class that you want to rename.
2. Click Rename.
3. Type the new name and press Enter.



Deleting a relationship class

1. Right-click the relationship class you want to delete.
2. Click Delete.



Exploring related objects in ArcMap

In ArcMap, you can explore what objects are related to any particular object in your geodatabase. When identifying features, the Identify Results dialog box allows you to navigate to a feature's related objects. When working with tables, you can navigate to a table of related objects.

Tip

Stacked relationships

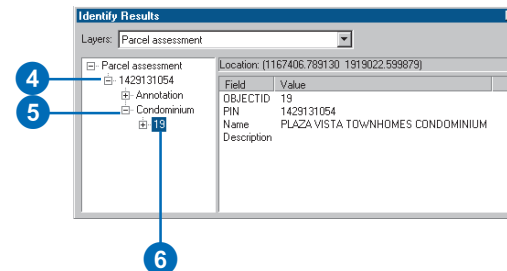
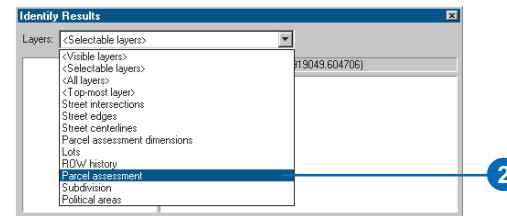
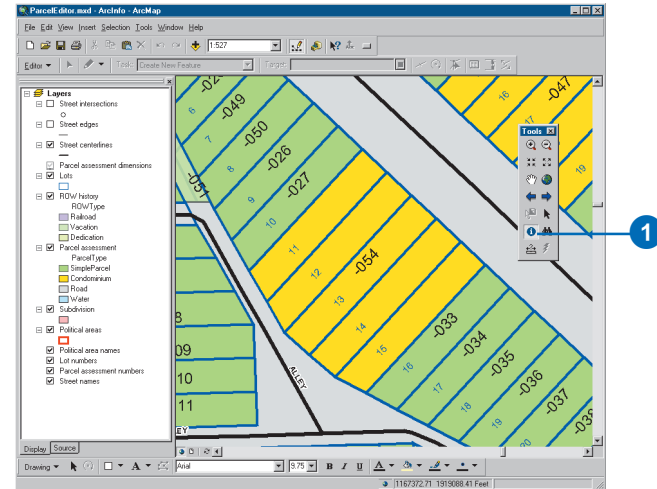
If the related object that you navigate to in the Identify Results dialog box has objects related to it through other relationships, you can continue to navigate to those related objects.

Exploring the related objects of a feature

1. Click the Identify tool in ArcMap.
2. Click the Layers dropdown arrow and click the layer in your map whose features you want to identify in the Identify Results dialog box.
3. Click the feature on the map.
4. Double-click the feature in the left panel of the Identify Results dialog box.
5. Double-click the relationship path label.

The related objects are listed below the path label.

6. Click the related object whose properties you want to explore.



See Also

If you are not already familiar with how to add data to your map, please refer to Using ArcMap.

See Also

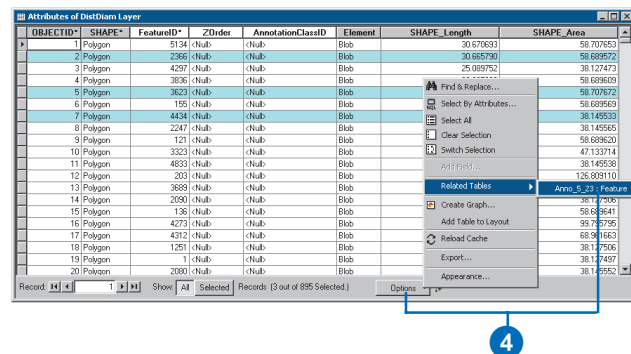
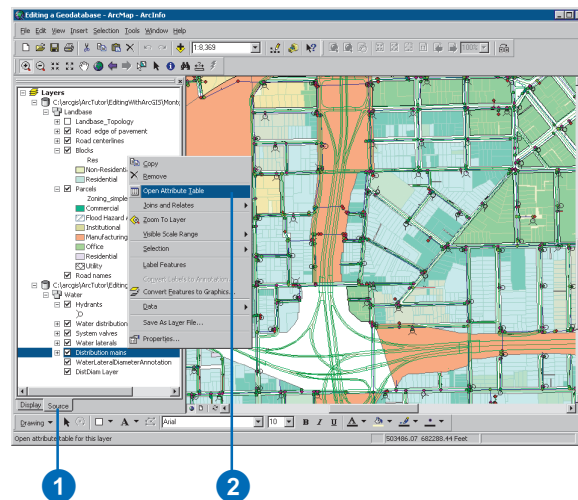
For more information on how to select records in a table, see Using ArcMap.

Exploring the related objects of an object in a table

1. Click the Source tab on the ArcMap table of contents.
2. Right-click the name of the object of interest and click Open Attribute Table.

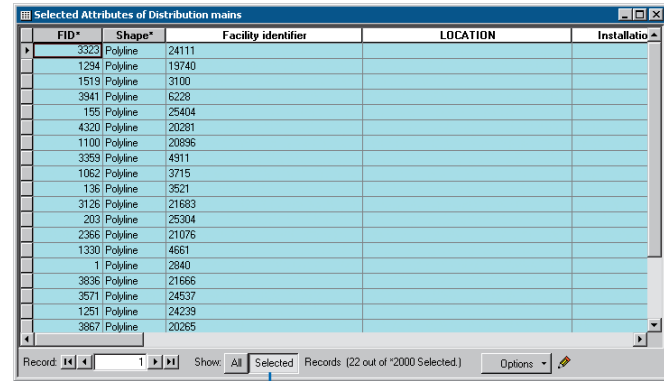
The table that contains the objects whose related objects you want to explore will open.

3. Select the objects whose related objects you want to explore.
4. Click Options, point to Related Tables, and click the path label for the relationship.



A new table dialog box opens for the related table.

5. Click Show Selected to display only those objects related to the selected objects in the first table.



FID*	Shape*	Facility identifier	LOCATION	Installation
3323	Polyline	24111		
1294	Polyline	19740		
1519	Polyline	3100		
3941	Polyline	6228		
155	Polyline	25404		
4320	Polyline	20281		
1100	Polyline	20896		
3359	Polyline	4911		
1062	Polyline	3715		
136	Polyline	3521		
3126	Polyline	21683		
203	Polyline	25304		
2366	Polyline	21076		
1330	Polyline	4661		
1	Polyline	2840		
3836	Polyline	21666		
3571	Polyline	24537		
1251	Polyline	24239		
3867	Polyline	20265		

Record: 1 Show: All Selected Records (22 out of *2000 Selected.) Options

5

Using related fields in ArcMap

In order for fields from a related object to be used for symbolizing and labeling, you must create a join between the feature class and its related feature class or table. Once you have created this join, the fields from the related feature class or table are added to your feature layer. You can use these fields for labeling, symbolizing, and querying your features.

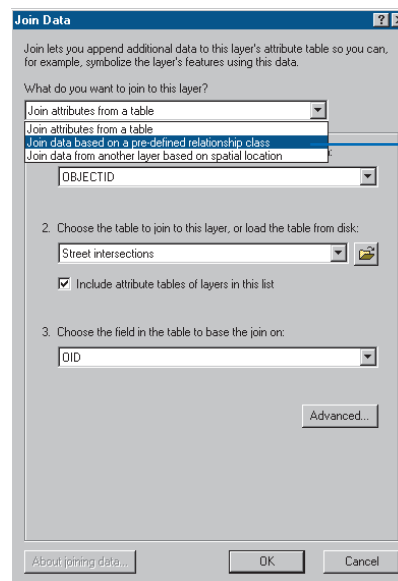
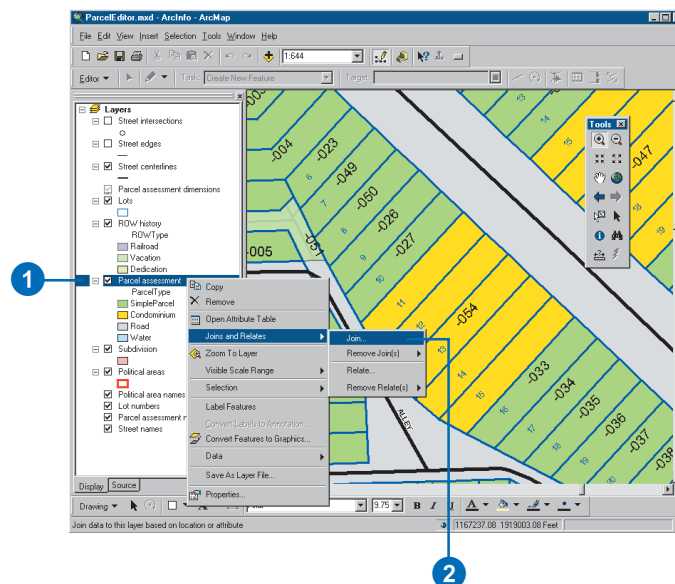
See Also

ArcMap has tools for editing relationships. To read more about relationships and editing in ArcMap, see Editing in ArcMap.

See Also

If you are not already familiar with how to add data to your map, please refer to Using ArcMap.

1. Right-click the feature layer in the ArcMap table of contents.
2. Point to Joins and Relates and click Join.
3. Click the Join options dropdown arrow and click Join data based on a predefined relationship class. ►



Tip

1-M and N-M relationships

If the relationship class is 1-M or N-M, each feature can have multiple, related objects. In this case, the attributes of the first related object are joined to the feature.

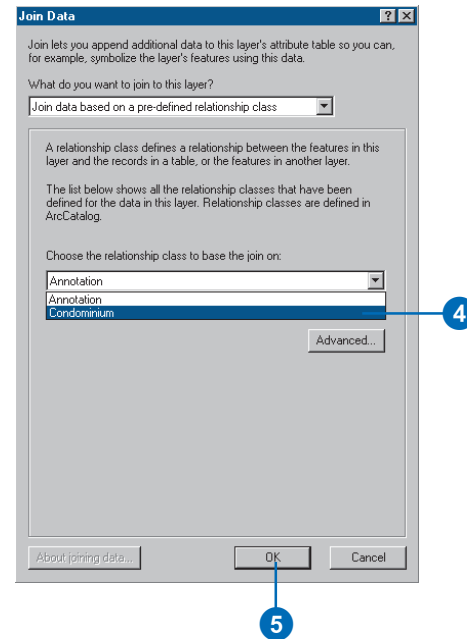
See Also

For more information about joins and using joined data in ArcMap, see Using ArcMap.

4. Click the dropdown arrow to get a list of relationship classes, then click the relationship class.

5. Click OK.

You can now use the related fields for labeling, symbolizing, and querying your features.



Geometric networks

7

IN THIS CHAPTER

- **What is a geometric network?**
- **Geometric networks and ArcCatalog**
- **Creating geometric networks**
- **Creating a new geometric network**
- **Building a geometric network from existing simple feature classes**
- **Adding new feature classes to your geometric network**
- **Network connectivity: defining the rules**
- **Establishing connectivity rules**
- **Managing a geometric network**

When you model automated mapping/facilities management (AM/FM) networks, or transportation networks, features have connectivity relationships with other features around them. This connectivity is maintained in the geodatabase through an association called a geometric network.

Geometric networks are created and managed using ArcCatalog. This chapter highlights the key tasks for creating and managing geometric networks. Another type of topological association is discussed in the ‘Topology’ chapter in this book.

ArcView can view feature classes that use advanced geodatabase functionality. To create or edit feature classes that take advantage of advanced geodatabase functionality, you need an ArcEditor or ArcInfo license.

What is a geometric network?

The movement of people, the transportation and distribution of goods and services, the delivery of resources and energy, and the communication of information all occur through definable network systems. In the geodatabase, networks are modeled as a one-dimensional nonplanar graph, or geometric network, that is composed of features. These features are constrained to exist within the network and can, therefore, be considered network features. The geodatabase automatically maintains the explicit topological relationships between network features in a geometric network. Network connectivity is based on geometric coincidence, hence the name geometric network.

A geometric network has a corresponding logical network. The geometric network is the actual set of feature classes that make up the network. The *logical network* is the physical representation of the network connectivity. Each element in the logical network is associated with a feature in the geometric network.

Once a geometric network is in place, ArcMap and ArcCatalog have tools that treat the network features in a special way. Editing and *tracing* on the network, as well as managing the feature classes participating in the network, are all handled automatically by the ArcGIS 8 system.

Network feature types

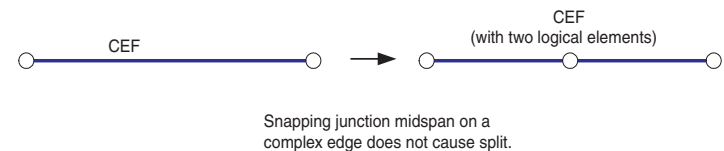
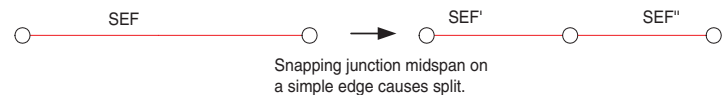
Geometric networks consist of edge network features and junction network features. An example of an edge feature is a water main, while a junction feature might be a valve. Edges must be connected to other edges through junctions. Edge features are related to *edge elements* in the network, and junction features are related to *junction elements* in the logical network.

There are two broad categories of network feature types: simple and complex. Simple network features correspond to a single network element in the logical network. A complex network feature

corresponds to more than one network element in the logical network.

A simple edge feature corresponds to a single network edge element in the logical network. Simple edges are always connected to exactly two junction features, one at each end. If a new junction feature is snapped midspan on a simple edge (thereby establishing connectivity), then that simple edge feature is physically split into two new features.

Complex edges correspond to one or more edge elements in the logical network. Complex edges are always connected to at least two junction features at their endpoints but can be connected to additional junction features along their length. If a new junction feature is snapped midspan on a complex edge, that complex edge remains a single feature. Snapping the junction does cause the complex edge to be split logically—for example, if it corresponded to one edge element in the logical network before the junction was connected, it now corresponds to two edge elements.

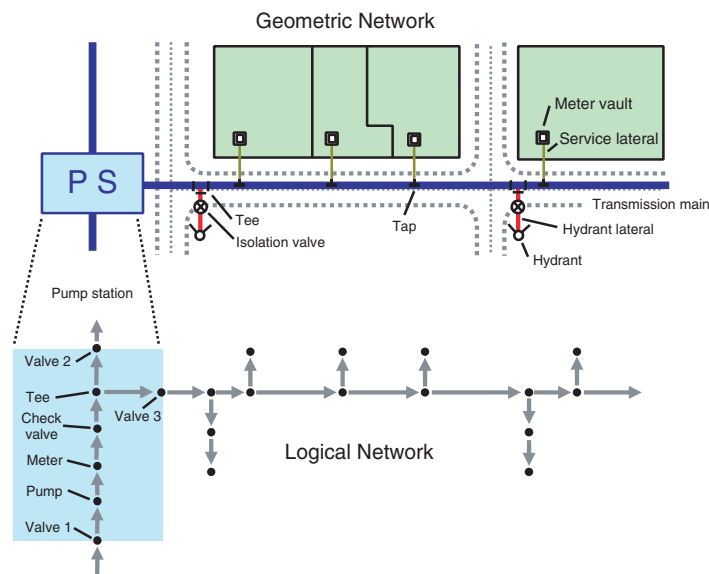


- CEF Complex edge feature
- SEF Simple edge feature
- Junction feature

*Simple edge features are connected to exactly two junction features.
Complex edges can be connected to two or more junction features.*

A complex junction is a single feature that corresponds to any number of edge and junction elements in the logical network. For example, a complex junction may be a water pump station in a water network. While the pump station feature itself is stored as a single complex junction feature in the geodatabase, its representation in the logical network may include a set of pipes, pumps, meters, and valves that all affect the flow through the pump station. The combination of these devices may be represented as a set of seven junction elements and six edge elements.

Complex junctions can be implemented as custom features only. To learn more about custom features, see *Exploring ArcObjects*.



In this example, a pump station is a single polygon in the geometric network but it represents a set of pipes, valves, meters, and pumps in the logical network. It is a complex junction feature.

Sources and sinks

Networks are often used to model real-world systems in which the direction of movement through the network is well defined. For example, the flow of electricity in an electrical network is from the power generation station to the customers. In a water network, the flow direction may not be as well defined as in an electric network, but the flow of water may be from a pump station to a customer or from customers to a treatment plant.

Flow direction in a network is calculated from a set of sources and sinks. In the above examples, electricity and water flow are driven by sources and sinks. Flow is away from sources, such as the power generation station or a pump station, and toward sinks such as a water treatment plant (in the case of a wastewater network).

Junction features in geometric networks can act as sources or sinks. When you create a new junction feature class in a network, you can specify whether the features stored in it can represent sources, sinks, or neither in the network. If you specify that these features can be sources or sinks, a field called *AncillaryRole* is added to the feature class to record if the feature is a source, sink, or neither a source nor a sink. When you calculate the flow direction for a geometric network in ArcMap, the flow direction will be calculated based on the sources and sinks in the network.

For example, you may have a tank in your water network that is down for maintenance, so its role in the network will be changed (temporarily) from source to none. The flow for the network is recalculated by the system; any traces on the network will be affected by the change in flow direction caused by the state of the tank. For more information on calculating flow direction and using flow direction in network analysis, see *Using ArcMap*.

Network weights

A network can also have a set of weights associated with it. A weight can be used to represent the cost of traversing an element in the logical network. For example, in a water network, a certain amount of pressure is lost when traveling the length of a transmission main due to surface friction within the pipe.

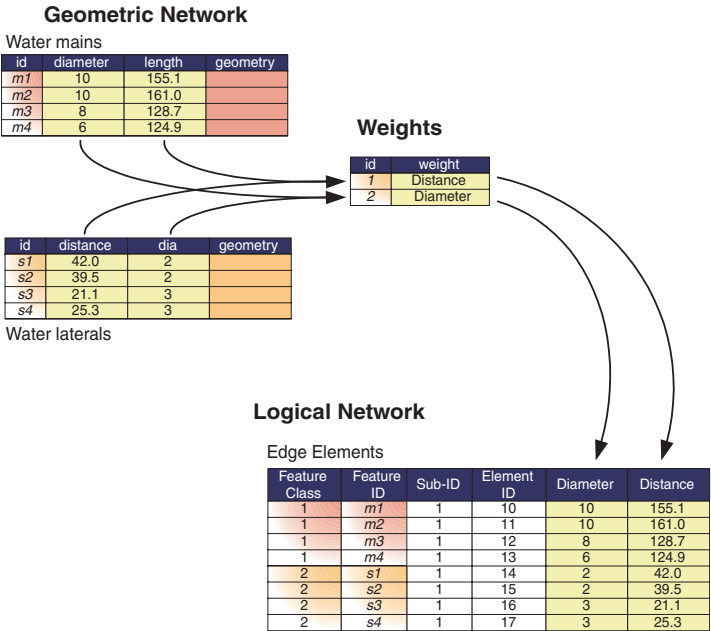
Weights are calculated based on some attribute of each feature. In the transmission main example above, an attribute that affects the weight would be the length of the feature.

A network can have any number of weights associated with it. Each feature class in the network may have some, all, or none of these weights associated with its attributes. The weight for each feature is determined by some attribute for that feature. Each weight can be associated with one attribute in a feature but, at the same time, can be associated with multiple features. For example, a weight called Diameter can be associated with the attribute Diameter in water main features and can also be associated with the attribute Dia in water lateral features.

Enabled and disabled features

Any edge or junction feature in a geometric network may be enabled or disabled in the logical network. A feature that is disabled in the logical network acts as a barrier. When the network is traced, the trace will stop at any barriers it encounters in the network including disabled network features.

The enabled or disabled state of a network feature is a property maintained by an attribute field called Enabled. It can have one of two values: true or false. When building a geometric network from simple feature classes, this field is automatically added to the input feature classes. When you use ArcCatalog to create a network feature class, Enabled is a required field for the feature class.



A network can have any number of weights associated with it. Each feature class in the network may have some, all, or none of these weights associated with its attributes. The weight for each feature is determined by some attribute for that feature.

For a discussion on required fields, see the chapter ‘Creating new items in a geodatabase’ in this book. When adding new features to a network, they are enabled by default. For more information on editing geometric networks, see *Editing in ArcMap*.

Values stored in the network weight, ancillary role, and enabled fields are the user’s view of the state of the feature in the logical network. When analysis—such as tracing and flow direction calculation—is performed against a network feature, the value of these fields within the feature is not directly referenced to determine the enabled, ancillary role state of the feature or its

weight. Instead, these states of the feature are stored in the logical network, which is queried during these operations. This is done for performance reasons.

When you edit a network feature and change the value of the enabled, ancillary role or a weight field, the state of the feature in the internal topology tables is modified to remain in sync with the field values of the feature.

Performance considerations

Geometric networks can be composed of a number of edge and junction feature classes. When editing geometric networks in ArcMap, topological relationships between features are maintained while editing on the fly. The benefit of this model is that there is no need to perform a postediting process to build topology for the geometric network. The cost of continually maintaining network connectivity imposes overhead on the time it takes to add or modify features in network feature classes.

Topological connectivity in a network feature class is based on geometric coincidence. If a junction is added along an edge, the edge and junction will become topologically connected to one another. When a new feature is added to a network feature class, this geometric coincidence must be discovered. So, each feature class in the network must be analyzed by performing a spatial query against it to determine if the new feature is coincident with network features. If coincidence is discovered, then network connectivity is established.

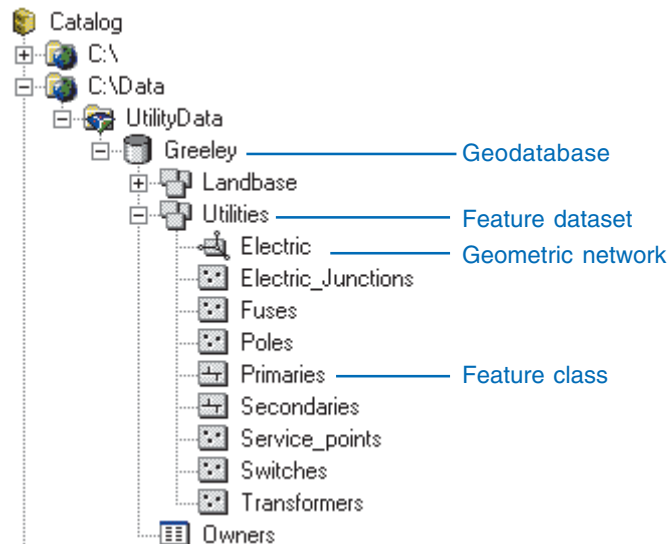
When discovering connectivity, a separate spatial query must be executed on the server for each feature class in the network. If you use the *edit cache* while editing the network, these spatial queries do not need to go against the server and are much faster. You will not pay as much of a penalty in performance for having a large number of feature classes in your network if you use the edit cache. Using the edit cache when editing your network

features will significantly improve performance when adding new features or connecting and moving existing features. For more information on editing geometric networks and using the cache, see *Editing in ArcMap*.

Try to reduce the number of feature classes you have in your geometric network by lumping feature classes together using subtypes. If your feature classes carry different attributes, you can use relationships to manage subtype-specific attributes in different tables in the database; or you can keep all the attributes in the same table using nulls for those that don't apply to a particular subtype.

Geometric networks and ArcCatalog

In ArcCatalog, you can view and manage geometric networks in geodatabases that you have access to. Because all geometric networks must be inside a *feature dataset*, they appear in the ArcCatalog tree under their feature dataset.



It is not immediately evident in the ArcCatalog tree which feature classes participate in the network, which participate in which network, and which participate in none. However, by examining the properties of both geometric networks and feature classes, the network feature classes can be determined.

ArcCatalog also contains various tools to create, delete, and manage both geometric networks and the feature classes that participate in geometric networks. These tools are discussed in more detail later in this chapter.

Creating geometric networks

A geometric network is a topological relationship between a collection of feature classes in a feature dataset. Each feature has a role in the geometric network of either an edge or a junction. Multiple feature classes may have the same role in a single geometric network.

The basic methodology for creating a geometric network is to determine which feature classes will participate in the network and what role each will play. Optionally, a series of network weights can be specified, as can other more advanced parameters.

Two different methods are available for creating a network. These are creating a new, empty geometric network and building a geometric network from existing simple features.

Creating a new, empty network

In ArcCatalog you can create, design, and build a geometric network from scratch. You can then use editing tools in ArcMap or custom Visual Basic® (VB), Visual Basic for Applications (VBA), or C++ code to add features to the geometric network.

The process of creating a network can be summarized in the following steps:

1. Use ArcCatalog to create the feature dataset that will contain the geometric network and its feature classes.
2. Use ArcCatalog to create an empty geometric network in the feature dataset.
3. Use ArcCatalog to create new feature classes in the feature dataset and assign each a role in the geometric network.
4. Use ArcCatalog to establish *connectivity rules* for elements of the geometric network.
5. Use custom scripts or ArcMap editing tools to add features to the network.

Building a geometric network from existing data

You may already have data from which you want to create a geometric network in your geodatabase. ArcCatalog and ArcToolbox contain tools to create a geometric network from that data.

The process of building a geometric network from existing data can be summarized in the following steps:

1. Use ArcCatalog or ArcToolbox to convert and load your data into a geodatabase.
2. Use ArcCatalog or ArcToolbox to build a geometric network from existing simple feature classes.
3. Use ArcCatalog to add any additional empty feature classes to the geometric network.
4. Use ArcCatalog to establish connectivity rules for the geometric network.

How networks are built

Building networks from existing features is a computationally intense operation that may take a considerable amount of time and system resources, depending on the number of input features. If those features require snapping, the network building operation will spend most of its time in the feature snapping phase. The network building process proceeds in the following sequence:

1. If snapping is specified, snap simple features.
2. If snapping is specified, snap complex features.
3. Create an empty logical network.
4. Create the network schema in the database.
5. Extract attributes from the input feature classes for weight calculations.

6. Create the topology.
7. Create orphan junctions as required, add input junction features to the logical network, and initialize the junction-enabled values.
8. Set weight values for the junction elements.
9. Add edges to the logical network.
10. Set weight values for the edge elements.
11. Create all necessary indexes in the database.

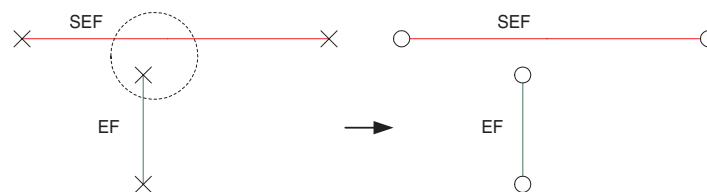
Network snapping models

Ideally, your data should be clean before you build a network. Clean data means that all features that should be connected in the network are geometrically coincident—that is, no *overshoots* or *undershoots*. However, if this is not the case, the data may be snapped during the network building process.

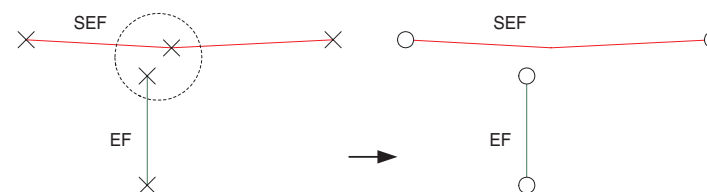
It is important to understand how connectivity is established based on snapping during the network building process and how feature geometries are adjusted to establish that connectivity. The following is a series of examples of how connectivity is established in given scenarios. In these diagrams, use the key below to identify what types of features are illustrated in each scenario:

<u>EF</u>	Edge feature (simple or complex)
<u>CEF</u>	Complex edge feature
<u>SEF</u>	Simple edge feature
×	Vertex
● ○	Junction features
○	Snapping tolerance

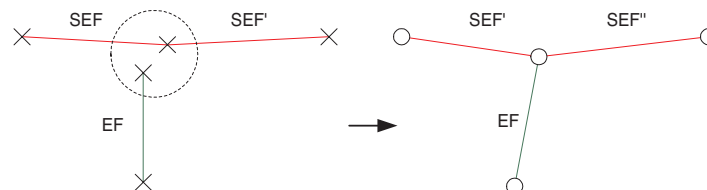
Simple edges: Connectivity against simple edges is established only at the ends of edge features. Midspan connectivity will not be established, even if there is a vertex along the simple edge feature.



No connectivity is established.



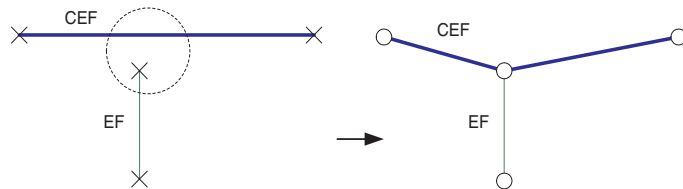
No connectivity established. Midspan connectivity on simple edge features is not established in snapping.



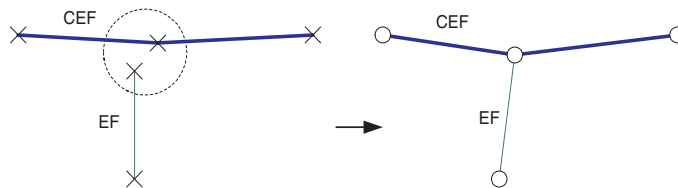
Connectivity is established. With simple edge features, only endpoint vertices are considered when establishing connectivity in snapping.

Simple edge connectivity models

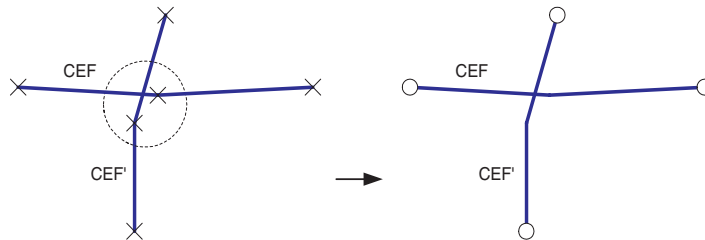
Complex edges: Connectivity against complex edges is established both at the ends of features and midspan. If there is no vertex along the complex edge where connectivity is established, a new vertex is created. When snapping complex edges, connectivity must be at the endpoint of at least one of the edges. Connectivity will not be established between the midspan of one edge and the midspan of another edge.



Connectivity established. Intersection detection is performed along complex edges, and new vertices are inserted as required.



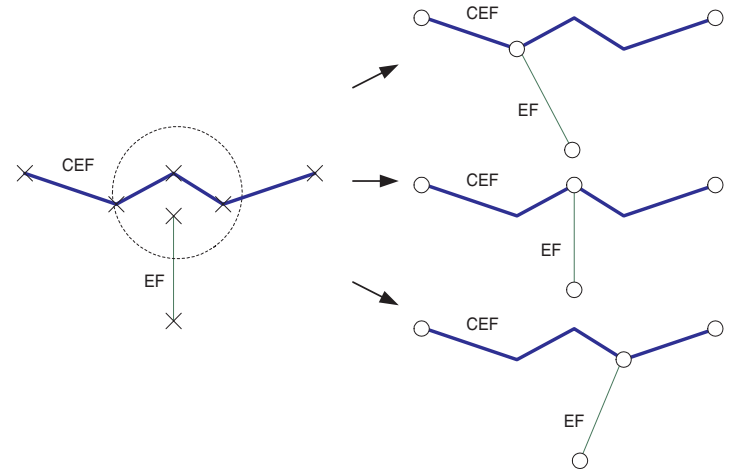
Connectivity established. Midspan connectivity on complex edge features is established in snapping.



No connectivity established. Connectivity must be at an endpoint of one of the two edge features.

Complex edge connectivity models

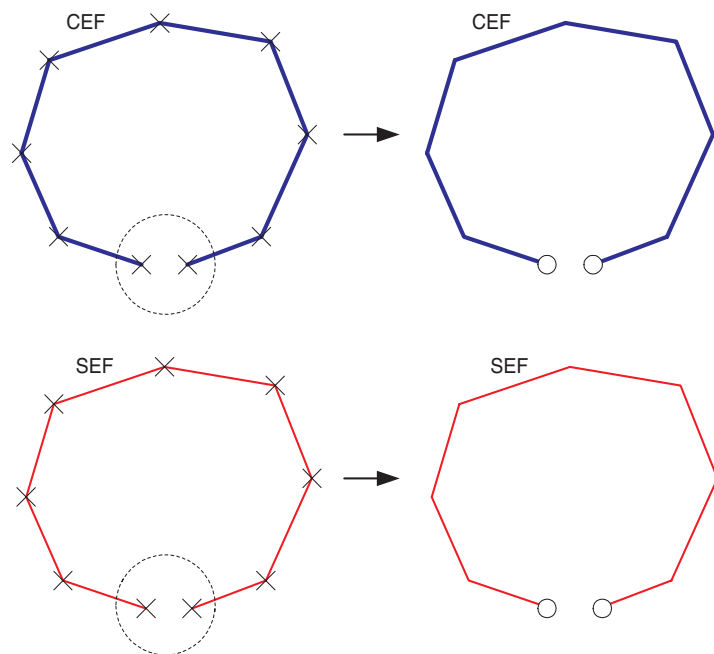
Vertex clustering: When snapping two features, if there is more than one vertex within the *snapping tolerance*, then those vertices are treated as a cluster. Snapping will occur to one of the vertices in the cluster, but not necessarily the closest.



Connectivity established. One of the vertices within the snap tolerance is snapped to, but not necessarily the closest.

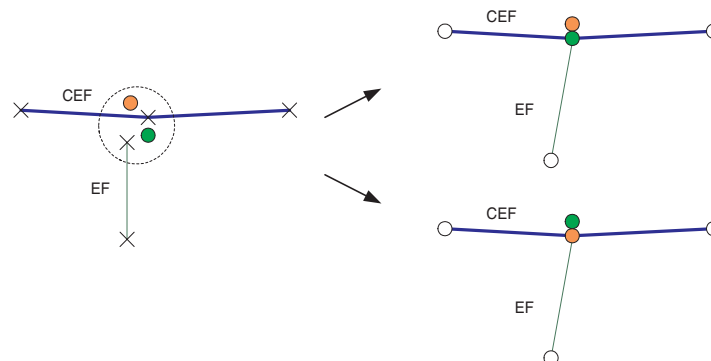
Network snapping vertex clustering does not guarantee the closest vertex is snapped to—it may be any of the vertices.

Connecting features to themselves: When the endpoints of a single edge feature come within the snapping tolerance of itself, the endpoint will not be snapped and no connectivity will be established. Connectivity is not established between a feature and itself.



No connectivity established. Connectivity is not created between features and themselves.

Coincident junctions: When the network building process encounters coincident junctions, or when the snapping process results in coincident junctions, the resulting connectivity will be nondeterministic. That is, connectivity will only be established to one of the coincident junctions.



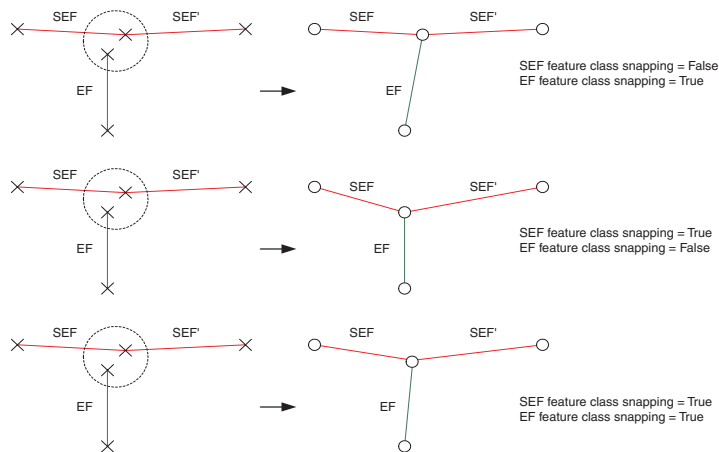
Connectivity is nondeterministic when coincident junctions are encountered.

Adjusting features

When snapping features during network building, it is important to understand how the geometry of features is adjusted when snapping. All or part of any feature in a feature class that was specified as being adjustable in the Build Geometric Network wizard can potentially be moved. Those features that are in feature classes that are not adjustable will remain fixed throughout the network building process.

All features in all feature classes have equal weights when being adjusted during snapping. This means that if the endpoints for two edges need to be snapped and both features can be adjusted, then they will move an equal distance to snap together. If one of

the features is not adjustable, then only the adjustable feature will move to snap to the static feature.

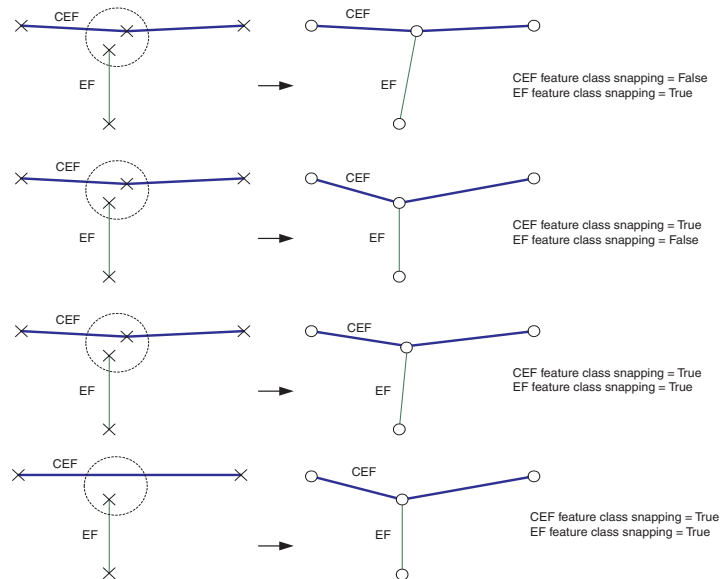


How simple edge features are adjusted depends on whether the features they are snapping to can or cannot be adjusted.

Schema locking

An exclusive lock is required on all of the input feature classes when building a geometric network. If any of the input feature classes have a shared lock, the network will not be built.

If any of the feature classes in a network have a shared or exclusive lock, that lock is propagated to all of the other feature classes in the network. For more information on exclusive locks and schema locking, see the chapter 'Creating new items in a geodatabase' in this book.



How complex edge features are adjusted depends on whether the features they are snapping to can or cannot be adjusted.

Creating a new geometric network

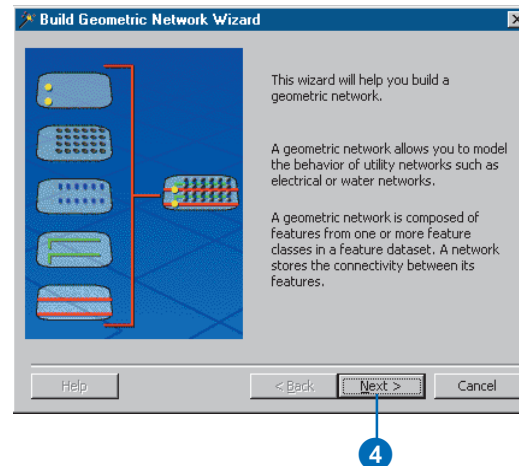
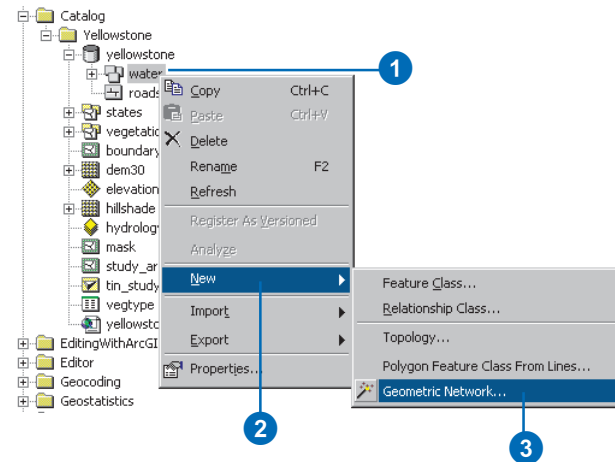
Geometric networks are created inside feature datasets. Once a geometric network has been created, you must add feature classes to the feature dataset and assign them roles in the network.

New feature classes can be added to a geometric network at any time.

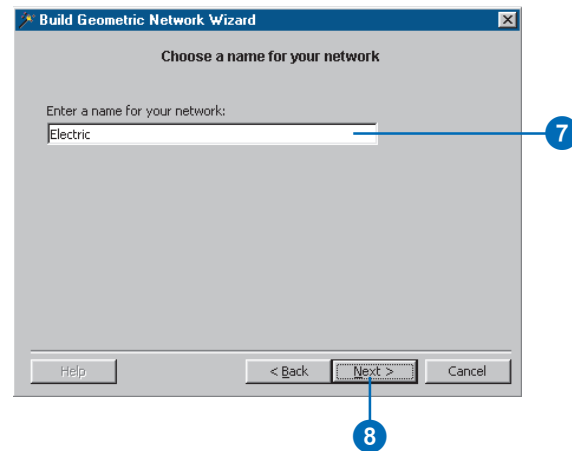
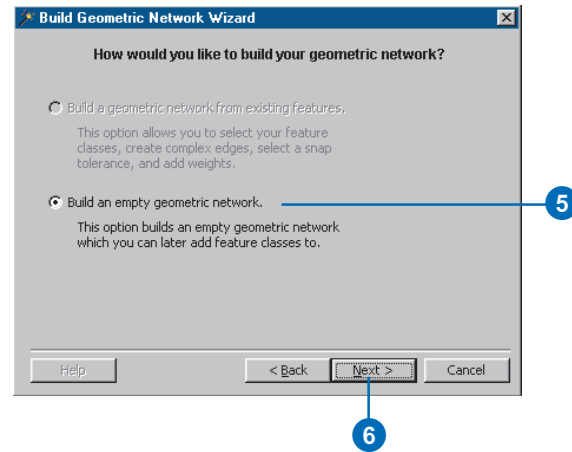
See Also

For more information on creating feature datasets and feature classes, see the chapter 'Creating new items in a geodatabase' in this book.

1. Right-click the feature dataset that will contain the network.
2. Point to New.
3. Click Geometric Network.
4. Read the information on the first panel and click Next.



5. Click the second option to build an empty geometric network.
6. Click Next.
7. Type a name for the new geometric network.
8. Click Next. ►



Tip

Network weights

Network weights apply to all elements in the network. You can assign which weights are associated with which field on each feature class when you create the network feature class.

You can't remove or add weights once the geometric network is created.

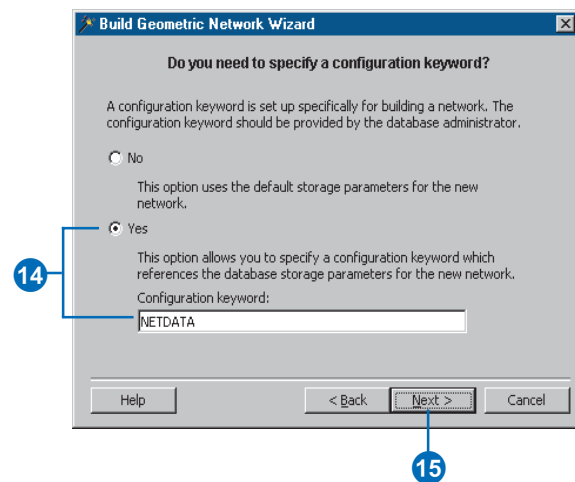
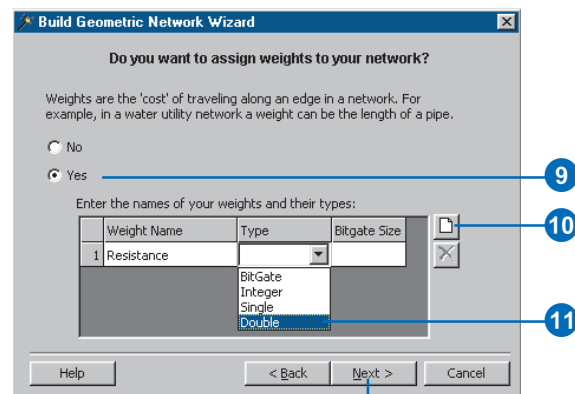
See Also

For more information on geometric networks and network weights, see Modeling Our World.

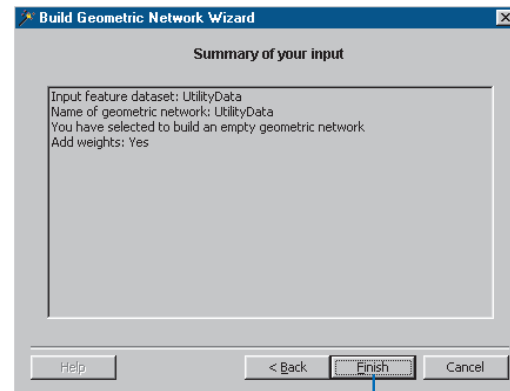
See Also

For details about using storage keywords with ArcSDE, see Managing ArcSDE Services.

9. Click Yes if you want to include weights in the network; otherwise, skip to step 13.
10. Click the New button and type a name to add a new weight.
11. Click the dropdown arrow and click the weight type.
12. Repeat steps 10 and 11 until all of the network's weights have been defined.
13. Click Next.
14. Click Yes and type the name of the keyword if your geodatabase is stored in an ArcSDE database and you have a configuration keyword for the network storage. If not, skip to step 15.
15. Click Next. ►



16. Review the options you specified for your new network. If you want to change something, you can go back through the wizard by clicking the Back button.
17. Click Finish to create the new geometric network.



Building a geometric network from existing simple feature classes

An alternative to creating and populating an empty geometric network is to build a geometric network from existing simple feature classes.

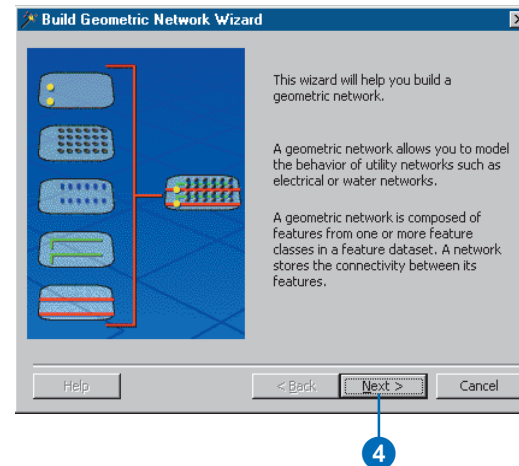
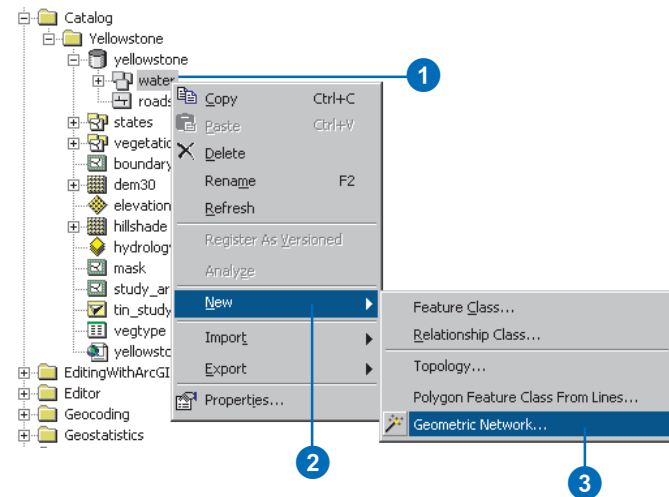
The Build Geometric Network wizard discovers the connectivity for a group of feature classes in a feature dataset and promotes them from simple feature types (lines and points) to network feature types (edges and junctions).

When you build a geometric network, the feature classes must already exist in the feature dataset. However, they can be empty. After the network has been built, you can add new empty network feature classes.

Geometric networks can be built using either ArcCatalog or ArcToolbox.

Building a geometric network using ArcCatalog

1. Right-click the feature dataset that will contain the network.
2. Point to New.
3. Click Geometric Network.
4. Read the information on the first panel and click Next.



Tip

Versioned data

When building a geometric network from simple feature classes in an ArcSDE geodatabase, the input feature classes cannot be versioned.

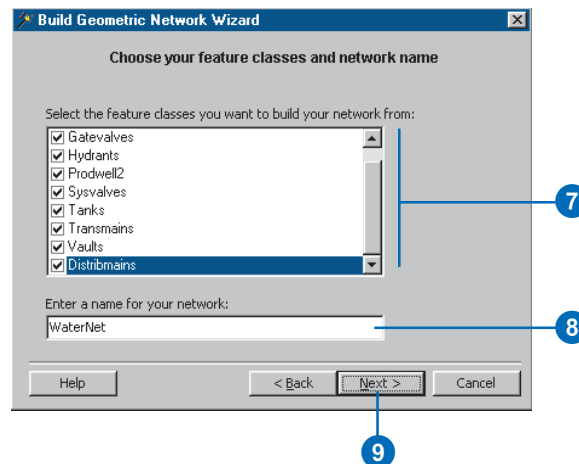
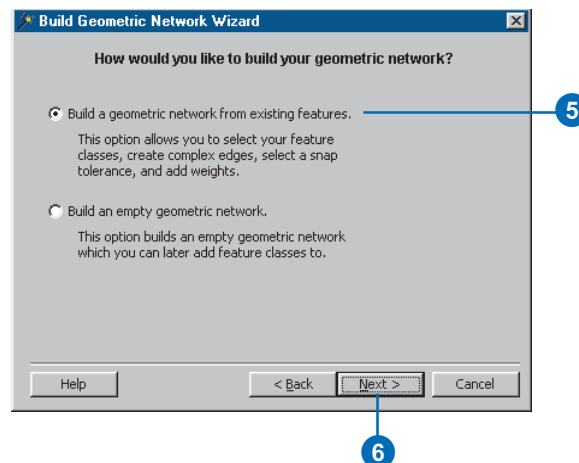
For more information on versions, see the chapter 'Working with a versioned geodatabase' in this book.

Tip

Input schema

All network feature classes require a short integer field named Enabled to record whether or not a feature is enabled or disabled in the logical network. The network building wizard will automatically add this field to your input feature classes.

5. Click the first option to build a geometric network from existing features.
6. Click Next.
7. Click the feature classes that you wish to include in this geometric network.
8. Type a name for the new geometric network.
9. Click Next. ►



Tip

Complex edges

When you build a geometric network from existing simple feature classes, the line feature classes become simple edges by default. However, you can specify that you want some of the line feature classes to be complex edges in the geometric network.

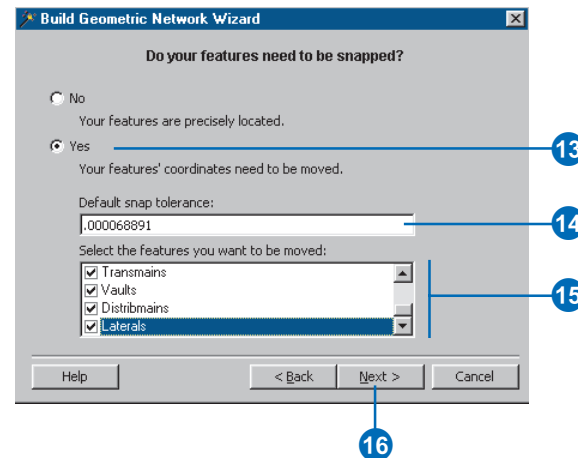
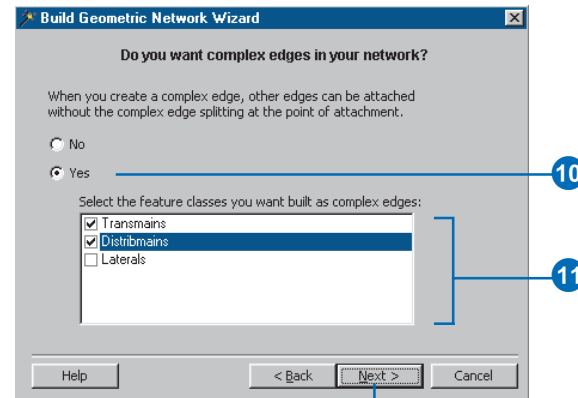
Tip

Snapping features

The geometric network builder can automatically adjust features in the input feature classes to correctly snap to connecting features. The default snapping tolerance is $1.5 * 1/XY$ scale of the feature dataset's spatial reference.

If snapping, you cannot use a value smaller than the default. Large snapping tolerances may cause unanticipated results. For best results, examine your data and provide an appropriate tolerance. Snapping (geometry changes) cannot be undone.

10. Click Yes if you want some of the input line feature classes to become complex edges; otherwise, skip to step 12.
11. Check the line feature classes that you want to become complex edges. Those that are not selected will become simple edges.
12. Click Next.
13. Click Yes if you want features in some of the input feature classes to be automatically adjusted and snapped during the network building process; otherwise, skip to step 16.
14. Type a *snapping tolerance* if you don't want to use the default tolerance.
15. Check the feature classes whose features you want automatically adjusted and snapped. Feature classes that are not selected are not adjusted.
16. Click Next. ►



Tip

Sources and sinks

If you specify that you want to store sources and sinks in a junction feature class, a field called *AncillaryRole* will automatically be added to the feature class.

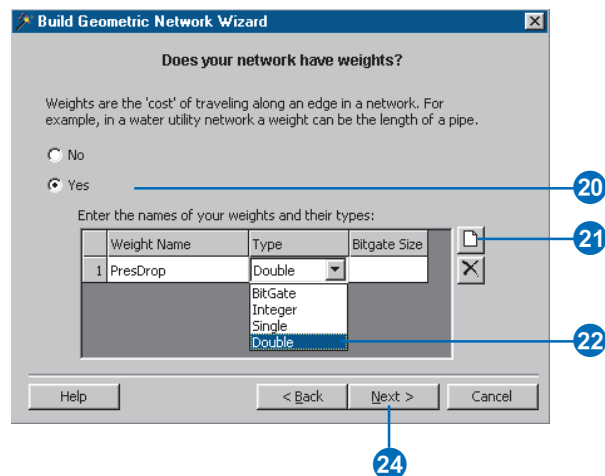
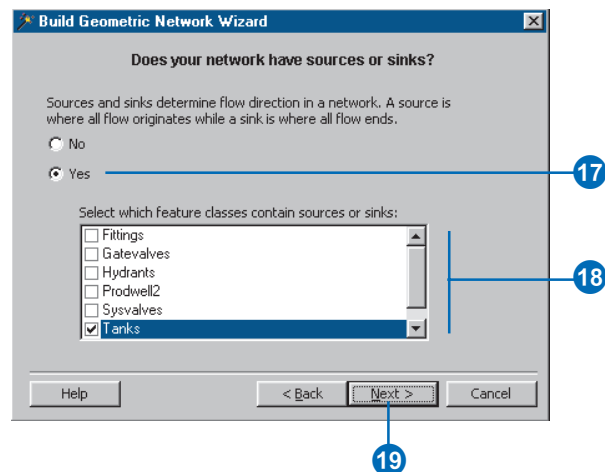
Tip

Network weights

Once the geometric network is built, no additional weights can be added to it. Also, the feature classes with which each weight is associated can't be altered.

When you add a new feature class to the geometric network, it can be associated with the existing network weights.

17. Click Yes if you want features in some of your junction feature classes to be able to act as sources or sinks; otherwise, skip to step 19.
18. Check those junction feature classes that you want to store sources and sinks.
19. Click Next.
20. Click Yes if you want to add weights to your network. Otherwise, skip over steps 25 through 29.
21. Click the New button to add a new weight.
22. Type a name for the new weight, click the dropdown arrow, and click a weight type.
23. Repeat steps 21 and 22 until all of the network's weights have been defined.
24. Click Next. ►



Tip

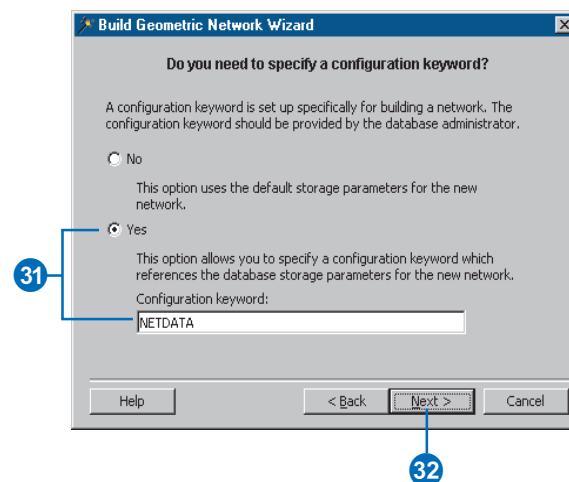
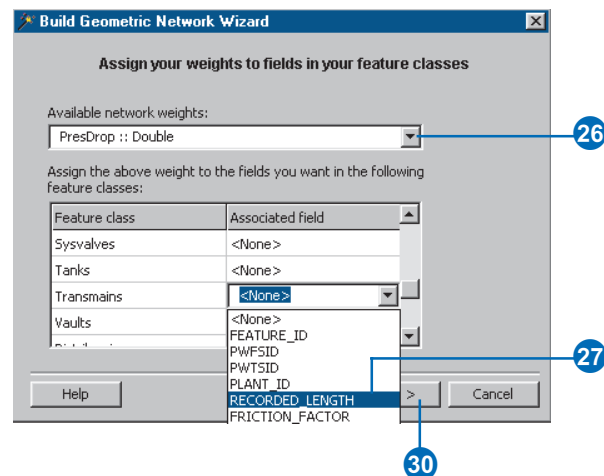
Building progress

You will be notified of the building progress with a series of progress bars indicating the progress of each step along the way.

See Also

For details about using storage keywords with ArcSDE, see Managing ArcSDE Services.

25. Assign the weights, if you added any, to specific fields in each feature class.
26. Click the dropdown arrow and click the network weight to which you will assign an attribute.
27. Click the dropdown arrow and click the field you want associated with this weight.
28. Repeat step 27 for each feature class that you want to associate with this weight.
29. Repeat steps 26 through 28 until you are finished associating network weights with feature class attributes.
30. Click Next.
31. Click Yes and type the name of the keyword if your geodatabase is stored in an ArcSDE database and you have a configuration keyword for the network storage. If not, skip to step 34.
32. Click Next. ►



Tip

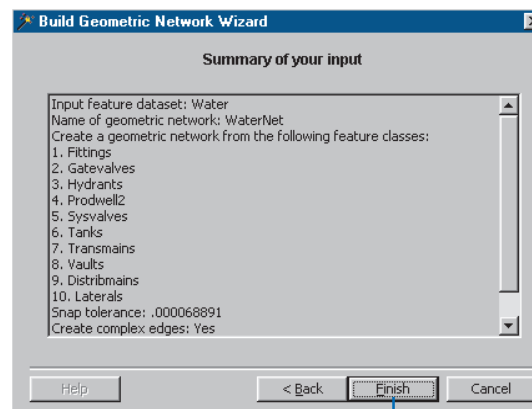
Aborting

At any time during the building process, you can abort by clicking Abort on the Progress dialog box.

When you abort the build, the system deletes any network tables created and sets the database to the state it was before the build started.

If snapping was already complete, that change is permanent and will not be restored.

33. Review the options you specified for your new network. If you want to change something, you can go back through the wizard by clicking the Back button.
34. Click Finish to create the new geometric network.

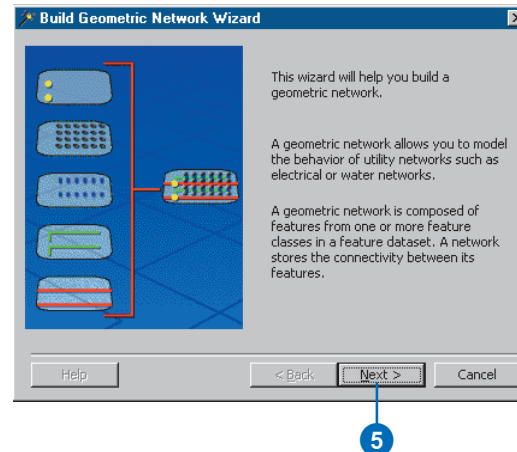
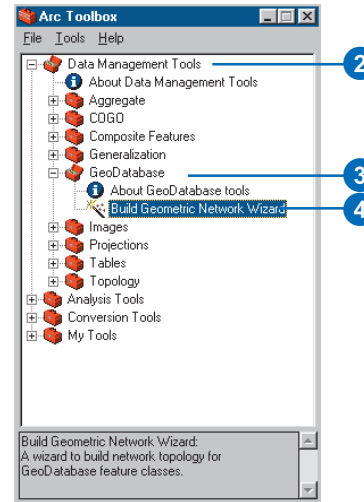


See Also

For more information on starting and using ArcToolbox, see Using ArcToolbox.

Building a geometric network using ArcToolbox

1. Start ArcToolbox from the Start menu.
2. Double-click Data Management Tools.
3. Double-click GeoDatabase.
4. Double-click Build Geometric Network Wizard.
5. Click Next. ►



Tip

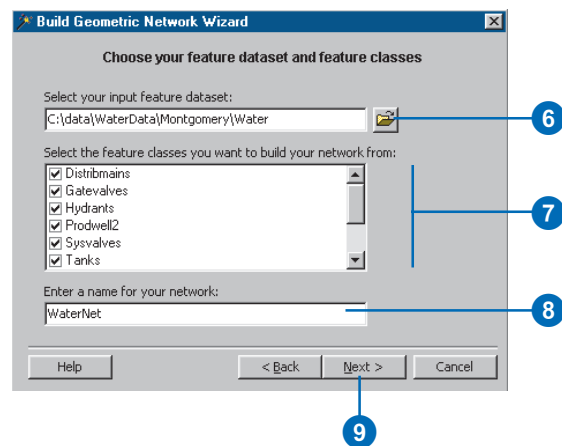
Input feature dataset

An alternative to browsing for the input feature dataset is to drag it from ArcCatalog and drop it in the feature dataset text box.

See Also

For more information on using the minibrowser to select data, see Using ArcCatalog.

6. Click the Browse button to browse for the feature dataset that contains the feature classes from which you want to build your network.
7. Check the feature classes that you wish to include in the geometric network.
8. Type a name for the new geometric network.
9. Click Next.
10. Follow steps 10 through 34 for 'Building a geometric network using ArcCatalog'.



Adding new feature classes to your geometric network

At any time in a geometric network's life, you can add new edge and junction feature classes. These new feature classes are empty—you cannot add populated feature classes to an existing geometric network. Adding a new feature class to a geometric network is similar to the task of creating a new feature class to store *custom features* (see the chapter 'Creating new items in a geodatabase' in this book).

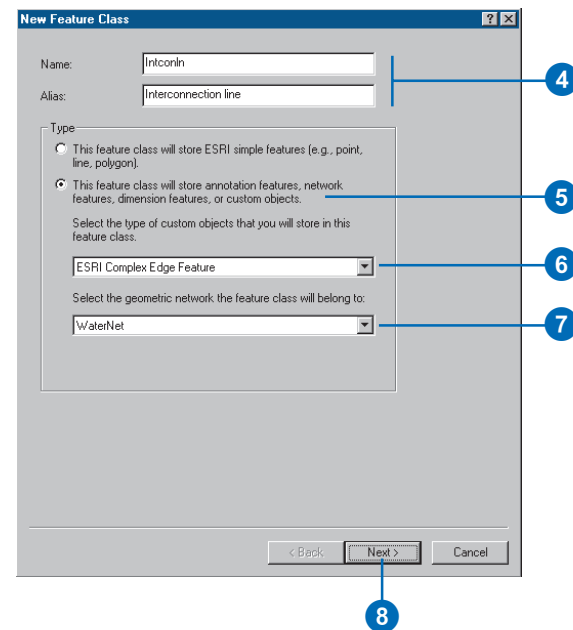
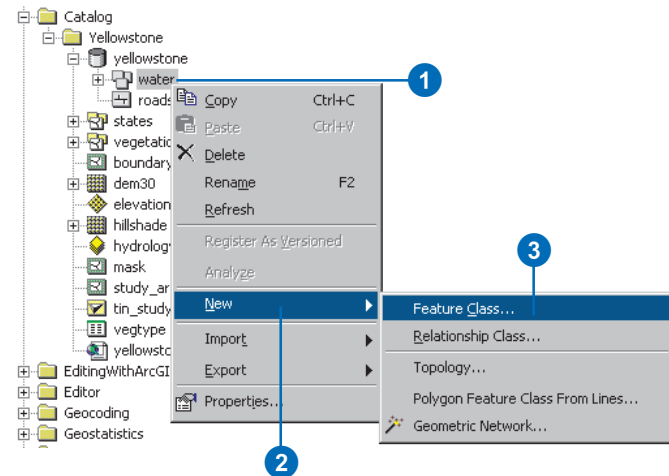
When creating a new network feature class, you must specify a feature type other than simple as well as specify the geometric network in which that feature class will participate. The new feature class must be created in the same feature dataset as the geometric network.

If you create a new junction feature class, you can specify if you want its features to be able to act as sources or sinks.

All network feature classes have the same required fields as simple feature classes—OBJECTID and Shape. In addition to this, network ►

Creating a new network edge feature class

1. Right-click the feature dataset that contains the network.
2. Point to New.
3. Click Feature Class.
4. Type a name and an *alias* for the new feature class.
5. Click the second option to store network objects in the feature class.
6. Click the dropdown arrow and click ESRI Simple Edge Feature to create a feature class that stores simple edges. Click ESRI Complex Edge Feature to create a feature class that stores complex edges.
7. Click the dropdown arrow and click the geometric network in which this feature class will participate.
8. Click Next. ►



The Enabled field records whether or not a feature in the feature class is enabled or disabled in the logical network. This field has a fixed *attribute domain* automatically associated with it.

For more information on how sources, sinks, and enabled and disabled features affect flow in a network, see *Using ArcMap*.

- [illegible]

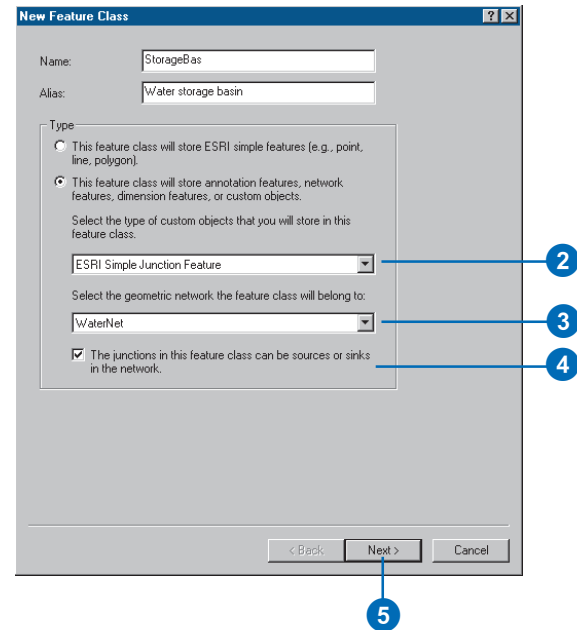
Tip

Sources and sinks

If you specify that you want to store sources and sinks in a junction feature class, a field called `AncillaryRole` will automatically be added to it. If not, then the `AncillaryRole` field will not be included in the feature class.

Creating a new network junction feature class

1. Follow steps 1 through 5 for 'Creating a new network edge feature class'.
2. Click the dropdown arrow and click ESRI Simple Junction Feature to create a feature class that stores network junctions.
3. Click the dropdown arrow and click the geometric network in which this feature class will participate.
4. Check the box to allow the junctions in this feature class to be able to act as sources or sinks in the network.
5. Click Next.
6. Follow steps 9 through 13 for 'Creating a new network edge feature class'.



Network connectivity: defining the rules

In most networks, you do not want all edge types to be able to logically connect to all junction types. Similarly, not all edge types can logically connect to all other edge types through all junction types. For example, in a water network, a hydrant can connect to a hydrant lateral but not to a service lateral. Similarly, in the same water network, a 10-inch transmission main can only connect to an 8-inch transmission main through a reducer.

Network connectivity rules constrain the type of network features that may be connected to one another and the number of features of any particular type that can be connected to features of another type. By establishing these rules, along with other rules such as attribute domains, you can maintain the integrity of the network data in the database. At any time, you can selectively validate features in the database and generate reports as to which features in the network are invalid—that is, are violating one of the connectivity or other rules.

There are two types of connectivity rules: edge–junction and edge–edge rules. An *edge–junction rule* is a connectivity rule that establishes that an edge of one type may connect to a junction of another type. An *edge–edge rule* is a connectivity rule that establishes that an edge of one type may connect to an edge of another type through a set of junction types. Edge–edge rules always involve a set of junctions.

You can establish and modify the connectivity rules for a network from within ArcCatalog by modifying the geometric network properties. You can establish connectivity rules between two feature classes, a feature class and the *subtype* of another feature class, or a subtype of one feature class and a subtype of another. In the water network example above, a connectivity rule would be established between two subtypes of the same edge feature class and a subtype of a third junction feature class (10-inch and 8-inch transmission mains and reducer valves).

Default junctions

Both edge–edge and edge–junction connectivity rules can have default junctions associated with them. Default junctions are automatically inserted by ArcMap when creating connectivity in a network.

When an edge pair has an edge–edge connectivity rule defined in the database and you create a new edge that connects to an existing edge, the default junction is automatically inserted. For an edge–junction connectivity rule, ArcMap automatically inserts the default junction at the free end of new edges that are created in the network.

Establishing connectivity rules

Connectivity rules are established and modified using the geometric network's Properties dialog box in ArcCatalog.

The two examples given here describe how to establish an edge–junction rule and an edge–edge rule. For simplicity, each is done separately, but any number of rules can be established or modified for the network at a single time.

Tip

Junction rules

If an edge–junction rule does not yet exist between one of the edge subtypes or feature classes and one of the junction subtypes or feature classes, a rule is automatically created.

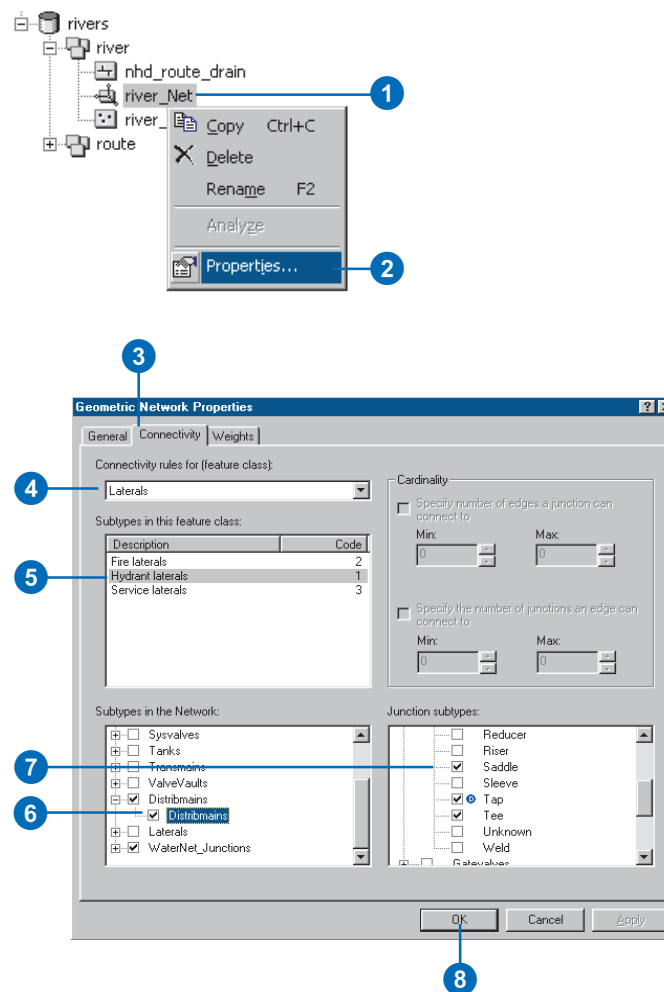
Tip

Default junction type

To set a default junction type, right-click the junction subtype or feature class in the Junction subtypes list, then click Set as default.

Adding an edge–edge rule

1. Right-click the geometric network.
2. Click Properties.
3. Click the Connectivity tab.
4. Click the dropdown arrow and click the feature class for which you want to create a rule.
5. Click the subtype of the feature class if your feature class has subtypes.
6. Navigate to and check the edge feature class or subtype you want to make connectable to this edge subtype or feature class.
7. Browse for and check the junction feature classes and subtypes through which these edge feature classes or subtypes will be permitted to connect.
8. Click OK to create the rule in the database.



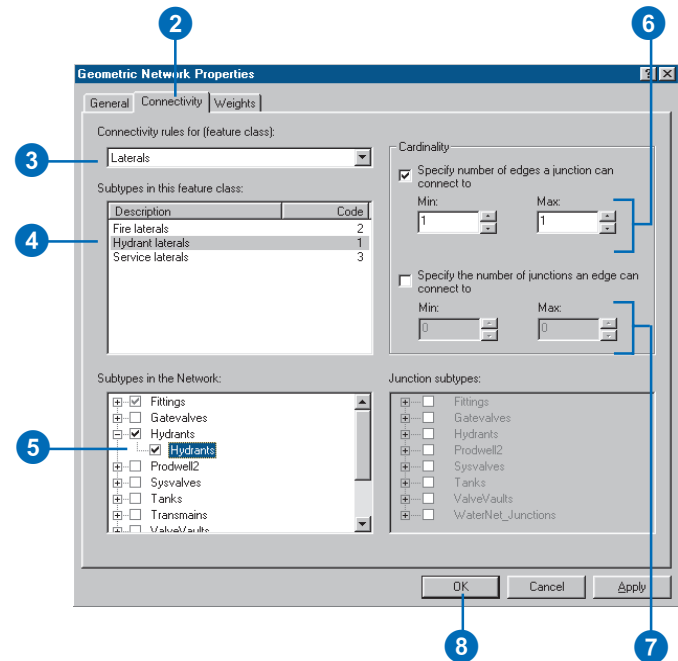
Tip

Default junction type

To set a default junction type, right-click the junction subtype or feature class in the Junction subtypes list, then click Set as default.

Adding an edge–junction rule

1. Follow steps 1 and 2 for 'Adding an edge–edge rule'.
2. Click the Connectivity tab.
3. Click the dropdown arrow and click the feature class for which you want to create a rule.
4. Click the subtype of the feature class if your feature class has subtypes.
5. Navigate and check the junction feature class or subtype you want to make connectable to this edge feature class or subtype.
6. Click the check box and enter the minimum and maximum number of permissible edges if you want to restrict the number of edges of this type that can connect to a single junction of this type.
7. Check the check box and type the minimum and maximum number of permissible junctions if you want to restrict the number of junctions of this type that can connect to a single edge of this type.
8. Click OK to create the rule in the database.



Managing a geometric network

You can manage geometric networks using ArcCatalog. Unlike most items that appear in ArcCatalog, the geometric network does not represent a single entity, such as a table, *shapefile*, or feature class. A geometric network is actually an association among several feature classes and is represented by several tables in the database. Managing a geometric network is different from managing other items in ArcCatalog.

Managing the geometric network itself

Some of the standard operations on the geometric network are handled the same way as with other items in ArcCatalog. A geometric network can be renamed or deleted. Renaming the geometric network doesn't affect any of its member feature classes or the topology of the network itself. However, deleting the geometric network does affect both.

You can delete a geometric network in two ways. The first is to delete the entire feature dataset that contains the network. This action deletes from the geodatabase all of the participating feature classes, all of the network topology tables, and any other objects stored inside that feature dataset. The second method is to simply delete the geometric network itself and leave the rest of the feature dataset intact.

All of the feature classes participating in a network store network feature types. A feature class can't store network features if it's not participating in a network. This means that when the network is deleted, all of the feature classes in the network are demoted to simple feature types. Edge feature classes become simple feature classes with line geometry, and junction feature classes become simple feature classes with point geometry. Deleting the network will also delete all of the related topology tables from the geodatabase.

Managing network feature classes

Managing network feature classes is more restrictive than managing simple feature classes. Although you can easily rename a network feature class, deleting one is more difficult. To delete the network feature class, you must first delete the geometric network; this action converts the network feature class to a simple feature class that can then be deleted. The alternative is to delete the entire feature dataset, which deletes the network and all of the feature classes.

Schema locking

An exclusive lock is required to modify a geometric network's connectivity rules or to rename or delete a geometric network. An exclusive lock can only be acquired for a geometric network if the feature classes that participate in the network can also be locked. Therefore, if another user has an exclusive or shared lock on any of the feature classes in a geometric network, then the properties of the geometric network cannot be edited.

For more information on exclusive locks and schema locking, see the chapter 'Creating new items in a geodatabase' in this book.

Managing annotation

8

IN THIS CHAPTER

- **Annotation in the geodatabase**
- **Annotation and ArcCatalog**
- **Creating annotation classes**
- **Converting labels to annotation**
- **Converting coverage annotation to geodatabase annotation**

In addition to geometry and location, geographic features may also have some descriptive text associated with them. For example, a feature class that contains streets may have text with the street's name associated with it. Annotation may also be a geographically located piece of text that exists independently of any other feature such as the name of a mountain range on a physical map.

Annotation refers either to the process of automating text placement or to the text itself. This chapter describes how to create annotation for your feature classes and how to convert annotation that you have in coverages to geodatabase annotation.

You can create and store annotation in a personal geodatabase with ArcView. To take advantage of advanced annotation functionality, you need an ArcEditor or ArcInfo license.

Annotation in the geodatabase

Annotation in the geodatabase is stored in special feature classes called annotation classes. Unlike points, lines, and polygons, which are stored as ESRI Simple Features, annotation is stored as ESRI Annotation Features.

What is geodatabase annotation?

Like other feature classes in the geodatabase, all features in an annotation class have a geographic location and attributes and can either be inside a feature dataset or a standalone annotation class. Each annotation feature has its own symbology including font, color, and so on. Annotation need not only be text—it can also include shapes such as boxes and arrows.

There are two kinds of annotation in the geodatabase—feature-linked annotation and nonfeature-linked annotation. Nonfeature-linked annotation is geographically placed text strings that are not associated with features in the geodatabase. An example of nonfeature-linked annotation is the text on a map for a mountain range. No specific feature represents the mountain range but it is an area you would want to mark.

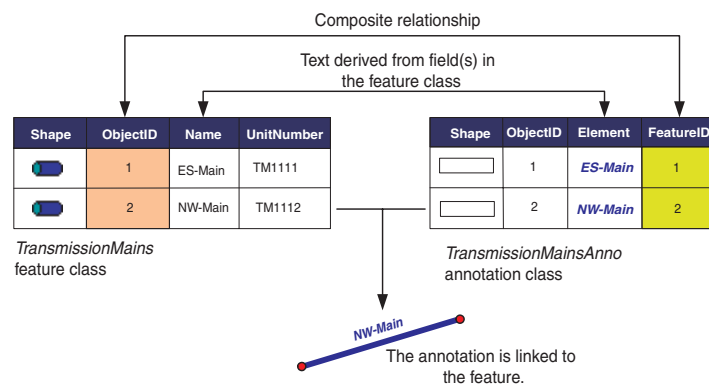
Feature-linked annotation is associated with a specific feature in another feature class in the geodatabase. The text in feature-linked annotation reflects the value of a field or fields from the feature to which it is linked. The annotation feature class participates in a composite relationship with the feature class that it is annotating. The annotation feature class is the destination class in the relationship, while the feature class it is annotating is the origin class. This means the feature controls the location and lifetime of the annotation (see the chapter ‘Defining relationship classes’ in this book).

As an example of feature-linked annotation, a hydrant in a water network may be annotated with its pressure, which is stored in a

field in the feature class. In the same network, the water transmission mains may be annotated with their names.

As with other composite relationships, the origin feature controls the destination feature. Therefore, when the origin feature is moved or rotated, the linked annotation moves or rotates with it. When the origin feature is deleted from the database, its linked annotation feature is also deleted. If the value of a field from which the annotation text is derived changes in the feature, the annotation feature has special behaviors to respond to those changes and automatically update its text string.

In the water network example, a hydrant may be too close to a busy intersection and may need to be moved by 50 feet. When the hydrant is moved, its linked annotation moves with it. In the same network, the name of a transmission main may change. When the value in its name field is modified, the text stored in its linked annotation feature is automatically changed to reflect this.



The link between a feature and its annotation is maintained through a composite relationship. Special behaviors in the ESRI Annotation Features allow you to derive an annotation feature's text from a field or combination of fields in the feature class.

Creating annotation

ArcCatalog contains tools to create both feature-linked and nonfeature-linked annotation classes. You have two options when creating feature-linked annotation classes. You can specify the field in the origin feature class on which to base your annotation. Alternatively, you can use more advanced labeling methods to derive the annotation from multiple fields and specify different labeling rules for different groups of features. Once you have created your annotation class, you can use ArcMap to populate it either on its linked feature or interactively with the drawing tools.

When a new feature is created in a feature class that has linked annotation, an annotation feature is automatically created in the annotation class and is linked to that feature. If the feature has default values associated with the fields from which the annotation is derived, its text will automatically be generated and placed.

Converting annotation

ArcMap also lets you convert labels and annotation stored in a coverage to a geodatabase annotation class. A geodatabase annotation class is created for the labels, and annotation features are inserted into it. When you convert labels to annotation, you can link them to the feature class that was used to create the labels or you can convert them to nonfeature-linked annotation.

When converting coverage annotation, you must first create an annotation class in ArcCatalog. You must specify the symbology of the annotation that will be stored in the geodatabase annotation class by setting up symbology for the coverage annotation in ArcMap. You can specify a different annotation symbol for all the values of the \$SYMBOL pseudo item in the coverage annotation class. During the process of setting the symbology, you should create test plots of your maps to ensure

that the symbology for the annotation is correct before you load the annotation features into the geodatabase. Once the symbology is set, you can then convert the coverage annotation class into the new annotation class.

Pseudo items in the coverage annotation class override the symbology properties you assign your annotation features in ArcMap. Your annotation symbology in ArcMap will be disregarded if it differs from your pseudo items. To avoid this conflict, when loading annotation from a coverage be sure that the symbology in the geodatabase annotation class matches the symbology established in ArcMap for the coverage annotation layer. Alternatively, make sure your pseudo items have values of zero in the coverage. To learn more about symbolizing map layers, see *Using ArcMap*.

You can make your coverage annotation feature-linked when you convert it to geodatabase annotation if you have an item in your coverage annotation's text attribute table (TAT) that relates to a field in the feature class you want to link them to. To create feature-linked annotation for annotations you import from a coverage, you must follow this sequence:

1. Create a new annotation class in ArcCatalog linked to the feature class to which you want to link the annotations.
2. Add your coverage annotation class to ArcMap.
3. Adjust the text symbology, size, and scale. This should match the symbology you specified for the feature-linked annotation in ArcCatalog.
4. Use the Convert Coverage Annotation command in ArcMap, specifying the annotation class you created in step 1 as the target for the coverage annotation.
5. Use Structured Query Language (SQL) to link the features with the loaded annotation.

6. Register your data as versioned (if your geodatabase is stored in ArcSDE).

When creating annotation in an ArcSDE geodatabase by converting labels or converting coverage annotation, you should always try to convert the annotation before registering your data as versioned. Since the data is not versioned, all of the data will be loaded directly into the base tables and a database compress will not be required. For more information about data loading strategies and their versioning impacts, see the chapter ‘Migrating existing data into a geodatabase’ in this book. Additional information on using SQL to link features with loaded annotation can be found at <http://support.esri.com>.

If you are creating annotation in an annotation class that is linked to a network feature class, you should create the annotation after building the geometric network. When features are snapped in the network building process, their geometry is modified at a level at which messages are not sent to linked annotation features to update themselves. So, after building the network, if a feature’s geometry was changed in the snapping process, the feature and its linked annotation may no longer correspond correctly. To learn more about geometric networks, see the chapter ‘Geometric networks’ in this book.

Converting ArcSDE annotation to geodatabase annotation

Annotation stored in an ArcSDE layer that was not created with ArcGIS 8 can also be converted to geodatabase annotation. If you add the ArcSDE annotation layer to ArcMap, you can use the Convert Coverage Annotation command to convert ArcSDE annotation to geodatabase annotation in the same way that you convert coverage annotation to geodatabase annotation.

Performance considerations

Annotation features are both expensive to draw and expensive to retrieve from the database. When working with annotation classes, you should always apply a scale suppression so that the annotation features only draw when you are zoomed in enough to read their text. To learn more about applying scale suppression to map layers, see *Using ArcMap*.

Managing annotation classes

Annotation classes can be managed in the same way as other feature classes and tables. They can be renamed and deleted using ArcCatalog.

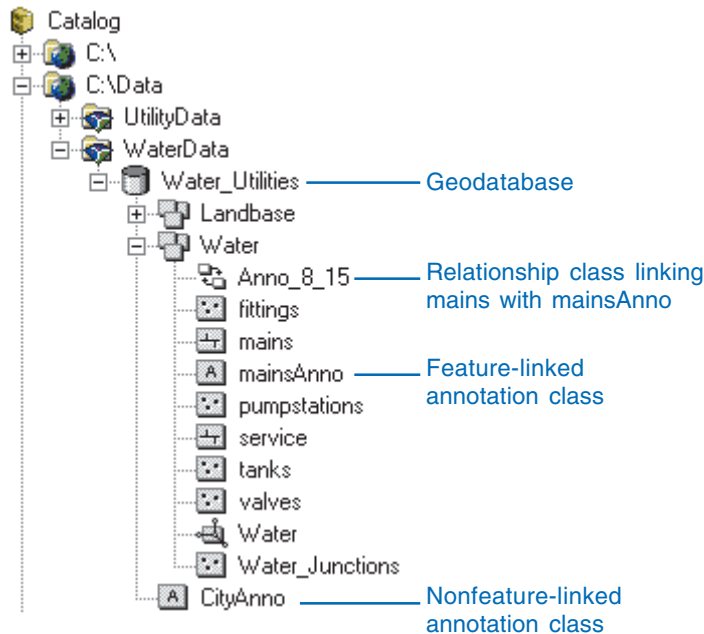
Similarly, the relationship class that links a feature-linked annotation class to its feature class is also managed like any other relationship class. For information on managing relationship classes, see the chapter ‘Defining relationship classes’ in this book.

It is important to note that if the relationship class that links the annotation to its features is deleted, the annotation will no longer be feature-linked but will behave as a nonfeature-linked annotation class. To learn how to re-create this relationship class, see the chapter ‘Defining relationship classes’ in this book.

Annotation and ArcCatalog

If you have access to an annotation class in a geodatabase, you can automatically access it in ArcCatalog. Annotation classes can exist both inside feature datasets and at the root level of the geodatabase. Feature-linked annotation classes that are created inside a feature dataset must be linked to another feature class within the same dataset. Standalone nonfeature-linked annotation classes can be linked to other standalone feature classes.

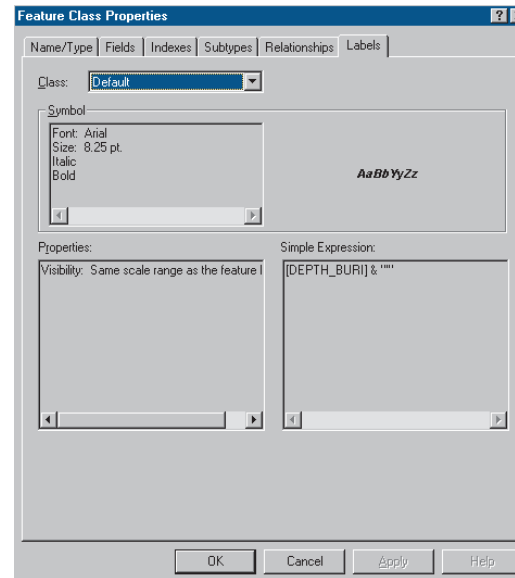
A feature-linked annotation class can only be linked to a single feature class. However, a feature class can have any number of linked annotation classes.



Annotation classes appear in ArcCatalog at either the database or feature dataset level. Feature-linked annotation classes also have a relationship class linking them to the feature class.

When you look at the display of annotation classes in the ArcCatalog tree, it's not immediately evident which ones are feature-linked and which ones are nonfeature-linked. Moreover, simply looking at the Catalog tree won't tell you to which feature class the feature-linked annotation classes are linked. However, examining the properties of the annotation class helps you determine if it is the destination class in a composite relationship. If it is, this indicates it is a feature-linked annotation class.

The Feature Class Properties dialog box of a feature-linked annotation class displays special information about its default symbology, placement relative to the feature, and how its text is derived.



The Labels tab on the Feature Class Properties dialog box displays feature-linked annotation information such as the default symbol and the expression used to derive the annotation text.

Creating annotation classes

Annotation is stored in annotation classes and can either be linked to features or not linked to features. Creating a nonfeature-linked annotation class is similar to creating a feature class that stores custom features.

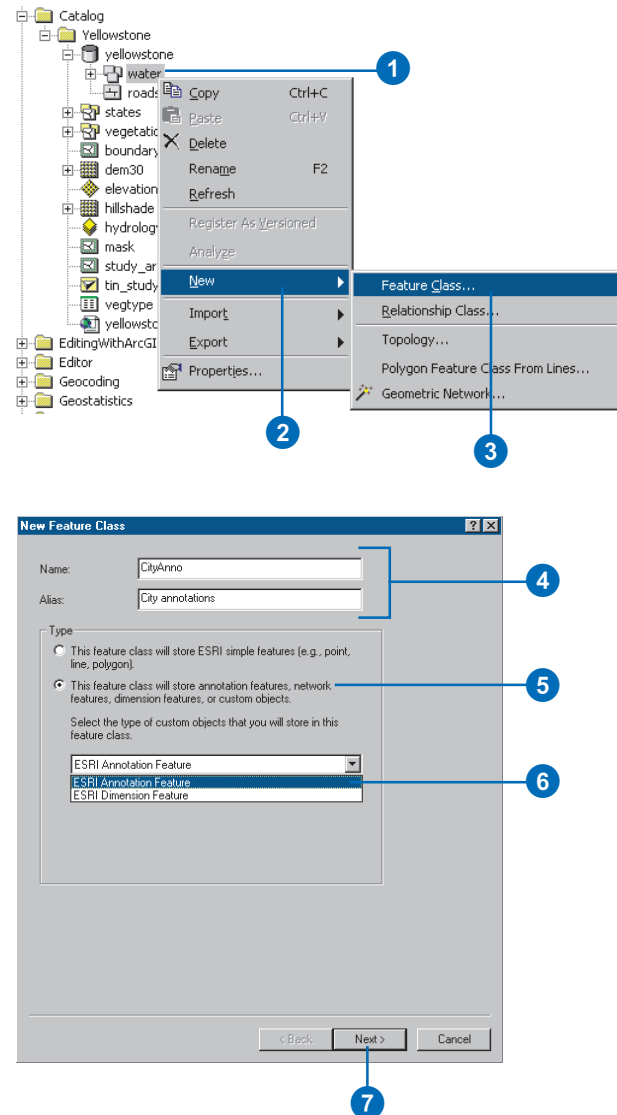
When creating feature-linked annotation, however, you need additional information to create the composite relationship that links the annotation to the features. You must also describe how the annotation is derived based on the feature class's fields.

When you have created a feature-linked annotation class, you can use ArcMap to automatically populate the annotation class with annotation. This annotation is created and placed based on the features to which they are linked. The process of converting feature-linked annotation is described later in this chapter.

Nonfeature-linked annotation does not have a linked feature from which to derive its text and placement. You can still use ArcMap to interactively create and place annotations in the ►

Creating a nonfeature-linked annotation class

1. Right-click the geodatabase or feature dataset in which you want to create the new annotation class.
2. Point to New.
3. Click Feature Class.
4. Type the name for the new annotation class. To create an alias for this annotation class, type the alias.
5. Click the second option to store custom objects in the feature class.
6. Click the dropdown arrow and click ESRI Annotation Feature.
7. Click Next. ►



annotation class. For details on how to create and edit individual annotation features, see *Editing in ArcMap*.

Tip

Required fields

Annotation features require more fields than simple features in order to behave correctly. These additional fields are *Element* and *FeatureID*. Like any required field, these cannot be modified or deleted.

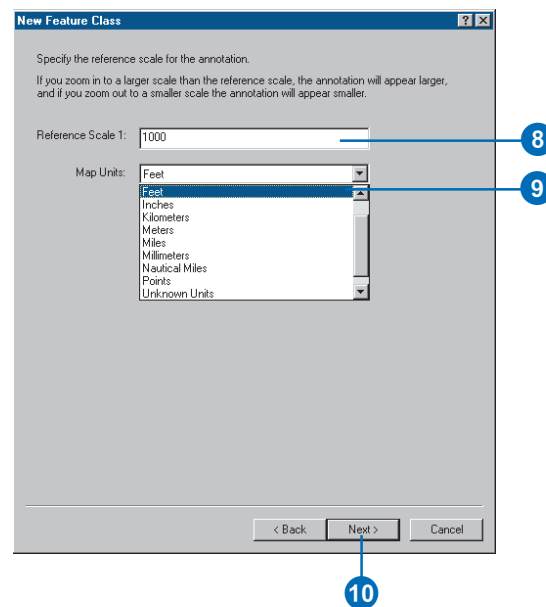
Tip

Reference scale

The reference scale describes the scale at which the annotation text is displayed at the font size specified. As you zoom out, the text will get smaller, and as you zoom in, the text will get larger.

The reference scale should always be in the same units as the spatial reference of the annotation class.

8. Type the scale at which you want the font size displayed.
9. Click the *Map Units* dropdown arrow and click the units for your data.
10. Click Next.
11. Follow steps 6 through 14 for 'Creating a feature class in a feature dataset' as described in the chapter 'Creating new items in a geodatabase' of this book if you are creating this annotation class inside a feature dataset. Follow steps 2 through 12 for 'Creating a standalone feature class' as described in the chapter 'Creating new items in a geodatabase' of this book if you are creating a standalone annotation class.

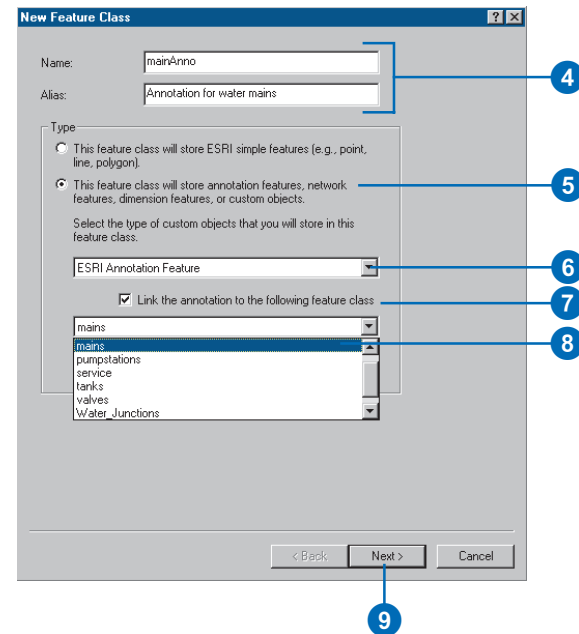
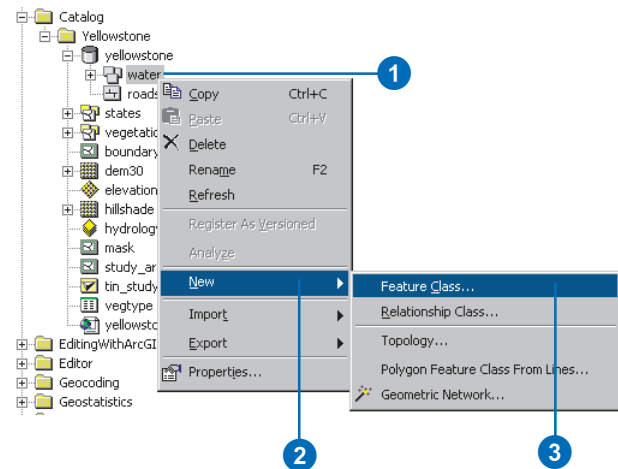


See Also

For a detailed discussion about labeling maps and the advanced labeling methods you can use, see Using ArcMap.

Creating a feature-linked annotation class

1. Right-click the geodatabase or feature dataset in which you want to create the new annotation class.
2. Point to New.
3. Click Feature Class...
4. Type a name for the new annotation class. To create an alias for this annotation class, type the alias.
5. Click the second option to store custom objects in the feature class.
6. Click the dropdown arrow and click ESRI Annotation Feature.
7. Check the check box to link the annotation to a feature class.
8. Click the dropdown arrow and click the feature class to which you want to link this annotation class.
9. Click Next.



Tip

Relationship class

A relationship class will automatically be created to link the annotation class with the feature class it is annotating. The name of the relationship class will be Anno_<feature class ID>_<annotation class ID>.

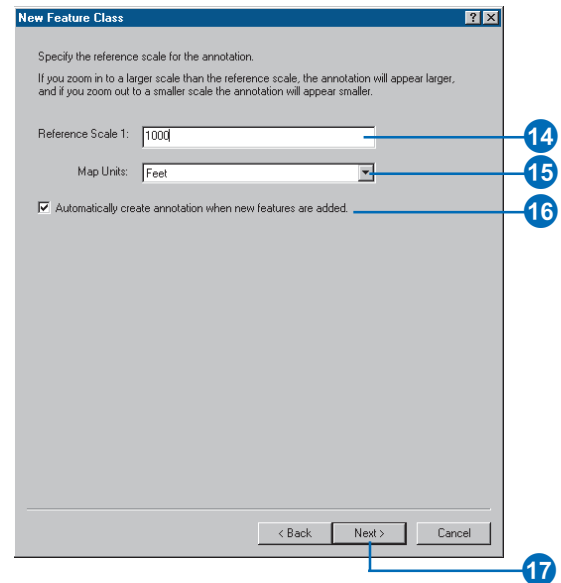
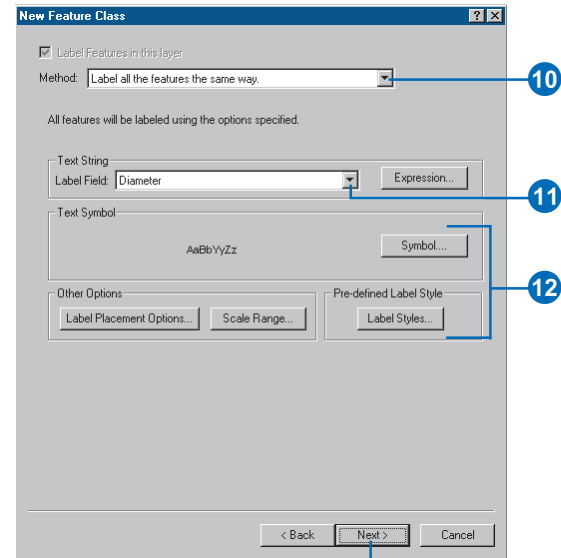
Tip

Reference scale

The reference scale describes the scale at which the annotation text is displayed at the font size specified. As you zoom out, the text will get smaller, and as you zoom in, the text will get larger.

The reference scale should always be in the same units as the spatial reference of the annotation class.

10. Click the Method dropdown arrow and click the method you want to use to create the annotation.
11. Click the dropdown arrow and click the field in the feature class from which you want to create the annotation text or click Expression to create the annotation text from multiple fields.
12. Click Label Styles, Symbol, Label Placement Options, and Scale Range to set some more advanced parameters for your annotation.
13. Click Next.
14. Type the scale at which you want the font size displayed.
15. Click the Map Units dropdown arrow and click the units for your data.
16. Check the check box if you want an annotation feature to be created automatically every time a new feature is created in the feature you selected in step 8.
17. Click Next.
18. Follow steps 6 through 14 for 'Creating a feature class in a feature dataset' as described in the chapter 'Creating new items in a geodatabase' of this book.



Tip

Versioning

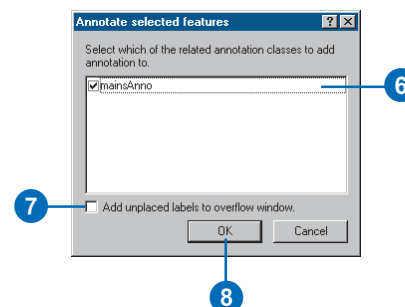
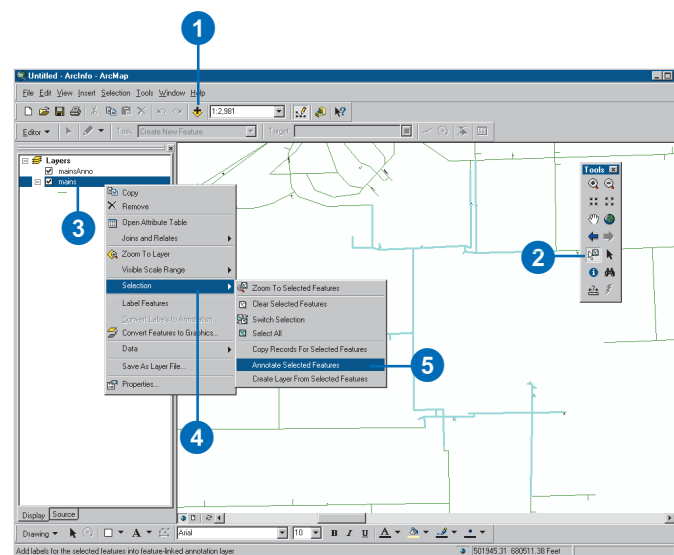
When possible, generate your annotation before you version your data.

See Also

For more information on how to use ArcMap to add feature classes to maps and how to select features, see Using ArcMap.

Generating feature-linked annotation

1. Click the Add Data button in ArcMap to add a feature class and its linked annotation class to your map.
2. Click the Select Features button to select the features for which you want to generate annotation. To create annotation for all of the features, select all of the features.
3. Right-click the feature class in the table of contents.
4. Point to Selection.
5. Click Annotate Selected Features.
6. Check the related annotation classes in which you want to store the annotation.
7. Check the check box to add unplaced labels to the overflow window.
8. Click OK.



Converting labels to annotation

ArcMap lets you label features stored in a feature class. Once you have created labels, you can store them within the ArcMap document or as graphics in a database, or you can convert them to annotation features.

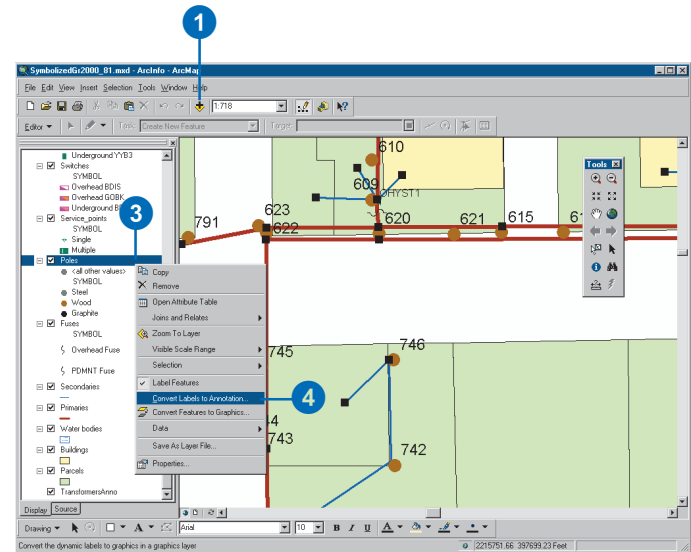
If you choose to convert them to annotation features, ArcMap will create an annotation class to store the annotation. If you specify that you want them linked to the features, ArcMap will also create the relationship class to maintain the link.

The annotation class and relationship class are created inside the same feature dataset in which the feature class is stored or at the geodatabase level for a standalone feature class.

See Also

For a detailed discussion about labeling maps and the different advanced labeling methods you can use, see Using ArcMap.

1. Click the Add Data button in ArcMap to add the feature class for which you want to create annotation to your map.
2. Label the features in your map as described in *Using ArcMap*.
3. Right-click the feature class in the table of contents.
4. Click Convert Labels to Annotation.



Tip

New annotation class

When all of the labels have been converted to annotation, the new annotation class is automatically added to the map.

Tip

Versioning

When possible, convert your labels to annotation before you version your data.

Tip

Annotation storage

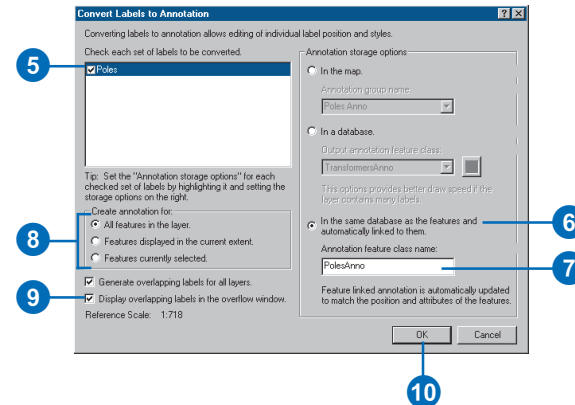
When storing geodatabase annotation in a DBMS, the row length is between 80 and 100 bytes.

5. Click the feature class for which you want to save labels.
6. Click the third option under Annotation storage options.
7. Type a name for the new annotation class that will be created to store the annotation.
8. Click All features in the layer to create annotation for all of the features.

Click Features displayed in the current extent to create annotation for features displayed in the current extent of the map.

To create annotation for the selected features only, click Features currently selected.

9. Click Display overlapping labels in the overflow window to display annotation that cannot be created without overlapping others.
10. Click OK.



Converting coverage annotation to geodatabase annotation

ArcMap lets you convert annotation stored in a coverage annotation feature class to geodatabase annotation. You must convert the annotation into an existing annotation class in your geodatabase.

If the coverage annotation feature class contains attributes, when you convert it to geodatabase annotation, these attributes will automatically be converted as well.

The command to convert coverage annotation is available through the Customize dialog box in ArcMap.

See Also

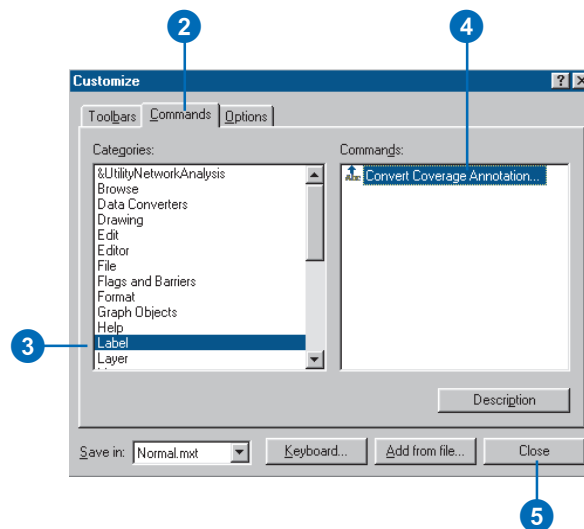
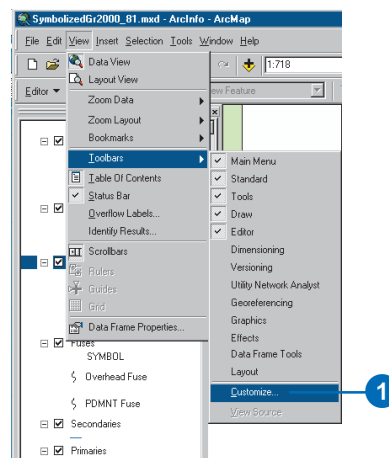
For more information on how you can customize ArcMap, see *Using ArcMap and Exploring ArcObjects*.

Adding the Convert Coverage Annotation command to ArcMap

1. Click View in ArcMap, point to Toolbars, and click Customize.
2. Click the Commands tab in the Customize dialog box.
3. Click the Label category.
4. Drag the Convert Coverage Annotation 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.

Tip

Annotation class

You can only convert coverage annotation into existing annotation classes. To create an annotation class, see the 'Creating a nonfeature-linked annotation class' and the 'Creating a feature-linked annotation class' tasks in this chapter.

Tip

Annotation storage

When storing geodatabase annotation in a DBMS, the row length is between 80 and 100 bytes.

Tip

Target annotation class

The annotation class into which you convert your coverage annotation must exist in the geodatabase before you perform the conversion. You can create a new annotation class using ArcCatalog.

See Also

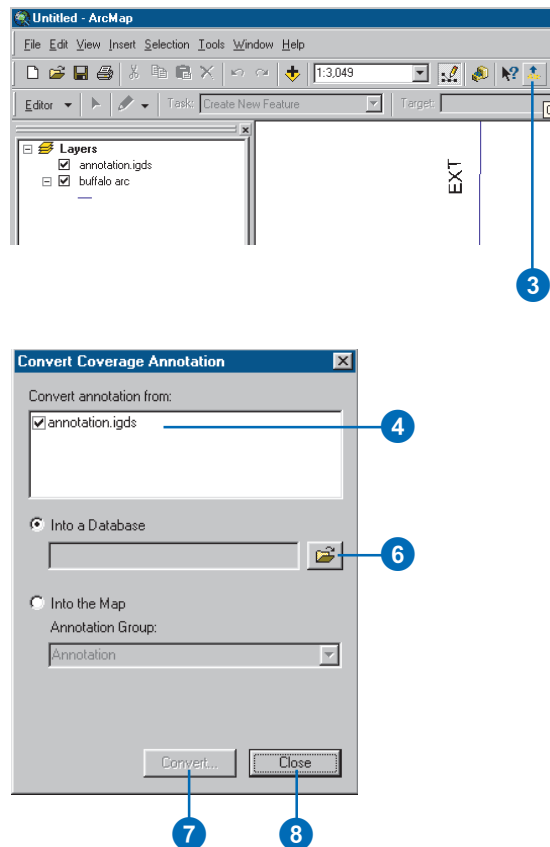
For more information on using ArcMap to add feature classes to maps, symbolize annotation, and create plots, see Using ArcMap.

See Also

For more information on symbolizing layers in ArcMap, see Using ArcMap.

Converting coverage annotation

1. Add to your map the coverage annotation feature classes you want to convert.
2. Set the symbology for the coverage annotation feature class. Create test plots of your data to ensure the symbology is correct before continuing to step 3.
3. Click the Convert Coverage Annotation command.
4. Check the annotation classes that you want to convert. You can convert multiple coverage annotation classes into a single geodatabase annotation class.
5. Click the second option to convert the annotation into a database.
6. Click the Browse button to browse for an existing annotation class within your geodatabase.
7. Click Convert.
8. Click Close to close the Convert Coverage Annotation dialog box when your conversion is complete.



Dimensioning

9

IN THIS CHAPTER

- Dimensions in the geodatabase
- Dimensions and ArcCatalog
- Creating dimension feature classes
- Creating and managing dimension styles

For many applications, plotting a map that shows a feature's shape and some descriptive annotation isn't sufficient for showing precise measurements or distances. Dimensions are a special kind of map annotation for showing just that—specific lengths or distances. A dimension may indicate the length of a side of a building or land parcel, or it may indicate the distance between two features—for example, a fire hydrant and the corner of a building.

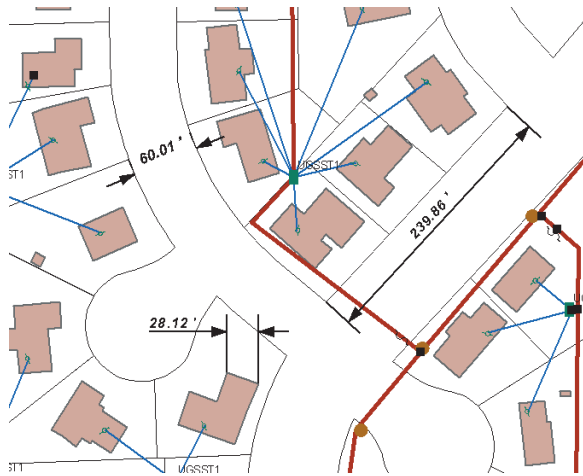
ArcInfo provides tools that allow you to store dimensions in your geodatabase. You can also create dimension styles to apply to your *dimension features* so they are consistent with your application standards. ArcInfo supports a number of dimension types and methods for creating dimensions. Dimensions can be created automatically from existing features, or you can use the rich set of construction tools available in ArcMap.

This chapter describes how to create *dimension feature classes* and *dimension styles*. *Editing in ArcMap* contains a chapter dedicated to using the editing capabilities of ArcMap to create and modify dimension features.

You can view dimension feature classes in ArcView. To take advantage of other dimensioning functionality, you need an ArcEditor or ArcInfo license.

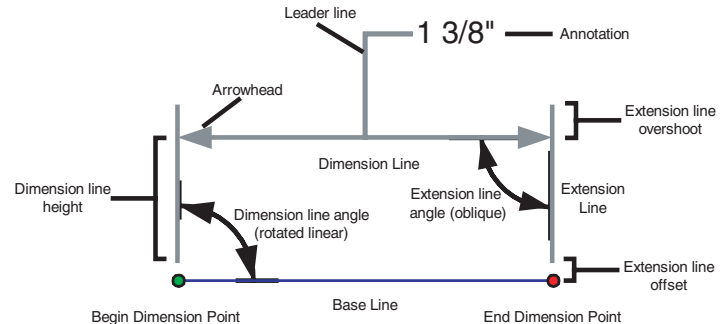
Dimensions in the geodatabase

Dimensions are a special kind of map annotation that show specific lengths or distances on a map. A dimension may indicate the length of a side of a building or land parcel or it may indicate the distance between two features such as a fire hydrant and the corner of a building. Dimensions can be as simple as a piece of text with a leader line or as elaborate as the diagram shown opposite.



Dimensions show specific lengths or distances on a map. For example, a dimension may indicate the length of a side of a building, the width of a street, or the length of a parcel.

A dimension feature is composed of several parts that may or may not be displayed, depending on the application. The following is an illustration of the anatomy of a dimension feature (throughout this chapter, these parts will be referred to by the names in this diagram):

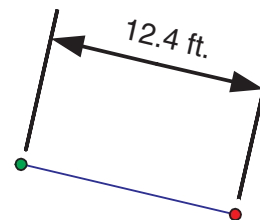


A dimension feature is composed of many parts. Each dimension feature can represent each part differently by using different symbology and placement rules. Dimensions may also display a subset of these parts.

Dimension types

ArcInfo supports two types of dimensions: aligned and linear. *Aligned dimensions* run parallel to the baseline and represent the true distance between the begin and end dimension points.

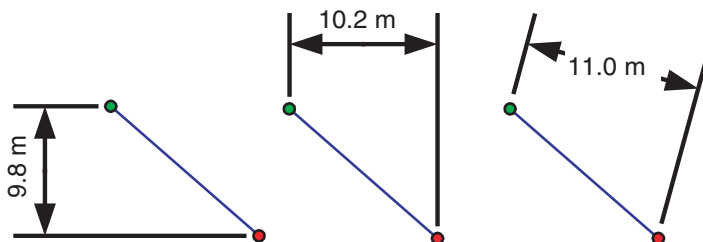
Unlike aligned dimensions, *linear dimensions* don't represent the true distance between the begin and end dimension points. Linear dimensions can be vertical, horizontal, or rotated. A vertical dimension's line represents the vertical distance between the begin and end dimension points. A horizontal linear dimension's



An aligned dimension has its dimension line parallel to the baseline, and its length represents the true distance between the begin and end dimension points.

line represents the horizontal distance between the begin and end dimension points. A rotated linear dimension is a dimension whose line is at some angle to the baseline and whose length represents the length of the dimension line itself, not the baseline.

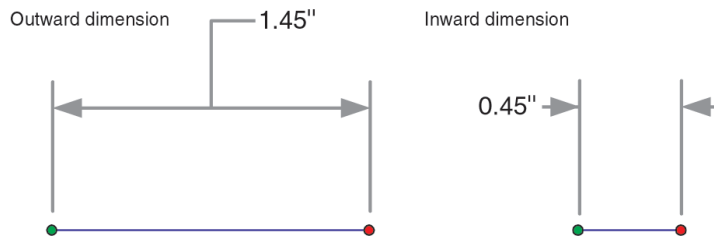
All dimensions can be oriented either outward or inward. Outward dimensions have dimension lines pointing to the outside of the feature and represent the distance being measured. Inward dimensions have arrows pointing in from the outside of the



Linear dimensions may be vertical, horizontal, or rotated. In each case the dimension's length represents something other than the true distance between the begin and end dimension points.

feature and measure the distance between these two arrows. Whether a dimension is outward or inward is determined by the distance that the dimension represents and whether that distance on the map is sufficient to display all of the elements of the dimension between the extension lines.

To learn more about the tools and construction methods used to create all of these types of dimensions while editing in ArcMap, see the chapter ‘Editing dimension features’ in *Editing in ArcMap*.



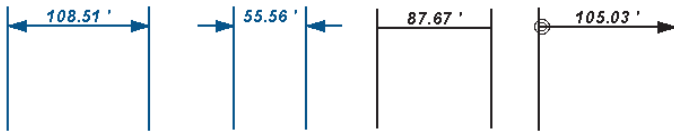
A dimension feature can be either outward or inward. Whether a dimension is inward or outward usually depends on its length and symbology.

Dimension feature classes

In the geodatabase, dimensions are stored in dimension feature classes. Like other feature classes in the geodatabase, all features in a dimension feature class have a geographic location and attributes and can either be inside or outside of a feature dataset. Like annotation features, dimension features are “smart”; they act within the parameters of a predefined style and are able to determine their own symbology and how the dimension features themselves should be drawn.

Dimension styles

A collection of dimension styles is associated with a dimension feature class. A dimension feature’s style describes its symbology, what parts of it are drawn, and how it is drawn. Every time you create a new dimension feature, it is assigned a particular style. All dimension features of a particular style share certain characteristics, some of which can be changed on a feature-by-feature basis. Styles for a dimension feature class are created, copied, and managed using ArcCatalog. Using the editing capabilities in ArcMap, styles are then assigned to individual dimension features.



Dimension style examples. All dimension features are associated with a style. The style describes how the dimension feature is symbolized and the content of the dimension's text.

The following are the properties of a dimension feature that can be assigned based on a style:

- Dimension line symbol: the symbol used for the dimension line.
- Begin symbol: the symbol used for arrow at the end of the begin dimension line.
- End symbol: the symbol used for the arrow at the end of the end dimension line.
- Dimension line display: indicates which of the dimension lines should be displayed—both, begin only, end only, or neither.
- Arrow display: indicates which dimension line arrows should be displayed—both, begin only, end only, or neither.
- Extension line symbol: the symbol used for the extension lines.
- Extension line display: indicates which of the extension lines should be displayed—both, begin only, end only, or neither.
- Offset and overshoot: the distance that the extension lines are drawn from the dimension points and how far they extend beyond the dimension line, respectively.
- The value to display for the text: The actual string that is displayed for the dimension text. This may be derived from the feature itself or from a user-specified value or string. The value may be in map units or converted and displayed in other units.

- Arrow fit and text fit: The adjustment in the display of the arrows and text when the dimension's length is too short to show the arrows and text between the extension lines.

ArcInfo provides tools that allow you to store dimensions in your geodatabase. You can also create dimension styles to apply to your dimension features so they are consistent with your application standards. ArcInfo supports a number of dimension types and methods for creating dimensions. Dimensions can be created automatically from existing features, or you can use the rich set of construction tools available in the ArcMap editing environment.

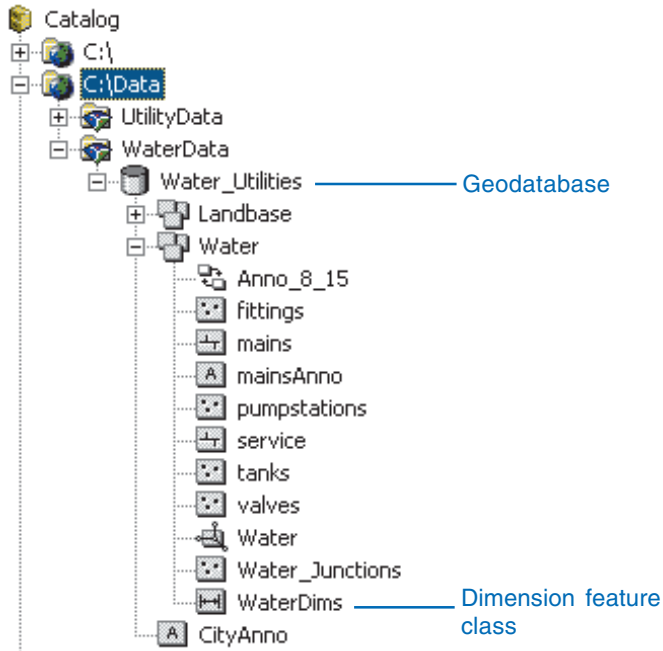
Once a style is created in a dimension feature class, it cannot be modified. If you want to modify the properties of an existing dimension style, you must create a new style with the new properties. You can create new styles based on the properties of an existing style, or you can import styles from dimension feature classes in other geodatabases. For more information on ArcMap editing capabilities, see *Editing in ArcMap*.

Performance considerations

Dimensions are a type of map annotation. Like regular annotation, the information that dimension features convey is not very useful unless you have projected your map at a scale in which you can visualize the dimension features clearly. Like annotation features, dimension features are costly to retrieve from the database and draw on the map display. When working with dimension feature classes in ArcMap, you should always apply scale suppression so that the dimension features only draw at scales in which they can be visualized. For more information on layers and scale suppression, see *Using ArcMap*.

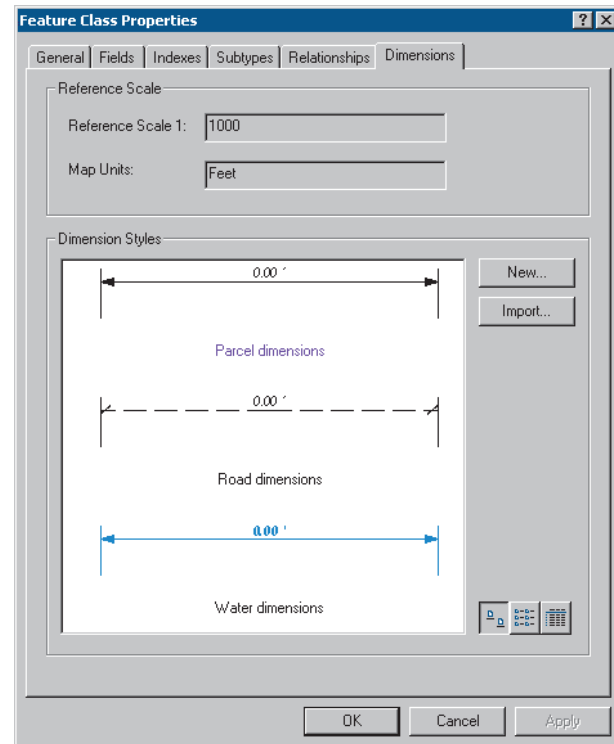
Dimensions and ArcCatalog

In ArcCatalog, you can work with a dimension class in any accessible geodatabase. Dimension feature classes can exist both inside a feature dataset and at the root level of the geodatabase.



Dimension feature classes appear in ArcCatalog at either the database or feature dataset level.

You can use ArcCatalog to create and manage dimension feature classes. The Feature Class Properties dialog box displays special information about the dimension feature's styles and at what scale those styles are displayed with their specific symbol sizes. You can use the Feature Class Properties dialog box to create new styles, delete styles, and import dimension styles from other dimension feature classes.



The Dimensions tab on the Feature Class Properties dialog box displays dimension information, such as the default style and the reference scale, for the dimension features.

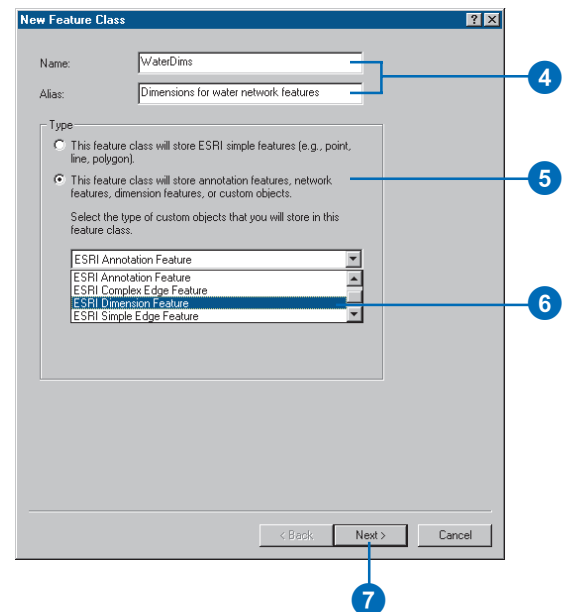
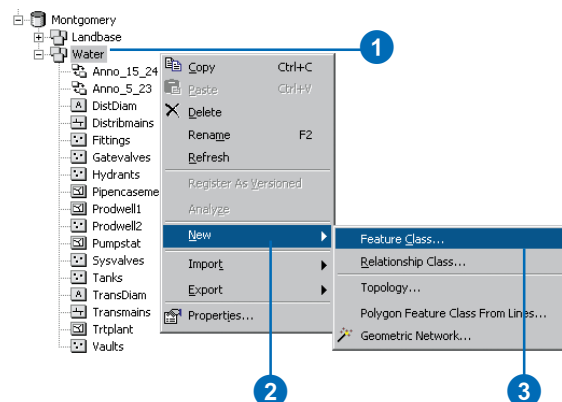
Creating dimension feature classes

Dimension features are stored in dimension feature classes. When creating a dimension feature class, you must create at least one style for the dimension features you will create. You can specify the properties of the style yourself, import the style from another dimension feature class, or let the wizard create a default style for you.

Once you have created a dimension feature class, you can use ArcCatalog to create and import additional styles.

Creating a dimension feature class with the default style

1. Right-click the geodatabase or feature dataset in which you want to create the new dimension class.
2. Point to New.
3. Click Feature Class.
4. Type the name for the new dimension feature class. To create an alias for this feature class, type the alias.
5. Click the second option to store custom objects in the feature class.
6. Click the dropdown arrow and click ESRI Dimension Feature.
7. Click Next. ►

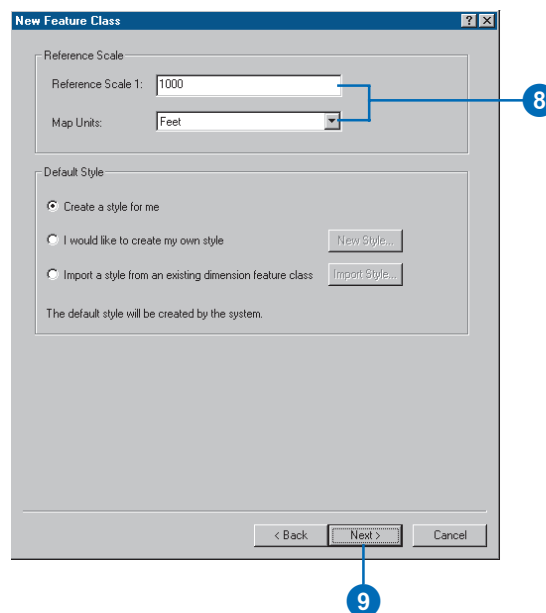


Tip

Reference scale

The reference scale describes the scale at which the symbology of the dimension feature is the same size as described in the style. For example, if your dimension text is Arial 12 pt. and your reference scale is 1:1,000, then your text will be 12 pt. at 1:1,000. As you zoom out from this scale, the text becomes smaller, and as you zoom in from this scale, the text becomes larger.

8. Type a reference scale. The reference scale units will automatically match the spatial reference's units if the dimension class is being created in a feature dataset. If this is a standalone dimension class, then you should pick the units for your spatial reference, which you will specify later in the wizard.
9. Click Next.
10. Follow steps 6 through 14 for 'Creating a feature class in a feature dataset' in the chapter 'Creating new items in a geodatabase' in this book if you are creating this dimension feature class inside a feature dataset. Follow steps 2 through 12 for 'Creating a standalone feature class' in the chapter 'Creating new items in a geodatabase' in this book if you are creating a standalone dimension feature class.

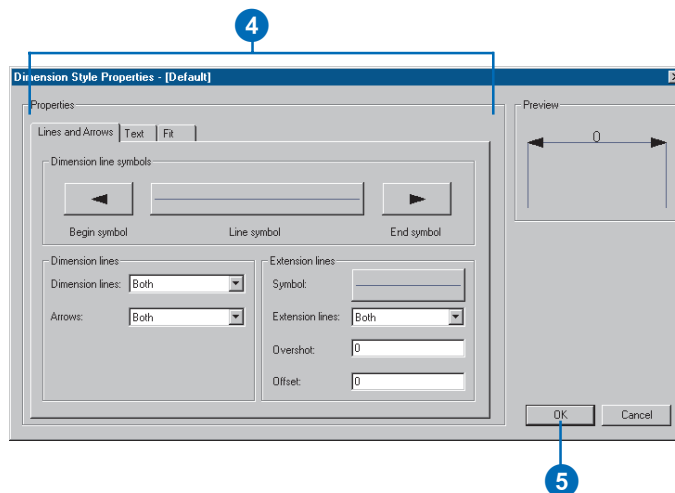
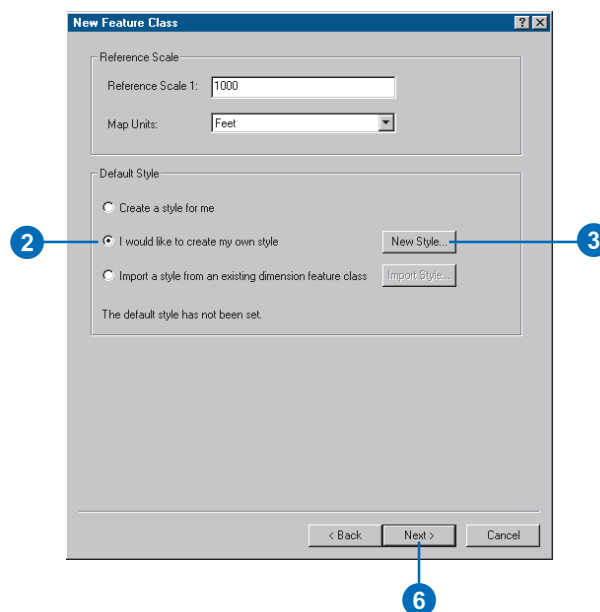


See Also

For more information on what each style element is and how to create styles, see 'Creating and managing dimension styles' later in this chapter.

Creating a dimension feature class with a custom style

1. Follow steps 1 through 8 for 'Creating a dimension feature class with the default style'.
2. Click the second option to create your own style.
3. Click New Style to open the style properties dialog box.
4. Use the Dimension Style Properties dialog box to set the characteristics of your dimension style.
5. Click OK.
6. Click Next.
7. Follow steps 6 through 14 for 'Creating a feature class in a feature dataset' in the chapter 'Creating new items in a geodatabase' in this book if you are creating this dimension feature class inside a feature dataset. Follow steps 2 through 12 for 'Creating a standalone feature class' in the chapter 'Creating new items in a geodatabase' in this book if you are creating a standalone dimension feature class.



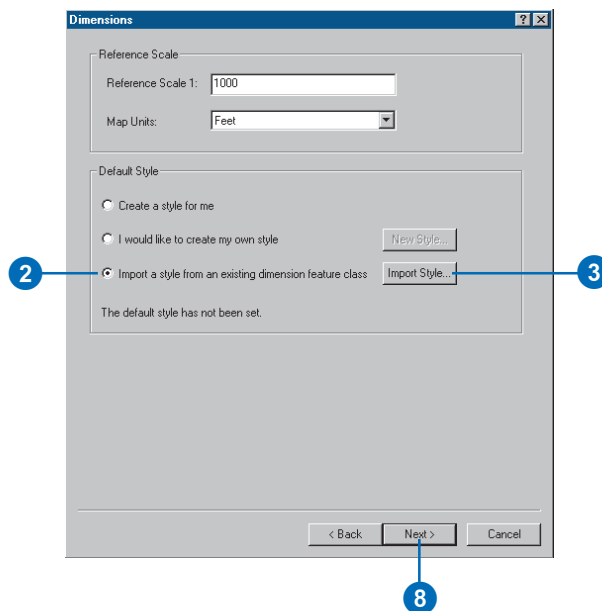
Tip

Browsing styles

You can browse styles by looking at an example dimension; or you can also click the View Options button at the bottom of the Import dialog box to switch the view to a list of the style names or the style names and style IDs.

Creating a dimension feature class by importing a style

1. Follow steps 1 through 8 for 'Creating a dimension feature class with the default style'.
2. Click the third option to import a style from an existing dimension feature class.
3. Click Import Style to browse for the dimension feature class from which you want to import a style. ►

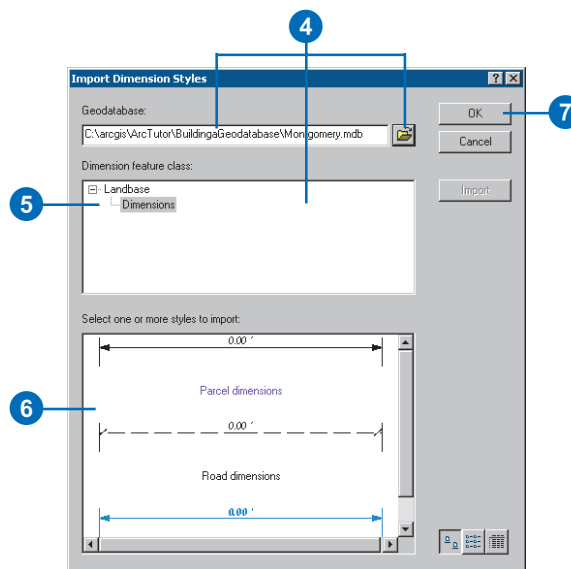


Tip

Importing more than one style

You can import multiple styles from multiple dimension classes by opening the property page for an existing dimension class. See 'Creating and managing dimension styles' later in this chapter.

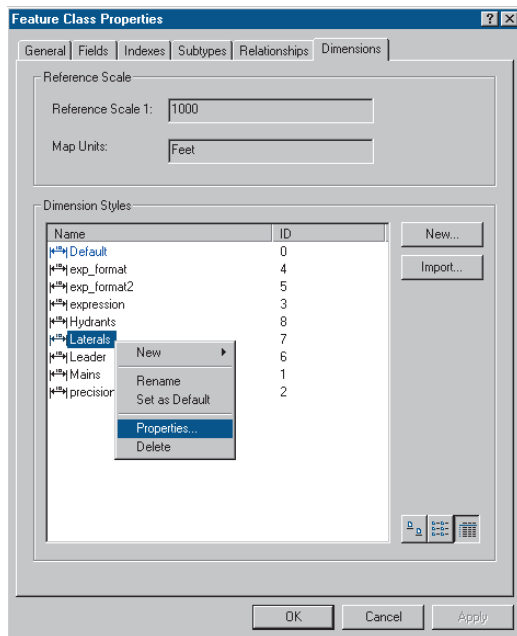
- Click the Browse button to browse for a geodatabase. Once a geodatabase is selected, the dimension feature classes and feature datasets containing dimension feature classes are listed in the tree view.
- Click the dimension feature class that contains the style you want to copy.
- Click the dimension style that you want to copy.
- Click OK.
- Click Next.
- Follow steps 6 through 14 for 'Creating a feature class in a feature dataset' in the chapter 'Creating new items in a geodatabase' in this book if you are creating this dimension feature class inside a feature dataset. Follow steps 2 through 12 for 'Creating a standalone feature class' in the chapter 'Creating new items in a geodatabase' in this book if you are creating a standalone dimension feature class.



Creating and managing dimension styles

A dimension style describes how a dimension feature is displayed including its symbology, label font, and label text. Each dimension feature class has at least one style. Dimension features within the dimension feature class are associated with a particular style. All dimension features of a particular style have certain characteristics that are the same, while other characteristics can be overridden on a feature-by-feature basis.

Dimension styles are created and managed in ArcCatalog in the Feature Class Properties dialog box. You can create, delete, rename, and import dimension styles, and you can specify the default style for a dimension feature class.



Dimension styles are created and managed in ArcCatalog in the Feature Class Properties dialog box.

The dimension styles properties are defined in the Dimension Style Properties dialog box. This dialog box has three tabs: Lines and Arrows, Text, and Fit. The Lines and Arrows tab allows you to set the properties for the dimension lines, line arrows, and extension lines. The Text tab allows you to control the content of the dimension text as well as its symbology. The Fit tab allows you to define how the dimension and dimension text adjust when the dimension's length is too short to display the arrows and text between the extension lines. Each tab and property on the Dimension Style Properties dialog box is discussed in the following pages.

When creating dimension features and assigning them a style, some properties can be overridden on a feature-by-feature basis. The properties that you can override for each feature are:

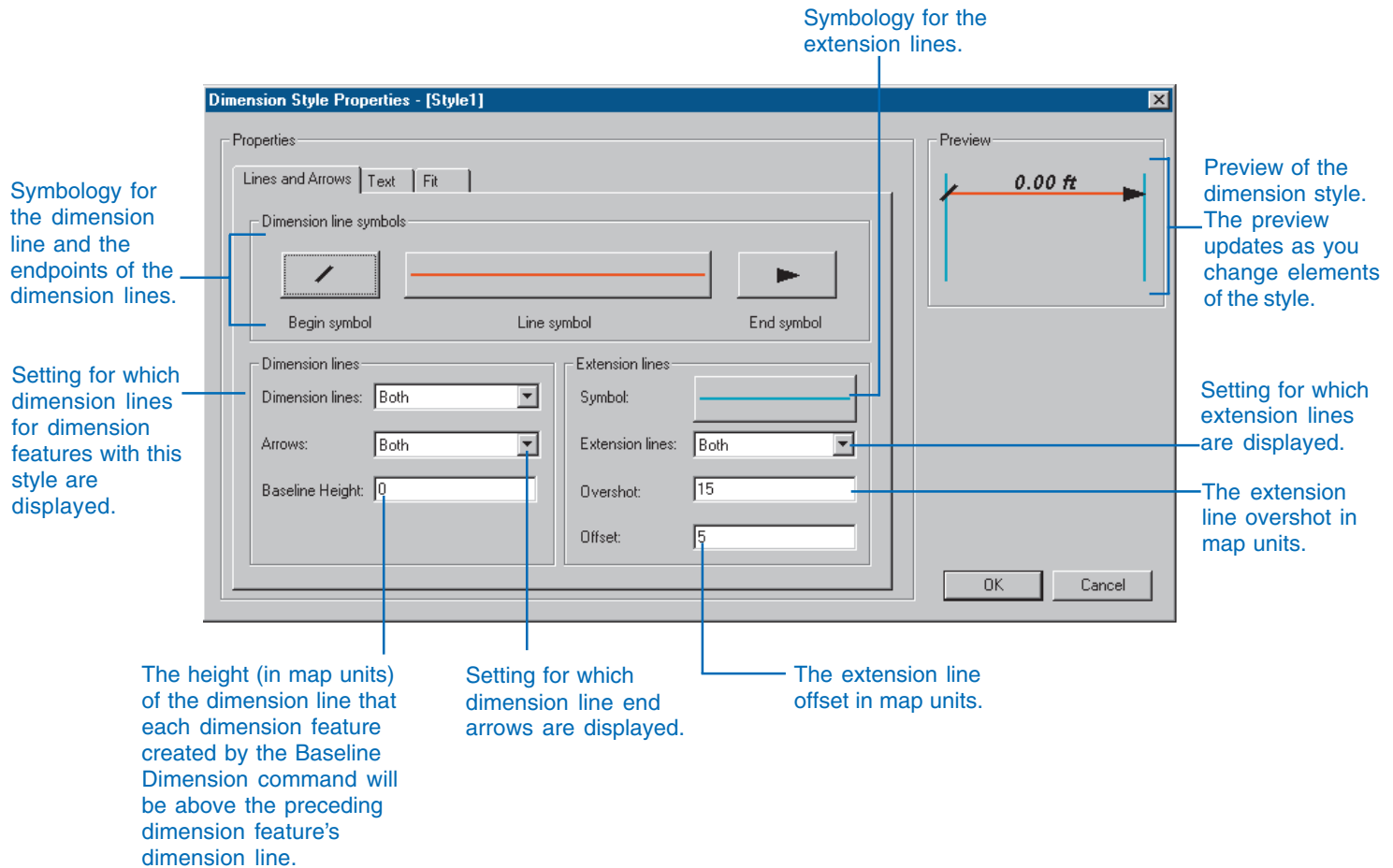
- Dimension line display
- Dimension line arrow symbol display
- Dimension text value
- Extension line display

For more information on editing dimension features, see *Editing in ArcMap*.

Schema locking

An exclusive lock is required when creating, renaming, or deleting styles in a dimension feature class. For more information on schema locking, see the chapter 'Creating new items in a geodatabase' in this book.

Lines and Arrows tab



Text tab

The screenshot shows the 'Dimension Style Properties - [Style1]' dialog box with the 'Text' tab selected. The 'Display' section on the left has four radio buttons: 'Value only' (unselected), 'Add prefix and/or suffix' (selected), 'Advanced' (unselected), and 'Do not display text' (unselected). Below these are fields for 'Prefix' and 'Suffix' (set to 'ft'), and an 'Expression...' button. A checked checkbox 'Align to dimension line' is at the bottom of this section. The 'Symbol' section in the center shows a preview of the dimension text 'AaBbYyZz 0123456789' and a 'Symbol...' button. The 'Units' section at the bottom right has a 'Display units' dropdown set to 'Map units' and a 'Precision' dropdown set to '0.00'. A 'Preview' window on the top right shows a dimension line with the text '0.00 ft'. Blue callout lines point from descriptive text to specific UI elements: 'The actual string that is displayed for the dimension text. The text label can be turned off.' points to the 'Display' radio buttons; 'Setting for whether to align the text with the dimension line. If this is checked, the text will always be parallel to the dimension line. If this is unchecked, the text will be oriented as is described by the properties of its symbol.' points to the 'Align to dimension line' checkbox; 'Setting for the precision of the value displayed by the dimension text. The value will be rounded off to the precision selected.' points to the 'Precision' dropdown; 'The units of the dimension text. If set to Map units, the dimension text value will be in the units described by the dimension feature class's coordinate system. If a value other than map units is specified, then the dimension value will automatically be converted to those units at display time.' points to the 'Display units' dropdown; and 'Symbology for the dimension text.' points to the 'Symbol...' button.

The actual string that is displayed for the dimension text. The text label can be turned off.

Setting for whether to align the text with the dimension line. If this is checked, the text will always be parallel to the dimension line. If this is unchecked, the text will be oriented as is described by the properties of its symbol.

Setting for the precision of the value displayed by the dimension text. The value will be rounded off to the precision selected.

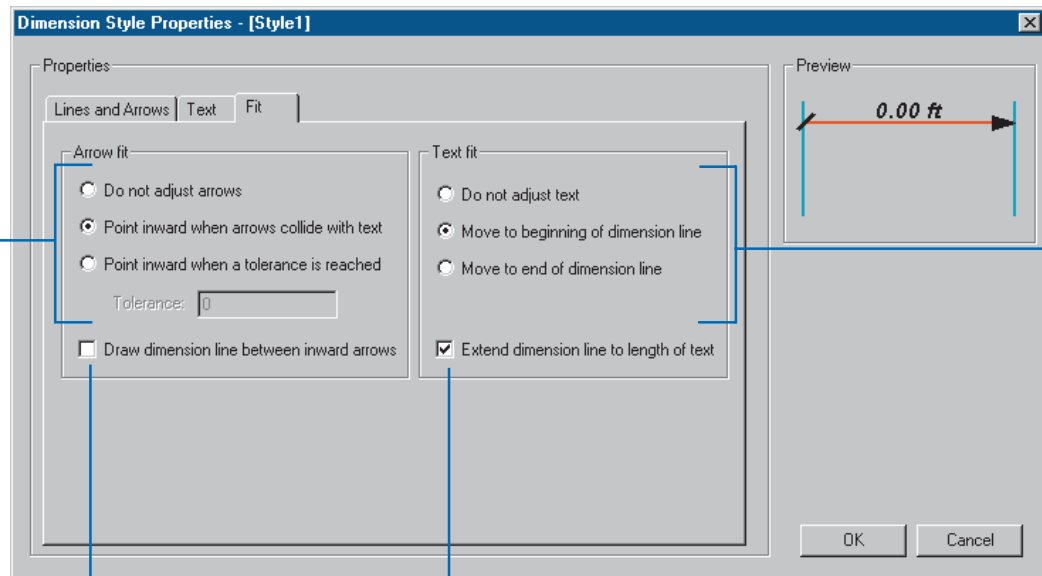
The units of the dimension text. If set to Map units, the dimension text value will be in the units described by the dimension feature class's coordinate system. If a value other than map units is specified, then the dimension value will automatically be converted to those units at display time.

Symbology for the dimension text.

Fit tab

The behavior of the dimension when its length is too short for the dimension line arrows and the text to fit between the extension lines.

Setting for how to resolve the case where the dimension is too short for the dimension text to fit between the extension lines.



Setting for whether to draw the dimension line between the arrows for inward dimensions.

Setting for whether to extend the dimension line to underline the dimension text when it is moved to the outside of the dimension.

Tip

New dimension styles

Newly created dimension styles' names are colored red, indicating that they have not yet been written to the database. Once you click *Apply* or *OK*, these dimension styles' names are colored black to indicate they are in the database.

Tip

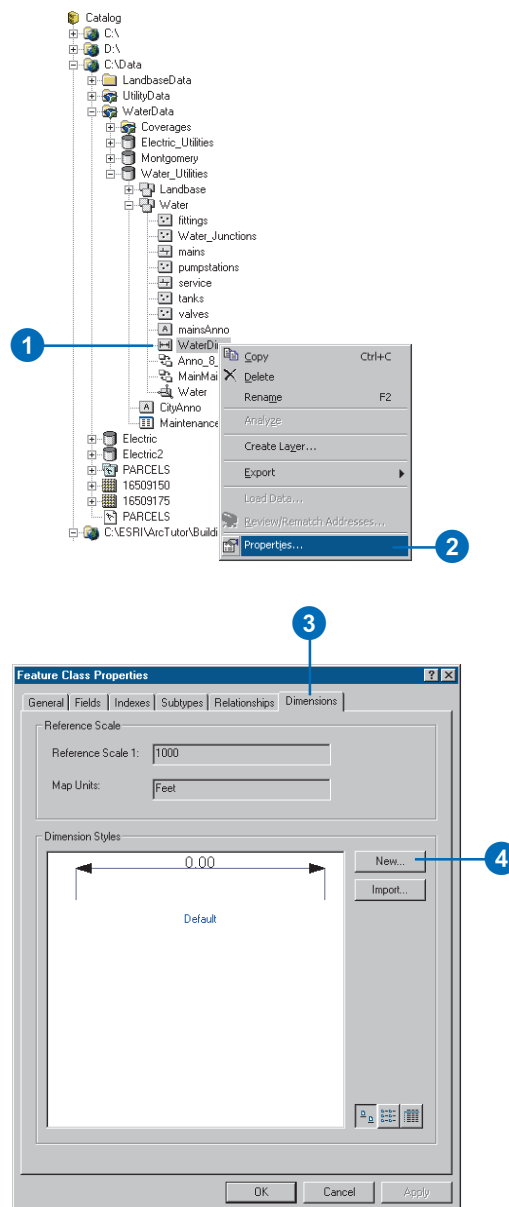
Style IDs

Each dimension style has an ID. A dimension feature stores this ID in a field called *STYLEID*. You can create and apply default values and domains to this field at the feature class level or the subtype level. You can do this as an alternative to using the *Dimension* toolbar in *ArcMap* to assign styles to your dimension features.

For more information on default values, domains, and subtypes, see the chapter 'Subtypes and attribute domains'; and for more information on the *Dimensioning* toolbar and editing dimension features, see *Editing in ArcMap*.

Creating a new dimension style

1. Right-click the dimension feature class.
2. Click *Properties*.
3. Click the *Dimensions* tab.
4. Click *New*.



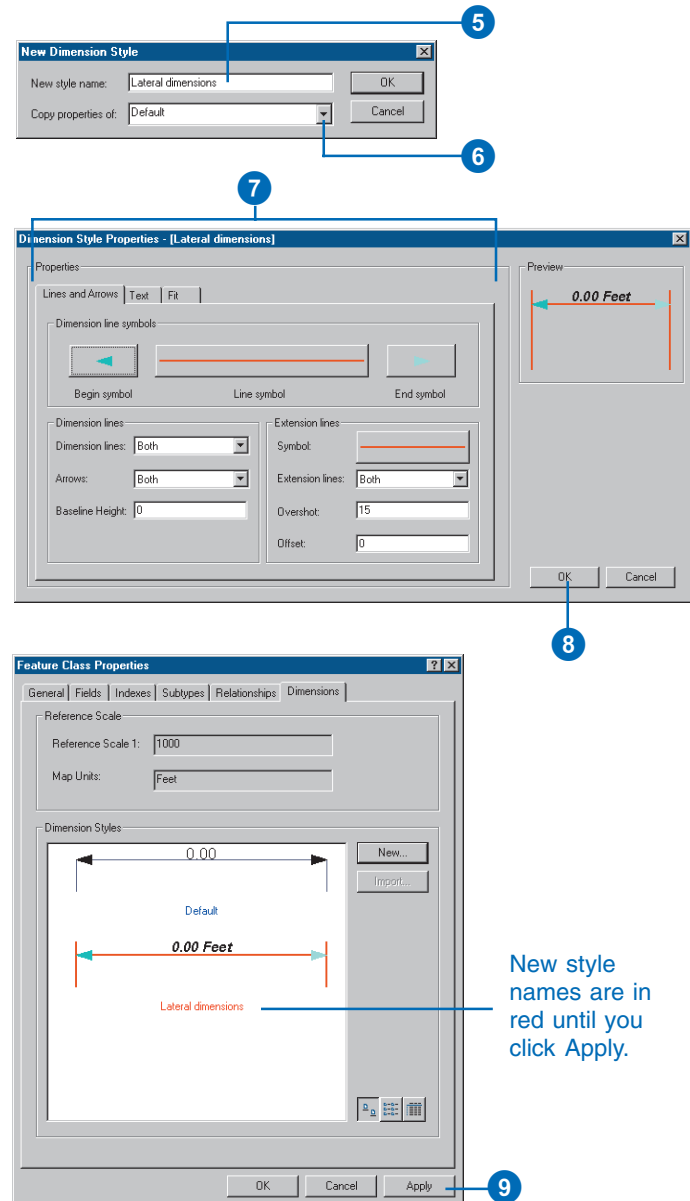
Tip

The template style

You can choose to copy the characteristics of an existing style to your new style, then use the Dimension Style Properties dialog box to modify the style elements.

You cannot use styles that have not yet been stored in the database as a template for your new style. Any styles whose names are in red cannot be used as templates.

5. Type a name for the new style.
6. Click the dropdown arrow and click the style in the dimension feature class whose properties you want to copy into the new style.
7. Modify those elements of the dimension style you wish to change in the Dimension Style Properties dialog box.
8. Click OK.
9. Click Apply.

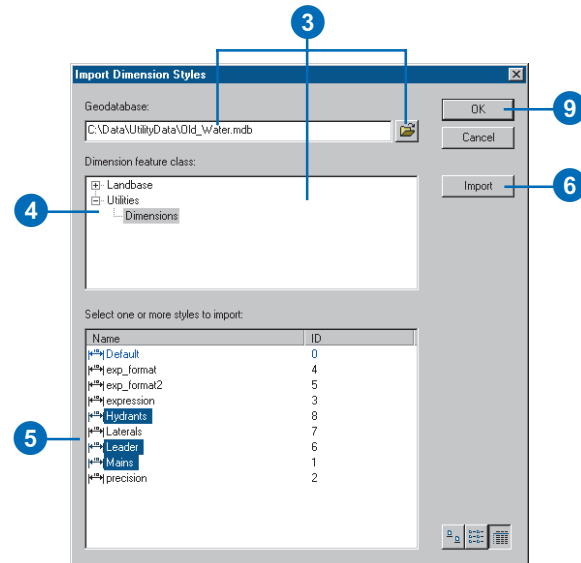
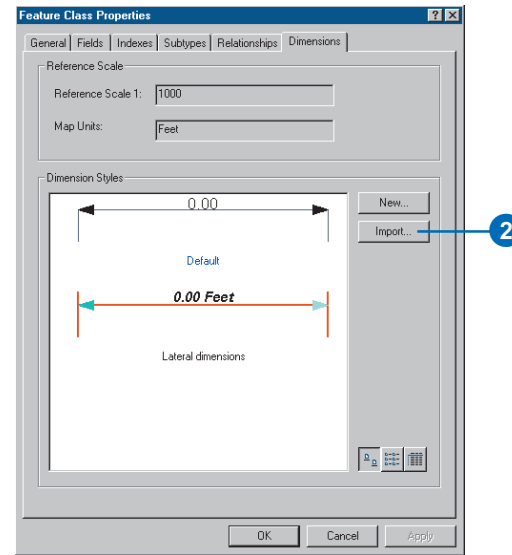


New style names are in red until you click Apply.

Importing dimension styles

1. Follow steps 1 through 3 for 'Creating a new dimension style'.
2. Click Import.
3. Click the Browse button to browse for a geodatabase. Once a geodatabase is selected, the dimension feature classes and feature datasets containing dimension feature classes are listed in the tree view.
4. Click the dimension feature class that contains the styles you want to import.
5. Hold down the Ctrl key and click the dimension styles that you want to import.
6. Click Import.
7. Click OK.
8. Repeat steps 4 through 8 to import more styles from the same geodatabase or steps 3 through 8 to import styles from a different geodatabase.
9. Click OK.

A dialog box displays whether the import was successful.



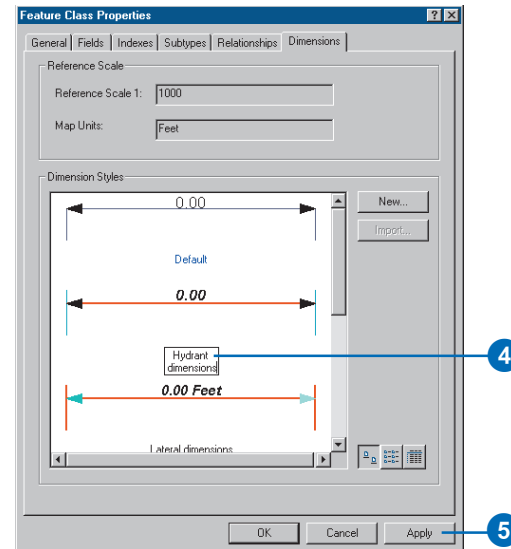
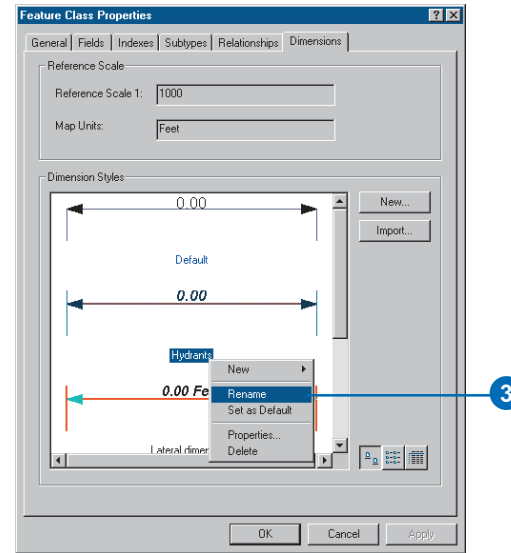
Tip

Dimension features

If dimension features that reference the style you want to rename already exist in your dimension feature class, they will not be affected. Those features will still reference the same style after it has been renamed.

Renaming a dimension style

1. Follow steps 1 through 3 for 'Creating a new dimension style'.
2. Right-click the dimension style you want to rename.
3. Click Rename.
4. Type the new name for the style and press Enter.
5. Click Apply.



Tip

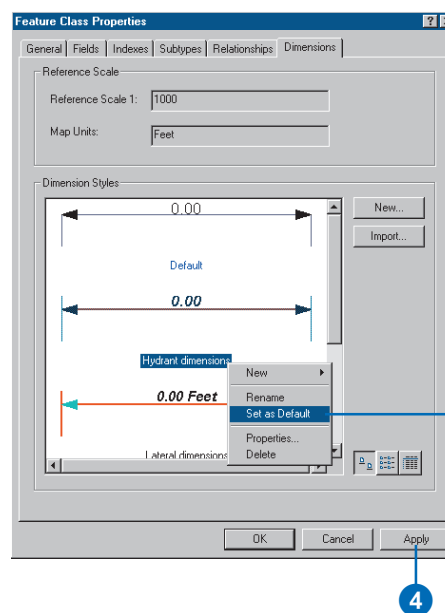
The default style

Any style can be the default style. The default style is the style that is to be assigned to all new dimensions when they are created in ArcMap, unless another style has been selected in the Dimensioning toolbar.

For more information on creating dimension features and the Dimensioning toolbar, see Editing in ArcMap.

Setting the default dimension style

1. Follow steps 1 through 3 for 'Creating a new dimension style'.
2. Right-click the dimension style you want to set as the default.
3. Click Set as Default.
4. Click Apply.



Tip

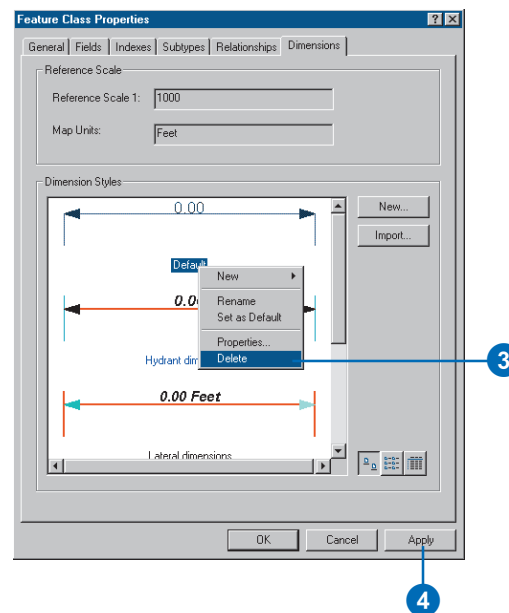
Deleting styles

All features that reference a style that has been deleted are symbolized as raw lines with a box for the text. You can use ArcMap to assign a different style to those dimension features.

For more information on editing dimension features, see Editing in ArcMap.

Deleting a dimension style

1. Follow steps 1 through 3 for 'Creating a new dimension style'.
2. Right-click the dimension style you want to delete.
3. Click Delete.
4. Click Apply.



Geocoding services

10

IN THIS CHAPTER

- **Geocoding services**
- **Geocoding services in ArcCatalog and ArcMap**
- **Preparing reference data for a geocoding service**
- **Creating a geocoding service**
- **Maintaining geocoding indexes**
- **Working with geocoding indexes**
- **Preparing address data for geocoding**

When you want to map the locations of addresses, you need to create spatial descriptions of these locations from the textual descriptions contained in the address. This process is known as *geocoding*. ArcGIS uses geocoding services to perform the task of creating geometry from textual descriptions of locations.

Geocoding services are created and maintained in ArcCatalog. You can use geocoding services to geocode addresses in both ArcCatalog and ArcMap. This chapter describes the key concepts of creating and maintaining geocoding services inside a geodatabase.

ArcView users can use preexisting ArcSDE and client-side geocoding services. Any other geocoding functionality requires an ArcEditor or ArcInfo license.

Geocoding services

What is a geocoding service?

A feature is an object that has geometry. In most cases, this geometry is captured by digitization or scanning of paper maps. In many cases, however, geographic data exists that indirectly captures geometry by describing locations such as street addresses, city names, or even telephone numbers. While humans understand what these descriptions mean and how they relate to locations on the earth's surface, computers do not. In order to display these locations on a map and perform analyses with them, a computer must be given geometric representations (such as point features) of these locations.

Geocoding (also commonly known as address matching) is the process of creating geometric representations for descriptions of locations. A geocoding service defines a process for converting alphanumeric descriptions of locations into geometric shapes.

ArcGIS 8 provides tools and a framework for creating, managing, and using geocoding services. In ArcGIS 8, a geocoding service defines paths to *reference data*; algorithms for standardizing alphanumeric descriptions of places and matching them to the reference data; and parameters for reading address data, matching the address data to the reference data, and creating output.

Client-side and server-side geocoding services

In ArcGIS 8, you can create and use both client-side and server-side geocoding services. Client-side geocoding services are stored on the same machine as the ArcGIS desktop installation that created them. Server-side geocoding services are stored in an ArcSDE geodatabase on a server.

Client-side geocoding services do not require an ArcSDE server. They are created and used through the ArcCatalog and ArcMap interfaces or the ArcObjects™ application programming interface

(API). By default, client-side geocoding services are not shared among users.

Server-side geocoding services do require an ArcSDE server and are easily shared among users in an organization. Server-side geocoding services can be created and used through the ArcCatalog and ArcMap desktop applications, the ArcObjects API, or the ArcSDE API.

Geocoding reference data

In order to find the geographic location of an address, a geocoding service must refer to at least one data source that has both address information (attributes) and spatial information (geometry). A feature class is an example of a data source that includes both types of information. When geocoding an address, a geocoding service searches through the features in the reference data feature class to find the feature with address attributes that most closely match the address. The geometry of the matching feature is then used to create geometry for the address.

Geocoding services can use other types of reference data such as alternate street name tables and place name alias tables. For more information about these types of reference data, see 'Preparing reference data for a geocoding service' in this chapter.

Input address tables

You can use a geocoding service to geocode an entire table of addresses. Address tables can be in any format supported by ArcGIS 8, such as INFO, dBASE, or geodatabase tables. Address tables store one address in each record and the components of each address are contained in several fields in each record.

Output feature classes

When you geocode a table of addresses, ArcGIS creates a feature class for the geocoded features. This feature class contains the geometry for the geocoded features, the status of the geocoded address (whether it was matched or not), the score with which the address was matched to a feature in the reference data and, for geocoding service styles that can match an address to a particular side of a street, the side of the street to which the address was matched. Optionally, the geocoding service can create attributes in the output feature class for the x,y coordinates of the geocoded feature; the standardized form of the input address; the feature ID from the *geocoding reference data* to which the address was matched; and, for geocoding reference data with line geometry, the percent along the reference feature at which the geocoded feature is located.

When you geocode a table of addresses and create your output feature class in the same geodatabase as the input address table, you can choose to create a relationship between the address table and the output feature class. When you choose to create this relationship, ArcGIS creates a relationship class in the geodatabase that relates features in the output feature class to records in the input address table. After geocoding the table of addresses, this relationship class maintains the geocoded feature class to reflect any edits you make to the input address table. If you add or delete rows in the address table, the corresponding features are added or deleted in the geocoded feature class. If you edit the address information in a row of the address table, the corresponding feature in the geocoded feature class is updated to reflect the new address.

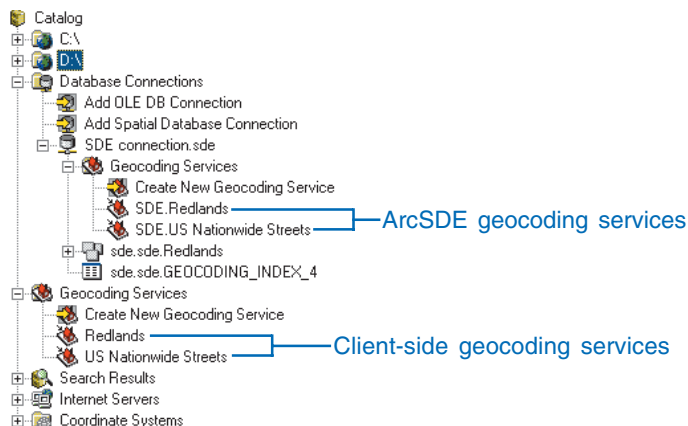
Alternatively, you can copy the attributes from the input address table to the output feature class. In this case, no relationship is created between the address table and the geocoded feature class, and edits made to the address table are not reflected in the geocoded feature class.

Geocoding services in ArcCatalog and ArcMap

Geocoding services in ArcCatalog

You can use ArcCatalog to create geocoding services, modify the properties of geocoding services, and geocode tables of addresses.

In the ArcCatalog tree, geocoding services are stored in Geocoding Services folders. Client-side geocoding services are stored in the top-level Geocoding Services folder. In addition, each ArcSDE database connection contains a Geocoding Services folder that contains the geocoding services that are stored on that ArcSDE server.

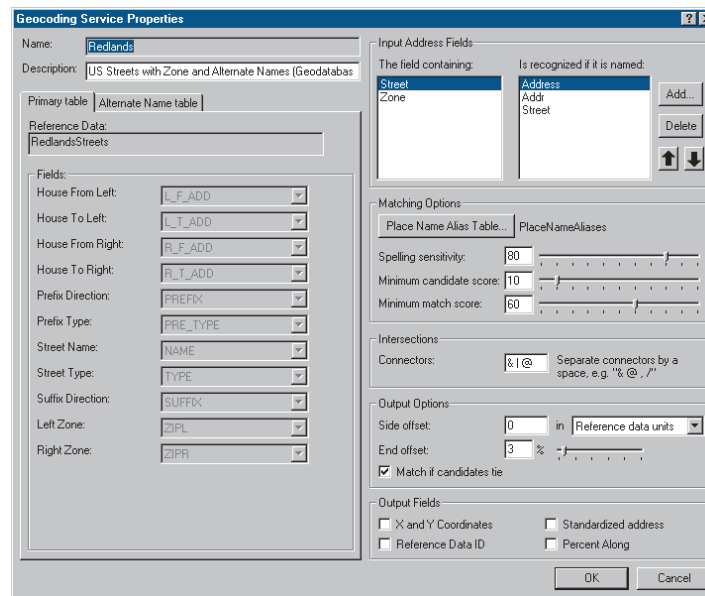


Geocoding services appear in the ArcCatalog tree.

The Geocoding Services folders also contain items called Create New Geocoding Services. You can click these items to create a new geocoding service in the Geocoding Services folder.

The Geocoding Service Properties dialog box contains information about the reference data used by the geocoding service, parameters for matching addresses, and options for

writing output to geocoded feature classes. The Geocoding Service Properties dialog box can be opened by right-clicking a particular geocoding service in ArcCatalog and clicking Properties.

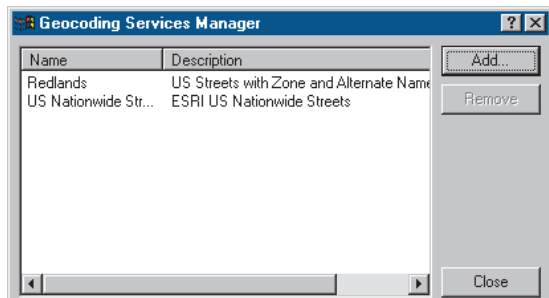


In ArcCatalog, you can also geocode tables of addresses using a geocoding service. For more information about geocoding tables of addresses, see *Using ArcCatalog*.

Geocoding services in ArcMap

Once you have created geocoding services in ArcCatalog, you can use them in ArcMap to find addresses and to geocode tables of addresses. In order to use a geocoding service in ArcMap, it must be added to the ArcMap document. The Geocoding Services Manager dialog box is used to manage the set of

geocoding services that are loaded in an ArcMap document. For more information on finding addresses in ArcMap, see *Using ArcMap*.



The Geocoding Services Manager dialog box lets you manage the set of geocoding services that are loaded in an ArcMap document. It is accessed by clicking the Tools menu in ArcMap, clicking Geocoding, then clicking Geocoding Services Manager.

Preparing reference data for a geocoding service

ArcGIS 8 comes with several predefined geocoding service styles that you can use immediately to create geocoding services. These geocoding service styles cover some of the most common styles of addresses that you might want to geocode. Each geocoding service style has specific requirements for the reference data that it can use to match addresses.

You can use ESRI StreetMap™ data, feature classes, and tables as reference data for geocoding services. When you use feature classes and tables as reference data, they may contain some common pieces of information that can be used for geocoding. This information includes:

- Prefix direction (a direction that precedes the street name), as in “W. Redlands Blvd.”
- Prefix type (a street type that precedes the street name), as in “Avenue B”
- Street name
- Street type (a street type that follows the street name), as in “New York St.”
- Suffix direction (a direction that follows the street name), as in “Bridge St. W.”
- Zone (additional information used to resolve ambiguity between addresses by identifying a region in which the address is located), as in a ZIP Code or city name

Each geocoding service style has its own requirements for reference data that it can use. Each geocoding service style that is provided with ArcGIS 8 is discussed in this section.

StreetMap

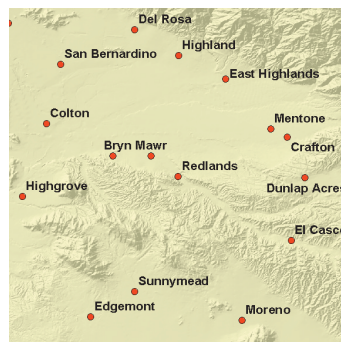
ArcGIS 8 can use StreetMap data as a reference data source for geocoding. ArcSDE includes a StreetMap license so that you can create server-side geocoding services that use StreetMap data as

reference data. However, if you want to create client-side geocoding services that use StreetMap data as reference data, you need to purchase a StreetMap license. Refer to the StreetMap installation guide for information on installing StreetMap for server-side and for client-side geocoding.

The Streets directory on the StreetMap CD contains one .edg file for each state. To create a StreetMap geocoding service for a single state, you can use the state's .edg file as reference data for your geocoding service. To create a geocoding service for the entire United States, use the usa.edg file as your reference datafile.

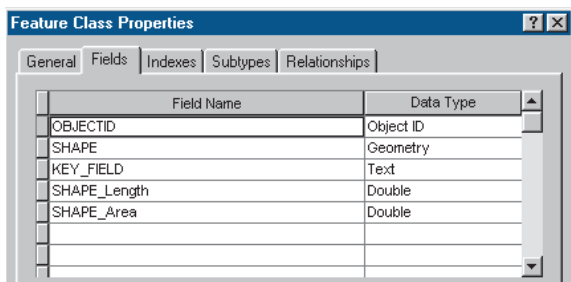
Single Field

The Single Field geocoding service style lets you create geocoding services for addresses that contain the address information in a single field. You could use a Single Field geocoding service style to geocode addresses such as place names, city names, and state names.



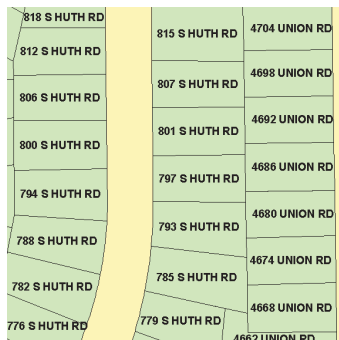
Although Single Field geocoding services can use feature classes with any type of geometry, they typically use feature classes with point or polygon geometry as reference data. In addition to an

ObjectID field and SHAPE field, feature classes that you can use as reference data for a Single Field geocoding service must have a key field that contains the unique “address” for that feature.



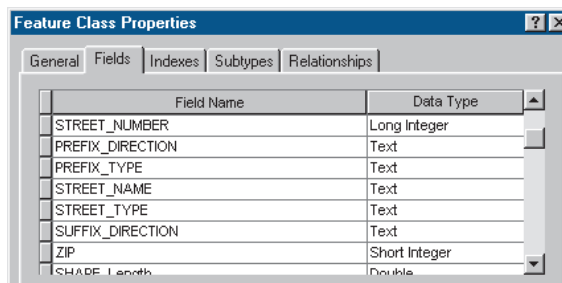
US One Address

The US One Address geocoding service style lets you create geocoding services for U.S. addresses. US One Address geocoding services can use feature classes with polygon or point geometry as reference data. Each feature in the reference data corresponds to a single address. For example, you could use a feature class containing parcel polygons or parcel *centroids* (the



center points of parcel polygons) as reference data for a US One Address geocoding service.

To use a feature class as reference data for a US One Address geocoding service, it must have fields that contain street number and street name information, in addition to an ObjectID field and a SHAPE field. Optionally, you can use fields that contain the street's prefix direction, prefix type, street type, suffix direction, or zone.



US One Range

The US One Range geocoding service style lets you create geocoding services for U.S. addresses. This geocoding service style can use feature classes with any type of geometry but



typically uses feature classes with line or polyline geometry. Each feature in the reference data represents a street segment with a range of addresses that fall along that street segment.

To use a feature class as reference data for a US One Range geocoding service, it must have fields that contain from address, to address, and street name information in addition to an ObjectID field and a SHAPE field. In addition, you can optionally specify fields that contain the street’s prefix direction, prefix type, street type, suffix direction, or zone.

Feature Class Properties

Field Name	Data Type
FROM_ADDRESS	Long Integer
TO_ADDRESS	Long Integer
PREFIX_DIRECTION	Text
PREFIX_TYPE	Text
STREET_NAME	Text
STREET_TYPE	Text
SUFFIX_DIRECTION	Text
ADDRESS_ZONE	Text

US Streets

The US Streets geocoding service style lets you create geocoding services for U.S. addresses. This geocoding service



style can use feature classes with any type of geometry but typically uses feature classes with line or polyline geometry. Each feature in the reference data represents a street segment with two ranges of addresses that fall along that street segment, one for each side of the street.

To use a feature class as reference data for a US Streets style of geocoding service, it must have fields that contain from address and to address information for each side of the street, street name information, and an ObjectID field and SHAPE field. Optionally, you can specify fields that contain the street’s prefix direction, prefix type, street type, suffix direction, or zone.

Feature Class Properties

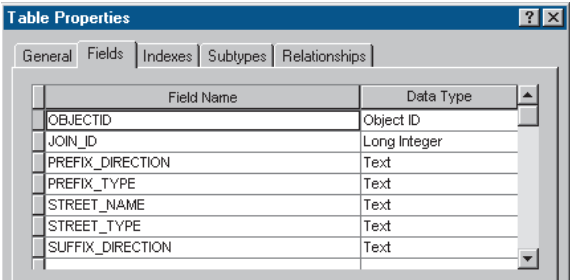
Field Name	Data Type
LEFT_FROM_ADDRESS	Long Integer
LEFT_TO_ADDRESS	Long Integer
RIGHT_FROM_ADDRESS	Long Integer
RIGHT_TO_ADDRESS	Long Integer
PREFIX_DIRECTION	Text
PREFIX_TYPE	Text
STREET_NAME	Text
STREET_TYPE	Text

Alternate street names

For the US One Address, US One Range, and US Streets geocoding service styles, you can use a table to define alternate street names for the features in your reference data feature class. Using alternate street names allows you to match an address to a feature using one of many names for the feature. For example, if “Bridge Street” is also known as “Slash Road”, then you can also find the address “266 Bridge Street” using “266 Slash Road”.

Tables that you use to specify alternate street names must have an ID field, a JOIN_ID that specifies the feature in the reference

data to which the alternate name applies, and an alternate street name field. Optionally, the table can contain fields that contain prefix direction, prefix type, street type, or suffix direction information. You can specify multiple alternate names for the same feature in your reference data by creating records in the alternate street name table with the same JOIN_ID, referencing the same feature in the reference data feature class.



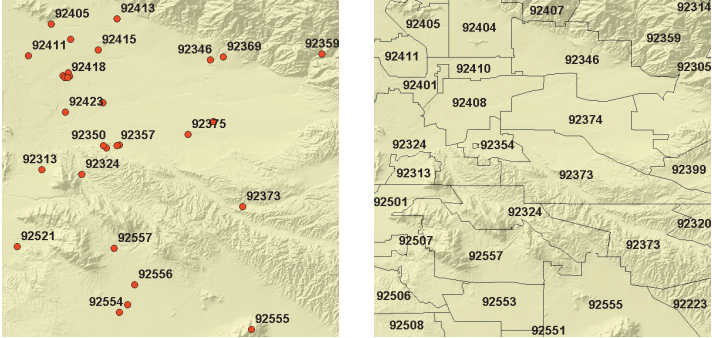
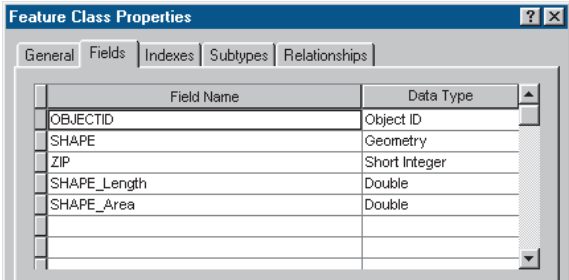
Field Name	Data Type
OBJECTID	Object ID
JOIN_ID	Long Integer
PREFIX_DIRECTION	Text
PREFIX_TYPE	Text
STREET_NAME	Text
STREET_TYPE	Text
SUFFIX_DIRECTION	Text

Each record in an alternate street name table applies to only one feature in your reference data feature class. In order to specify an alternate street name for all features that make up a particular street in your reference data feature class, you must create a record in the alternate street name table for each feature in your reference data feature class.

ZIP

The ZIP geocoding service style lets you create geocoding services for U.S. ZIP Codes. This geocoding service style can use feature classes with point or polygon geometry. Each feature in the reference data represents a ZIP polygon or its centroid.

To use a feature class as reference data for a ZIP style geocoding service, it must have a field that specifies the five-digit ZIP Code for the feature, in addition to an ObjectID field and a SHAPE field.

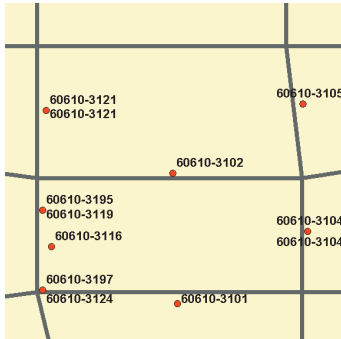



Field Name	Data Type
OBJECTID	Object ID
SHAPE	Geometry
ZIP	Short Integer
SHAPE_Length	Double
SHAPE_Area	Double

ZIP+4

The ZIP+4 geocoding service style lets you create geocoding services for U.S. ZIP+4 Codes. This geocoding service style can use feature classes with point or polygon geometry. Each feature in the reference data represents a ZIP+4 polygon or its centroid.

To use a feature class as reference data for a ZIP+4 style of geocoding service, it must have fields that specify the five-digit ZIP Code for the feature and the four-digit add-on code for the feature in addition to an ObjectID field and a SHAPE field.



General Fields Indexes Subtypes Relationships	
Field Name	Data Type
OBJECTID	Object ID
SHAPE	Geometry
ZIP	Short Integer
FROM_ZIP4	Short Integer
TO_ZIP4	Short Integer
SHAPE_Length	Double
SHAPE_Area	Double

Place name aliases

For all geocoding services, you can specify a place name alias table. Using a place name alias table, you can geocode addresses by their common names (for example, “Town Hall”) instead of by their street addresses.

In order to use a place name alias table for a geocoding service, it must contain an alias field that contains the common name by which an address is referred, in addition to all of the required fields for the particular geocoding service style. Optionally, the place name alias table can use any or all of the optional address information fields for the particular geocoding service style.

General Fields Indexes Subtypes Relationships	
Field Name	Data Type
OBJECTID	Object ID
SHAPE	Geometry
ZIP	Short Integer
ZIP4	Short Integer
SHAPE_Length	Double
SHAPE_Area	Double

ZIP+4 Range

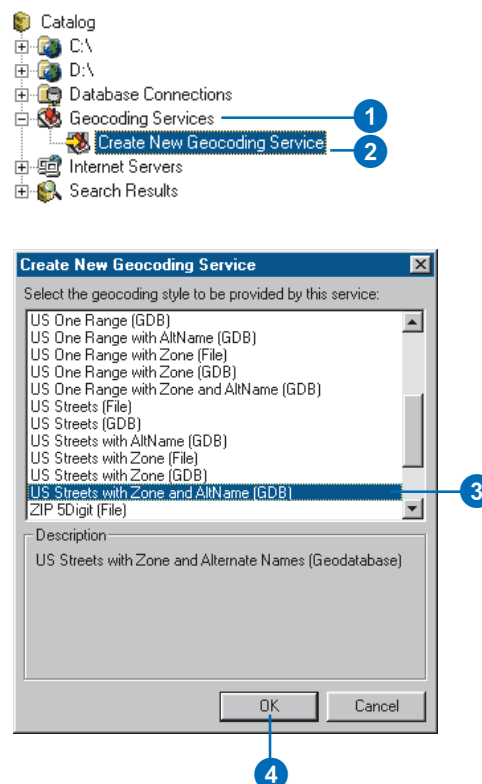
The ZIP+4 Range geocoding service style lets you create geocoding services for U.S. ZIP+4 Codes. This geocoding service style can use feature classes with point or polygon geometry. Each feature in the reference data represents a contiguous block of ZIP+4 Codes.

To use a feature class as reference data for a ZIP+4 Range style of geocoding service, it must have fields that specify the five-digit ZIP Code for the feature and lower and upper bounds for the four-digit add-on code in addition to an ObjectID and a SHAPE field.

Creating a geocoding service

You can create new geocoding services using ArcCatalog. Geocoding services appear in the ArcCatalog tree either in the top-level Geocoding Services folder (client-side geocoding services) or in the Geocoding Services folder in an ArcSDE database connection (server-side geocoding services). You can also view and modify the settings for a geocoding service in ArcCatalog.

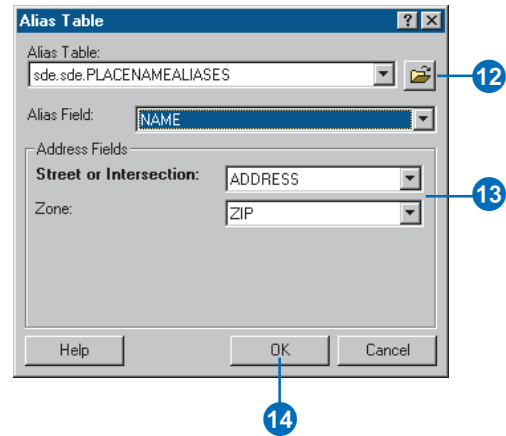
1. Click on a Geocoding Services folder in the ArcCatalog tree.
2. Double-click the Create New Geocoding Service item.
3. Click the geocoding service style that you want to use to create the new geocoding service.
4. Click OK. ►



5. Type a name for the new geocoding service in the Name text box.
6. Click the Primary table tab and click the Browse button to navigate to the feature class that the geocoding service will use as reference data. Click Add under the Input Address Fields.
7. Choose the column name from each dropdown list that contains the specified address information. The names of the required address attributes are shown in bold.
8. Click the Alternate Name table tab if your geocoding service will use an alternate street name table.
9. Click the Browse button to navigate to the table that the geocoding service will use as an alternate street name table, then click Add under the Input Address Fields.
10. Choose the column name from each dropdown list that contains the specified alternate street name information. The names of the required address attributes are shown in bold.
11. Click Place Name Alias Table if your geocoding service will use a place name alias table. ►

12. Click the Browse button to navigate to the table that the geocoding service will use as a place name alias table, then click Add.
13. Choose the column name from each dropdown list that contains the specified place name alias information.

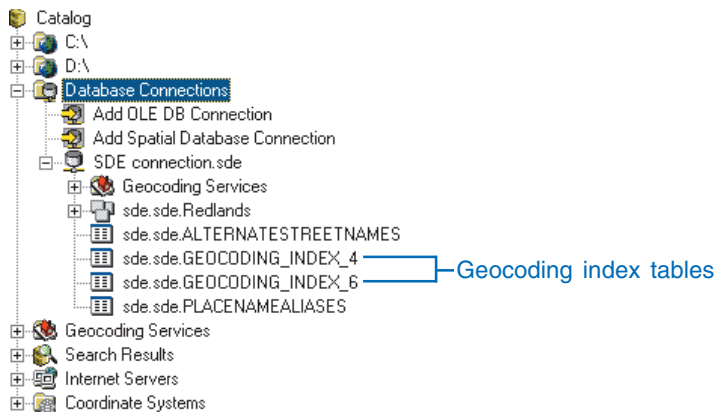
The names of the required address attributes are shown in bold.
14. Click OK.
15. Review the other settings for the new geocoding service, then click OK to create the new geocoding service.



Maintaining geocoding indexes

Geocoding indexes

When you create a new geocoding service, ArcGIS creates *geocoding indexes* for the reference data that the new geocoding service uses. These geocoding indexes allow ArcGIS to quickly find features that may match the addresses that you geocode using the geocoding service. Which information is contained in the geocoding indexes is determined by the style on which the geocoding service is based. By default, the geocoding service styles that are provided with ArcGIS build geocoding indexes on fields containing street name information, both for the primary reference data feature class and the alternate street name table, if any.



If your geocoding service uses feature classes and tables in a geodatabase as reference data, then the geocoding indexes are implemented as tables in the same geodatabase.

Maintaining geocoding indexes manually

In order to be able to geocode addresses, there must be a one-to-one correspondence between rows in an index table and features in the primary reference data feature class, or between rows in the index table and rows in the alternate street name table. If you add or delete features or rows from the primary reference data feature class or alternate street name table, or if you edit the street names of the features or rows, then the geocoding index needs to be updated.

By default, ArcGIS does not update your geocoding index tables for you when you edit your geocoding reference data. In order to keep geocoding indexes current with your geocoding reference data, you can delete the geocoding index tables from the geodatabase and re-create the geocoding service. Alternatively, you can use ArcObjects to rebuild the geocoding indexes without re-creating the geocoding service. A developer sample is provided with ArcGIS that demonstrates how to do this.

Maintaining geocoding indexes automatically

If you want ArcGIS to maintain your geocoding indexes when you edit your geocoding reference data, you can take advantage of relationship classes within the geodatabase. A relationship class between your geocoding reference data and your geocoding index table can be used to add or update rows in your geocoding index table when you add or update features or rows in your geocoding reference data. A composite relationship class will delete rows in the geocoding index table when features or rows are deleted from the geocoding reference data.

ArcGIS provides two objects that you can use to automatically maintain your geocoding indexes. One is the geocoding index object, which responds to edits to the geocoding reference data through the relationship class. When you register your geocoding index table as an object class containing geocoding

index objects, the geocoding index objects respond to edits of features or rows in the geocoding reference data by recalculating their index values based on the values of attributes in the geocoding reference data.

The other object that is provided with ArcGIS for helping you maintain your geocoding indexes is the geocoding index class extension object. When you register this custom object class extension with your geocoding index object class, this object adds new geocoding index objects to your geocoding index when you add features or rows to the geocoding reference data.

The process for setting up your geocoding indexes so that they are maintained automatically is as follows:

- Register the geocoding index table as an object class.
- Register the geocoding index object class as containing custom geocoding index objects.
- Set the geocoding index object class to use a custom geocoding index object class extension.
- Create a relationship class between the geocoding reference data and the geocoding index object class.

A developer sample is provided with ArcGIS that demonstrates how to set up your geocoding indexes to be maintained automatically.

Working with geocoding indexes

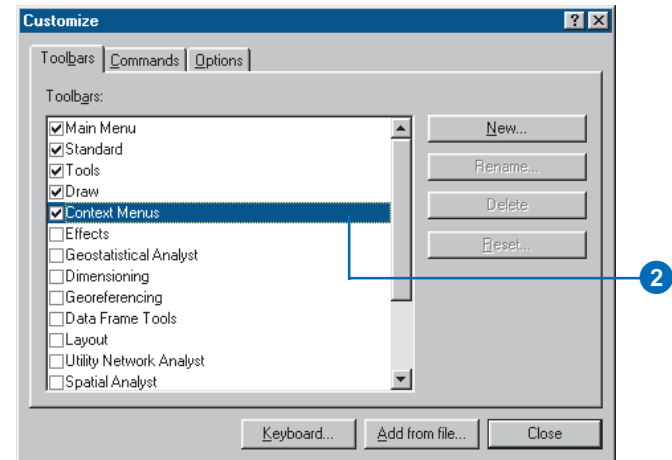
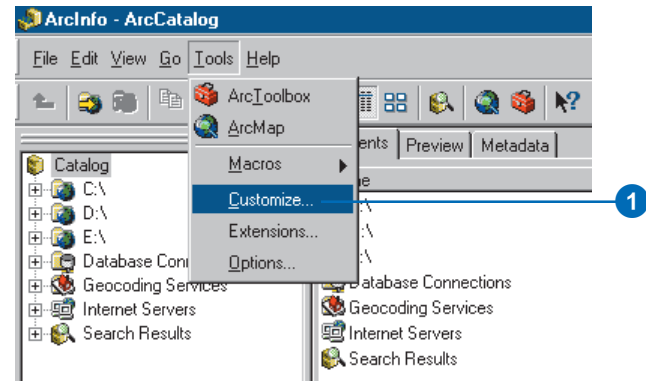
By default, geocoding indexes are not maintained if you edit your geocoding reference data. When you make edits to your geocoding reference data (either by adding or deleting features or rows, or by editing the street names that they contain), you need to update your geocoding indexes.

ArcGIS provides two developer samples that demonstrate how to maintain geocoding indexes. The first sample is a command that rebuilds the geocoding indexes for a geocoding service. The second sample is a command that sets up geocoding indexes so that they are maintained automatically by the geodatabase.

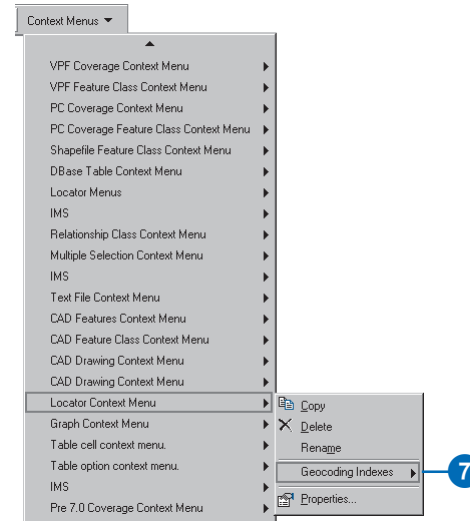
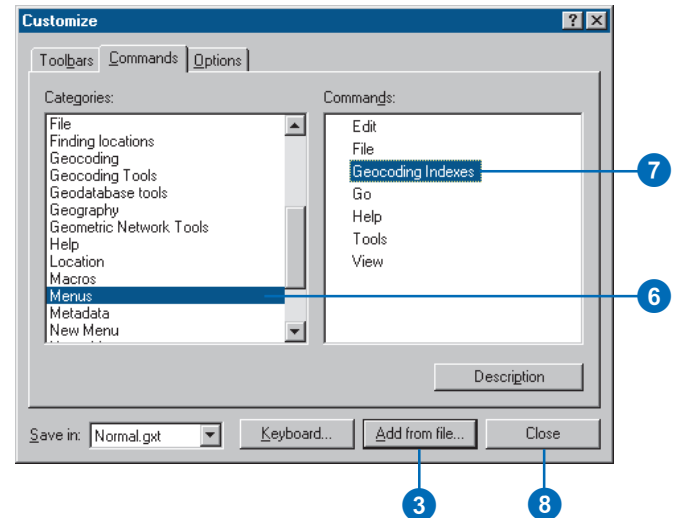
Manually rebuilding geocoding indexes

1. Click Tools and click Customize in ArcCatalog.
2. Click the Toolbars tab and check Context Menus.

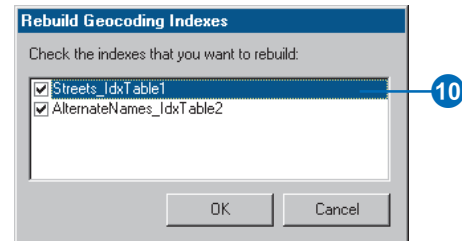
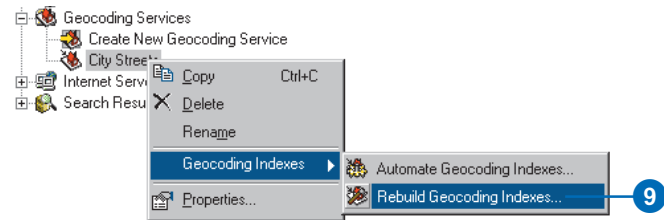
The Context Menus toolbar displays. ►



3. Click the Commands tab and click Add from file.
4. Navigate to the GCIIndexManagement.dll library in the Developer Samples folder and click Open.
5. Click OK.
6. Scroll through the Categories list and click Menus.
7. Click the Context Menus toolbar and locate the Locator Context Menu. Click and drag Geocoding Indexes onto the Locator Context Menu.
8. Click Close. ►



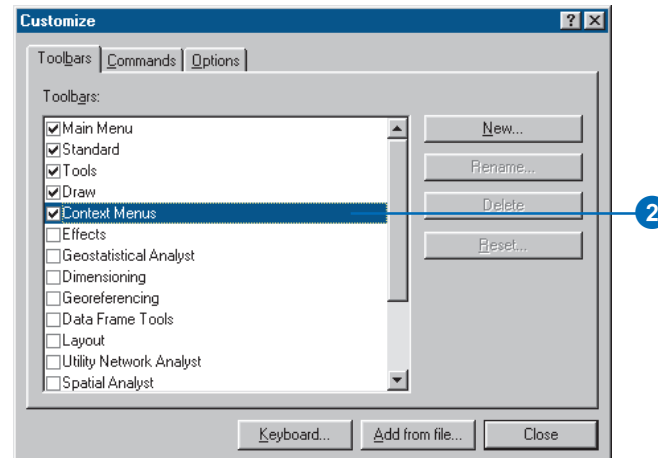
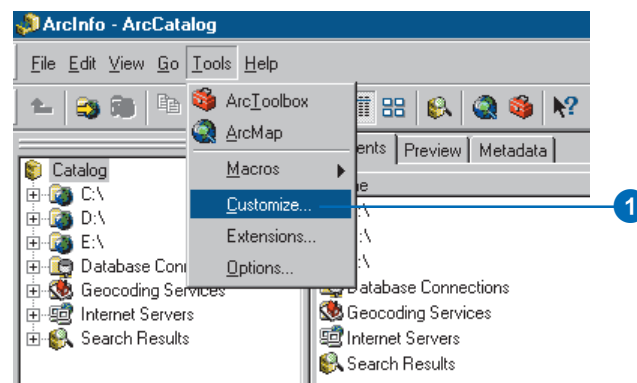
9. Right-click a geocoding service for which you want to rebuild the geocoding indexes, click Geocoding Indexes, and click Rebuild Geocoding Indexes.
10. Check the names of the indexes that you want to rebuild and click OK.



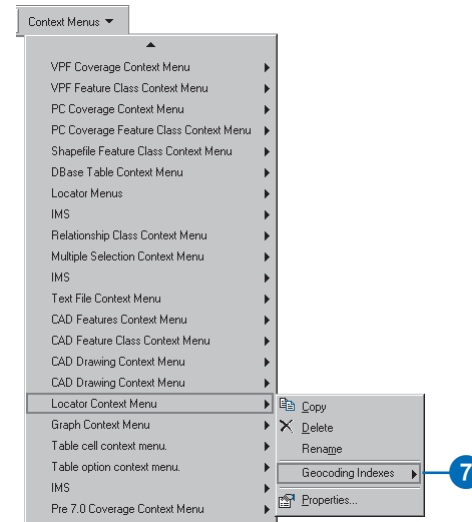
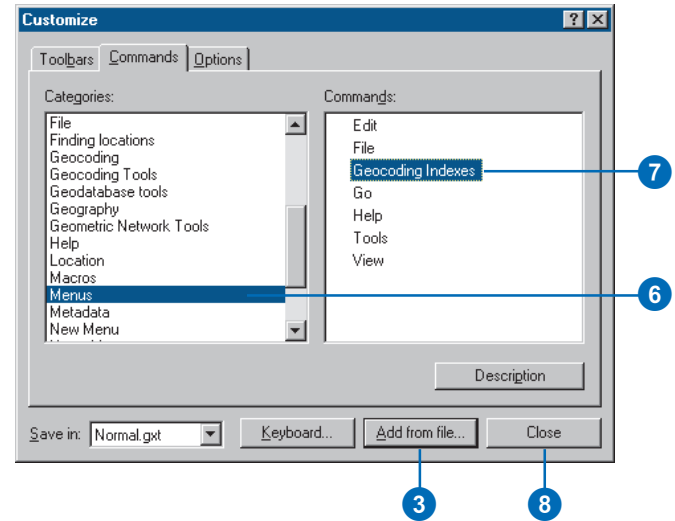
Automatically maintaining geocoding indexes

1. Click Tools and click Customize in ArcCatalog.
2. Click the Toolbars tab and check Context Menus.

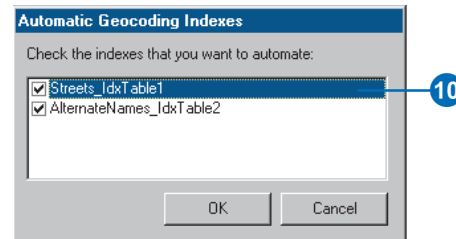
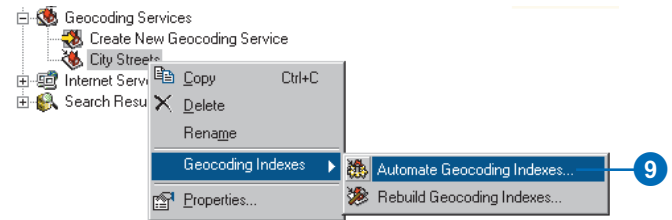
The Context Menus toolbar displays. ►



3. Click the Commands tab and click Add from file.
4. Navigate to the GCIndexManagement.dll library in the Developer Samples folder and click Open.
5. Click OK.
6. Scroll through the Categories list and click Menus.
7. Click the Context Menus toolbar and locate the Locator Context Menu. Click and drag Geocoding Indexes onto the Locator Context Menu.
8. Click Close. ►



9. Right-click a geocoding service for which you want to automatically maintain the geocoding indexes, click Geocoding Indexes, and click Automate Geocoding Indexes.
10. Check the names of the indexes that you want automatically maintained and click OK.



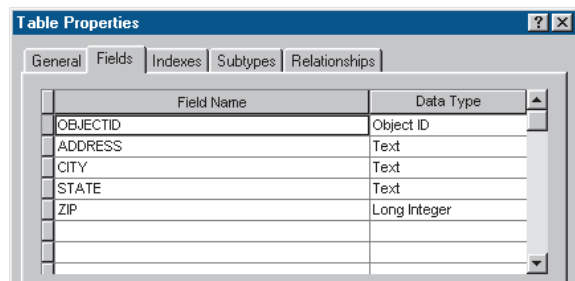
Preparing address data for geocoding

Just as each style of geocoding service has different requirements for the attributes of the reference data that it uses, each style also has different requirements for the information that tables of addresses must contain in order to be geocoded. For each style of geocoding service provided with ArcGIS 8, the requirements for address data are described below.

StreetMap

A StreetMap style of geocoding service allows you to geocode addresses anywhere in the United States. Tables of addresses that these geocoding services can match must include the following attributes:

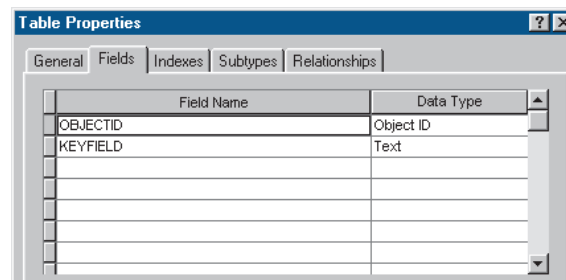
- Address—including the street number; street name; and the street's prefix direction, prefix type, street type, or suffix direction, if any; intersection descriptions (for example, "Hollywood Blvd. & Vine St.") can also be included in this field.
- City—the city in which the address is located.
- State—the state in which the address is located.
- ZIP—the address's five-digit ZIP Code.



Field Name	Data Type
OBJECTID	Object ID
ADDRESS	Text
CITY	Text
STATE	Text
ZIP	Long Integer

Single Field

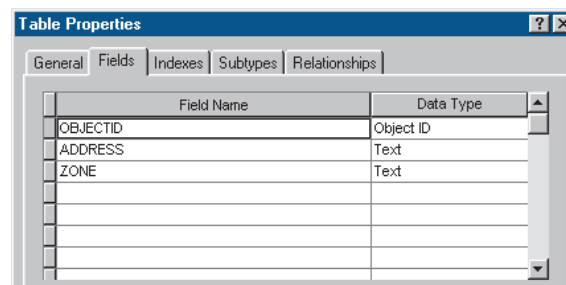
A Single Field style of geocoding service allows you to geocode addresses using only a single item of information such as a state name or place name. Tables of addresses that can be geocoded using these geocoding services must contain a field with this information that can be used to find the locations.



Field Name	Data Type
OBJECTID	Object ID
KEYFIELD	Text

US One Address, US One Range, and US Streets

Each of the U.S. address geocoding service styles, while having different requirements for reference data, has the same requirements for input address data. Tables of addresses that can be geocoded using these geocoding services must contain a field that has an address (as described for the StreetMap geocoding service style). Optionally, if the geocoding service reference data



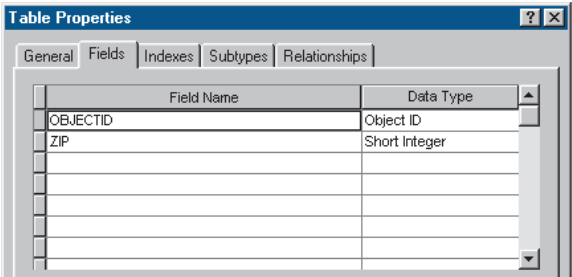
Field Name	Data Type
OBJECTID	Object ID
ADDRESS	Text
ZONE	Text

contains zone information, the table of addresses may contain a field having zone information (for example, city name or ZIP Code) that corresponds to the type of zone information in the reference data feature class.

ZIP

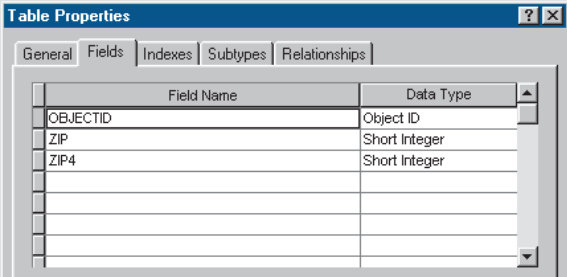
A ZIP style of geocoding service allows you to geocode addresses that contain five-digit ZIP Code information. Tables of addresses that can be geocoded using these geocoding services must contain a field that has five-digit ZIP Code information.

ZIP+4 and ZIP+4 Range



Field Name	Data Type
OBJECTID	Object ID
ZIP	Short Integer

The ZIP+4 geocoding service styles allow you to geocode addresses that contain ZIP+4 Code information. Tables of addresses that can be geocoded using these geocoding services must contain fields that have five-digit ZIP Code and four-digit add-on code information.



Field Name	Data Type
OBJECTID	Object ID
ZIP	Short Integer
ZIP4	Short Integer

Working with a versioned geodatabase

11

IN THIS CHAPTER

- Integrating versioning with your organization's work flow
- Registering data as versioned
- Creating and administering versions in ArcCatalog
- Working with versions in ArcMap
- Editing and conflict resolution
- Editing a version
- Versioning scenarios

With ArcGIS 8, multiple users can access geographic data in a geodatabase through *versioning*. Versioning lets users simultaneously create multiple, persistent representations of the database without data replication. Users can edit the same features or rows without explicitly applying locks to prohibit other users from modifying the same data.

An organization can use versioning to manage alternative engineering designs, solve complex “what if” scenarios without impacting the corporate database, and create point-in-time representations of the database.

Primarily, versioning simplifies the editing experience. Multiple users can directly modify the database without having to extract data or lock features and rows before editing. If, by chance, the same features are modified, a *conflict* resolution dialog box guides the user through the process of determining the feature's correct representation and attributes.

Versioned databases may contain topologies. For more information on how versioning affects topologies, see the chapter ‘Topology’ in this book.

Versioned databases may also be the checkout databases for disconnected editors. See the chapter ‘Disconnected editing’ in this book for more information on using versioned databases for disconnected editing.

You can view versioned geodatabases in ArcView. To take advantage of other versioning functionality, you need an ArcEditor or ArcInfo license.

Integrating versioning with your organization's work flow

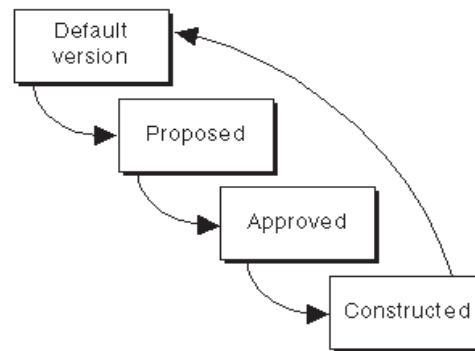
The geodatabase and versioning provide organizations with advanced data storage techniques that revolutionize the *work flow* process in many applications where spatial information is used. Engineers can generate design alternatives using the entire database. Spatial analysts can perform complex “what if” scenarios without affecting the current representation of the database. Database administrators can create “historical” snapshots of the database for archiving or database recovery.

In the long run, an organization benefits from implementing a versioned database. The data is centrally located in one corporate database. There is never a need to extract units of the database to update or lock map sheets or individual features. These factors simplify the administration process.

The work flow process

The evolution of the work flow process—how projects or *work orders* transpire over time—varies greatly from organization to organization and throughout each sector of the business community. Therefore, the geodatabase's versioning process has been designed to be flexible enough to accommodate the most basic of work flow processes as well as the most complex and to be sufficiently restrictive with or without additional application customization.

Common work flow processes usually progress in discrete stages. At each stage, different requirements or business rules may be enforced. Typically, during each stage of the process, the project or work order is associated with a named stage. For example, within the utility domain, common stages include “working”, “proposed”, “accepted”, “under construction”, and “as built”. The process is essentially cyclical. The work order is initially generated and assigned to an engineer, then modified over time as it progresses from stage to stage, and finally the changes are “posted” or applied back to the corporate database.



A common work flow process evolving through each stage of a project

This is one example of how versioning can help simplify the work flow process. Because the work flow process may span days, months, and even years, the corporate database requires continuous availability for daily operations. If a work order applied restrictive locks to the data involved in the process, other database users might not be able to perform their daily work assignments.

To implement your work flow in the geodatabase, versions can be created to correspond with each stage of the work flow process. Alternatively, you may want to create one version for each work order and modify the version's name to represent the current stage as the process proceeds through each step.

The current structure of your organization's work flow will significantly influence how you implement the geodatabase versioning process to manage your spatial transactions. The flexibility and openness of the system will allow you to determine the best solution to meet the requirements of your business processes.

The remaining sections of this chapter will help illustrate how to use ArcCatalog and ArcMap to perform various versioning tasks. In particular, the last section provides examples of how an organization can implement work flow processes using the geodatabase's versioning capabilities. For additional details on managing your organization's work flow with versions, read *Modeling Our World*.

Registering data as versioned

Before editing feature datasets, feature classes, and tables in a multiuser geodatabase, you must first register the data as versioned in ArcCatalog.

Making a feature class or table multiversioned requires a unique integer field. This is typically the OBJECTID field. Only the owner of the data may register or unregister the object as versioned.

When unregistering a dataset or feature class as versioned in ArcCatalog, a warning dialog box may appear informing you that outstanding edits still remain in existing versions. Unregistering the class as versioned will remove all the edits. To preserve the edits, you must compress the database before unregistering the versioned data.

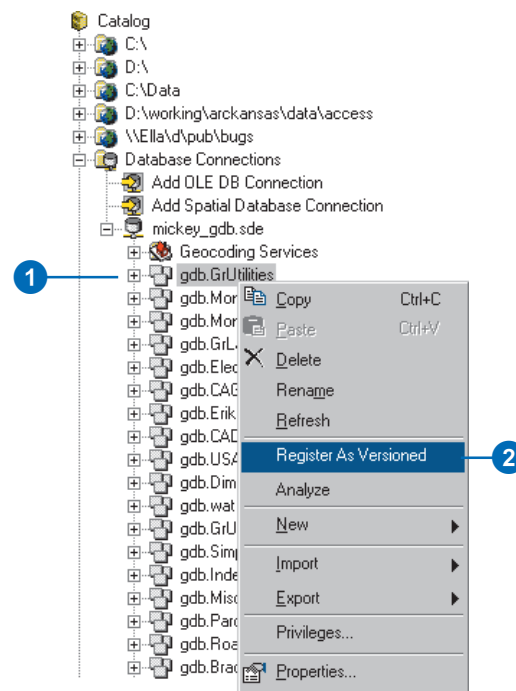
Personal geodatabases do not support versioning.

Tip

Registering data as versioned

Registering a feature dataset as versioned registers all feature classes within the feature dataset as versioned.

1. Right-click the feature dataset, feature class, or table you want to register as versioned in the ArcCatalog tree.
2. Click Register As Versioned.



Creating and administering versions in ArcCatalog

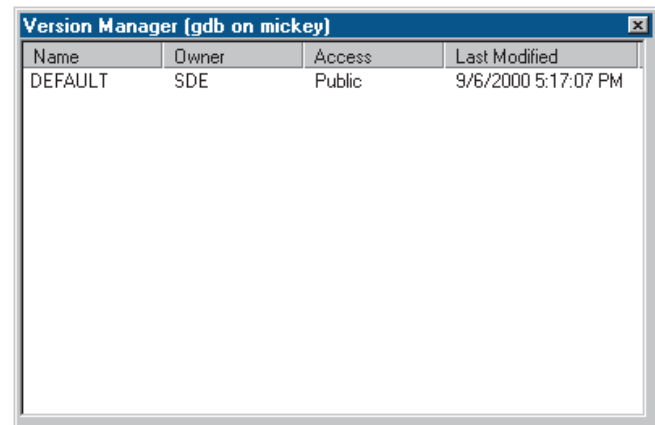
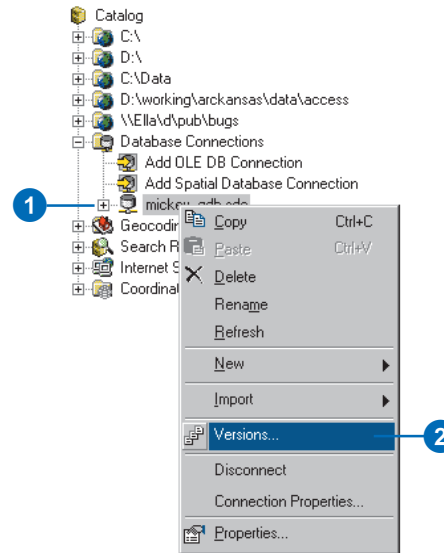
ArcCatalog lets you create new versions, rename existing versions, delete versions, and modify version properties. These administrative tasks are accomplished using the Version Manager dialog box.

Initially, the database consists of one version named “DEFAULT”, owned by the ArcSDE administrative user. The new versions that are created are always based on an existing version. When the new version is created, it is identical to the version from which it was derived. Over time, the versions will diverge as changes are made to the parent version and to the new version.

Each version has several properties: an alphanumeric name, an owner, an optional description, the creation date, the last modified date, the parent version, and the version’s permission. ►

Creating a new version

1. Create a new connection to the database in ArcCatalog with the Add SDE Connection dialog boxes described in the ‘Introduction’ chapter of this book.
2. Right-click your database connection in the Catalog tree and click Versions. ►



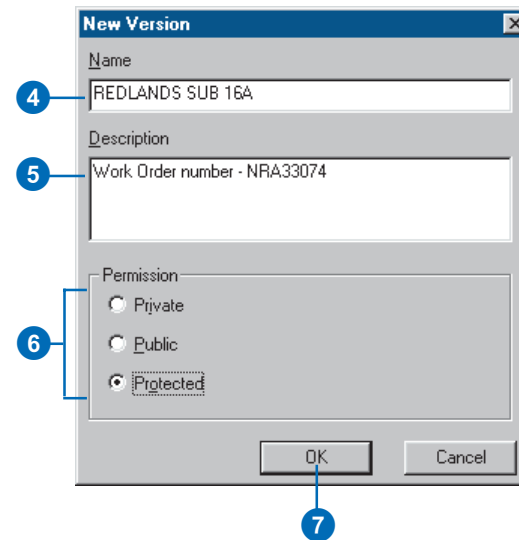
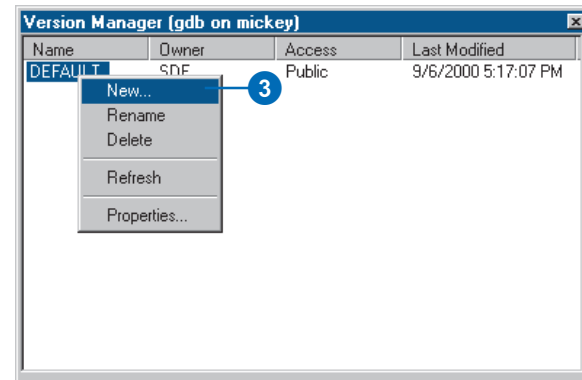
A version's permission can only be changed by its owner. The available permission settings are:

- Private—only the owner may view the version and modify available feature classes.
- Protected—any user may view the version, but only the owner may modify available feature classes.
- Public—any user may view the version and modify available feature classes.

Only the version's owner can rename, delete, or alter the version. A parent version cannot be deleted until all dependent child versions are first deleted.

To improve database performance, the database should be compressed periodically. Compressing the database removes all unreferenced database states and redundant rows. Only the ArcSDE administrator can perform this task. When the *Compress* command is executed, the database is unavailable until compression is completed. For additional details, see the versioning scenarios section at the end of the chapter. ►

3. Right-click a version and click New.
4. Type the new version's name.
5. Type a description.
6. Click the appropriate permission type; the default is Private.
7. Click OK.



Finally, after compressing the database or editing the data, the Analyze command should be executed to update the database statistics for each dataset or feature class. This will help improve display and query performance.

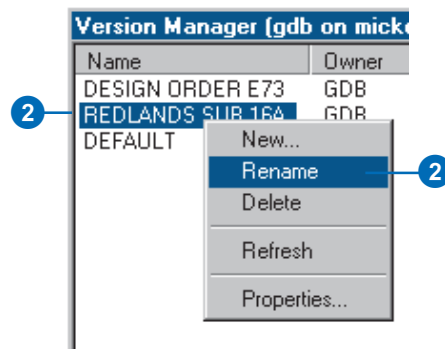
Tip

Descriptions

You can use the version description to provide additional information regarding the version's purpose.

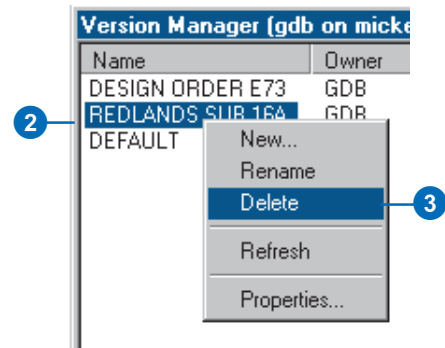
Renaming a version

1. Right-click your database connection and click Versions.
2. Right-click the version you want to rename and click Rename.
3. Type a new name and press Enter.



Deleting a version

1. Right-click your database connection and click Versions.
2. Right-click the version you want to delete.
3. Click Delete or press Delete on your keyboard.



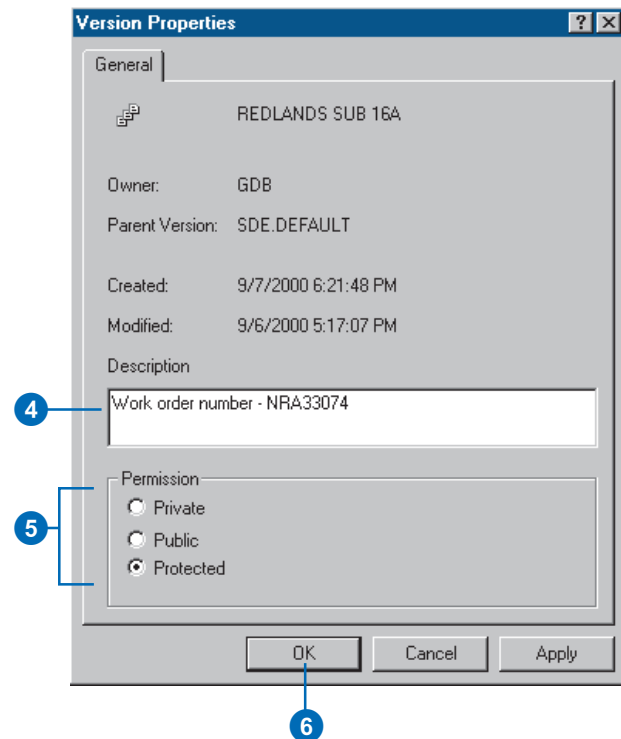
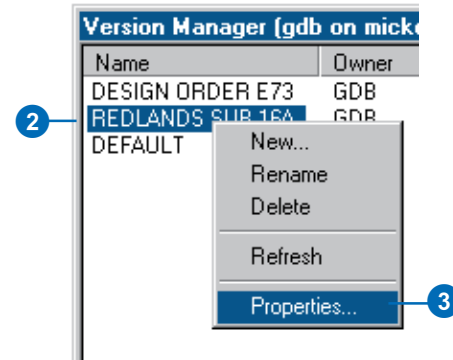
Tip

Refresh

Use the Refresh command to update the properties of each version with their current values.

Changing a version's properties

1. Right-click your database connection and click Versions.
2. Right-click a version.
3. Click Properties.
4. Type the new description.
5. Click the new permission type.
6. Click OK.

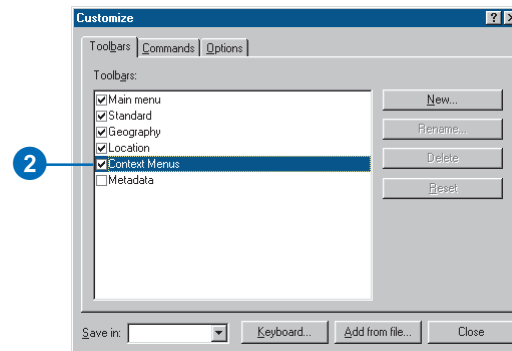
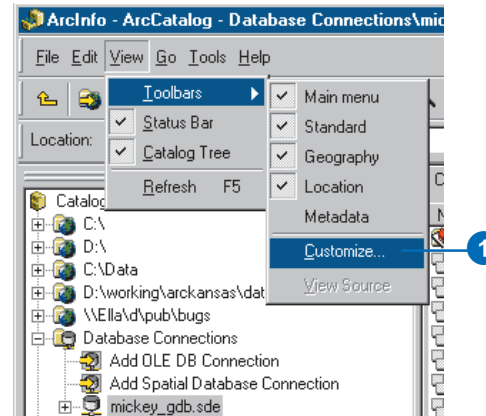


See Also

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

Adding the Compress command to ArcCatalog

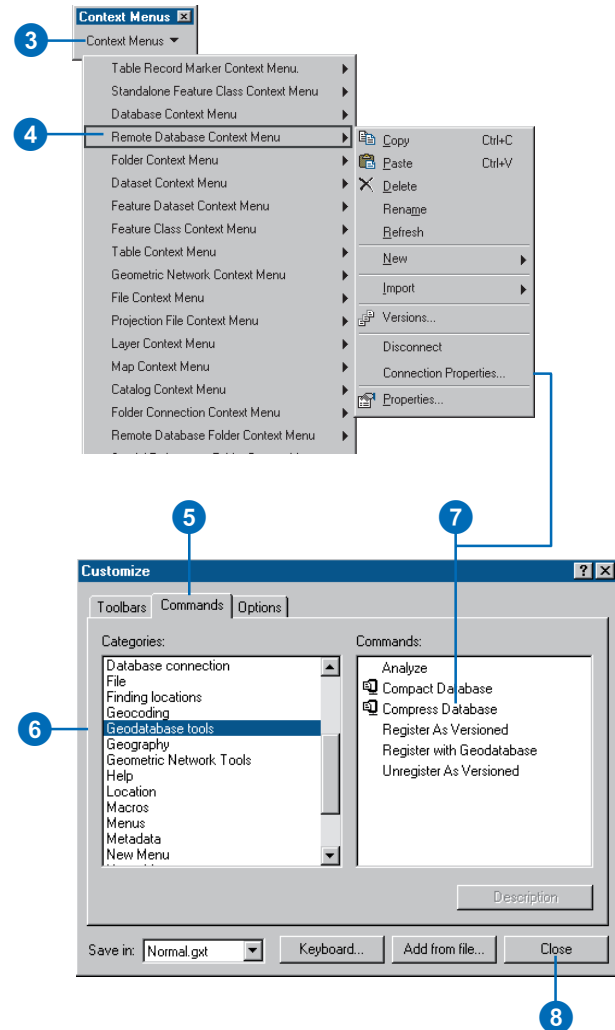
1. Click View, click Toolbars, and click Customize in ArcCatalog.
2. Check Context Menus in the list of toolbars. ►



3. Click the Context Menus menu.
4. Click the arrow next to the Remote Database Context Menu.
5. Click the Commands tab in the Customize dialog box.
6. Click Geodatabase tools.
7. Click and drag the Compress Database command from the Commands list and drop it on the context menu.

The command appears in the context menu.

8. Click Close on the Customize dialog box.



Tip

Analyze

After compressing a database, always analyze your data to update the database statistics. Use the same method to add the Analyze command to ArcCatalog as you use to add the Compress command.

See Also

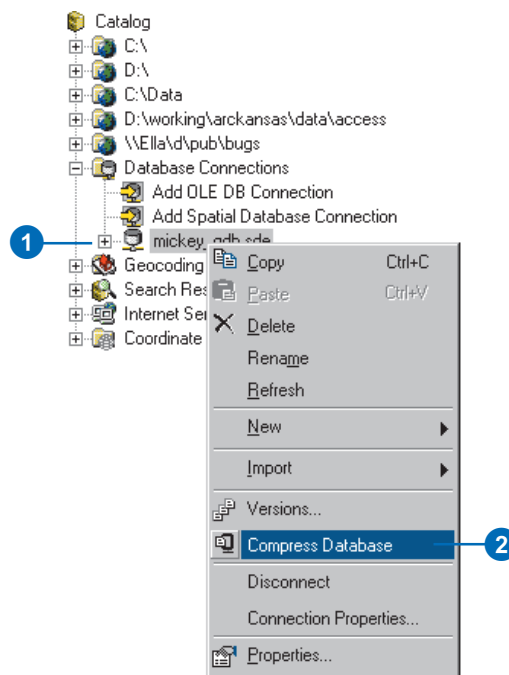
For more information on how to administer an ArcSDE database, see Managing ArcSDE Services.

See Also

For more information on how to create connections to databases in ArcCatalog, see the 'Introduction' chapter in this book.

Compressing the database

1. Create a new database connection in ArcCatalog as the ArcSDE administrative user.
2. Right-click the new database connection and click Compress Database.



Working with versions in ArcMap

In ArcMap, you can view and work with multiple versions simultaneously, create new versions, and change the feature classes or tables from one version to another version. You can also use the version manager, refresh a version's *workspace* connection, and modify available feature classes in ArcMap.

To create a new version, at least one version must be present in the map. If multiple versions are present, you will need to specify the parent version. The newly created version will then be identical to the parent version.

Changing versions allows you to quickly navigate between two versions by changing the feature classes currently in the map. This simplifies the process of viewing the differences between feature classes or performing an analysis with two versions. ►

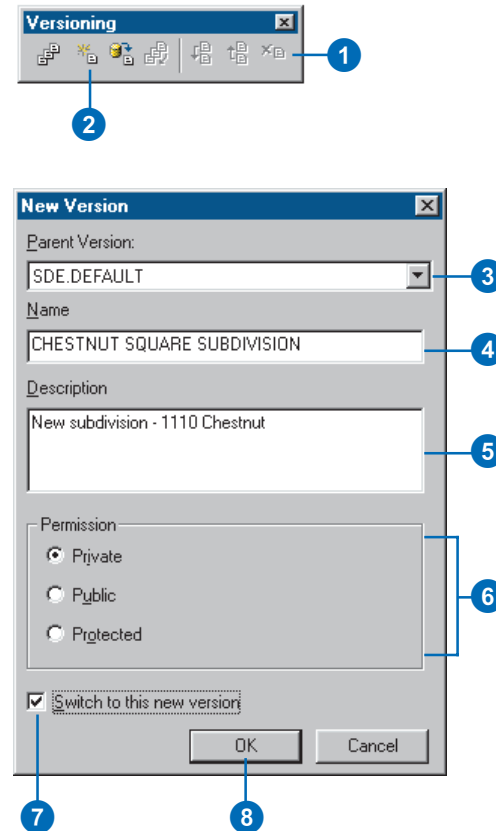
Tip

Creating backup versions

You can create alternative versions as online backups to the original version.

Creating a new version in ArcMap

1. Add the Versioning toolbar to the map.
2. Click the Create New Version button. At least one version must be in ArcMap before the button is enabled.
3. Click the Parent Version dropdown arrow and click the parent version from which you want to create the new version.
4. Type the new version's name.
5. Optionally, type a description.
6. Click the appropriate permission type.
7. Optionally, if you are not currently editing, check the check box to switch the parent version to the new version.
8. Click OK.



When a version workspace is changed to a different version, all feature classes present in the workspace will represent the new “target” version.

Two methods are available in ArcMap for changing versions. You can change versions from the Versioning toolbar or in the table of contents.

When you work in a multiuser environment, the database may be modified by another user at the same time you’re viewing the database. Therefore, the feature classes present in ArcMap may become outdated.

To update the feature classes in ArcMap, you can refresh one or all of the version workspaces present.

To refresh all the versions, click the Refresh button on the Versioning toolbar. To refresh an individual workspace, use the Refresh command on the table of contents context menu.

While you are editing, the Refresh button and the Refresh command for the version’s workspace are unavailable. ►

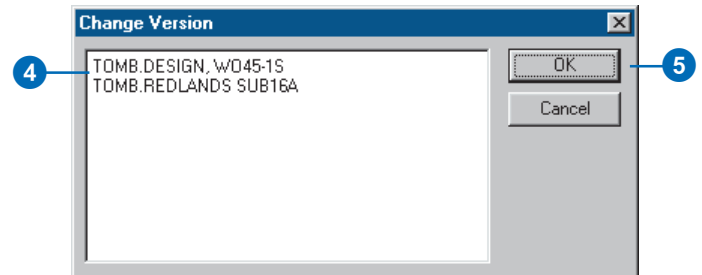
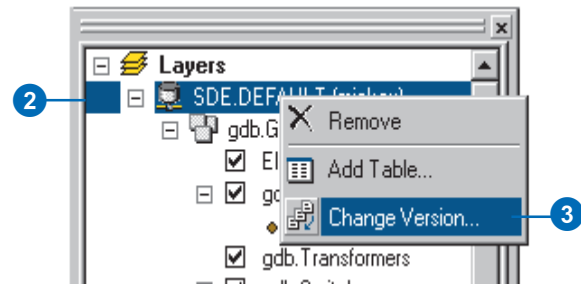
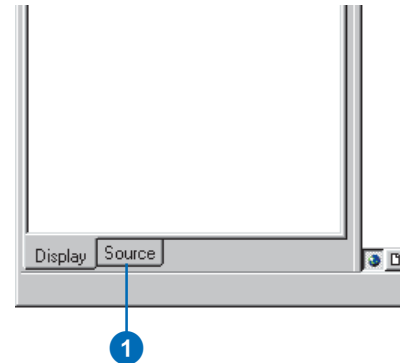
Tip

The Change Version command

Use the Change Version command instead of adding multiple version workspaces to your map document.

Changing versions

1. Click the Source tab at the bottom of the ArcMap table of contents to list the workspaces in your map.
2. Right-click a version workspace.
3. Click Change Version.
4. Click the version to which you want to change.
5. Click OK.



You can have as many versions in the map as needed, but you can only edit one version per *edit session*.

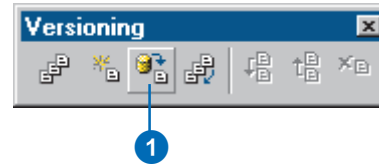
Tip

Preserving a version

If you need to preserve a current representation of the database, create a new version before refreshing.

Refreshing a workspace

1. Click the Refresh button on the Versioning toolbar.



Editing and conflict resolution

The geodatabase is designed to efficiently manage and support long *transactions* using versions. The geodatabase also allows multiple users to edit the same version at the same time. Each edit session in ArcMap is its own representation of the version until it is saved. Saving the edit session applies your modifications to the version, making these changes immediately accessible in the database.

When multiple users simultaneously edit a version or *reconcile* two versions, *conflicts* can occur. Reconciling is the process of merging two versions. Conflicts occur when the same feature or topologically related features are edited by two or more users, and the database is unclear about which representation is valid. Conflicts are rare but can occur when overlapping geographic areas in the database are edited. To ensure database integrity, the geodatabase detects when a feature has been edited in two versions and reports it as a conflict. ArcMap provides the necessary tools for you to investigate conflicts, though you make the final decision as to the feature's correct representation.

ArcMap provides tools to resolve conflicts and reconcile and *post* versions. ArcMap also provides the tools to detect errors created in topologies and to validate these issues. The next sections explain these capabilities in more detail.

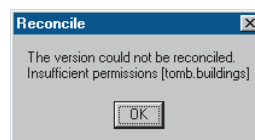
Reconcile

The Reconcile button in ArcMap merges all modifications between the current edit session and a target version you select. Any differences between the features in the target version and the features in the edit session are applied to the edit session. Differences can consist of newly inserted, deleted, or updated features. The reconcile process detects these differences and discovers any conflicts. If conflicts exist, a message is displayed, followed by the conflict resolution dialog box. You must reconcile edits with a target version before posting them to that version. A

target version is any version in the direct ancestry of the version such as the parent version or the DEFAULT version.

In addition, the reconcile process requires that you are the only user currently editing the version and you are the only user able to edit the version throughout the reconcile process until you save or post. If another user is simultaneously editing the version or attempts to start editing since you have reconciled, an error message will inform you that the version is currently in use.

The reconcile process requires that you have full permissions to all the feature classes that have been modified in the version being edited. If a feature class is modified in the version for which you do not have update privileges, an error message appears. You will not be able to reconcile the versions; a user with adequate permissions to perform the reconcile must do this for you.



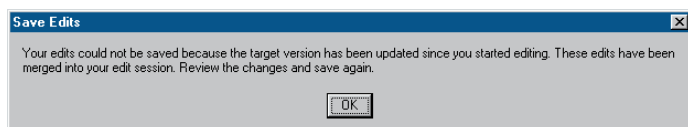
An error message appears when you do not have permission to a feature class to reconcile versions.

For example, suppose you have completed your changes in a version and need to post the version to the database. You must first reconcile the version with a target version you select, resolve any conflicts if necessary, then post.

Autoreconciliation

When multiple users are simultaneously editing the same version and one user has already saved his or her edit session, ArcMap

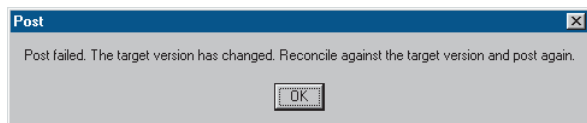
can notify you when you save that the edit session has been reconciled with the version's current representation. You also have the option to explicitly perform the reconcile and save without notification. You may want the notification to allow further inspection of the results from the reconciliation. It's an opportunity to review the difference introduced from the reconciliation between the edit session and the current version's representation. If conflicts are detected, you will be informed with a warning message, and the save process will fail.



The Save Edits dialog box appears when an edit session could not be saved because another user has modified and saved an edit session prior to the current user saving his or her edit session.

Post

You can post a version after you have performed a reconcile. Once the edit session has reconciled with a target version, clicking the Post button synchronizes the edit session with the reconciled version and saves the data. Posting cannot be undone, as you are applying changes to a version that you are not currently editing. If the reconciled version is modified between reconciling and posting, you will be notified to reconcile again before posting.



This message indicates that the target version has been modified since the reconciliation; reconcile again before posting.

Conflicts

Conflicts occur when the same feature (topologically related features or relationship classes) is modified in two versions—the current version being edited and a target version. Conflict detection only occurs during the reconciliation process. If conflicts are detected, a message appears, followed by the conflict resolution dialog box.

There are three categories of conflicts: when the same feature has been updated in each version, when the same feature has been updated in one version and deleted in the other or vice versa, and when the same feature has been deleted in one version and updated in the other version.

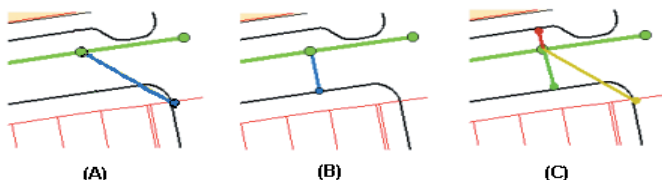
When conflicts are detected, the parent version's feature representation takes precedence over the edit session's representation. Therefore, all conflicting features in the current edit session are replaced by their representation in the parent version. If multiple users are editing the same version and conflicts are detected, the feature that was first saved, the current version's representation, is preserved by replacing the edit session's feature representation. ArcMap ensures database integrity by forcing you to interactively inspect each conflict and resolve the conflict by replacing the feature in the current version with your edit session's representation.

Conflict resolution

Once conflicts are detected, a conflict resolution dialog box appears containing all the conflict classes and their features or rows in conflict. The conflict resolution dialog box allows you to interactively resolve conflicts at the level of the feature class or individual feature. Resolving the conflict implies that you will make a decision as to the feature's correct representation; this could mean doing nothing at all if you are satisfied with the current feature's representation.

You can choose from three representations of the conflicting feature or row to resolve the conflict. The preedit version is the feature's representation when you initially started editing, before making any changes. The edit session version represents the feature as it existed before you performed the reconcile. The last representation is the conflict version, the feature's representation in the conflicting version.

Selecting a feature class or individual feature displays any of the three representations of the feature in the map. The preedit version is displayed in yellow, the edit session's version is displayed in green, and the conflict's version is displayed in red. You can also optionally enable or disable the display settings for each version (preedit, edit session, and conflict) by clicking the Display command on the context menu and checking or unchecking the corresponding version.



The lateral in blue as it existed prior to editing (A), the lateral after being modified (B), and the three representations during conflict resolution (C).

When you select a feature in the conflict resolution dialog box, each version's representation of the feature's or row's attributes is listed in the bottom half of the box. A red dot to the left of the field name identifies why the feature is a conflict. For example, if the feature's geometry was edited in each version, a red dot appears next to the shape field. The same principle holds true for attribute conflicts. If a feature has been deleted in either version, "<deleted>" appears for that version's attribute value. Therefore,

a red dot marks each column, signifying that each column is an update-delete or a delete-update conflict.

Resolving a conflict implies that you made a conscious decision about the feature's correct representation. You can select the feature in the conflict resolution dialog box and replace the current feature in the map with any of the three representations of the feature. This allows you to quickly update and replace conflicting features. If further modifications are required, you can simply use any of the ArcMap editing tools to update the feature.

Conflicts			
Conflicts - Mains 1864 1890 1866 - Services - Junctions			
Property	Conflict	Edit	Pre-Edit
FID	1	1	1
• SHAPE			
SEG_ID	33211070	33211070	33211070
• SYMBOL			
PIPE_SIZE	8	8	8
ACC_NO	16-A	16-A	16-A
SEW_NO	38952	38952	38952
MATERIAL	STEEL	STEEL	STEEL
SEW_SHAPE	CIRC	CIRC	CIRC
HEIGHT	0	0	0
WIDTH	0	0	0
INST_YEAR	1982	1982	1982
DRAIN_AREA	S.BRANCH	S.BRANCH	S.BRANCH
SEP_COMB	SEPTIC	SEPTIC	SEPTIC
PUB_PRI	PUBLIC	PUBLIC	PUBLIC

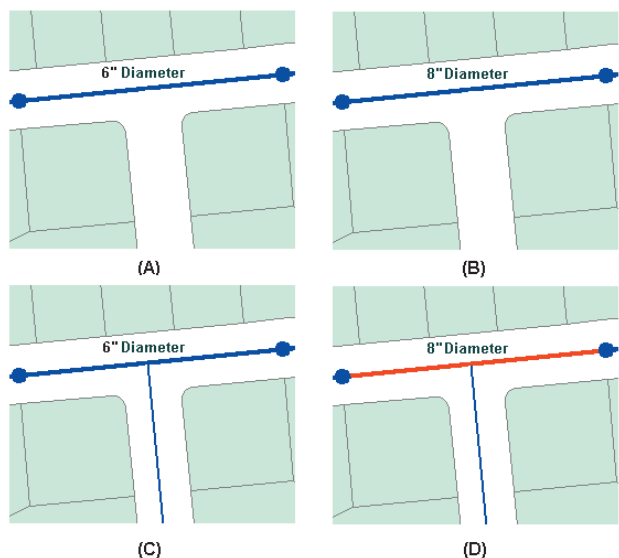
This conflict resolution dialog box shows two feature classes with conflicts and a feature with each of its version's attributes.

Conflicts with geometric networks, feature-linked annotation, and relationships

Resolving conflicts with features that are related to other features through *geometric networks*, feature-linked *annotation*, and *relationship classes* is different than resolving conflicts with

simple feature classes. Because each of these feature classes has specific geodatabase behaviors that can impact other feature classes, resolving a feature conflict may impact related features.

When you edit network features, changes to the geometric network and to the logical network may create conflicts. For example, when you add a service to a main, the main will not be physically split in the geometric network but will be split in the *logical network*. Therefore, while you have not directly edited the main's geometry, it has been edited logically. If the target version you are reconciling has also modified the main, then the new service you inserted will create a conflict with the main.



The original water main (A), the water main changed to an 8-inch diameter in the first edit session (B), a new service inserted in the second edit session (C), and the water main in red shown as a conflict (D).

Resolving a conflict involving geometric network feature classes requires understanding how the Replace With command in the conflict resolution dialog box will update the existing network topology present in the edit session.

In the previous example, two users modified the water main—one by changing an attribute and the second by connecting a new service. Resolving the conflict would merely require investigating the differences and seeing that the conflict is valid and no further resolution is required. Since the main contains the correct attribute for the diameter, the new service is correctly connected to the main. But there are cases when resolving conflicts involving a junction feature class will also update the connected network edge.

Working with feature-linked annotation requires that you remember one rule: when replacing a feature that has feature-linked annotation, both the feature and the annotation are replaced with the new feature and annotation. You may have to further edit the new annotation. For example, you may encounter a conflict in which you have moved a feature and repositioned its annotation. The conflict version has performed the same edit, moving the feature and rotating the annotation. Your decision is to replace the feature with the conflict version's feature. This action deletes the existing feature-linked annotation, inserts the conflict feature, and creates a new annotation. You will then need to further edit the new annotation by moving and rotating it as necessary.

Relationships have similar dependencies to feature-linked annotation. Deleting a feature from an origin relationship class may trigger a message to delete a feature from the destination relationship class. Therefore, be aware of the ramifications of simply replacing conflicts involving feature classes that participate in relationship classes.

An example of when a conflict can arise between relationship classes is if you were to update the origin class primary field, breaking the relationship in version A. At the same time, in version B, the destination class-related feature is also updated. When you reconcile the versions, since the destination class is dependent on the origin class, a conflict is detected. A similar example is if you were to delete a pole that has a relationship to a transformer, the transformer is also deleted. But in the conflict version, the transformer's attributes are edited. An update/delete conflict would be detected when reconciled.

Conflicts with topologies

Because features in feature classes that participate in a topology can share geometry with other features, the process of resolving conflicts between versions of topological feature classes is different from resolving conflicts with simple feature classes. It is also different from that used to resolve conflicts with geometric networks, relationship classes, and feature-linked annotation.

When a feature class that participates in a topology is edited, other topologically related features may be simultaneously changed. The changed features may belong to the same feature class or to one or more other feature classes. To manage the process of detecting new topology errors that may have been introduced by edits, topologies record where edits have been made as dirty areas. Editing features in a topology creates dirty areas in the topology.

New topology errors may occur when edited parent and child versions are reconciled, even when the dirty areas within each version have been validated and are free of errors. In order to detect such topology errors, the dirty areas in a child version are all returned to dirty status after changes from the parent version are brought into it during a reconcile. After the reconcile these areas can be revalidated and any errors will be detected.

Reconciling two versions that do not contain active dirty areas may still result in dirty areas. Any dirty area that has been present in the child version, whether it has been validated or not, will be a dirty area after the versions are reconciled. In general, when reconciling a version:

- Any dirty area the child version inherited from the parent version, whether it is validated in the child version or not, will be a dirty area after the reconcile.
- Any dirty area that was created for any feature that was created, updated, or deleted in the child version, whether it is validated or not, will be a dirty area after the reconcile.

For more information on reconciling versions with topologies, see the section on topology and versioning in the 'Topology' chapter in this book or in the chapter 'Editing topology' in *Editing in ArcMap*.

Editing a version

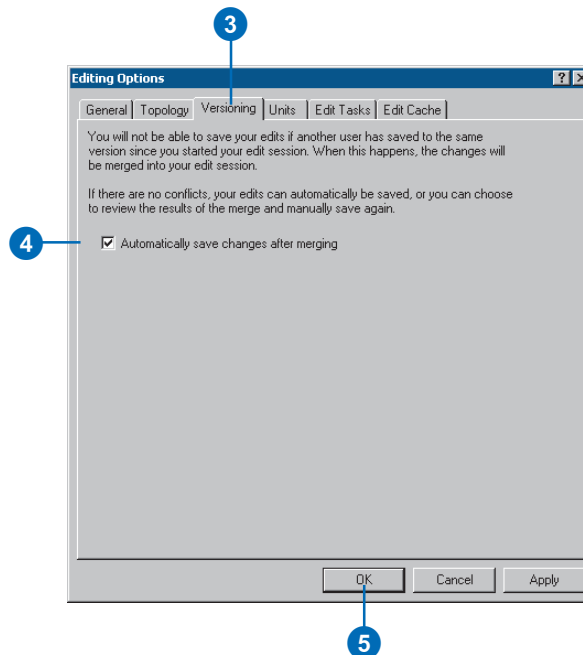
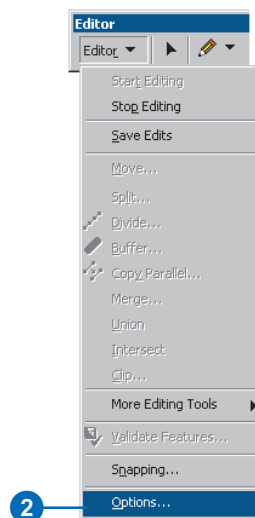
The ArcMap editing toolbar provides the arena for editing versions, reconciling versions, resolving conflicts, and posting versions.

When you start editing, if multiple versions are present in the map, you will have to select one version. Starting an edit session on a version creates a new, unnamed, temporary version that exists until you save or end the edit session. You are the only user who can see your changes until you explicitly save.

When saving an edit session, you have an option to enable or disable autoreconciliation. If enabled, autoreconciliation will automatically reconcile your edit session with the version's current database state and save, making your changes available to others using the database. If autoreconciliation is not enabled, then when you save, your edit session will be reconciled with the version's current database state. A message will inform you that the edit session has been reconciled but has not been saved. This will only occur if a second user has also edited the version and saved since you ►

Enabling and disabling autoreconciliation

1. Click Editor and click Start Editing.
2. Click Editor and click Options.
3. Click the Versioning tab.
4. Check the check box to enable autoreconciliation or uncheck the check box to disable autoreconciliation.
5. Click OK.



started editing. You will need to save again to make your changes available to others using the database.

Based on your organization's work flow, you may eventually need to reconcile two versions. Reconciliation is the process of merging features from a target version into the current edit session. Reconciliation must be done before posting changes to another version.

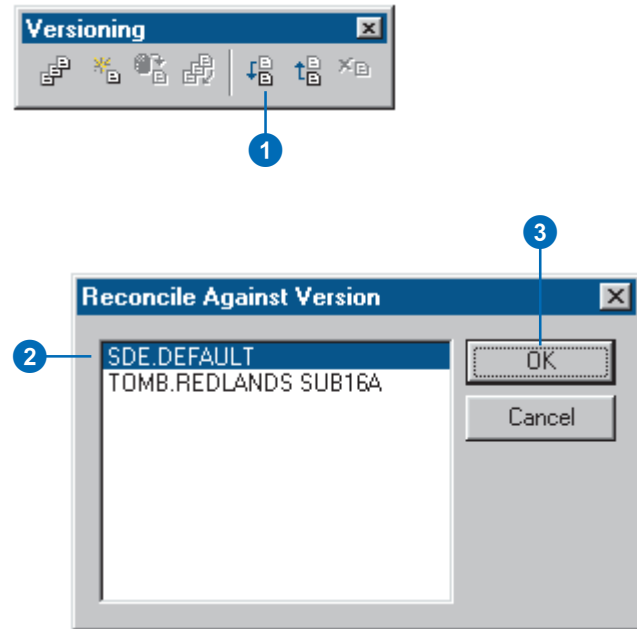
During reconciliation, conflicts may be discovered. Conflicts arise when the same feature is updated in each version or updated in one and deleted in the other.

When conflicts arise, an interactive conflict resolution dialog box will provide the tools necessary to resolve the conflicts. For each conflict, you can choose whether to replace the feature in your edit session with the conflict version, the version from your edit session, or the version as it existed at the beginning of your edit session.

Once you have successfully completed the reconciliation, you can post the version. The post operation synchronizes your edit session with the target version. They are then identical.

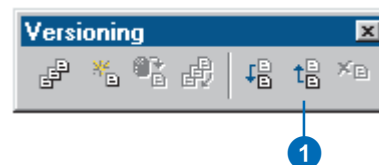
Reconciling

1. Click the Reconcile button on the Versioning toolbar.
2. Click the target version.
3. Click OK.



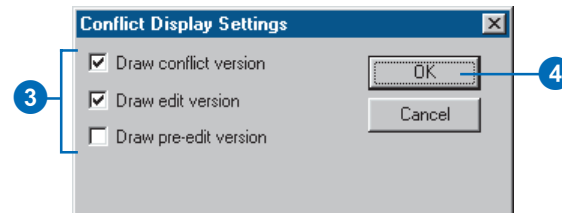
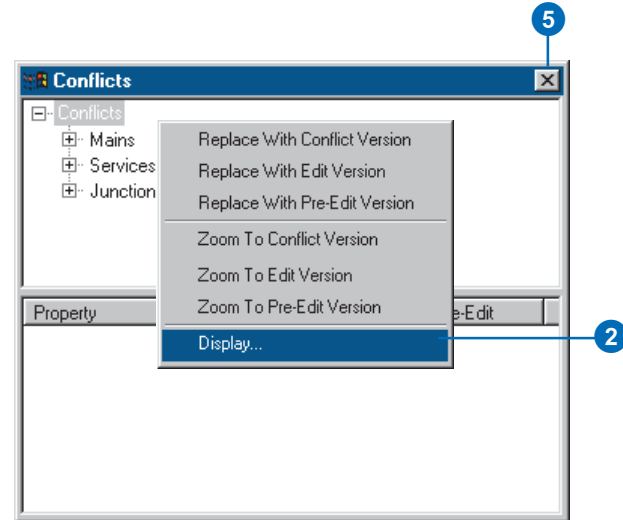
Posting

1. Click the Post button on the Versioning toolbar.



Displaying conflicts

1. Click the Conflicts button on the Versioning toolbar.
2. Right-click Conflicts and click Display.
3. Click the appropriate check box to display each conflict category.
4. Click OK.
5. Click the Close button to close the Conflicts dialog box.



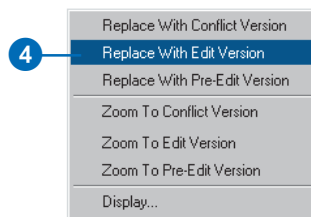
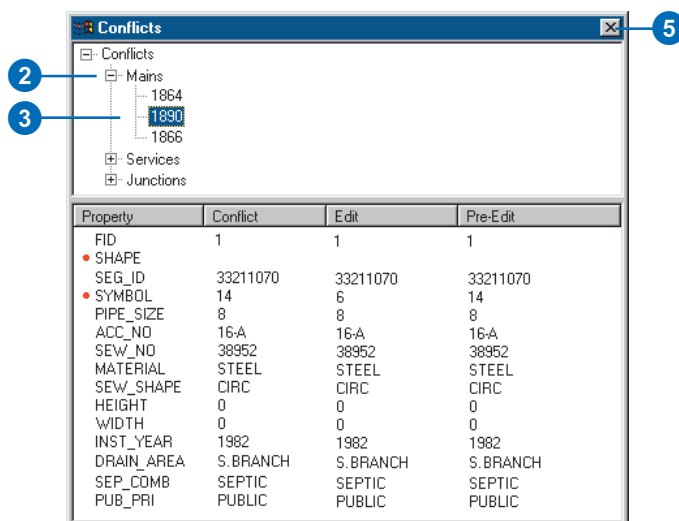
Tip

Topology errors

While editing a versioned geodatabase, the topology rules might have been violated. To validate these errors, use the validation techniques described in the 'Topology' chapter in this book or in the chapter 'Editing topology' in Editing in ArcMap.

Resolving conflicts

1. Click the Conflicts button on the Versioning toolbar.
2. Click a feature class.
3. Click a feature and right-click to display the context menu.
4. Click the appropriate Replace With command to resolve the conflict.
5. Click the Close button to close the Conflicts dialog box.



Versioning scenarios

The following scenarios show how an organization can implement its work flow process using a versioned database. These examples demonstrate several techniques available for performing long transactions in a multiuser environment. It is likely that organizations will, in some manner, use each of these techniques depending on the task.

Scenario 1: Simple database modifications

Task: Multiple users are concurrently editing the database, performing common map sheet changes such as inserting new features, updating attributes, and removing out-of-date facilities.

Solution: Users can simply connect to the DEFAULT version (simultaneously), start editing, and save their changes when their work is complete. Users do not have to create new versions to modify the database. If another user has edited the DEFAULT version since the current user has started editing, the user saving is notified that the version has been changed and, therefore, the version will need to be saved again. Users may bypass this warning message by enabling autoreconciliation in the ArcMap Options dialog box. Also, if two users modify the same feature during their edit sessions, the second user to save encounters a conflict. The user then has to decide what the feature's correct representation is and save the edit session.

Scenario 2: Transactions spanning multiple days

Task: Update the database to incorporate new and updated facilities in the field, which will likely require multiple edit sessions and a couple of days to complete.

Solution: A user creates and switches to a new version derived from the DEFAULT version. The user starts editing the new version and begins modifying features and saving as required. The user can resume the edit session, as appropriate, the following day or possibly the following week. When the changes

are complete and ready to be posted to the DEFAULT version, the user must first click the Reconcile button on the Versioning toolbar. If conflicts are detected, the user can resolve the differences and complete the transaction by clicking the Post button. The posting process applies all the changes in the user's version to the DEFAULT version. The user can then delete the version.

Scenario 3: A work flow process

Task: Create individual versions for each step or stage of the work order and work flow process and post the work order to the database.

Solution: A user or supervisor creates a new version derived from the DEFAULT version. The user starts editing the new version and begins modifying features or creating a new design. When the user has completed the design or proposed modifications, the work order can be submitted to a supervisor for review. At this time, a new version can be created to ensure the preservation of the initial design. The new version can then be further modified or adjusted as required. Once the work order has been approved for construction, another version can be created. The purpose of this version is to reflect any changes that may occur while the work order is being constructed in the field. Finally, as the construction is completed and the new facilities are in service, the work order must be posted to the database. A user can then start editing the work order, perform a reconcile with the DEFAULT version, resolve any conflicts if necessary, and post.

The solution allows the organization to create new versions of the work order for each step of the project—the initial design or proposed version, a working or accepted version, and a version for the construction phase. Each version is preserved and available to look back on for historical purposes. The final step is to post the constructed version to the database. The project

completes a full circle from start to finish, creating individual versions at each step.

Scenario 4: Restricting permissions to the database

Task: The organization’s supervisor has restricted write access to the DEFAULT version, requiring managerial review of each user’s edits prior to posting the changes to the database.

Solution: To restrict write permissions to the database (the DEFAULT version), the ArcSDE administrative user can set the permission of the DEFAULT version to “protected” using the version manager. This allows users to continue to view the DEFAULT version but does not allow users to start editing the version. Therefore, users will need to create new versions for editing the database, similar to Scenario 2. When a user has completed and saved the edit session, the ArcSDE administrator can reconcile the version with the DEFAULT version. To accomplish this task, the manager who connects to the database as the ArcSDE administrator starts editing the user’s version and clicks the Reconcile button. The process will merge all the changes in the user’s version and the DEFAULT version. If conflicts are detected, the manager can resolve the conflicts and save the edit session. Once the edits are acceptable to the manager, the version is ready to be posted to the DEFAULT version. The ArcSDE administrative user can then start editing the version, perform a reconcile, and post the version. The user’s version can then be deleted.

Scenario 5: Compressing the database

Task: The geodatabase has been edited for an extended time, and the number of database states and rows in each feature class’s delta tables has significantly increased. How do we improve performance?

Solution: The Compress command will remove all database states that are no longer referenced by a version and move all the rows in the delta tables, which are common to all versions, to the base table. To achieve the maximum benefit when running the Compress command, you will need to first reconcile, post, and delete each version with the DEFAULT version. Sometimes this may not be a reasonable option based on your organization’s work flow. At minimum, to improve performance, simply reconcile each version with the DEFAULT version and save, then perform the compress. This will ensure that all the edits in the DEFAULT version will be compressed from the delta tables to the business table. Remember, the Compress command can still be executed without first reconciling, posting, and deleting each version, but the benefits may not be as noticeable.

Disconnected editing

12

IN THIS CHAPTER

- **Disconnected editing**
- **Checking out data from a geodatabase**
- **Customizing a check-out**
- **Checking in data to a geodatabase**
- **Managing check-outs**

Exchanging and maintaining data between geographically dispersed users has become an integral part of the work flow process. Some of these remote users may be connected via a Wide Area Network (WAN) or Local Area Network (LAN) to a central office infrastructure; others will be more mobile, requiring only to download and upload data to and from central office databases on an infrequent basis.

These organizational requirements are supported in the geodatabase.

Disconnected editing functionality does not apply to ArcView licenses.

Disconnected editing

Many organizations have users who work at remote locations, such as a regional suboffice, but who still operate within the same infrastructure as the central office via a WAN or LAN. For those remote users, the overhead of maintaining an active connection to a remote database server may be a consideration. One solution would be to transfer data from the central database to the remote location, disconnect from that central database, and continue to work offline. This avoids the overhead of the remote database connection, but allows users to maintain connections to other network-based services such as e-mail.

More mobile users, such as a field survey crew, require similar functionality. They need to disconnect completely from an organization's infrastructure, often for a prolonged period of time. When preparing for a particular work order or project, the relevant data would be transferred to a portable device, such as a laptop. This device would then be disconnected from the network, enabling the user to operate independently.

Remote or mobile users may continue to work with and modify the data even though they are disconnected from the network. When a connection to the network is reestablished, any changes made to the data will be transferred to, and integrated with, data maintained in the central database.

Geodatabase disconnected editing allows organizations to disseminate their spatial data to other departments, associated agencies, or mobile workers and maintain the integrity and currency of that data.

How does disconnected editing work?

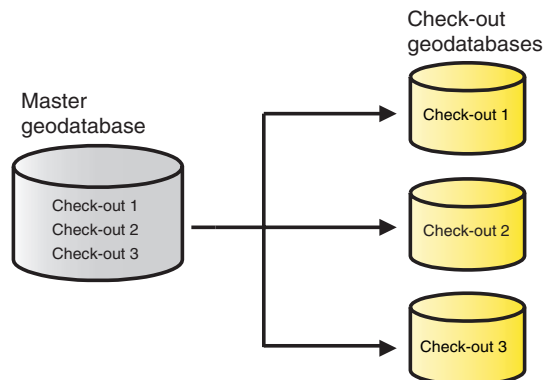
Disconnected editing is supported by two principal mechanisms—checking out and checking in data.

Checking out data from a geodatabase involves taking a copy of some data from one geodatabase, *the master geodatabase*, and transferring it to another geodatabase, the *check-out geodatabase*. This copy is undertaken within a framework that allows the data to be reintegrated with the master geodatabase at a later time.

Checking in the data from a check-out geodatabase involves establishing whether any changes have been made to the data in the check-out geodatabase, and transferring those changes back to the master geodatabase.

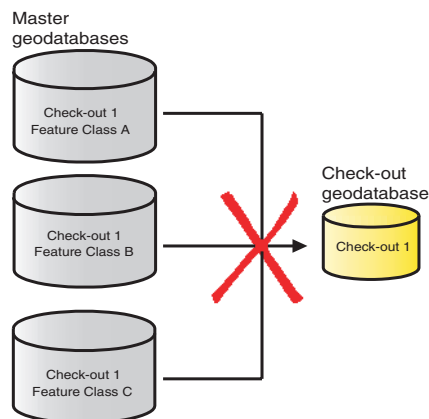
The master geodatabase

The master geodatabase must be an ArcSDE geodatabase. Data cannot be checked out from a personal geodatabase. Data in a master geodatabase must be registered as versioned before it can be checked out. A master geodatabase may also host multiple check-outs to multiple check-out geodatabases.



A master geodatabase hosting multiple check-outs

The creation of each check-out is performed on a per geodatabase connection basis, which means data from only one ArcSDE connection may be checked out to one check-out geodatabase; it is not possible to check out data from multiple master geodatabases to one check-out geodatabase at the same time.



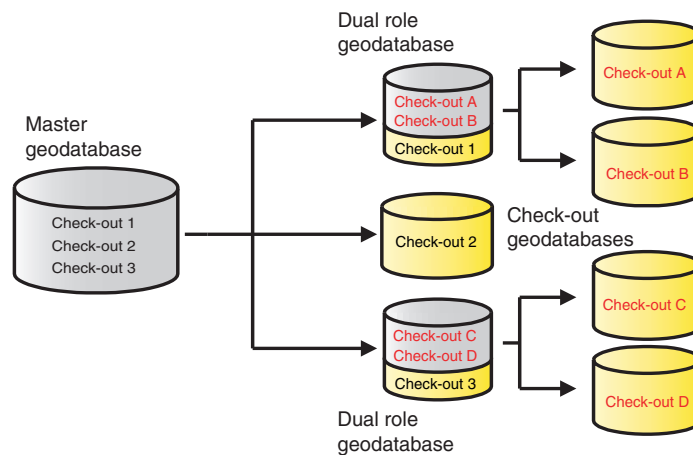
Checking out data from multiple geodatabases to one check-out geodatabase is not supported.

The check-out geodatabase

The check-out geodatabase may be either a personal geodatabase or another ArcSDE geodatabase. ArcSDE, on all the currently supported database management system (DBMS) platforms, is supported as a check-out geodatabase.

Once a geodatabase contains data checked out from a master geodatabase, it cannot be used again for checking out until the changes have been checked in. Each *check-out* geodatabase will support only one check-out from a master geodatabase.

However, any ArcSDE geodatabase may also adopt a dual role by acting as both a check-out and a master geodatabase from which further check-outs may be made. This database may contain one check-out from another master geodatabase and could also host multiple check-outs to other check-out geodatabases.



Dual role master and check-out geodatabase

Preparing a check-out

The first step in preparing a check-out is to identify what data is required. It is important that the data model and nature of edits to be made in the check-out geodatabase are understood and that the appropriate amount of data is included in the check-out. As a general rule, check out more data than is required for editing purposes. Edits made to features close to, or on the boundary of, the check-out area may affect other features not included in the check-out. Including more data in the check-out than you need will avoid potential problems or unexpected behavior when the data is checked in.

ArcMap provides the framework for checking out data in ArcGIS. An ArcMap document should be prepared to support each check-out operation. For example, if you wish to limit the extent of data to check out, zoom in to the area of interest before you start the check-out process. If you wish to check out certain features only, apply the selection or define the query on the relevant layer or table before you start.

Once the check-out is initiated, the following default behavior will determine what data is checked out:

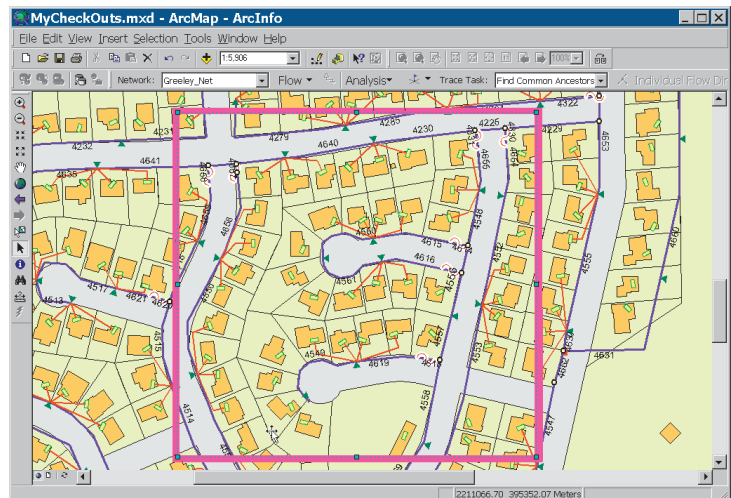
- All layers and tables present in the ArcMap document for the selected ArcSDE geodatabase will be included in the check-out.
- The following data filters will be applied:
 - The spatial extent of the data to check out will be determined by the current view extent of the ArcMap document or the boundary of a currently selected graphic.
 - Any selections that already exist on those layers or tables will be maintained.
 - Any definition queries already applied to those layers or tables will be maintained.

Note: The intersection of these data filters will determine what data will be checked out.

- To maintain data integrity, any related objects will automatically be included in the check-out, regardless of whether or not they are currently present in the ArcMap document.
- The list of data to check out will automatically be expanded to include dependent datasets. For example, all feature classes in a geometric network, topology, or feature dataset will be included if just one feature class in the network, topology, or feature dataset is selected for check-out.

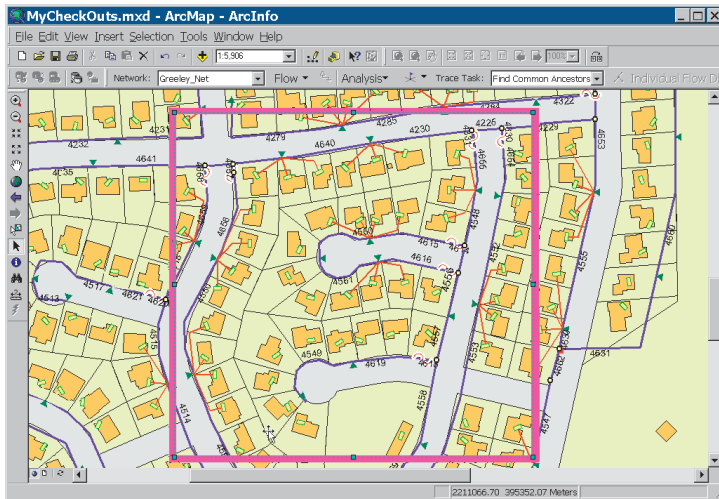
The following electric utilities' maintenance work orders will help illustrate some of the default behavior for checking out data.

A maintenance crew is preparing to inspect some of the electric utilities in a residential area. In order to do some field editing, they need to check out that part of the electric network that covers this residential area. To start the check-out process, the spatial extent of the inspection area is identified using a spatial filter (in this case, the extent is determined by a selected graphic).



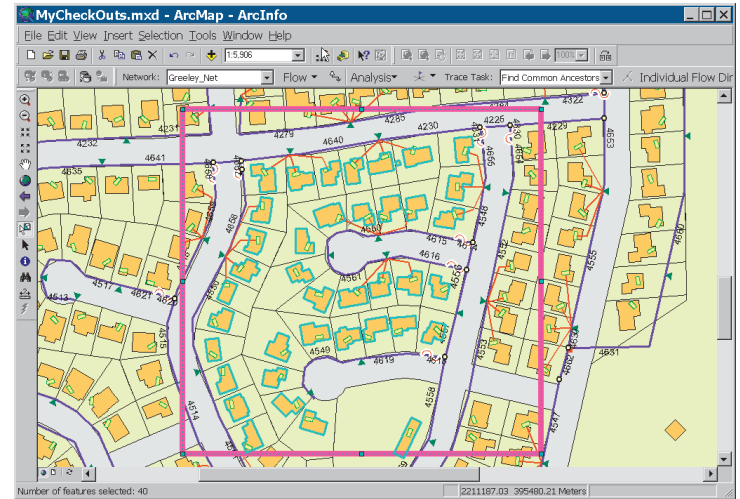
Spatial extent filter applied to some electric utility data

The crew is to concentrate on cables that have been insulated with a particular material. To identify these cables, a query is applied to the relevant dataset.



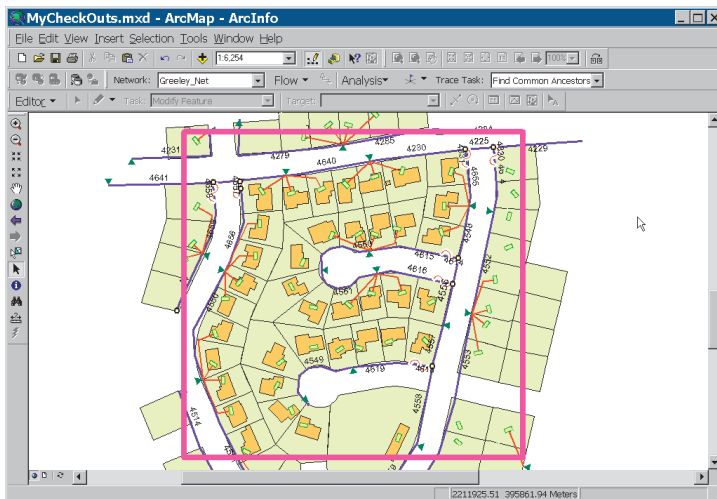
The results of a definition query—cables (in red) insulated with a particular material

Finally, as each maintenance crew can expect to visit only a certain number of properties in a day, the homes in one residential block are identified by a selection based on property numbers.



Attribute selection to identify certain properties

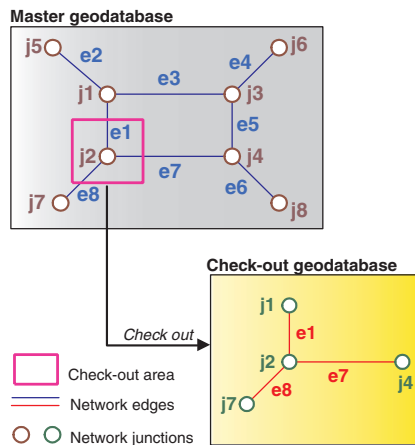
The selected features, features identified by a definition query, and features that intersect the chosen spatial extent will be checked out. Some additional network features have also been included. How geometric networks are checked out is explained in greater detail in the next section.



The data checked out to the check-out geodatabase

Checking out geometric networks

All the feature classes that participate in a geometric network are checked out together to maintain the connectivity of the network in the check-out geodatabase. Individual network classes cannot be excluded from the check-out. Where the check-out boundary intersects a network edge, the next junction in the network will automatically be included in the check-out.



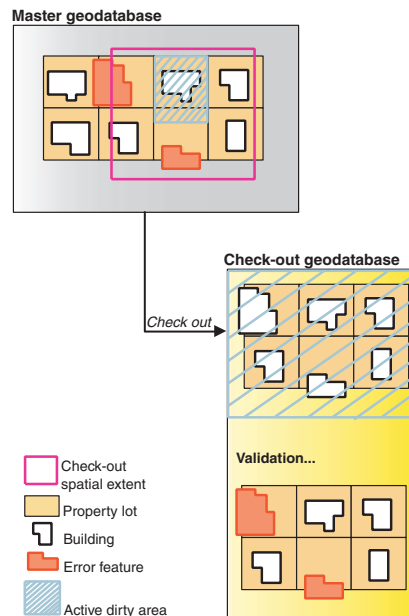
Checking out network features

In this example, all the network elements that intersect with the check-out extent are checked out from the master geodatabase—that is, junction *j2* and edges *e1*, *e7*, and *e8*. Junctions *j1*, *j7*, and *j4* will also automatically be included in the check-out to maintain connectivity in the check-out geodatabase. Again, it is important that the extent of the check-out is large enough to accommodate any modifications to features that may be close to or on the boundary of the check-out area. The implications of not including sufficient network features in the check-out will be discussed later on in this chapter.

Checking out topologies

As with geometric networks, all the feature classes that participate in a topology are checked out together; individual topology feature classes cannot be excluded from a check-out.

When topology feature classes are checked out, the whole extent of the topology in the check-out geodatabase will be marked as dirty. For more information on dirty areas and topologies in general, see the ‘Topology’ chapter in this book. To discover existing errors, the topology must first be validated. The topology will behave as it would in the master geodatabase: edits create dirty areas, and validation creates/deletes errors.

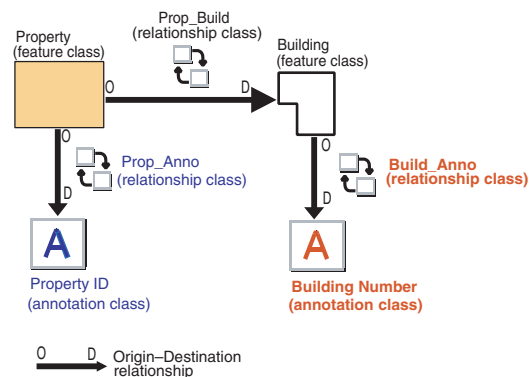


Checking out topological data. In this example, one topology rule is in effect—buildings must be contained by the parcels.

Checking out related data

To maintain data integrity and avoid unexpected behavior, related objects should always be checked out. The check-out process will automatically include related objects that may or may not be present in the check-out ArcMap document. For every geodatabase object that participates as an origin in a relationship class, the check-out will include directly related objects, one step away in the relationship, in the associated destination object. This destination object may itself act as an origin in some other relationship with a different geodatabase object; for performance reasons, the check-out process will not recursively include other objects that may be indirectly related to the related objects already included in the check-out.

The following examples will illustrate the default check-out behavior with respect to related objects. The data model used in these examples is a simple origin–destination relationship between properties, buildings, and their related annotation.

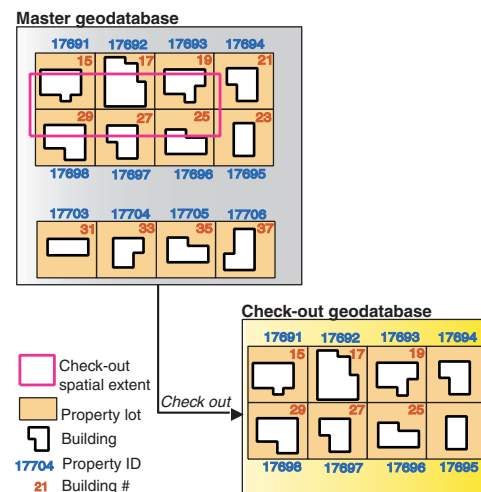


Simple data model for property and buildings

In this model, each property has related annotation, represented by the relationship class *Prop_Anno*. The annotation's objects

are one step away in the relationship with the properties. Each property also has a building—the relationship here is represented by *Prop_Build*. The buildings are, again, one step away from the properties in the relationship. Each building, in turn, has some related annotation, represented by *Build_Anno*. This building annotation is indirectly related to the properties—it is two steps away from the property in the relationship.

Some properties are identified for check-out using a spatial extent filter.

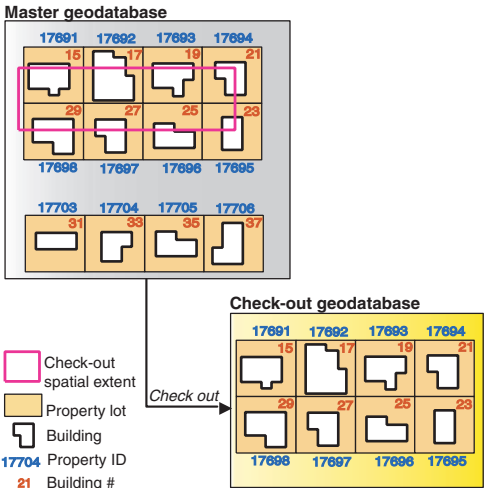


The features and related annotation checked out by default

Although the annotation related to the properties falls outside the spatial extent of the check-out, they are automatically included in the check-out—all features directly related to, or one step away from, the features in the check-out are included by default.

The check-out also includes two building features related to properties 17694 and 17695. Applying the one step related objects

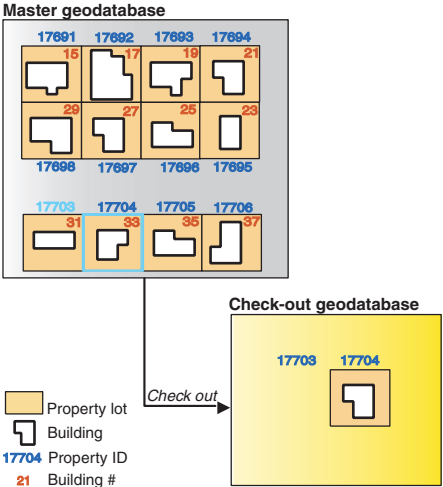
rule, these features are included in the check-out, but the related building annotation is not. To avoid this, include more objects in the check-out than required for editing purposes. In this example, extending the spatial extent of the check-out will ensure that all related features and annotation are included.



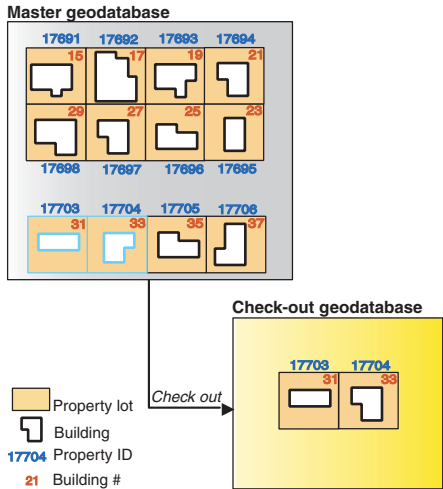
Extend the spatial extent of the check-out to include all related annotation

In the next example, two properties and related buildings are required in the check-out. The property 17704 is identified through attribute selection, along with the annotation for property 17703. In this case, while property 17704, its related annotation, and the related building feature are checked out, the annotation related to the building is excluded. This annotation feature is two steps away in the relationship to the property in the check-out. Because it was selected, the annotation relating to property 17703 is included in the check-out but not the property itself. When checking out data, relationships are traversed in one direction only, from the origin to the destination. As the annotation for the property 17703 represents the destination in relationship to the property, no related property is included.

DISCONNECTED EDITING

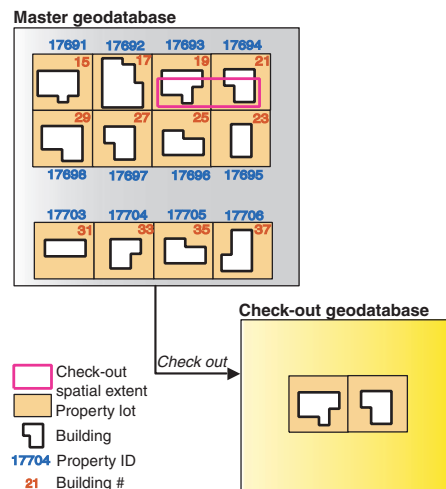


To include the features and the related annotation, redefine the selection to include both the lot and the related building features.



Including all related features in the check-out

In the previous examples, each check-out was made using the default behavior of including related objects. It is possible to override this behavior either at the global or local level to customize each check-out. At the global level, each check-out may be configured to not include any related objects associated with features identified for check-out.



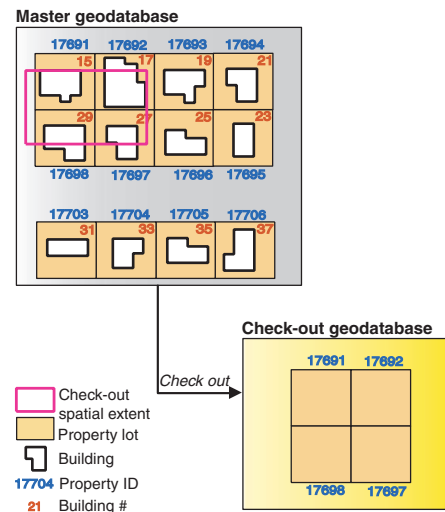
No related objects were required in this check-out—no annotation features are included.

Alternatively, the choice to include or exclude specific classes may also be controlled at an individual class level. The following general rules will apply in this case:

- If either an origin class (for example, lots) or destination class (for example, buildings) is excluded from a check-out, the associated relationship class and, therefore, the related objects will not be included.

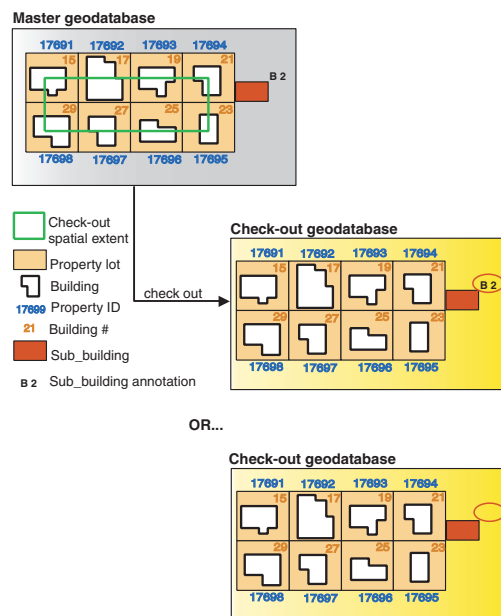
- If a relationship class is excluded from a check-out, the one-step related objects rule does not apply and no related objects will be included in the check-out.
- Nonattributed relationship classes cannot be checked out without the corresponding origin and destination classes; attributed relationship classes may be checked out on their own.

In the next example, although the spatial extent of the check-out includes four properties (17691, 17692, 17698, 17697) that have related buildings, all buildings have been explicitly excluded from the check-out. As the global default behavior to always include related objects is still in effect for the other feature classes, the property annotation will again be included in the check-out.



The features checked out to the check-out geodatabase after buildings have been excluded

In some cases it appears that you are getting more in the check-out than you had anticipated given the model for checking out related data. The order in which the data to be checked out is processed can determine which related data will be included in the check-out. In this next example, one of the properties, property lot 21 (an origin) also has an additional related sub-building feature. This sub-building feature itself is both a destination and an origin as it also has some related annotation (a destination). This related sub-building is included in the list of features to check out by virtue of being a destination object in relation to the property lot feature identified by the check-out extent.



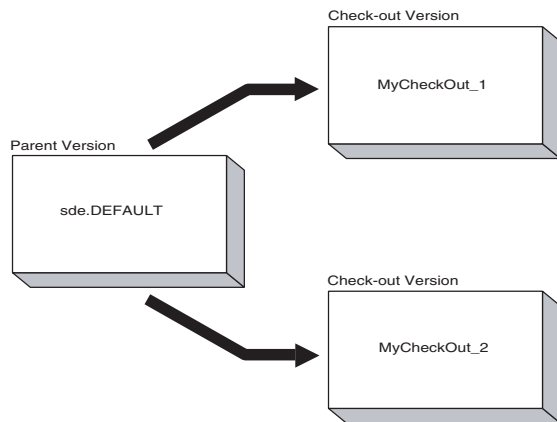
The features checked out to the check-out geodatabase after buildings have been excluded

If the property feature class was first in the list of datasets to check out, the check-out process would evaluate that feature class first, see what was related to the features selected, evaluate the related feature class next, and see what was related to that dataset. In this case, the property, its sub-building, and the sub-building's annotation would be checked out. If the sub-building was processed first, as no rows in that feature class had been explicitly selected for checking out (i.e., it's outside the check-out area), no data related to that sub-building (in this case, annotation) would be included in the check-out. With default behavior automatically including related data, you are always guaranteed to get the data related to the features you have identified for check-out via your check-out extent, selections, or definition queries, but in some cases, depending on the order of processing, you may get additional data related to those related objects as well.

The check-out process

Having identified the data to check out, the check-out process itself involves a number of steps. The selected data, along with any related objects, are copied to the designated check-out geodatabase. A new version of the data, the *master check-out version*, will be created in the master geodatabase as a child of the version to which you are currently connected. This will be created as a public version accessible to all users in the master geodatabase. This version may be set to private, or hidden from general access, if required.

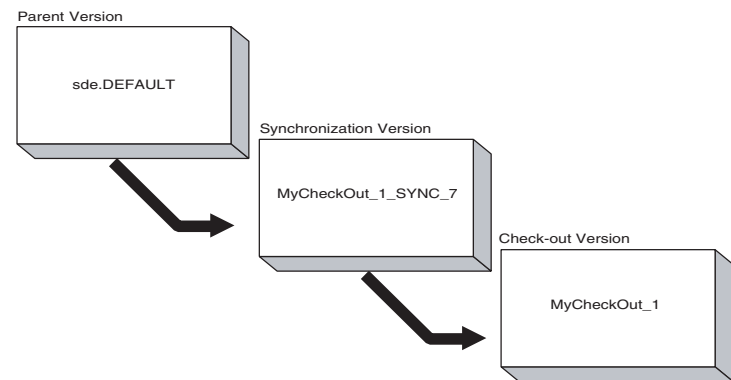
For example, if you are connected to the DEFAULT version, a new version with the same name as the check-out will be created as a child of DEFAULT. This version will receive the changes from a check-out geodatabase when the data is checked in.



The check-out version tree in a master geodatabase

If the check-out geodatabase is an ArcSDE geodatabase, then two new versions are created in the check-out geodatabase. The first version, the *synchronization version*, will be created as a child of the DEFAULT version. This synchronization version will reflect the state of the data at the time the check-out was created and should not be edited.

The second version, the *check-out version*, will be created as a child of the synchronization version. Only the edits made to this version can be checked back into the master geodatabase. Once data has been checked out to an ArcSDE geodatabase, it is automatically registered as versioned.



The check-out version tree in a check-out ArcSDE geodatabase

If the check-out geodatabase is a personal geodatabase that does not support versioning, changes to the data are maintained in a separate DBMS table.

The check-out process concludes by adding information to both the master and check-out geodatabases that describes each check-out. This information includes the name of the user who created the check-out, the data that has been checked out, the time the check-out was created, and the name of the new check-out version in the master geodatabase.

Details about the source of the data, the master geodatabase, are also recorded. This involves storing partial ArcSDE connection information, which will be used during the check-in process. Usernames and passwords are not stored in the check-out geodatabase.

Check-out restrictions

There are some important restrictions to remember when checking out data. First, the version created in the master geodatabase should not be edited while the data it represents is still checked out. As this check-out version in the master geodatabase is public, and therefore editable, when created, any changes made to this version may be overwritten during the check-in process, as no conflict reconciliation is undertaken at this point.

Second, it is not possible to append data to or refresh an existing check-out. You cannot check out an updated version of the same data to a check-out geodatabase while it contains a check-out. You must first check in the original check-out and then make a second check-out. Once data has been checked back in, the check-out geodatabase no longer participates in any check-out/check-in relationship with a master geodatabase. However, it is possible to reuse a geodatabase for a new check-out. You can reuse the existing schema in a geodatabase to form a template for a second check-out. If the data to be checked out is complex, reusing the schema in a geodatabase will afford some performance improvements. When creating the new check-out, if the geodatabase to be reused contains an older copy of the data, this will be deleted in preparation for receiving the latest copy of the data from the master geodatabase. If required, additional objects may be included to augment the check-out.

Finally, the check-out model does not support schema modifications to data that have been checked out. If the schema is altered in any way, either in the master or check-out

geodatabase, for example, by adding a field to a table or feature class, the check-out is rendered invalid and any attempts to check in the modified schema and data will fail. If a new table has been created in the check-out geodatabase, it will be ignored when the rest of the data is checked in.

Managing check-outs

Once a check-out has been created, any subsequent modifications to the properties of the check-out are made *asynchronously*. This means that changes made to the properties of a check-out in a master geodatabase are independent of the associated check-out in the check-out geodatabase and vice versa. For example, if a check-out is unregistered in the master geodatabase, this will not unregister the associated check-out in the check-out geodatabase. Similarly, if a check-out is renamed in the master geodatabase, that operation does not rename the check-out in the check-out geodatabase. The name of a check-out in a master geodatabase does not have to match the name of the check-out in the corresponding check-out geodatabase. If a check-out is renamed in the master but not the check-out geodatabase, the data can still be checked in.

Extracting data

Data may also be extracted from one geodatabase to another. Extracting data is similar to copying and pasting data between geodatabases, but it also supports the same data filters used when checking out data (selections, definition queries, and spatial extent). However, unlike checking out and checking in data, once data has been extracted, it cannot be merged back into the source geodatabase at a later time. For more information on extracting data, see the 'Migrating existing data into a geodatabase' chapter in this book.

Checking in data to a master geodatabase

Once a connection to the master geodatabase has been established, such as reconnecting a laptop to a network, data may be checked back in to the master geodatabase.

Two check-in models are supported; the *pull* model, in which the check-in is initiated from the master geodatabase, and the *push* model, in which the check-in is initiated from the check-out geodatabase. The choice of which approach to take is a matter of user preference. A data administrator checking in many check-outs to one master geodatabase might use the pull check-in model; open one connection to the master geodatabase and locate each check-out geodatabase in turn. Alternately, once an editor has completed editing the data in a check-out geodatabase, the push model would be the most convenient option for checking in the changes; checking out, editing, and checking in can all be accomplished within one ArcMap session without having to reopen a separate connection to the master geodatabase.

The master geodatabase does not record the location and type of the check-out geodatabase, so for a *pull check-in*, the check-out geodatabase has to be located as part of the check-in process. Changes are then pulled in from the check-out geodatabase to the master geodatabase.

For a *push check-in* from the check-out geodatabase to the master geodatabase, most of the connection information required to check in the data is already stored in the check-out geodatabase. All that is required is a username and password to complete the connection to the master geodatabase. Once this information has been supplied, the check-out geodatabase pushes the changes to the master geodatabase.

For both check-in models, any user who has write permissions to the data in the master geodatabase may check in the data.

However, only the user who created the check-out, or the Spatial Database Engine (SDE) user, can check data back in to a master geodatabase with reconcile and post as part of the same operation. Once the reconcile and post have successfully been completed, the check-out version in the master geodatabase is deleted—an operation that requires version owner or SDE permissions.

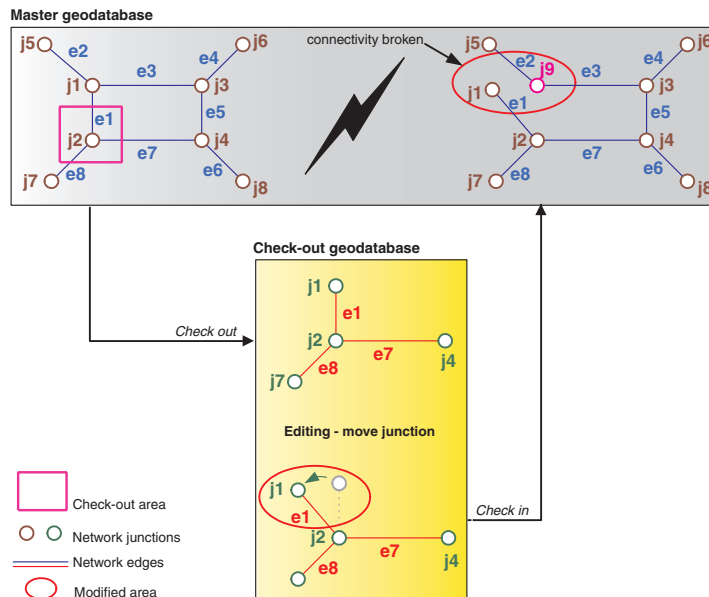
Delta databases

The edits made to data in a check-out geodatabase may optionally be exported to a delta database—a type of personal geodatabase that contains only changes exported from a check-out geodatabase. If connections to an ArcSDE server are made via a WAN with less bandwidth for data transfer or unreliable connections, this delta database may be transferred independently of the check-out/check-in process to a machine on the same LAN as the server. As delta databases are smaller than the original check-out geodatabase, data transfer times will be reduced.

The edits in the delta database may be checked in as a pull check-in from the master geodatabase. As with a check-in from a check-out geodatabase, when the check-in from the delta database succeeds, the check-out in the master geodatabase will be unregistered. Checking in changes in from a delta database does not automatically unregister the check-out in the associated check-out geodatabase; this must be done manually.

Checking in geometric networks

Checking in geometric networks involves transferring any changes to the network data to the master geodatabase and rebuilding connectivity in any part of the network that has been modified. Areas of the network that are unaffected by changes checked in are not rebuilt. If a junction has been moved in the check-out geodatabase, and that move results in two edges without a junction, to maintain the connectivity of the network in the master geodatabase a new junction will automatically be added at the old location.

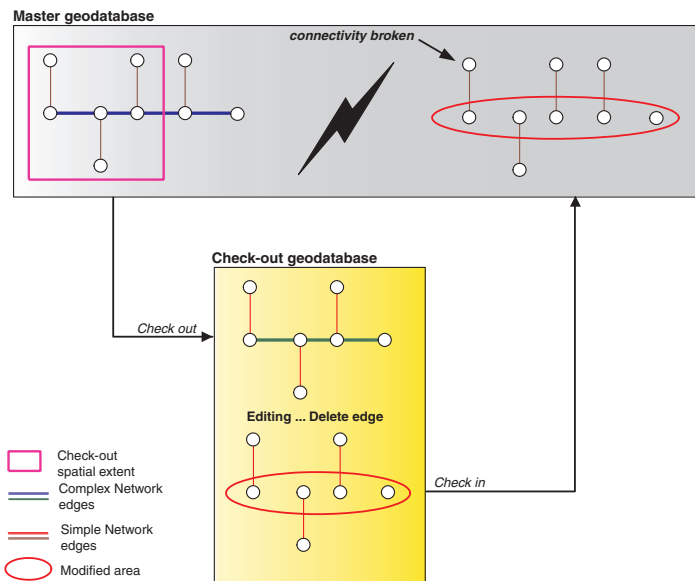


Checking in modified network junctions

In the example to the left, junction $j1$ is moved in the check-out geodatabase. As junction $j1$ was checked out as a result of edge $e1$ being checked out to maintain valid connectivity, it is no longer connected to its other edges, $e2$ and $e3$, in the check-out geodatabase. As these edges were not connected to junction $j1$ at the time it was moved, network connectivity is broken when the data is checked in to the master geodatabase.

Once the changes are checked in, the connectivity in the modified section of the network is rebuilt. This will result in a new junction, $j9$, being created automatically to maintain the correct pre-check-out network connectivity between edges $e2$ and $e3$.

In the next example, a collection of edges and junctions in a network is checked out. In the check-out geodatabase, a complex edge that connected numerous simple edges to the rest of the network is deleted. The data is then checked back in to the master geodatabase. In this case, connectivity is again broken, but editing the data in the check-out resulted in the same connectivity as if the edge had been deleted in the master geodatabase.



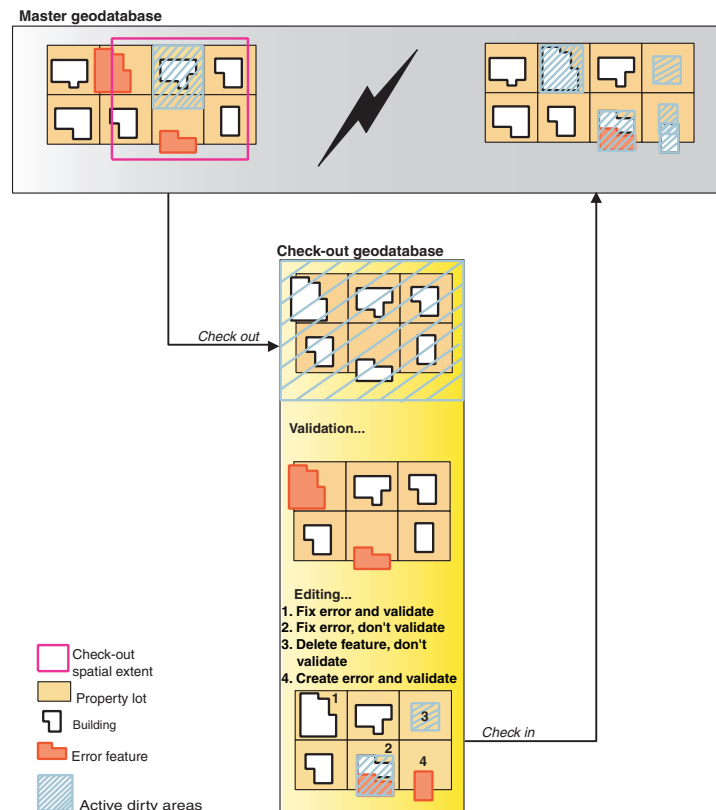
Checking in network features where one edge has been deleted

If the modifications in the check-out geodatabase result in network connectivity being broken in the master geodatabase, the responsibility for correcting this remains with the user or data administrator.

Checking in topologies

After topologies have been checked in, all the changes (inserts, updates, and deletes) are flagged as dirty areas that require revalidation.

If you have included reconcile and post as part of the check-in, the standard topology and version reconciliation rules will apply. For more information, see the 'Topology' chapter in this book.



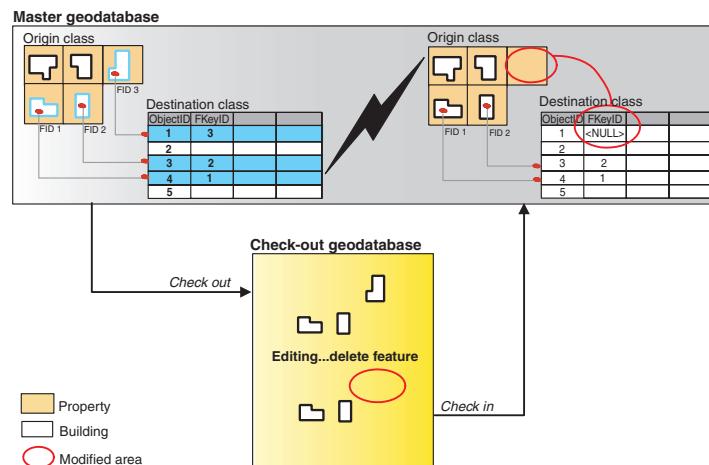
Checking in topological data

Checking in related data

When checking in data, there are some important considerations to remember if related data was not included in the check-out. The check-in process is a full geodatabase check-in, which means the process assumes that edits made in a check-out geodatabase were made with a geodatabase-aware editor, such as ArcMap, or any custom editing tool written using the geodatabase application programming interface (API). The check-in process also assumes that required related records were included in the check-out. For example, if a new feature was created in the check-out geodatabase, the foreign keys of a relationship involving a new feature are automatically updated on check-in.

However, if all the related objects were not included in the check-out, the check-in process will ensure that relationships pertaining to objects deleted in the check-out geodatabase are also deleted during check-in. This may result in the nulling of foreign keys in classes in the master geodatabase that were excluded from the check-out.

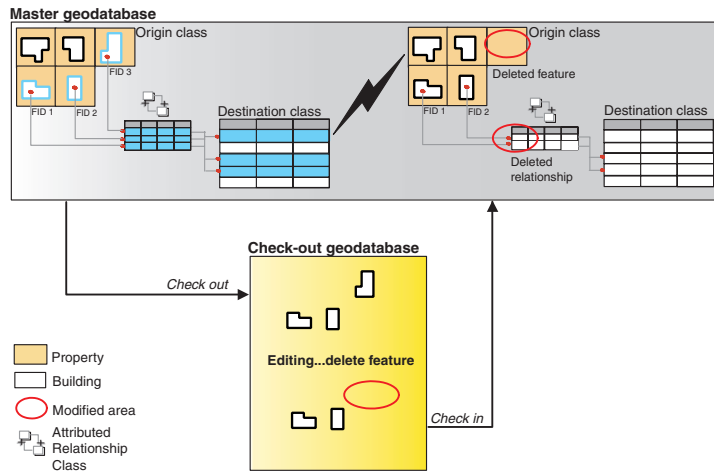
In this first example, some features in an origin class, buildings, were selected for checking out. The buildings are related, in a nonattributed relationship, to attribute records in a table that were excluded from the check-out. While the data was checked out, a building was deleted. On check-in, to void the relationship with the feature that was deleted, the corresponding entry in the foreign key field in the related destination class, the table, is set to NULL.



Checking in nonattributed relationships when related objects were excluded from the check-out

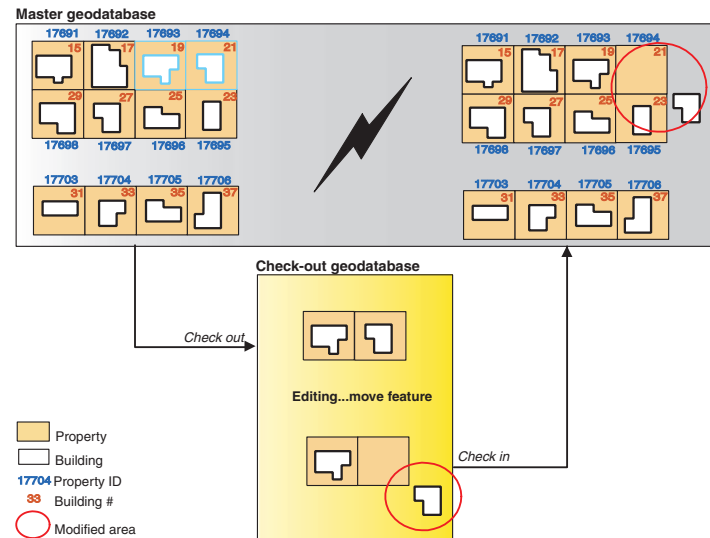
This check-in behavior may also result in the deletion of rows representing relationships in an attributed relationship class table. In the next example, the relationship between the origin feature class and the destination class table is attributed, which means the relationship itself has an associated table. Both the relationship and the destination class were excluded from the check-out. Edits made to the origin feature class resulted in one feature being deleted. On check-in, the row in the attributed relationship class table representing this feature's relationship to an object in the destination class is removed.

Only relationships are deleted on check-in; related objects themselves are never deleted.



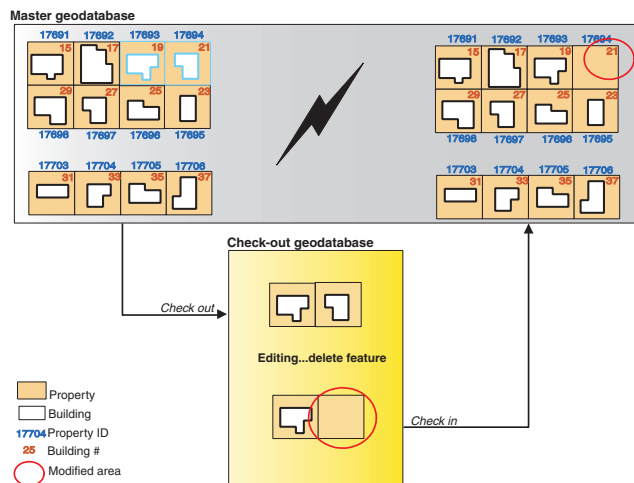
Managing attributed relationships on check-in when related objects are excluded in the check-out

In the next example, some features are selected for checking out without their related feature-linked annotation. The position of one feature is altered in the check-out geodatabase. On check-in, the annotation that relates to the modified feature in the master geodatabase is unaltered and remains in its original position.



Checking in modified features where the related annotation was excluded from the check-out

Similarly, not including related objects in the check-out and deleting a feature in the check-out geodatabase would result in orphaned annotation after the data has been checked back in.



Checking in deleted features where the related annotation was excluded from the check-out

The check-in process

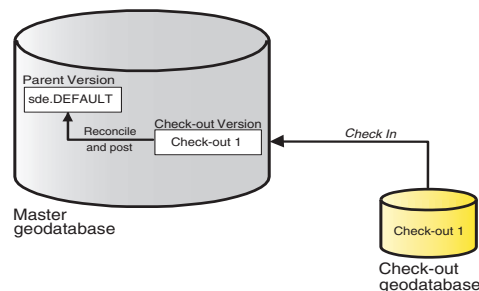
As with checking out data, the check-in process involves a number of automated steps. This process begins by establishing what changes have been made to the data in the check-out geodatabase; only these changes will be checked back in to the master geodatabase. If the check-out geodatabase is an ArcSDE geodatabase, a comparison is made between the edited check-out version and the static synchronization version to identify what is different. If the check-out geodatabase is a personal geodatabase, the changes are recorded in a separate DBMS table.

The changes are then transferred directly to the check-out version in the master geodatabase; there is no version

reconciliation at this point with the check-out version. If the check-out version has been modified since the data was checked out, these changes may be overwritten.

Once the data has been checked back in to the master geodatabase, all associated check-out information will be removed, such as the list of datasets checked out, from both the master and the check-out geodatabases. Any versions created in an ArcSDE check-out geodatabase will also be removed. However, the copy of data that was checked out is not removed from the check-out geodatabase. The responsibility for deleting any residual copies of the data once check-in has completed remains with the user or data administrator.

The check-in process may optionally involve an additional step of reconciling and posting the changes from the check-out version to its parent version in the master geodatabase.



The check-in with reconcile process

For further information on the reconcile and post processes, see the chapter 'Working with a versioned geodatabase'. If any conflicts are detected at this stage, they must be resolved using standard geodatabase version reconciliation tools. If no conflicts were detected, the check-out version in the master geodatabase is removed.

Checking out data from a geodatabase

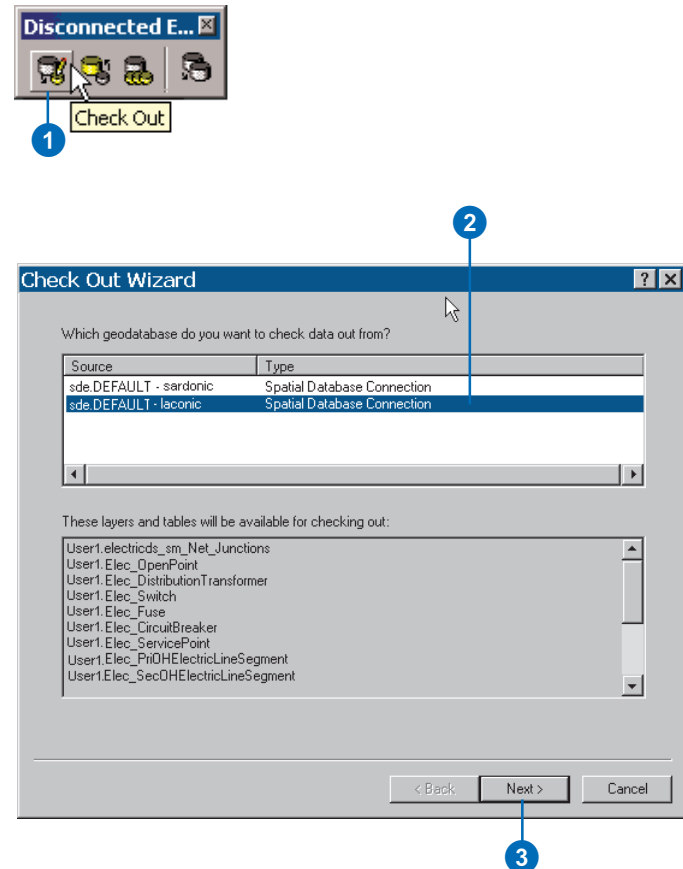
You can check out data from a geodatabase using the check-out wizard in ArcMap. The wizard supports a range of default and advanced check-out options.

The default behavior for each check-out will be:

- To include all the data, visible and invisible, in the active document for the chosen geodatabase.
- To maintain any selections and definition queries applied to that data.
- To restrict the spatial extent of the check-out to the current view extent of the document or the boundary of a selected graphic.
- To include any directly related objects.
- To expand the list of layers and tables to check out to include dependent datasets—for example, all feature classes in a geometric network will be included if just one feature class in the geometric network is selected. ►

Creating a check-out using the check-out wizard

1. Click the Check Out command on the Disconnected Editing toolbar to activate the check-out wizard in ArcMap.
2. You will be prompted to choose which ArcSDE geodatabase you want to work with if the current ArcMap document includes data from more than one geodatabase. You can only check out data from one ArcSDE geodatabase at a time.
3. Click Next. ►



- If there are several layers representing one feature class in the document, only the top layer in the Table of Contents (TOC) is used in the check-out.

If you just want to add new features or create a template for future check-outs, use the Schema Only option. This will create the tables for your data in the check-out geodatabase, but will not copy any data.

Each check-out name must be unique to the user creating it. User1 and user2 could create a check-out called MyCheckOut but neither could create multiple check-outs with that name. With each check-out a new version is created with the name of the check-out. The combination of user name and check-out name must be unique when creating versions.

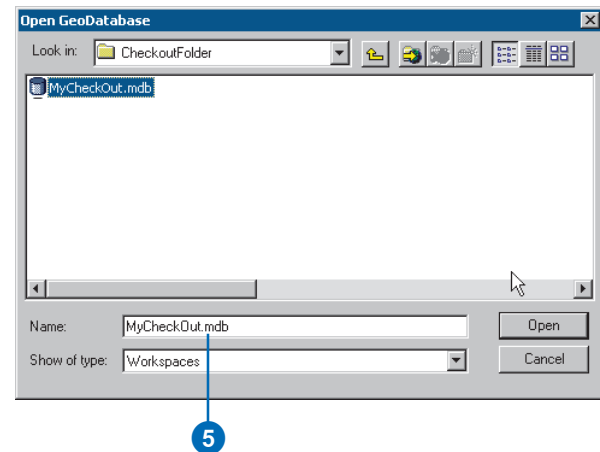
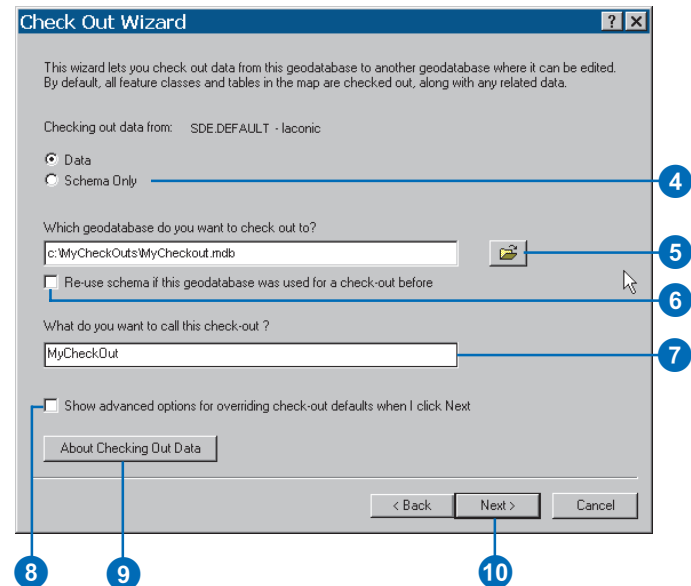
The reuse schema option allows you to reuse a geodatabase that contains the schema of the data you want to check out. This will reduce the amount of time required to check the data out.

Tip

Edit sessions

You may also check out data in an edit session. The check-out will represent the current state of the edit cache. Creating the check-out will not save pending edits to the master geodatabase.

4. Click Schema Only if a *schema-only check-out* is required. The first panel of options has one default setting: to check out data from the current ArcMap document.
5. Enter the name of, or click the Browse button to navigate to, a personal geodatabase or ArcSDE connection to check data out to. If the personal geodatabase does not already exist, it will be created.
6. Check Re-use schema if you wish to reuse schema in an existing geodatabase.
7. A default name for the new check-out is provided. The name of each check-out must be unique to the user creating the check-out.
8. Check Show advanced options if you wish to override the check-out defaults.
9. Click About Checking Out Data to learn more about checking out data.
10. Click Next. If you are not making any change to the default options, skip to step 17. ▶



Customizing a check-out

The advanced check-out options allow you to customize the check-out. The advanced panel will initially reflect the current check-out defaults and the expanded list of all the datasets that will be included in this check-out. Any data that is not versioned or for which you do not have read-write permissions will be automatically excluded from the list.

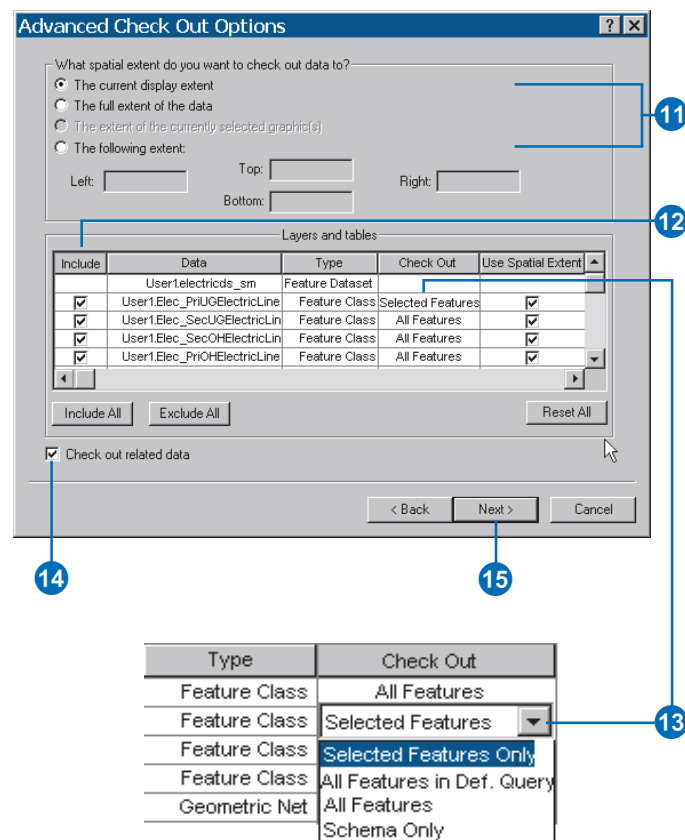
The extent of the check-out area may be determined by one of the following:

- The current view extent (the default)
- The full extent of the data
- The boundary of a currently selected graphic
- User-defined coordinates

If a schema-only check-out was selected previously, this top section will be grayed out.

The options in the grid determine how many records from each layer or table will be checked out. In addition to excluding layers and tables from the list of data to check out, you may also override the defaults for individual layers and tables. For example, if a selection set exists for a ►

11. Click the appropriate extent option to modify the spatial extent of the check-out.
12. Uncheck the check box associated with that layer/table to exclude individual layers/tables from the check-out.
13. Each entry in the Check Out column is a combo box of options. The options that you see will depend on how your ArcMap document is configured and will include All Features, Selected Features Only, All Features in Def. Query, and Schema Only. If a particular layer or table has no selection set or definition query defined, you will only see the All Features and Schema Only options listed. Select All features and uncheck the Use Spatial Extent box if you do not wish to apply any data filters.
14. Uncheck the related data check box if you do not wish to check out any related data.
15. Click Next. ►



layer, you may choose to disregard that selection for this check-out. By default, all layers will be filtered by geometry and/or selection and/or definition query. For tables, if no other filters have been applied (for example, a selection), the default filter is Schema Only; only the schema for the table will be checked out.

If you wish to exclude a geometric network or topology from a check-out, you must exclude all the participating layers; including just one layer will result in the whole network or topology being checked out.

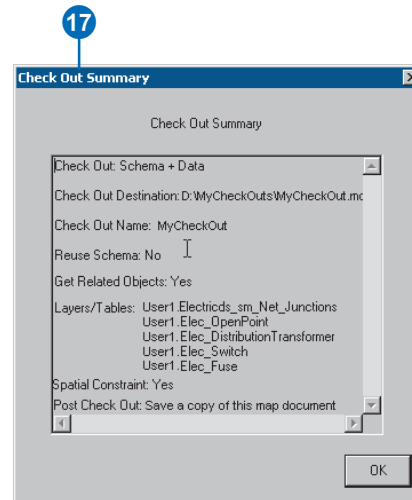
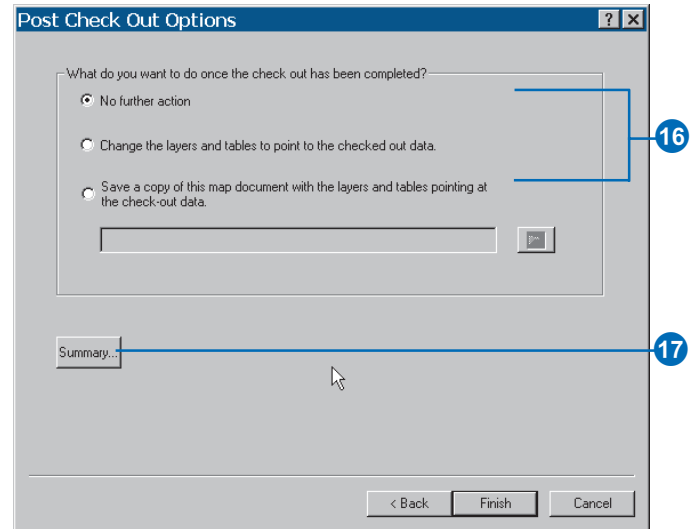
The final panel includes some post check-out options:

- No further action (the default)—the current document will not be modified and no new documents created.
- The current ArcMap document will be modified to point to the data in the check-out geodatabase, preserving all symbology.
- A new ArcMap document, referencing the data in the check-out geodatabase, again with symbology preserved, will be created.

The summary report is available for reference.

16. Select an appropriate post check-out option.

17. Click Summary to review the parameters for the current check-out. ►



Tip

Related objects

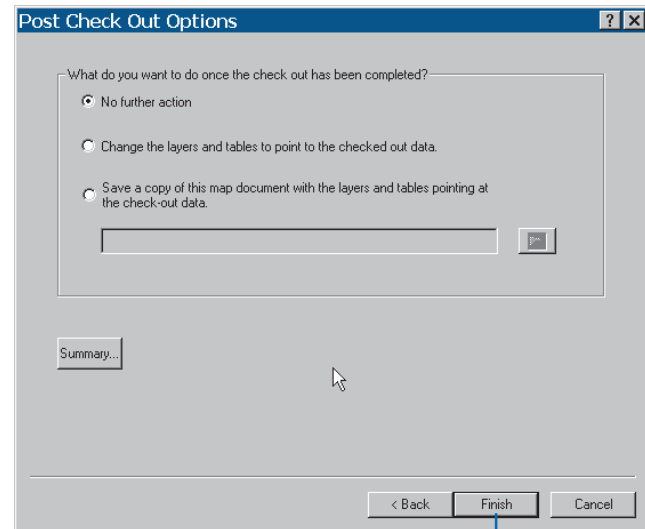
It is always recommended that you include related objects in the check-out. This will ensure the integrity of the data throughout the check-out/check-in process. If you choose not to include related objects, you may introduce some inconsistencies in the data that may result in unexpected behavior.

Tip

Saving new map documents

If you opt to save a new map document with your check-out, there are two settings you can use to make the new document more portable. First, modify your existing document to store relative pathnames, and second, under the map document page setup (click the File menu, then click Page Setup), uncheck the Same as Printer option. Unchecking this option prevents information about the local default printer from being saved with the new document.

18. Click Finish to start checking out the data. The status of the check-out operation will be monitored in a progress dialog box.



Checking in data to a geodatabase

You can check in data from a geodatabase using the check-in wizard. The wizard supports both the *pull* and *push* models for checking in data. This utility is available in both ArcCatalog and ArcMap.

You may also check in data from a delta database: delta databases contain changes exported from a check-out geodatabase.

Tip

Reconcile following check-in

If you choose to reconcile the changes with the parent version once the data has been checked in, any conflicts encountered must be resolved using standard geodatabase version reconciliation tools. If no conflicts are encountered, the changes will be posted to the parent version.

Checking in data—pull operation

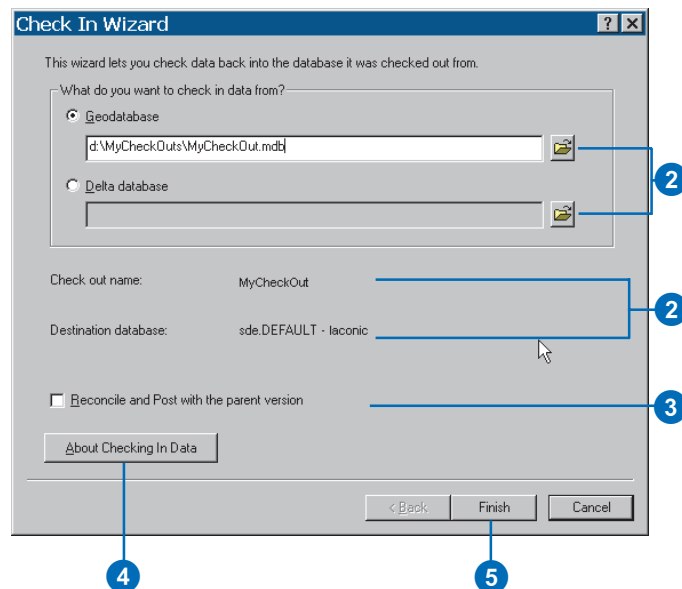
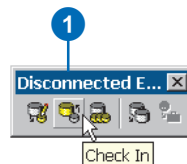
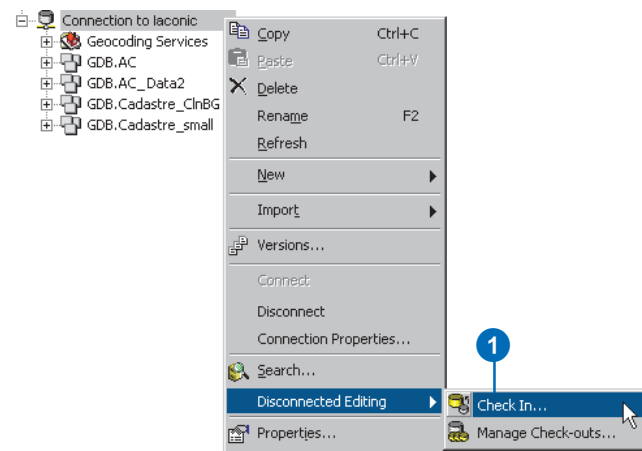
1. Browse for the master geodatabase in ArcCatalog to check data back in and right-click the connection item. Point to Disconnected Editing, then click Check In to activate the wizard.

In ArcMap, click the Check In command on the Disconnected Editing toolbar. If more than one master geodatabase is present, you will be prompted to choose one.

2. Enter the name of, or click the Browse button to navigate to, a check-out geodatabase or delta database from which you want to check in the changes.

The name of the check-out and the master geodatabase you are checking in to will be listed for reference.

3. Check the Reconcile and Post box to reconcile and post the changes to the parent version once the check-in is complete.
4. Click About Checking In Data to learn more about checking in data.
5. Click Finish. The check-in status will be monitored in a progress dialog box.



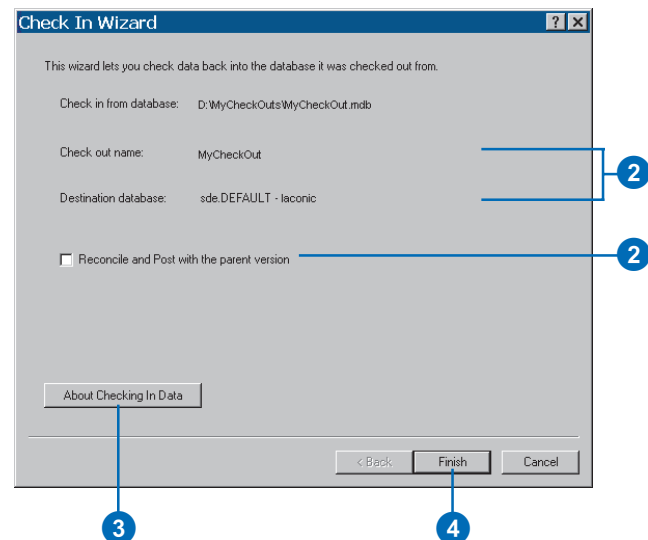
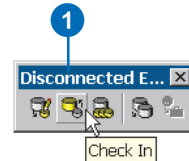
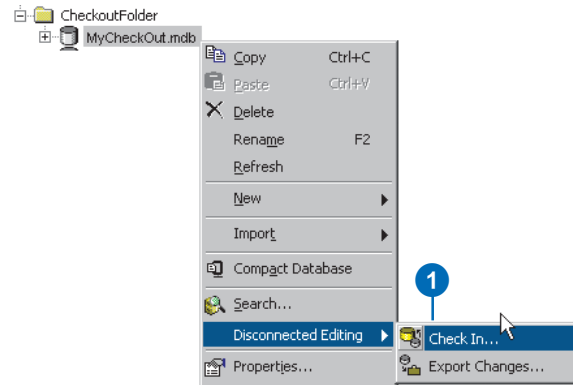
Checking in data—push operation

1. Select your check-out geodatabase in ArcCatalog and right-click the database object. Point to **Disconnected Editing**, then click **Check In**.

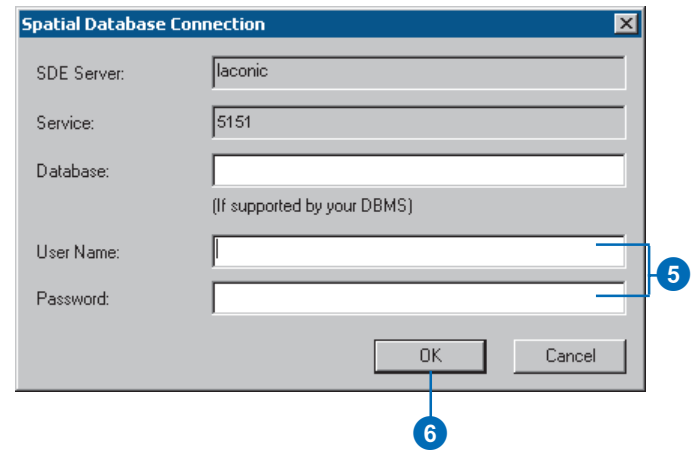
In ArcMap, click the **Check In** command on the **Disconnected Editing** toolbar. If more than one check-out geodatabase is present in your document, you will be prompted to choose one.

The name of the check-out and the master geodatabase you are checking in to will be listed for reference.

2. Check the **Reconcile and Post** box if you want to reconcile and post the changes to the parent version once the check-in is complete.
3. Click **About Checking In Data** to learn more about checking in data.
4. Click **Finish**.



5. Supply the necessary information to complete the connection to the master geodatabase in the SDE connection dialog box.
6. Click OK. The status of the check-in will be monitored in a progress dialog box.



Managing check-outs

You can manage the check-outs in a master geodatabase using the check-out manager. With this utility, you can rename, refresh, and review the properties of each check-out and unregister as required. This utility is available in both ArcCatalog and ArcMap.

The general geodatabase property dialog box provides the same management utilities as the check-out manager for a check-out geodatabase. From this dialog box, the check-out may be renamed, its properties may be reviewed, or the check-out may be unregistered.

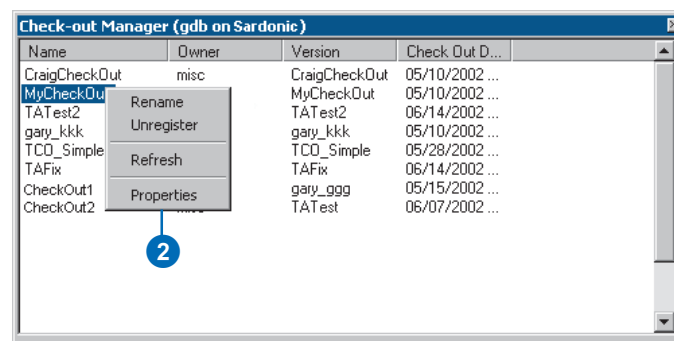
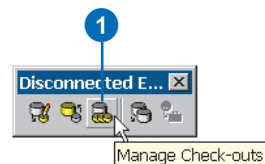
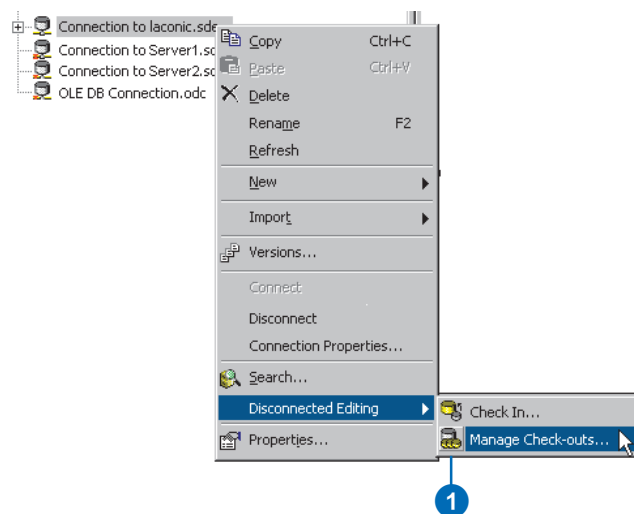
Managing check-outs in a master geodatabase

1. Right-click your database connection in ArcCatalog. Point to Disconnected Editing and click Manage Check-outs.

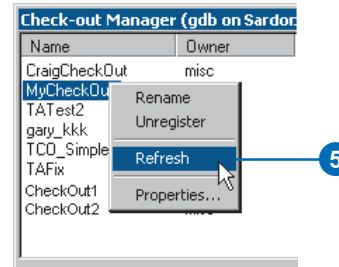
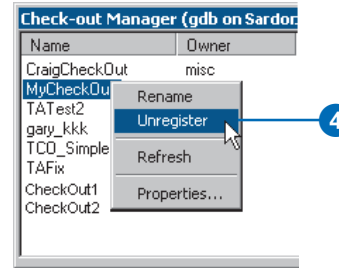
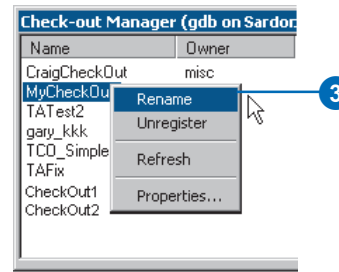
In ArcMap, click the Manage Check-outs command in the Disconnected Editing toolbar.

If more than one master geodatabase is present in your document, you will be prompted to choose one.

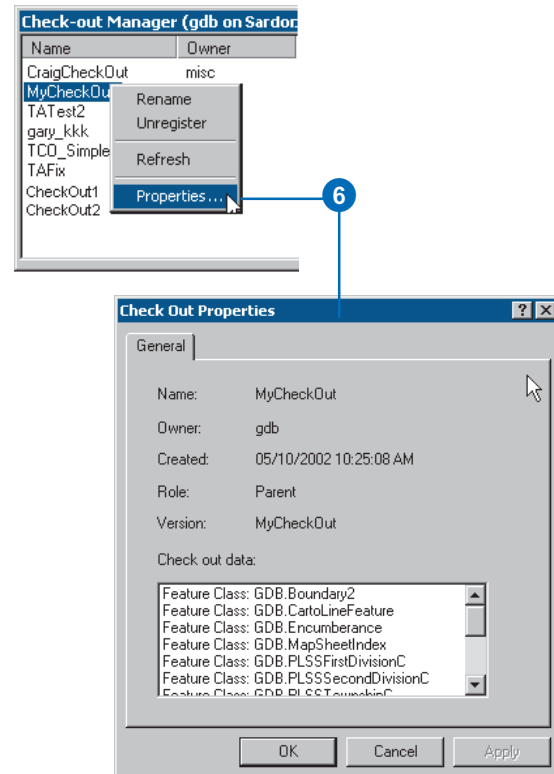
2. Select a check-out and right-click to review the options. ►



3. To rename a check-out, right-click the check-out you want to rename and click Rename. Enter a new name and press Enter. This will automatically rename the associated check-out version at the same time.
4. To unregister a check-out, right-click the check-out you want to unregister and click Unregister. The version associated with this check-out will also be removed at the same time.
5. To refresh a check-out, right-click the check-out you want to refresh and click Refresh. The latest state of the properties will be displayed. ►

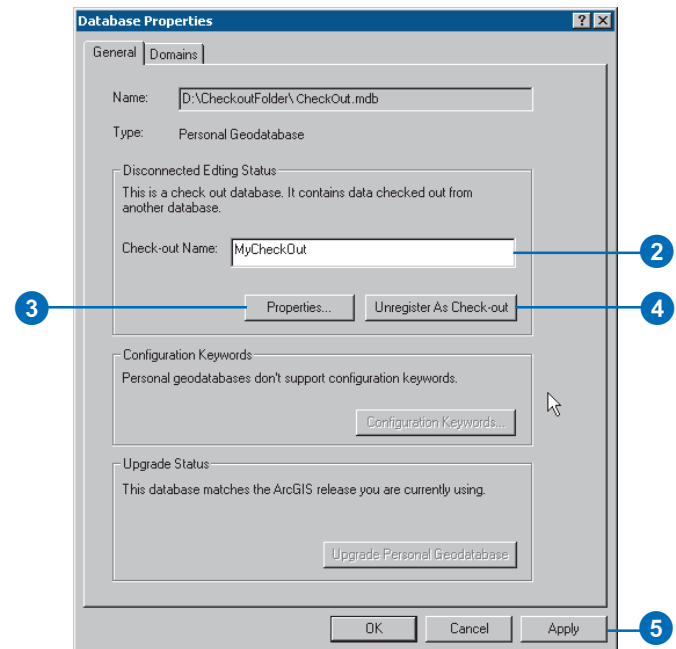
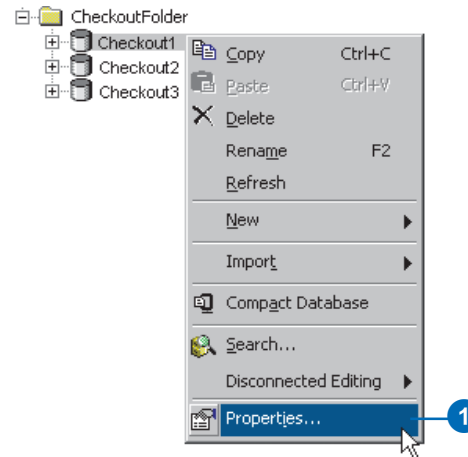


6. Right-click the check-out and click Properties to view the properties of a check-out. The properties include information on when the check-out was created, who created it, and the associated check-out version. A full list of the data checked out is also included.



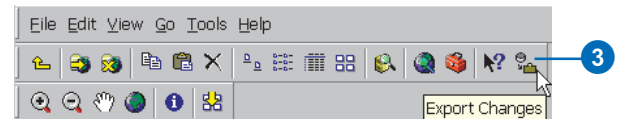
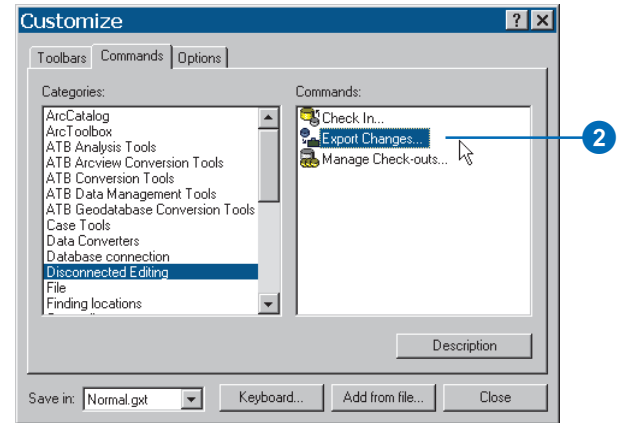
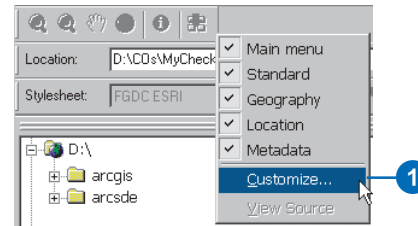
Managing a check-out in a check-out geodatabase

1. Right-click your check-out geodatabase in ArcCatalog and click Properties.
2. Enter a new name in the name field to rename a check-out.
3. Click Properties to review the properties of a check-out.
4. Click Unregister As Check-out to unregister a check-out in a check-out geodatabase. This will remove all the information about the check-out in the geodatabase; it will not delete the data that was checked out. If the check-out geodatabase is an ArcSDE geodatabase, the check-out version in that geodatabase will be removed.
5. Click Apply.



Exporting changes from a check-out geodatabase to a delta database

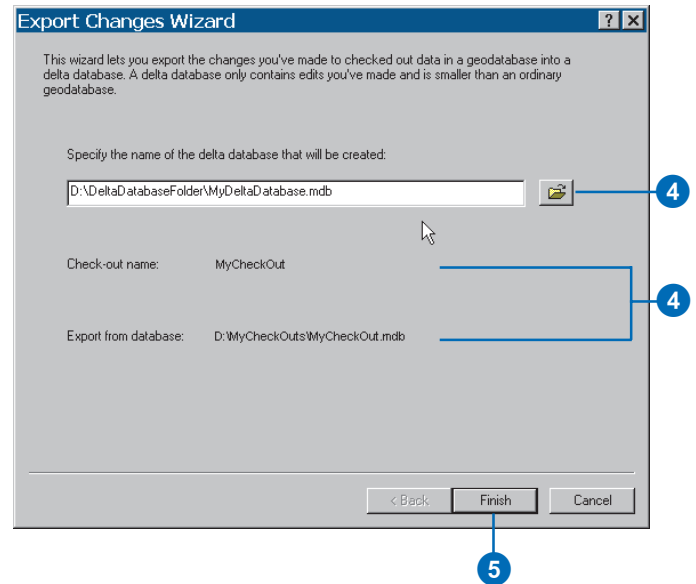
1. Right-click on one of the toolbars in ArcCatalog or ArcMap to bring up the Customize dialog box.
2. Click the Commands tab to see a list of all the commands. Navigate to the Disconnected Editing category and select the Export Changes command.
3. Drag the command onto a toolbar. To enable this command, select a check-out geodatabase. Once enabled, click to bring up the Export Changes dialog box. ►



4. Either navigate to or enter the path to a directory where the delta database will be created. Specify the name of the delta database you want to create.

The name of the check-out and the name of the check-out geodatabase from which you are exporting the changes are included for reference.

5. Click Finish.



Building geodatabases with CASE tools

13

IN THIS CHAPTER

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

The geodatabase brings the physical representations 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 chapter discusses how you can use Microsoft Visio to construct your UML and how you can use CASE tools in ArcCatalog to generate the schema for your geodatabase.

CASE tool functionality does not apply to ArcView licenses.

What are CASE tools?

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—have been discussed thus far in this book. This chapter 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

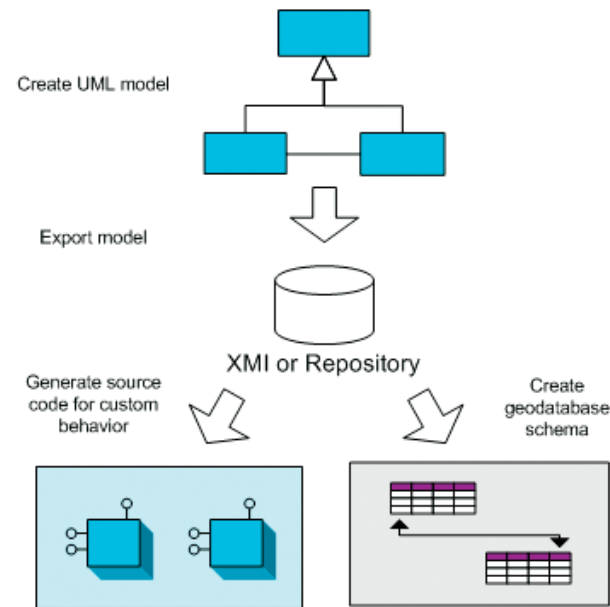
The general strategy for using UML and CASE tools to design and create your geodatabase involves using UML to define all of the schema for the geodatabase, generating that schema, then populating the schema with data. The steps for accomplishing this are outlined below:

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 throughout this book.



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 the organizational relationship between elements 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 <i>Integer fields: number of digits</i> <i>Double fields: total number of digits</i>
Scale	Integer value <i>Number of decimal places in single and double fields</i>
Length	Integer value <i>Width of character fields</i>
AllowNulls	True/False
Alias	String <i>The alias name for the field</i>
Feature class/Object class:	
Geometry type	esriGeometryPoint esriGeometryPolygon esriGeometryPolyline esriGeometryMultipoint <i>Valid only for feature classes</i>
Ancillary role	esriNCARNone esriNCARSourceSink <i>Valid only for junction feature classes</i>

ConfigKeyword	String value
HasM	True/False <i>Valid only for feature classes</i>
HasZ	True/False <i>Valid only for feature classes</i>
CLSID	GUID in registry format <i>If specified, the Schema Wizard will look for a Component Object Model (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.</i>
Alias	String <i>The alias name for the field</i>

Geometric network:

GNConfigKeyword	String
-----------------	--------

Relationship class:

Notification	esriRelNotificationBackward esriRelNotificationBoth esriRelNotificationForward esriRelNotificationNone
IsAttributed	True/False

OriginClass	Name of the origin
classOriginPrimaryKey	Name of the primary key field of the origin class
OriginForeignKey	Name of the foreign key field of the origin class <i>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.</i>
DestinationPrimaryKey	Name of the primary key field of the destination class <i>Valid for M–N/attributed relationship classes only.</i>
DestinationForeignKey	Name of the foreign key field of the destination class <i>Valid for M–N/attributed relationship classes only. This field lives in the auxiliary <<RelationshipClass>> class.</i>

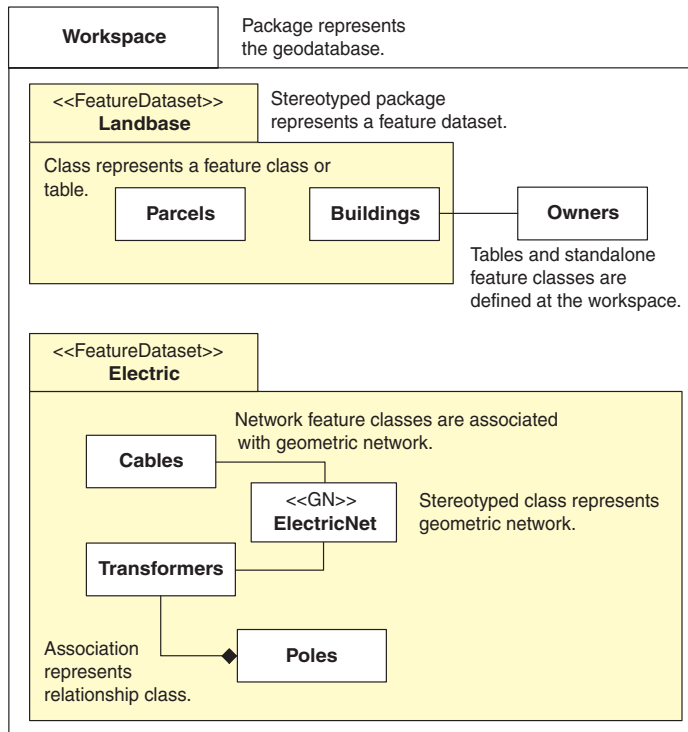
Domain:

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.



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, 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.

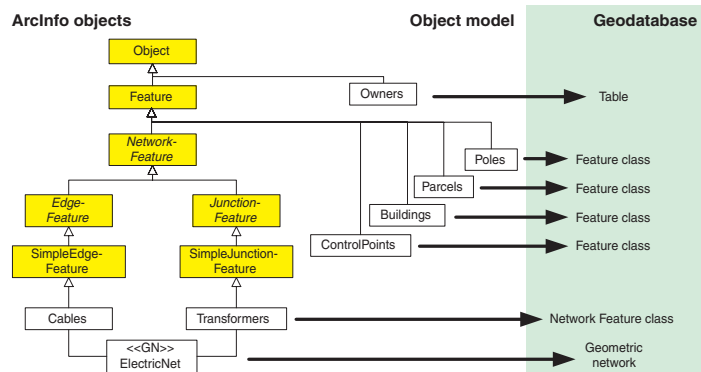
When generating schema, all feature classes the wizard creates use the coordinate system of the target feature dataset. Object classes (tables) are created at the workspace level.

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, while 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 the chapter ‘Geometric networks’ in this book.

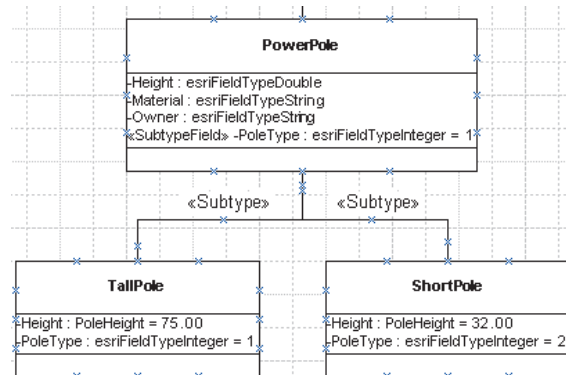


Tables are created for objects that inherit from the Object class, feature classes are created for objects that inherit from the Feature class, and network feature classes are created for objects that inherit from the Network feature class.

Subtypes and domains

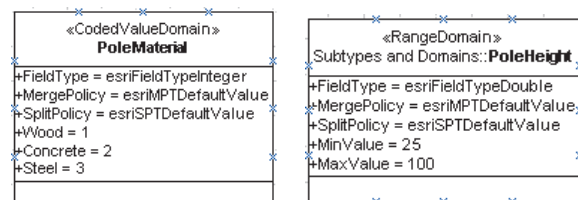
Attribute domains are modeled in UML as special classes. When the CASE tool generates the schema from the UML model, these domain classes are stored in the geodatabase as domains. Coded value and range domains, along with their valid values and split and merge policies, are all modeled in this way.

If your tables or feature classes require subtypes, these can also be modeled in UML and be 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.

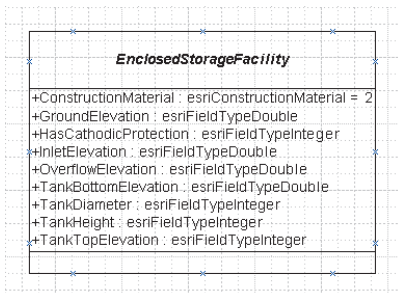


Subtypes are modeled in UML as classes related to the parent class through an association stereotyped as Subtype. You can specify domains and default values for the fields for that subtype.

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.



Attribute domains are modeled in UML as special classes.



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.

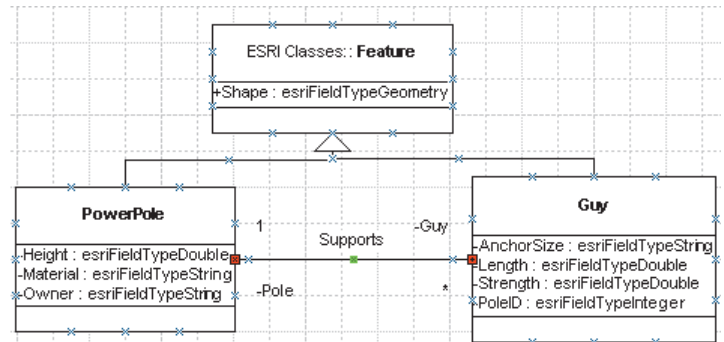
Relationships

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.

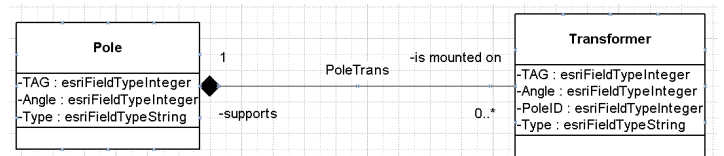
Relationship rules apply to subtypes in feature classes and tables. In UML, these relationship rules are modeled as associations between subtypes of the classes participating in the relationship class.



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 the chapter 'Defining relationship classes' in this book.

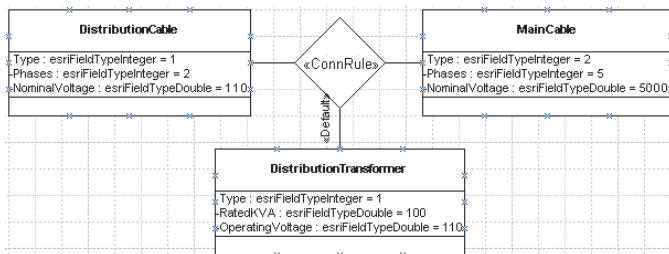


Composite relationship classes are modeled in UML as aggregations between classes.

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*.



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

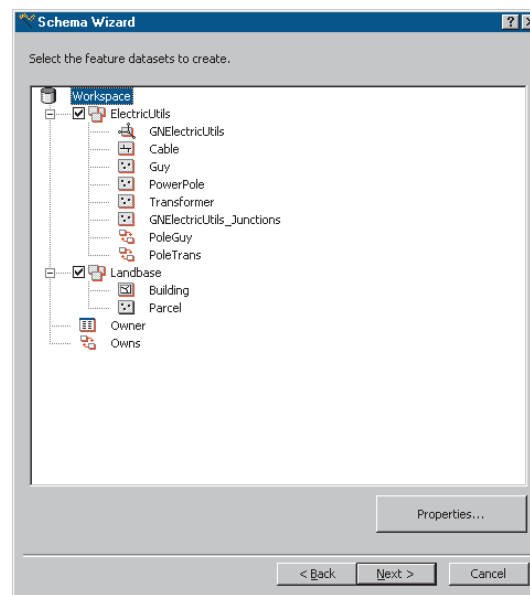
The CASE tools expect UML models to be created following a set of modeling rules. For example, a network feature class must be associated to a geometric network. The semantics checker can be used to verify that a model stored in the Microsoft Repository or XMI has been correctly defined. It will produce 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.

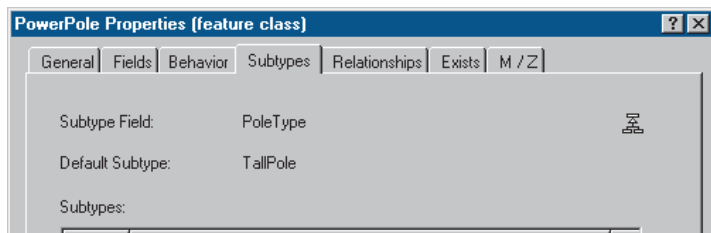
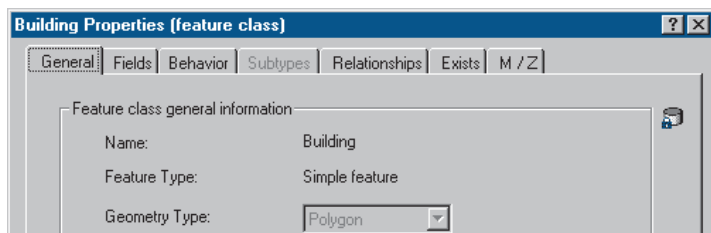
You can use the Schema Wizard to modify an existing geodatabase, even if data has already been loaded. To do so, select the target geodatabase and run the Schema Wizard. When the wizard starts, some elements in the model are searched for and matched to objects in the database—feature classes and tables, 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.



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.



Those properties that cannot be changed when reapplying the model will have a locked database icon on the right side of their properties dialog box such as the top image above, while those that can be modified will have a model icon such as the bottom image above.

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:

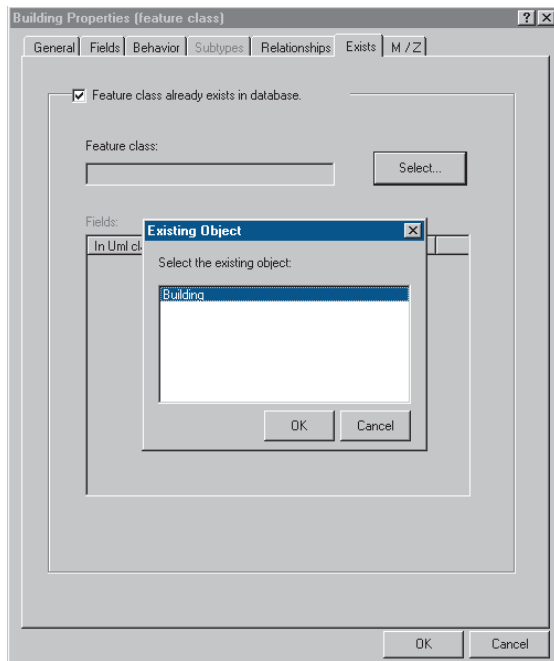
Feature datasets: The Schema Wizard uses the name of the UML package representing the feature dataset to search for an existing feature dataset in the target database. The spatial reference and spatial domain properties become read-only.

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.

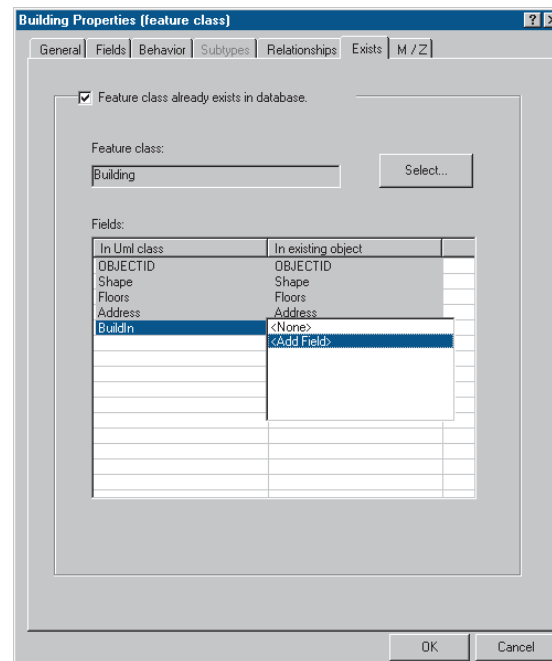
When the Schema Wizard is run, it updates the model name of both existing and new feature classes and fields. This ensures the matching will occur automatically the next time the model is applied.



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).



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.

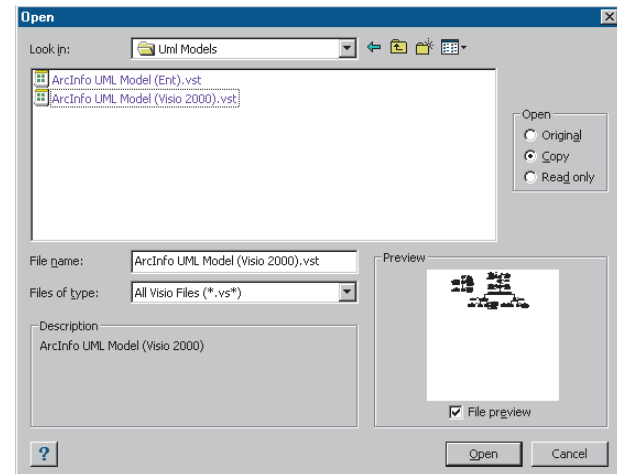
This chapter discusses how to create models using Microsoft Visio and how you can use CASE tools in ArcCatalog to generate the schema for your geodatabase. You will learn how to:

- 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.

Designing the object model in Microsoft Visio

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.

1. Start Visio.
2. Click File and click Open.
3. Browse to C:\arcgis\arcexe83\CaseTools\UML Models, which is the default installation path, and double-click the ArcInfo UML Model template file.



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. As 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.

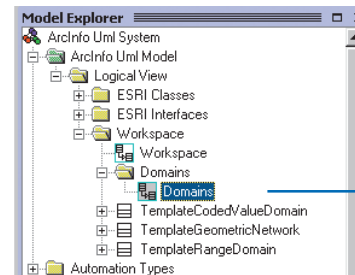
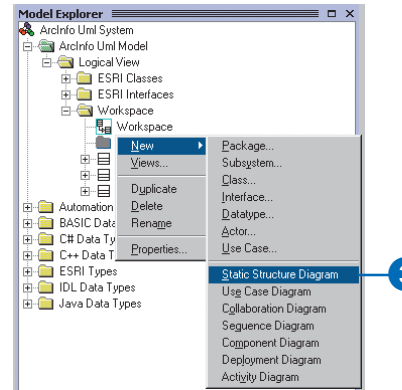
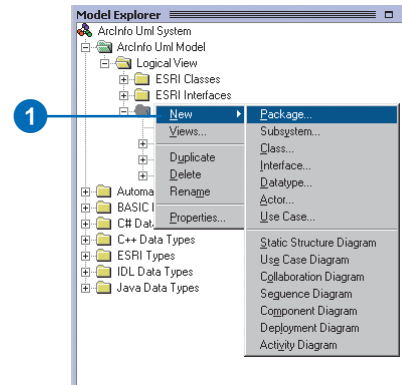
Before you begin, create a new diagram using the ArcInfo UML Model Visio templates. These templates are located under your ArcGIS installation in the casetools\UML Models directory.

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 chapter.

1. Right-click Workspace in the Model Explorer tree, 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.



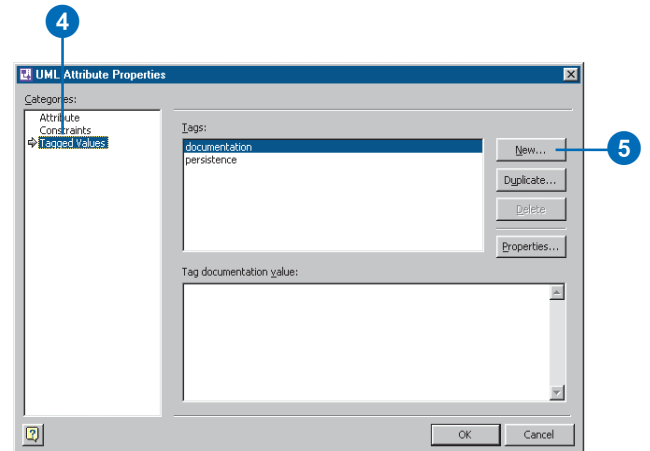
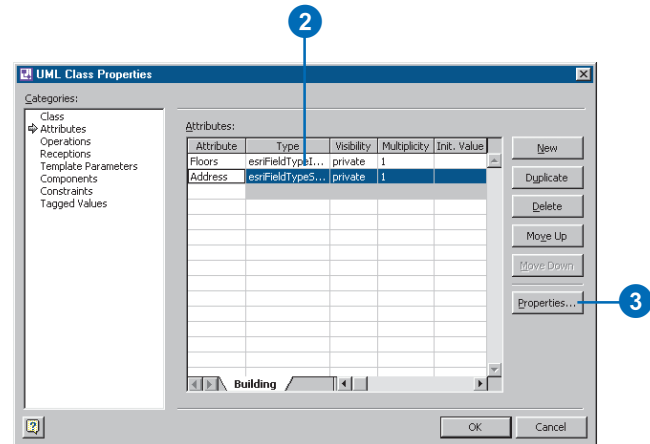
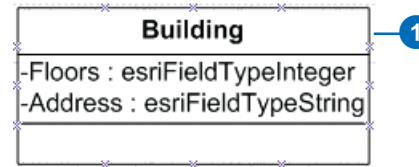
A new class diagram is created in the Model Explorer.

Setting tagged values

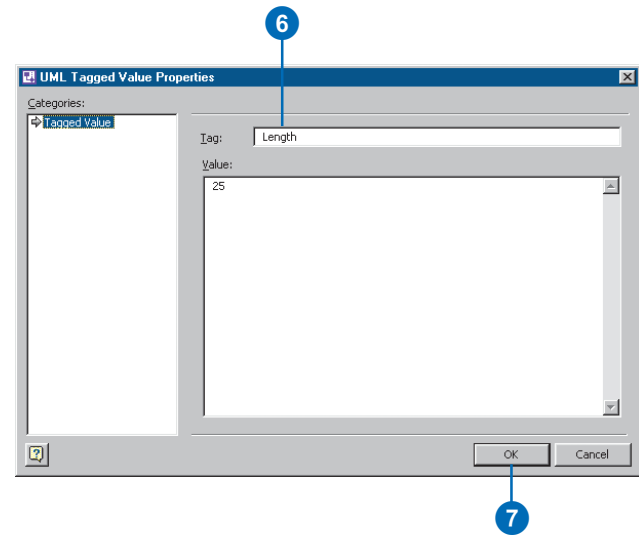
Tagged values are used to set additional properties of UML elements. For example, you can set the length (in characters) of a string field by using a tagged value.

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

1. Double-click the UML class in the Visio diagram.
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. ►



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



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.

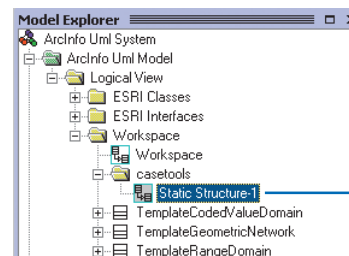
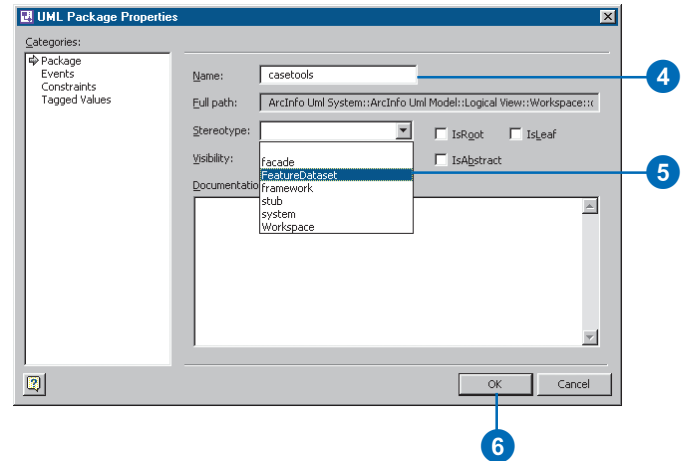
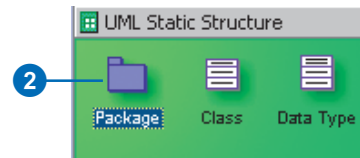
Tip

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. Double-click the Workspace diagram in the Model Explorer to open it.
2. Click Package in the UML Static Structure stencil 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.



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.

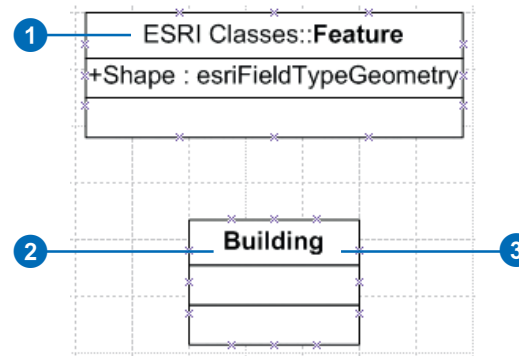
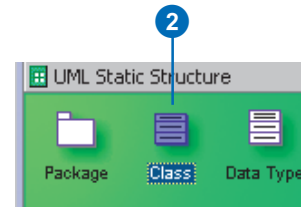
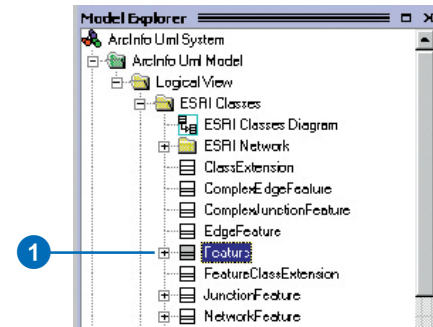
You can add fields to a feature class by adding attributes to the UML class.

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. Click the parent class in the Model Explorer tree under ESRI Classes and drag and drop it onto the diagram.
2. Click Class in the UML Static Structure stencil and drag and drop a new UML class onto the diagram.
3. Double-click the new class in the diagram. ►



Tip

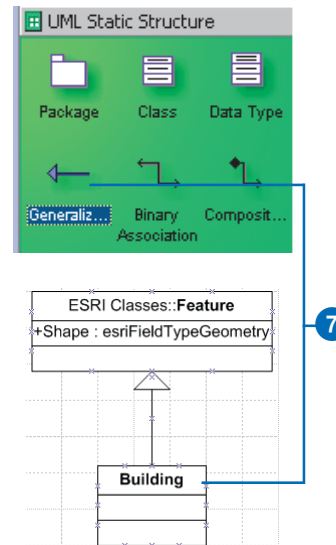
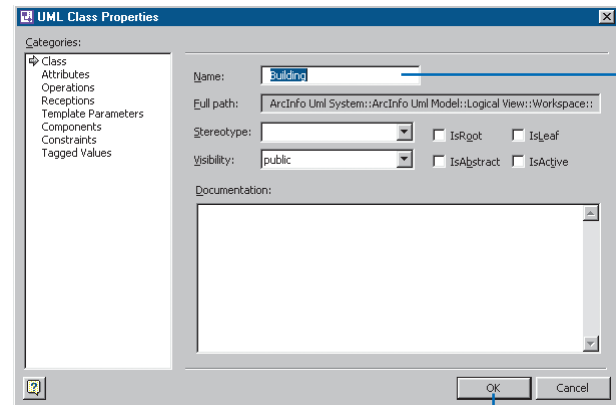
Abstract classes

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

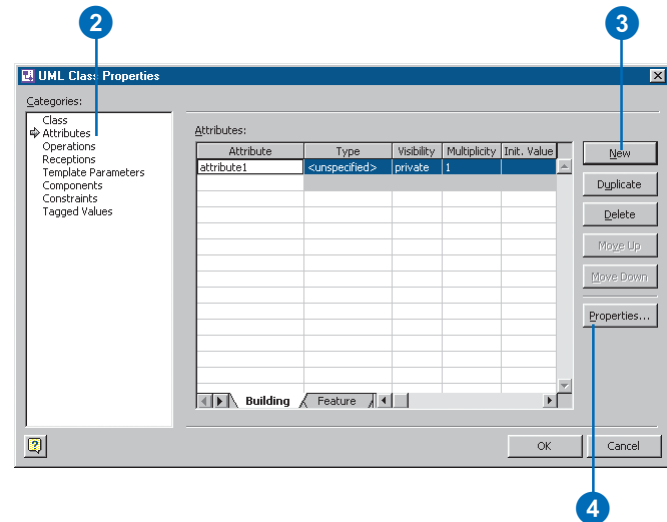
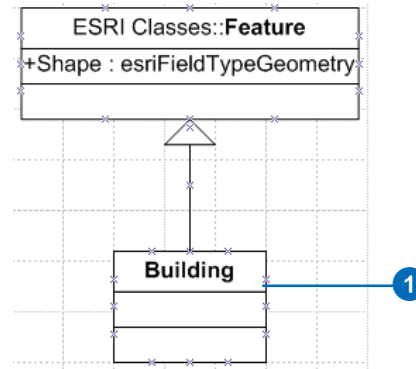
Tagged values allow you to fully specify UML elements in the model. See ‘Setting tagged values’ in this chapter.

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



Adding fields to a feature class

1. Double-click the class in the diagram.
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. ►



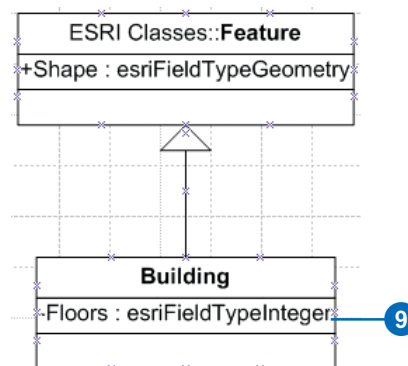
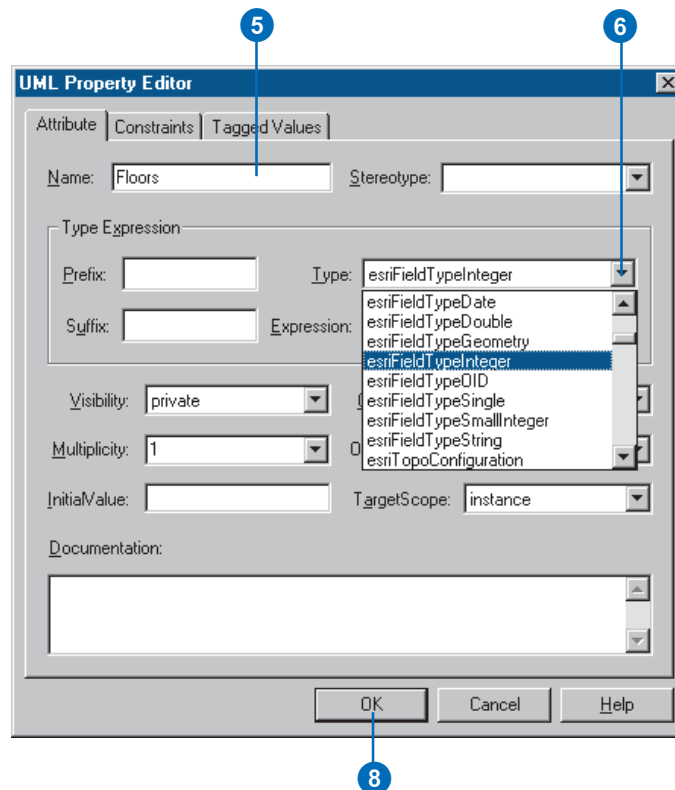
Tip

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. Follow steps 2 through 6 of 'Setting tagged values' in this chapter to set tagged values; 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.



Creating relationship classes

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

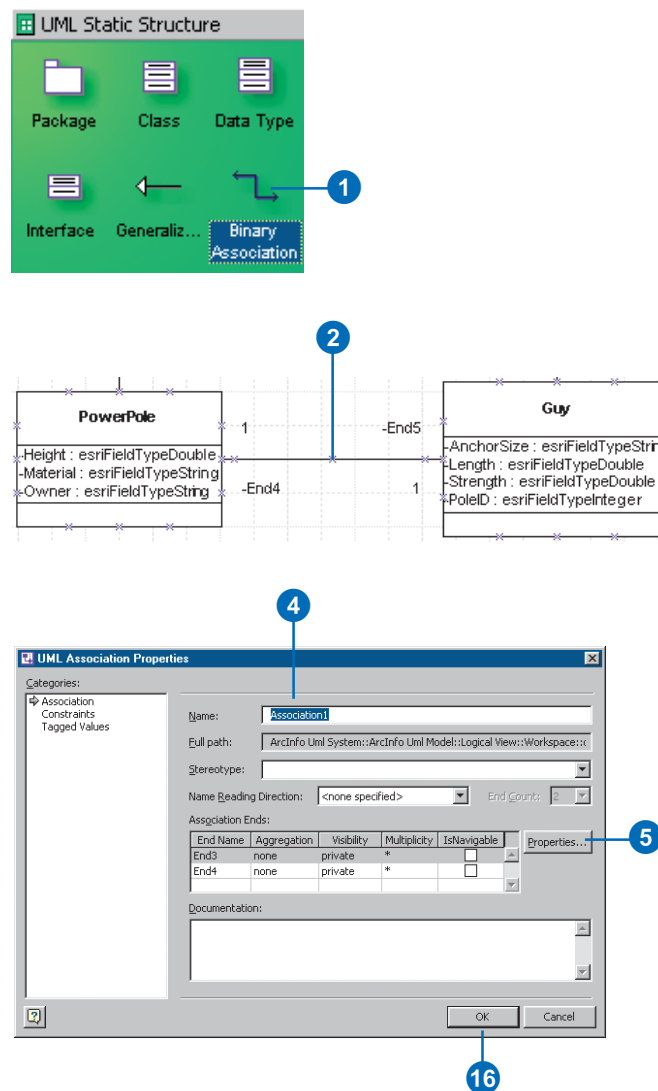
Relationship classes can be created only among leaf classes. Primary and foreign key fields must be present in the classes.

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. Click Binary Association in the UML Static Structure stencil 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. ▶



- Notification: one of the values of the esriRelNotification enumeration

Example:

Notification=esriRelNotificationBoth

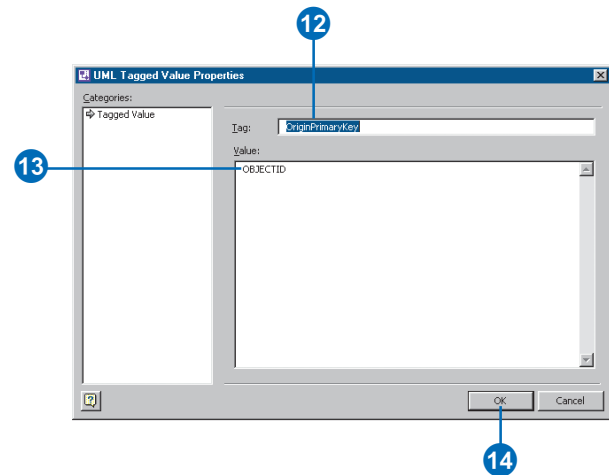
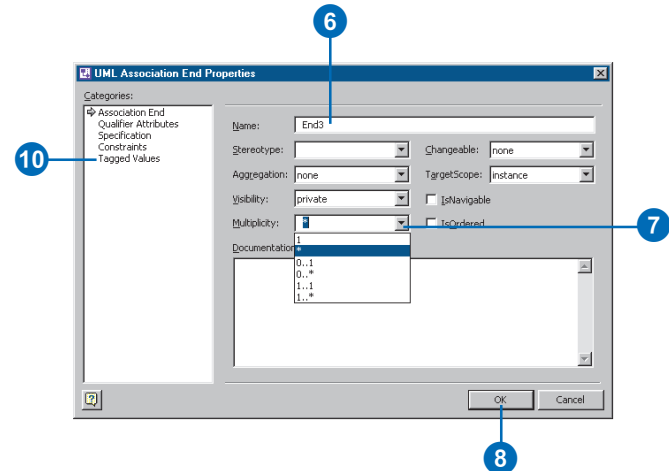
The cardinality of the relationship class is derived from the multiplicity of both association ends. Because the cardinality of a relationship class can be only 1–1, 1–M, or M–N, the only valid multiplicity values are 1 and * (many). The name of an association end becomes a relationship class path label.

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. ►



- **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`

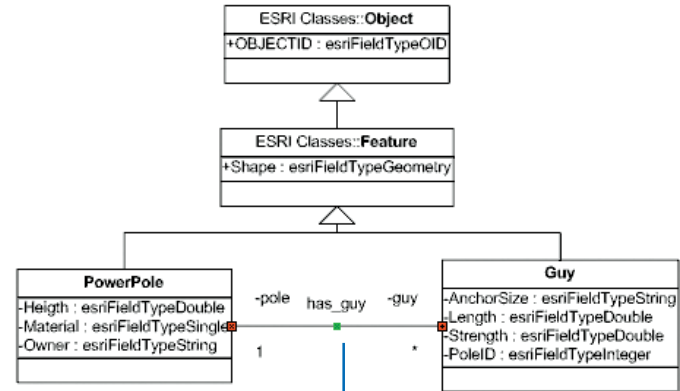
Tip

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.



The relationship class is shown as an association between classes.

Tip

Composite relationships

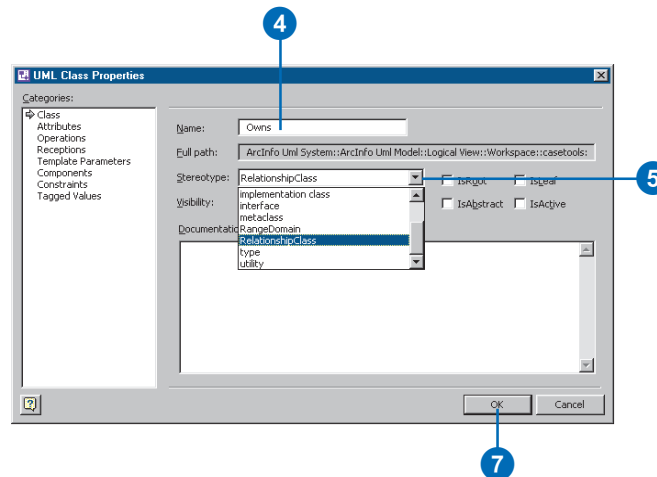
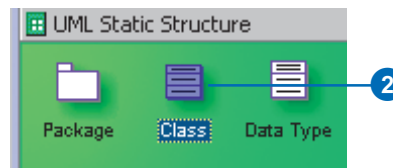
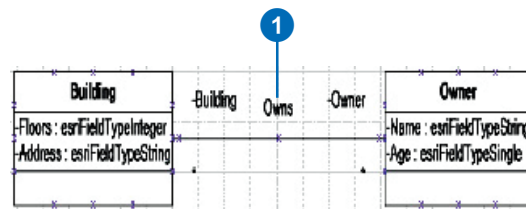
In a composite relationship, one of the objects controls the lifetime of the associated objects. You can create a composite relationship by using a Composition instead of a binary association.



Creating attributed relationship classes

1. Create the UML association representing the relationship class by following steps 1 through 16 of 'Creating nonattributed relationship classes', described in the previous task.
2. Click Class in the UML Static Structure stencil 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 chapter to add fields to the relationship class.
7. Click OK.

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



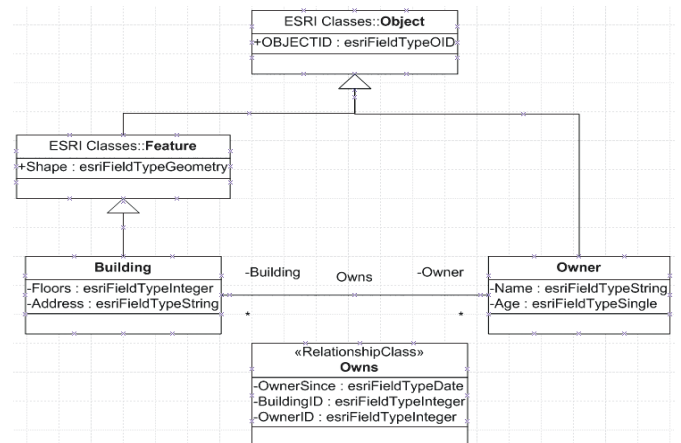
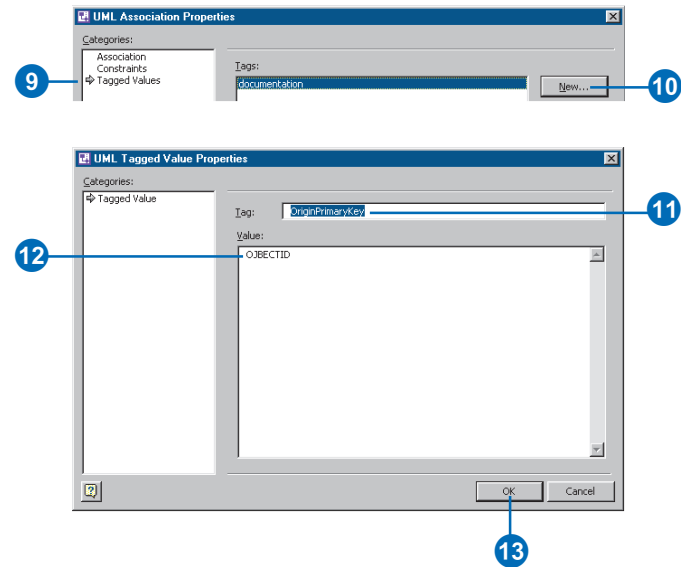
Tip

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 in the diagram.
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 to close the UML Tagged Value Properties dialog box.
14. Repeat steps 10 through 13 for the destination primary key, the origin and destination foreign keys, and the notification tags.
15. Click OK to close the UML Association Properties dialog box.

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



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.

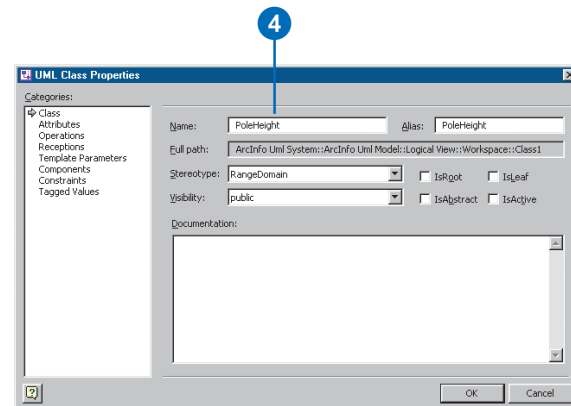
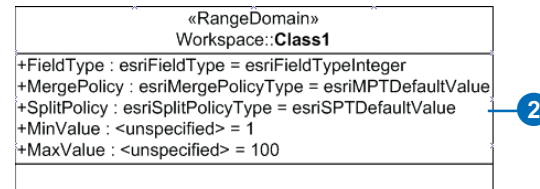
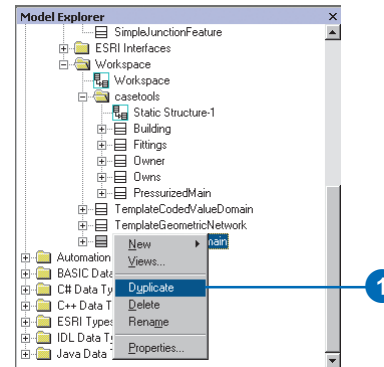
In addition to the standard attributes of domains (FieldType, MergePolicy, and SplitPolicy), range domains have MinValue and MaxValue. ►

Creating a range domain

1. Right-click TemplateRangeDomain under the Workspace package in the Model Explorer and click Duplicate.

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

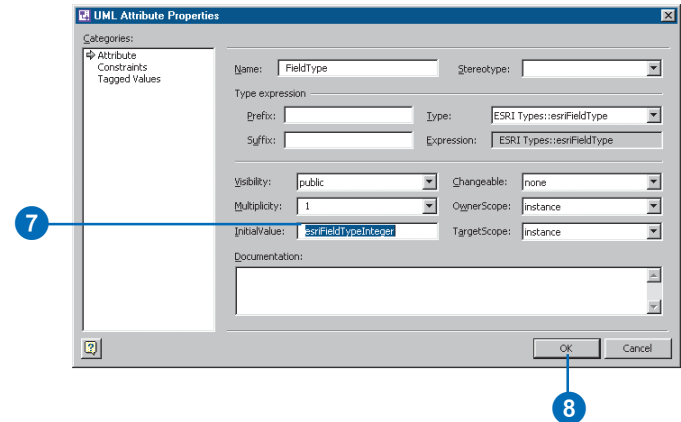
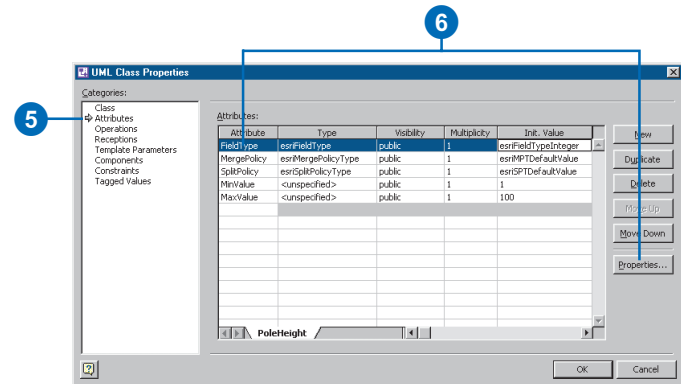
2. Drag and drop it on the diagram.
3. Double-click the new class.
4. Type the name of the domain. ►



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. Input 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. ►



15. Click the MinValue attribute and click Properties.

16. Type the minimum value for the range domain in the InitialValue text box.

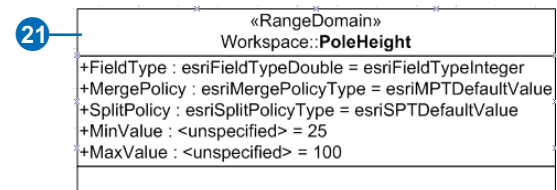
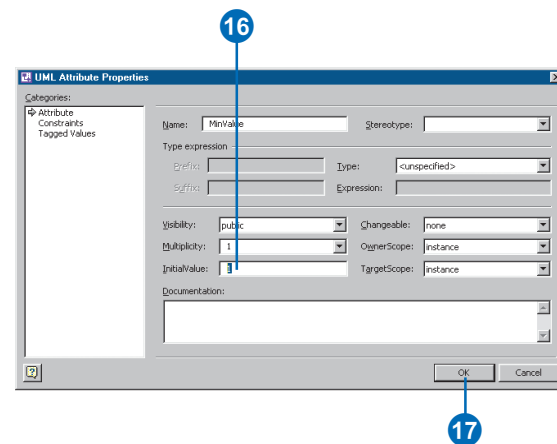
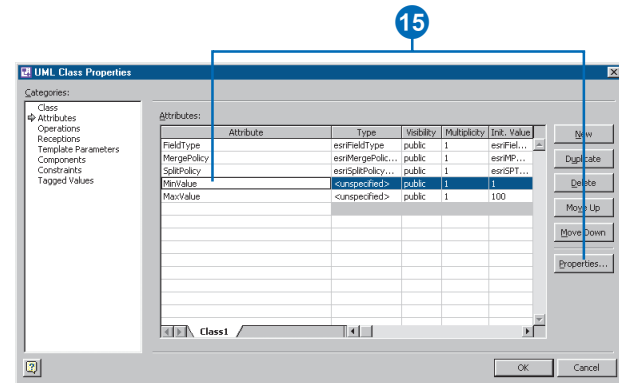
17. Click OK.

18. Click the MaxValue attribute and click Properties.

19. Type the maximum value for the range domain in the InitialValue text box.

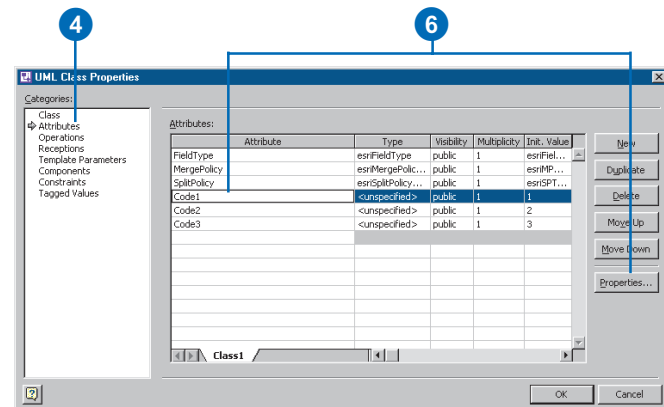
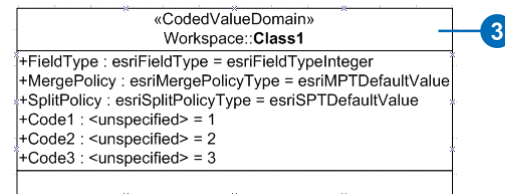
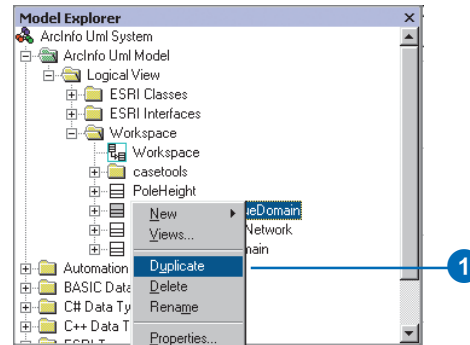
20. Click OK.

21. Right-click the domain in the diagram and click Shape Display Options. Click the Attribute check box to hide the attribute types.



Creating coded value domains

1. Navigate to and right-click `TemplateCodedValueDomain` in the Model Explorer under the Workspace package. Click **Duplicate**.
2. Drag and drop on the diagram the copy of the `TemplateCodedValueDomain` that is created under the Workspace package.
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 chapter to set the field type and the split and merge policies.
6. Click the `Code1` attribute and click **Properties...**.

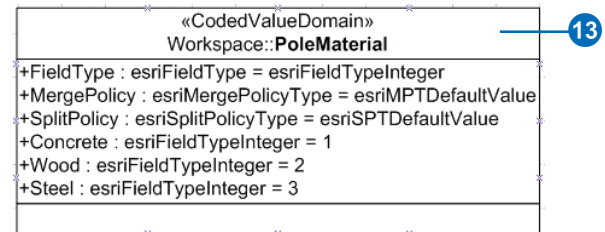
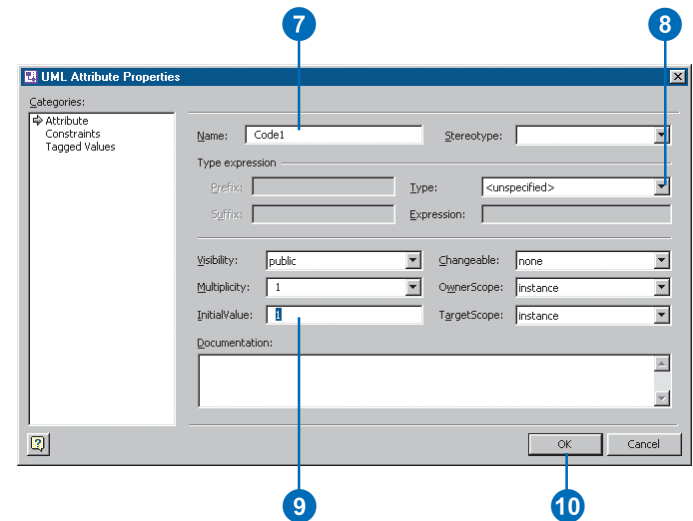


Tip

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.



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.

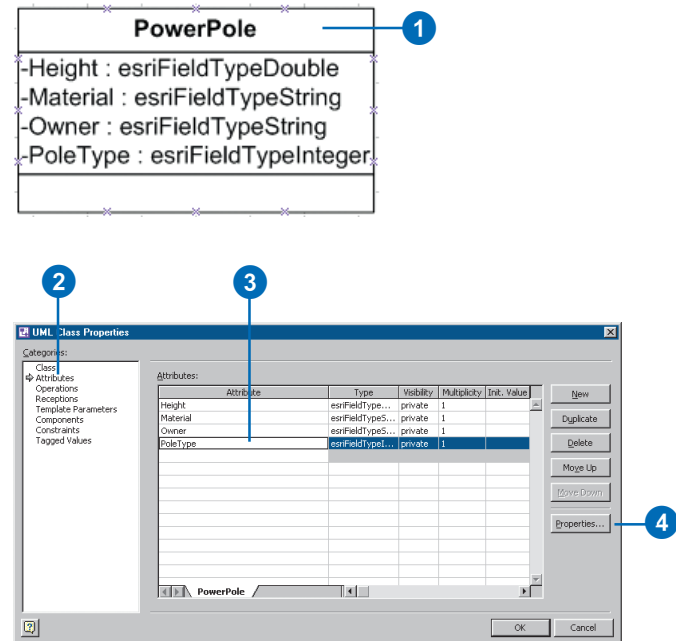
A field in the parent class must be stereotyped as SubtypeField. Its initial value is the subtype code of the default subtype. The subtype field must be typed as “esriFieldTypeInteger” in the parent class.

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.

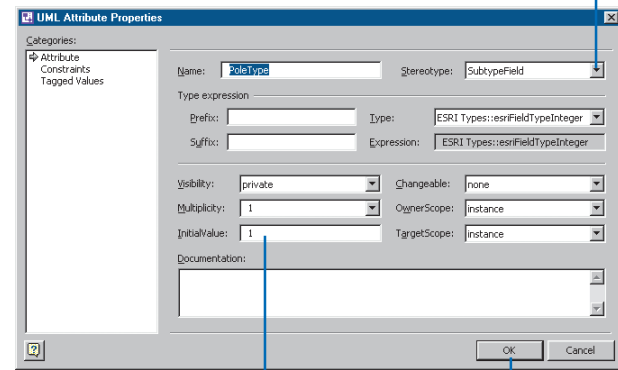
Defining the subtype field for the feature class

1. Double-click in the diagram 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. ►

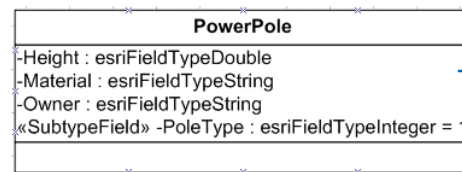


5. Click the Stereotype dropdown arrow and click SubtypeField.
6. Type the subtype code of the default subtype in the InitialValue text box.
7. Click OK.

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



The image shows the 'UML Attribute Properties' dialog box. It has a 'Categories' pane on the left with 'Attribute' selected. The main area contains fields for 'Name' (set to 'PoleType'), 'Stereotype' (set to 'SubtypeField'), 'Type expression' (with 'Prefix' and 'Suffix' fields), 'Type' (set to 'ESRI Types::esriFieldTypeInteger'), 'Expression' (set to 'ESRI Types::esriFieldTypeInteger'), 'Visibility' (set to 'private'), 'Changeable' (set to 'none'), 'Multiplicity' (set to '1'), 'OwnerScope' (set to 'instance'), 'InitialValue' (set to '1'), 'TargetScope' (set to 'instance'), and a 'Documentation' text area. At the bottom right are 'OK' and 'Cancel' buttons. Blue callout circles with numbers 5, 6, and 7 point to the 'Stereotype' dropdown, the 'InitialValue' field, and the 'OK' button respectively.



The subtype field is stereotyped as SubtypeField.

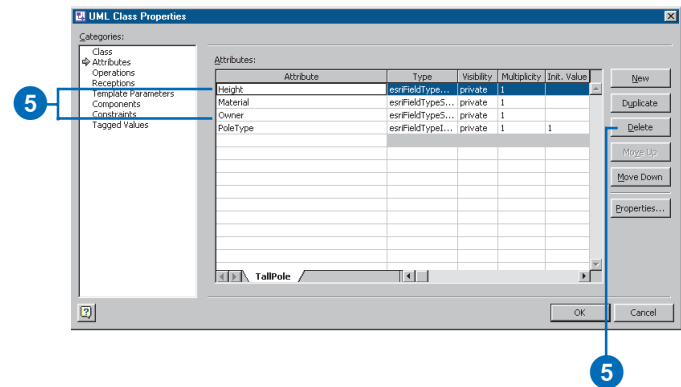
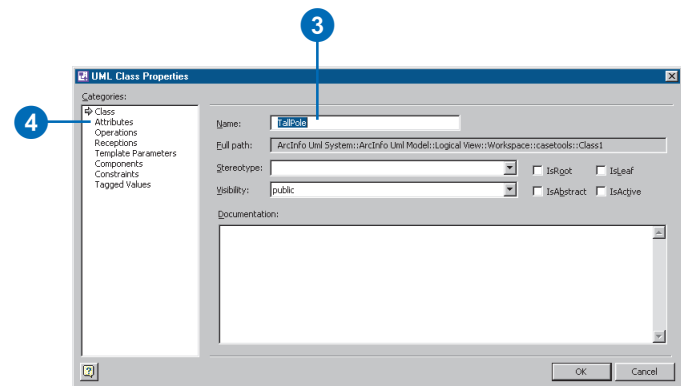
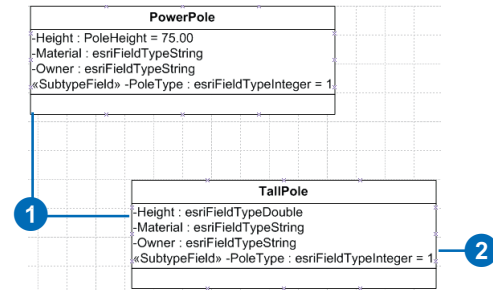
Tip

When to include a field in a subtype

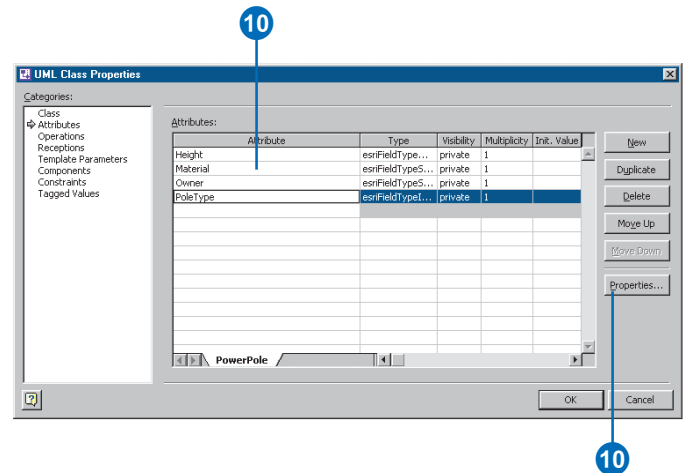
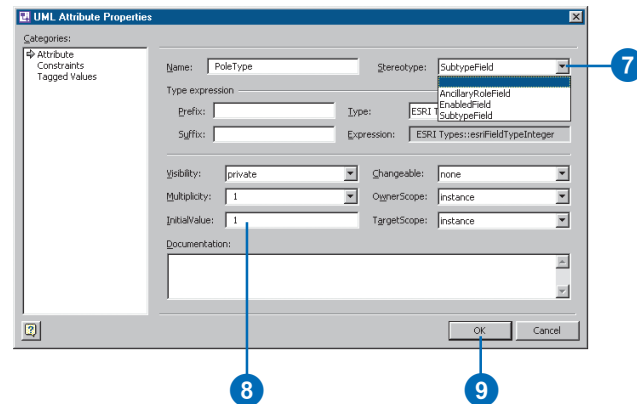
A field should be included in a subtype only when you want to associate a default value or domain with it. Fields inherited from abstract classes can be included in subtypes.

Creating a subtype

1. Copy the class in the diagram 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. ►

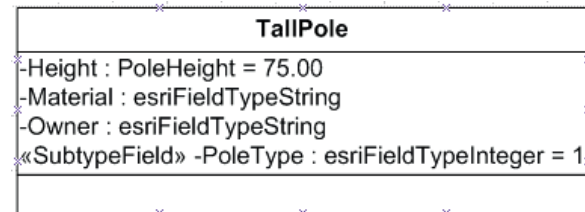
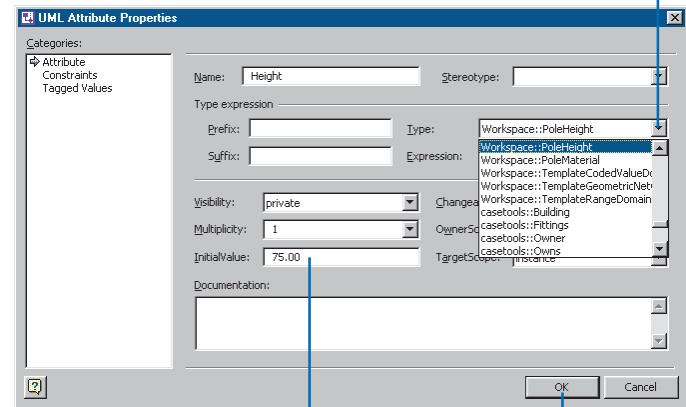


6. Click the subtype field and click Properties.
 7. Click the Stereotype dropdown arrow and click the blank stereotype to set its stereotype to nothing.
 8. Type the code for this subtype in the InitialValue text box.
 9. Click OK.
- Repeat steps 6 through 9 until you have associated default values and domains with all the attributes.
10. Click the attribute and click Properties to set default values and domains for a field. ►



11. Click the Type dropdown arrow and click the domain you want to associate with the attribute as the type.
12. Type the default value in the InitialValue text box to associate a default value with this attribute.
13. Click OK to close the UML Attribute Properties dialog box.
14. Click OK to close the UML Class Properties dialog box.

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



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

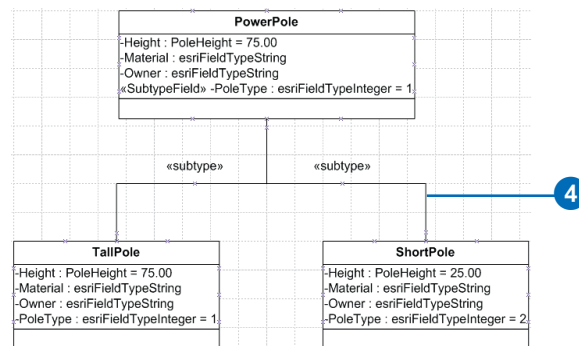
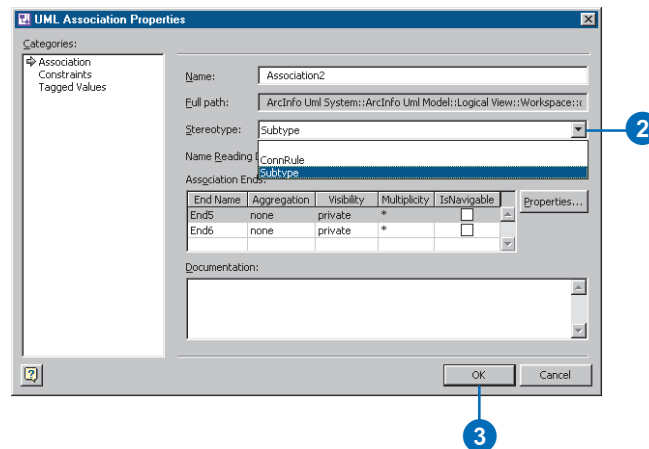
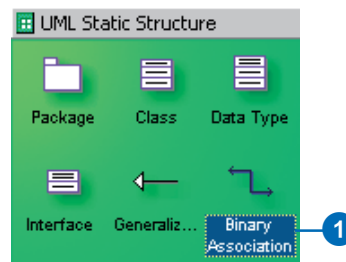
Tip

Setting the initial value for the subtype field

Be 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. Click Binary Association in the UML Static Structure stencil and drag and drop it on the diagram. Connect the parent and subtype classes.
2. Double-click the association in the diagram. 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.



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.

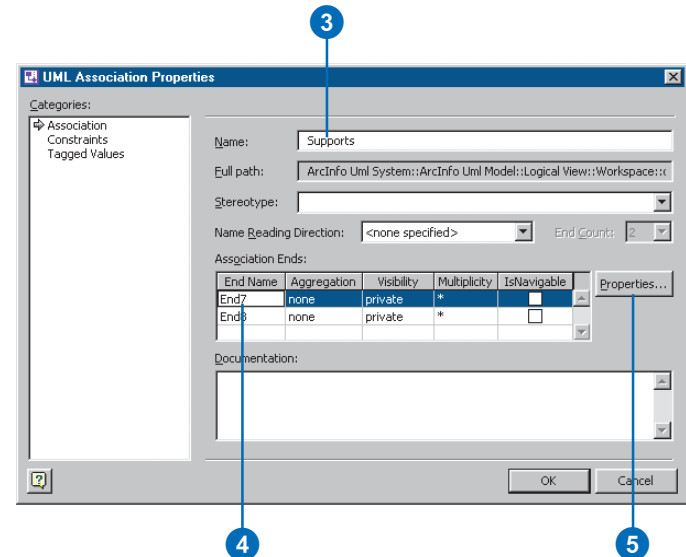
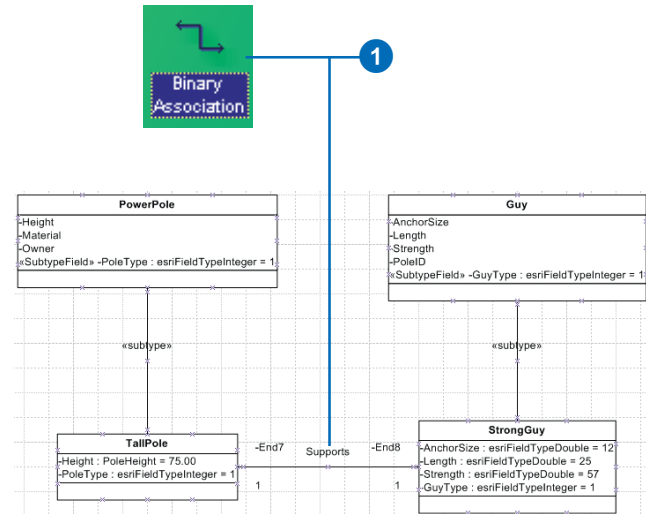
A value of 1..3 means at least one and no more than three objects of this subtype can be related.

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).

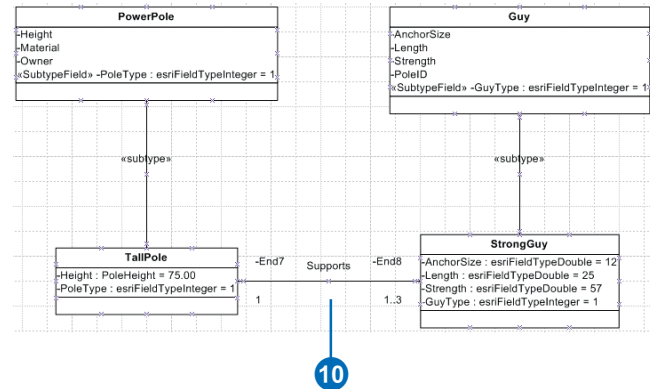
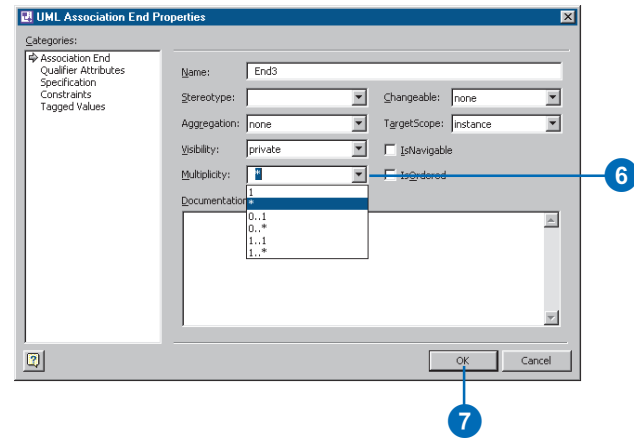
See Also

For more information on relationship rules, see the chapter 'Defining relationship classes' in this book.

1. Click Binary Association in the UML Static Structure stencil 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. ►



6. Select the valid cardinality as the multiplicity of the association end.
7. Click OK to close the UML Association End Properties dialog box.
8. Repeat steps 4 through 7 for the other association end.
9. Click OK to close the UML Association Properties dialog box.
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.



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.

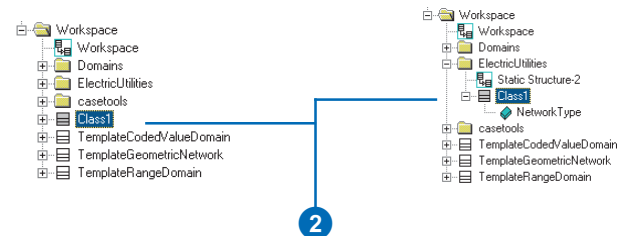
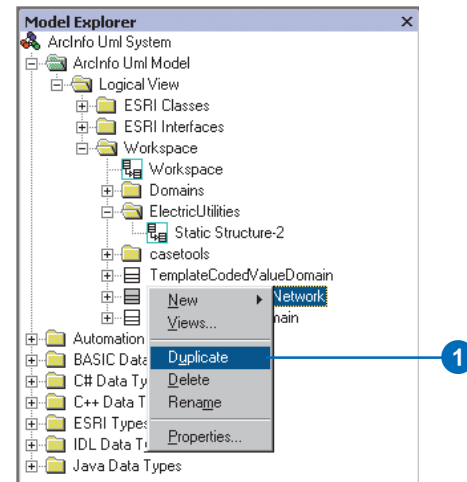
The only attribute in a geometric network class defines the network type. When you copy the template, the attribute is already in the class.

Tip

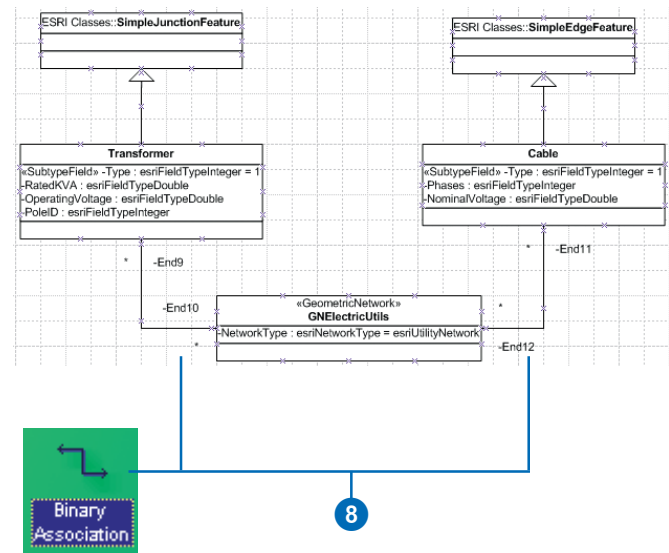
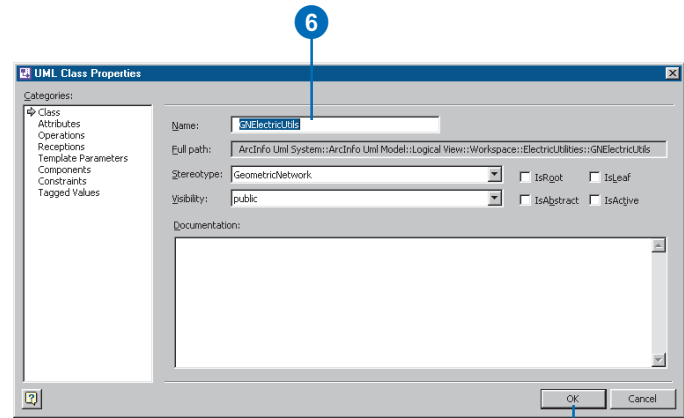
Feature datasets

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

1. Right-click TemplateGeometricNetwork under the Workspace package in the Model Explorer and click Duplicate.
2. Drag and drop the copy of TemplateGeometricNetwork, which is created in the Model Explorer under the Workspace package, 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. Click Binary Association in the Model Explorer and drag it on the diagram. Connect the network feature classes to the geometric network.



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.

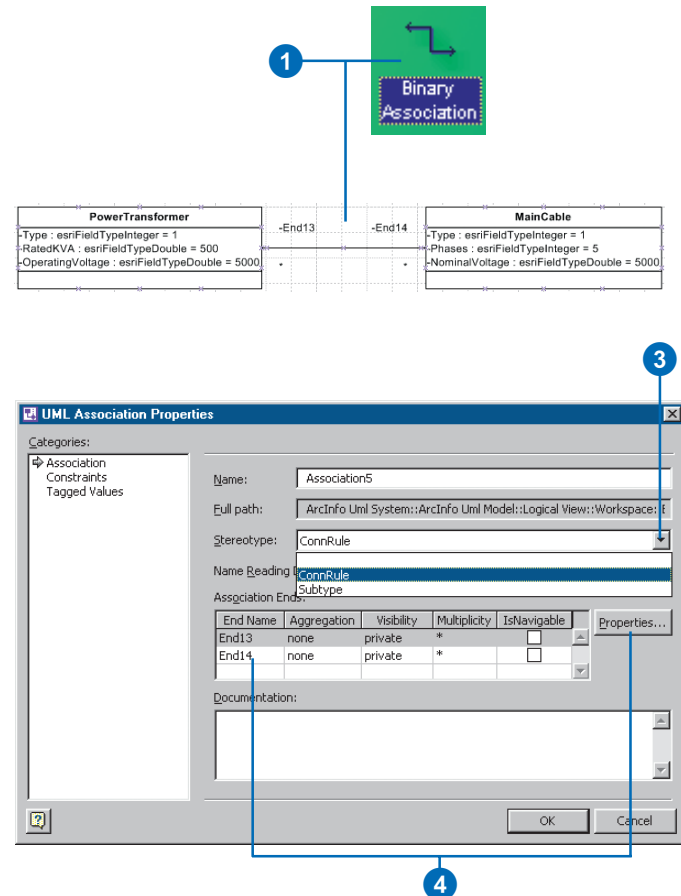
Edge–junction connectivity rules can have specific cardinalities for each subtype involved. In this example, one power transformer can be connected to up to two main cables. One of the junction subtypes can be marked as the default junction by stereotyping the association end as Default.

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

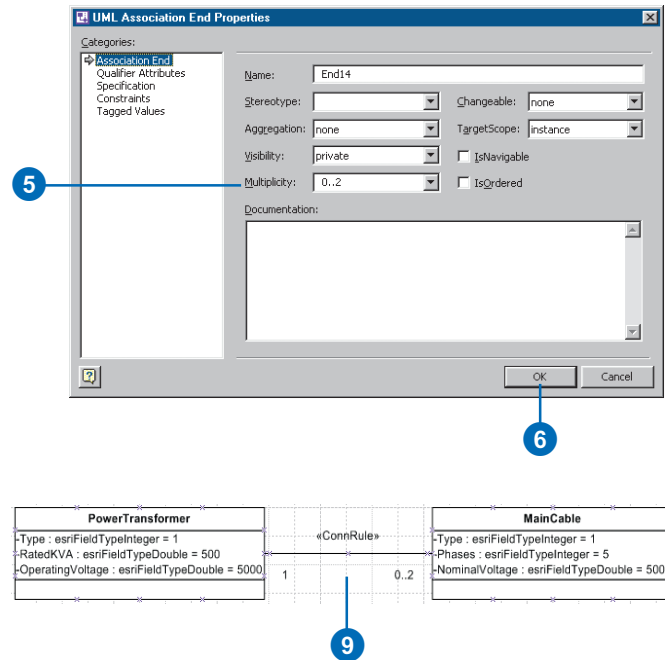
1. In the UML Static Structure stencil, click Binary Association and drag and drop it on the diagram. Connect the edge subtypes and junction subtype.
2. Double-click the association.
3. Click the Stereotype dropdown arrow and click ConnRule.
4. Click one of the association ends and click Properties.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



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.

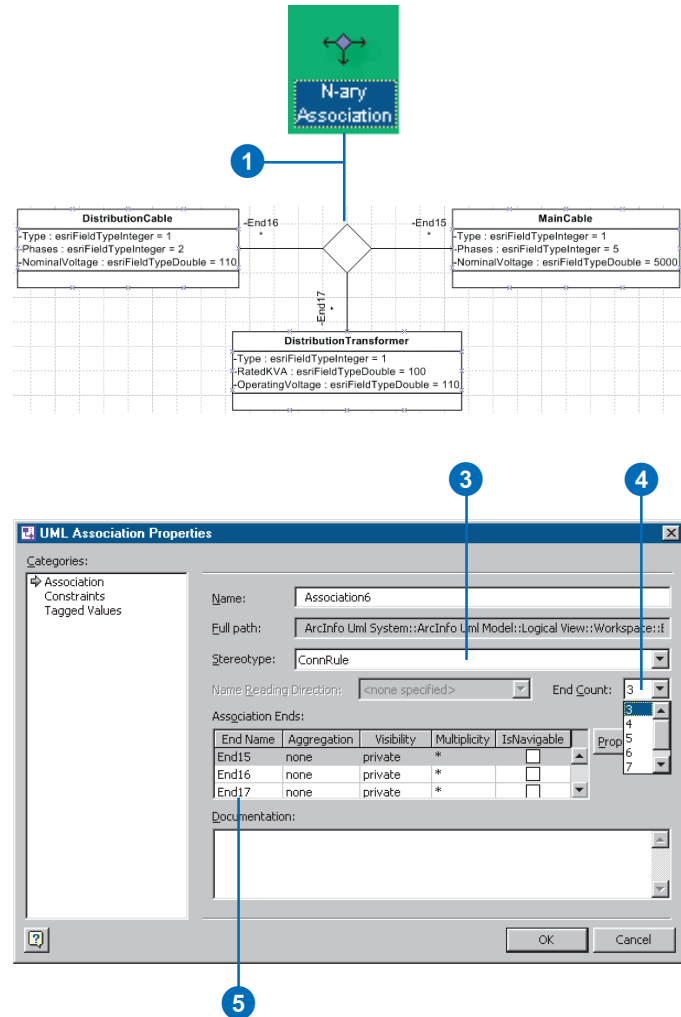
Both edge-junction and edge-edge rules can involve the generic junction.

5. Type the cardinality for the subtype.
6. Click OK to close the UML Association End Properties dialog box.
7. Repeat steps 4 through 6 to set the cardinality for the second subtype.
8. Click OK to close the UML Association Properties dialog box.
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.

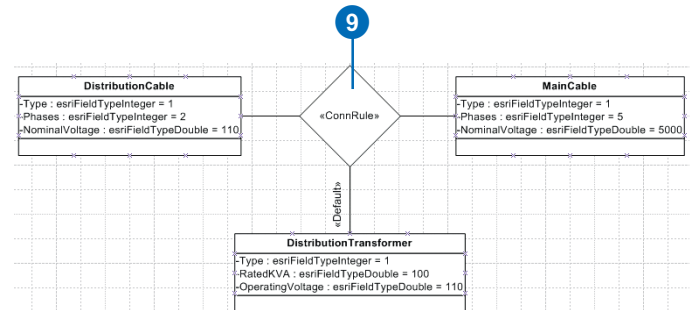
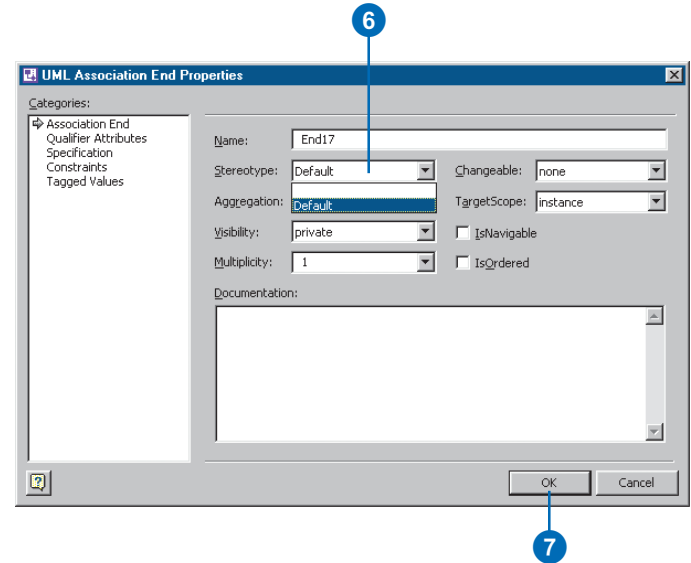


Creating an edge-edge rule

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

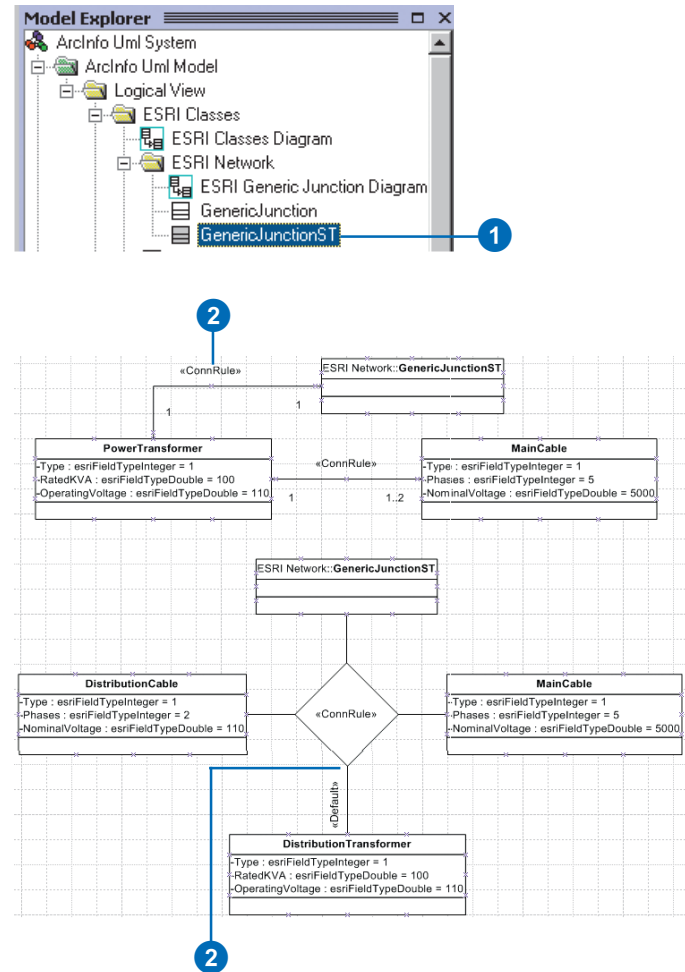


6. Click the Stereotype dropdown arrow and click Default to mark the junction subtype as the default junction subtype in the edge-edge rule.
7. Click OK to close the UML Association End Properties dialog box.
8. Click OK again to close the UML Association Properties dialog box.
9. Right-click the association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



Using the generic junction subtype

1. Navigate to and click GenericJunctionST in the Model Explorer. 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', as described in this chapter.



Extending classes with custom behavior

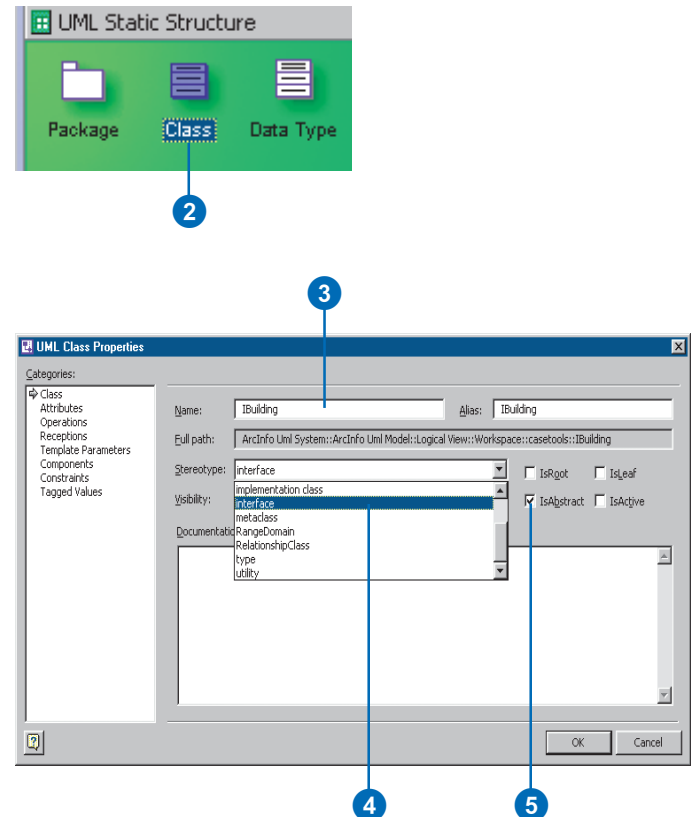
In addition to designing schema, you can use UML and CASE tools to generate code to create custom feature behavior. There are two modeling tasks associated with this: creating new interfaces and creating class extensions. It is important to understand that interfaces and class extensions in UML models 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. Click Class in the UML Static Structure stencil and drag 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. ►



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`.

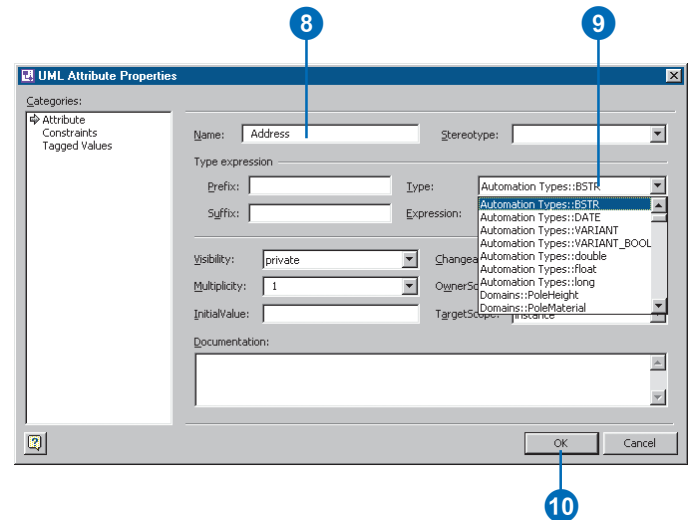
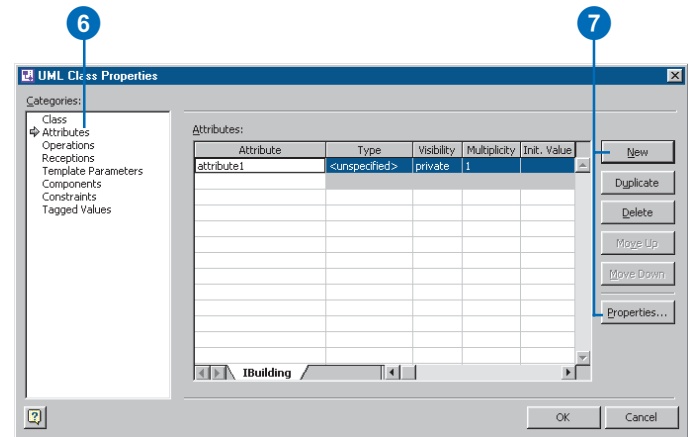
Class extensions for tables are derived from `ObjectClassExtension`. Likewise, class extensions for feature classes are derived from `FeatureClassExtension`.

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

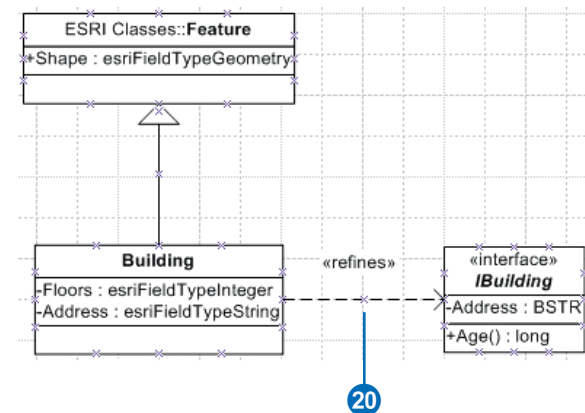
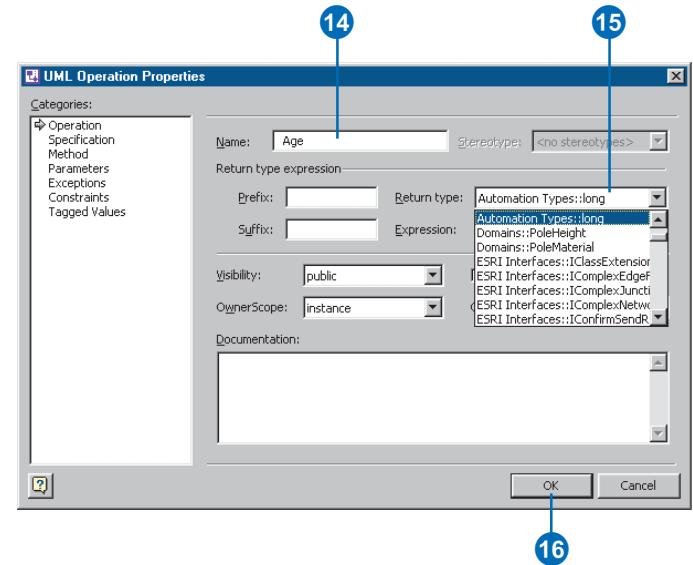
See Also

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. ►



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. Click Refinement in the UML Static Structure stencil and drag and drop it on the diagram.
20. Connect the custom feature and the interface.



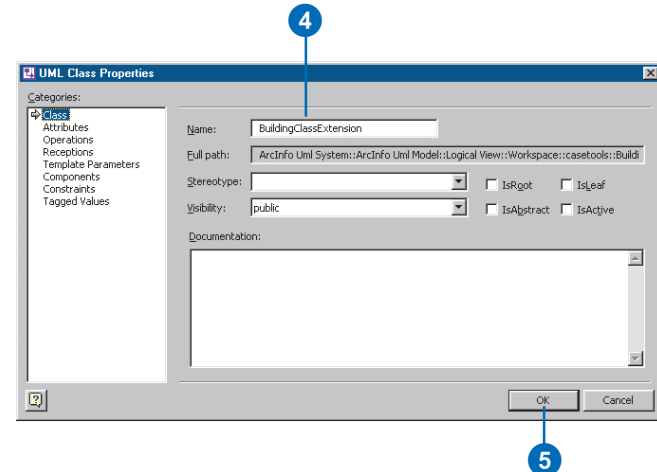
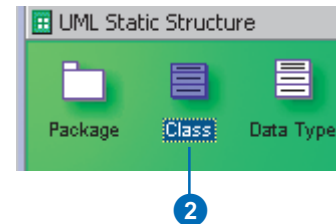
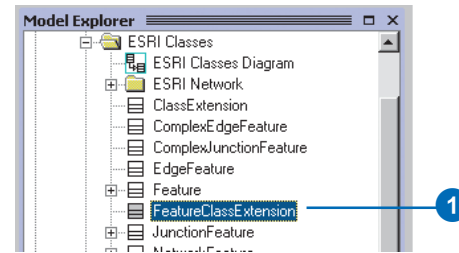
Tip

Class extension naming convention

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

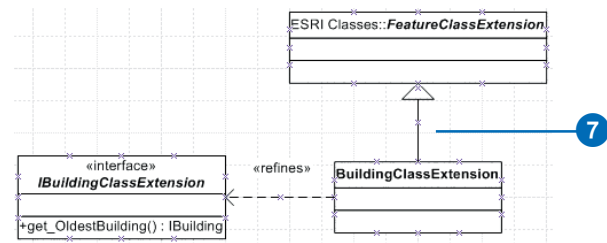
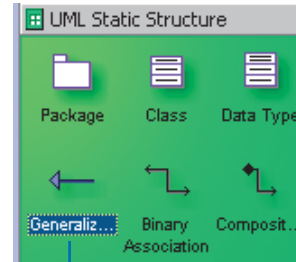
Creating a class extension

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



6. Click Generalization in the UML Static Structure stencil 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' in this chapter. There are a number of predefined optional interfaces you can implement in your class extension. ►

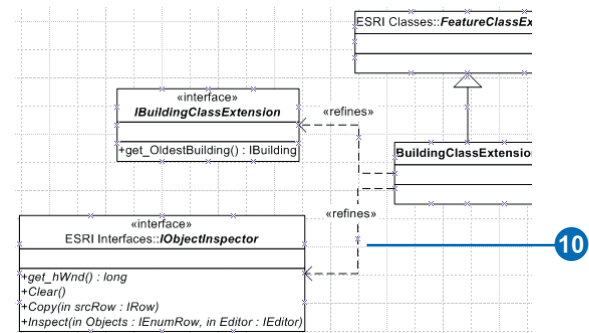
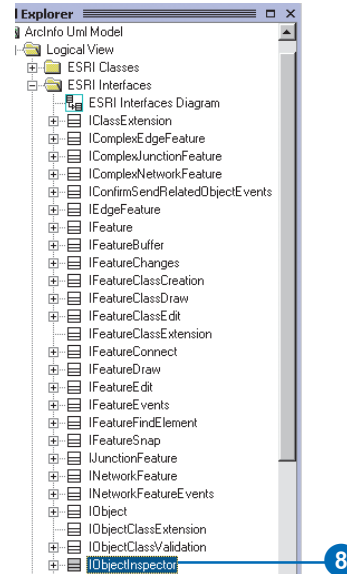


Tip

Optional interfaces

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

8. Click the optional interface in the Model Explorer tree under ESRI Interfaces to select it. Drag and drop it onto the diagram.
9. Click Refinement in the UML Static Structure stencil 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.



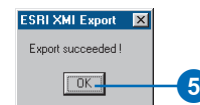
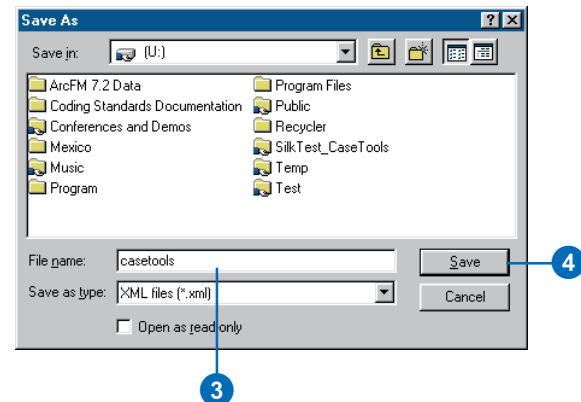
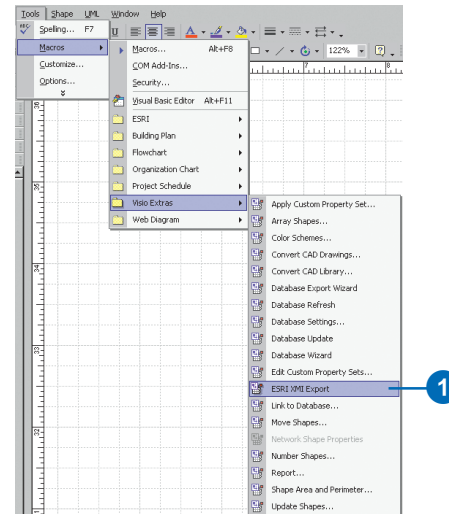
Exporting a model to XMI

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. Click Tools in Visio, 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.



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 XML.

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

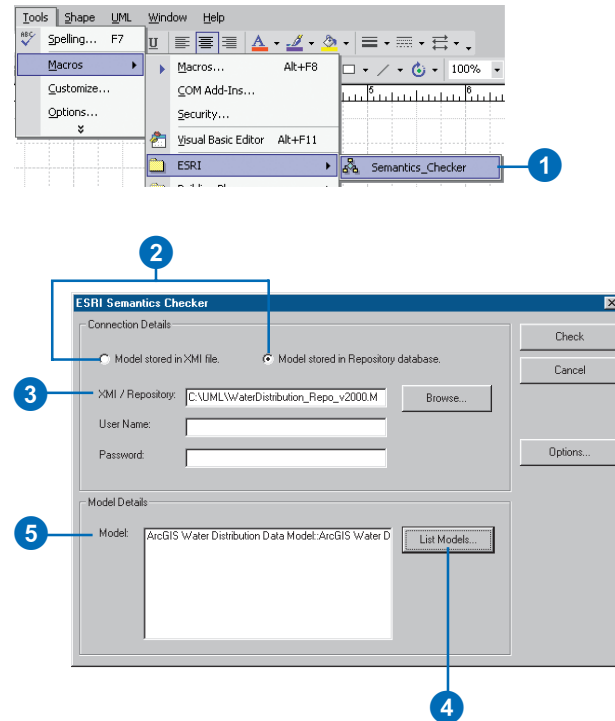
Tip

Options

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

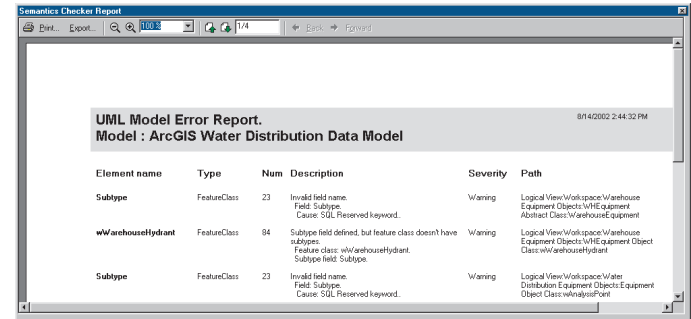
Running the semantics checker

1. Click Tools in Visio, point to Macros, point to ESRI, then click Semantics_Checker.
2. Select the source of your model. Models can be stored in XML files or repository databases.
3. Type the path to the XML file or repository database or click Browse to navigate to it.
4. Click List Models if you are using a repository.
5. Select the model you want to verify. ►



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
8/14/2002 2:44:32 PM

Element name	Type	Num	Description	Severity	Path
Subtype	FeatureClass	23	Invalid field name. Field: Subtype. Cause: SQL Reserved keyword.	Warning	Logical View/Workspace/V/warehouse Equipment Objects/VHE/equipment Abstract Class/Vwarehouse/equipment
wWarehouseHydant	FeatureClass	84	Subtype field defined, but feature class doesn't have subtypes. Feature class: wWarehouseHydant. Subtype field: Subtype.	Warning	Logical View/Workspace/V/warehouse Equipment Objects/VHE/equipment Object Class/wWarehouseHydant
Subtype	FeatureClass	23	Invalid field name. Field: Subtype. Cause: SQL Reserved keyword.	Warning	Logical View/Workspace/V/water Distribution Equipment Objects/equipment Object Class/wWarehousePost

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.

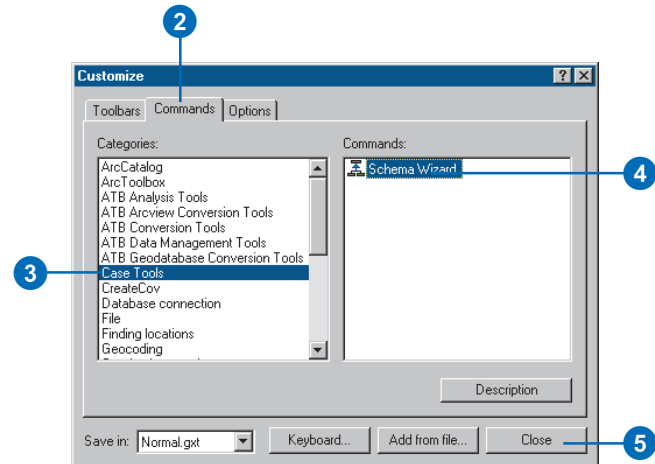
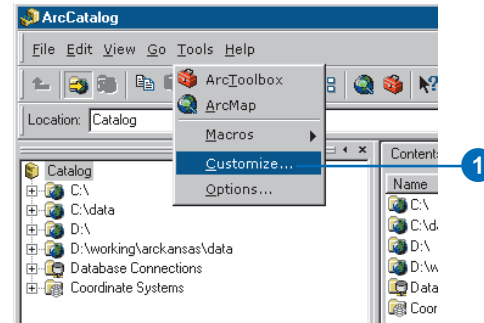
The examples here involve objects, features, and network features. Many of these objects and features contain subtypes with attribute domains and default values. The examples ►

Adding the CASE Tools wizard to ArcCatalog

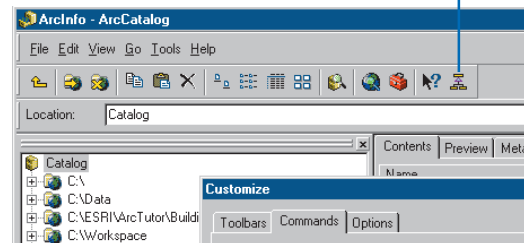
1. Click Tools in ArcCatalog and click Customize.
2. Click the Commands tab in the Customize dialog box.
3. Click Case Tools.
4. Drag the Schema Wizard command from the Commands list and drop it on the Standard toolbar.

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.

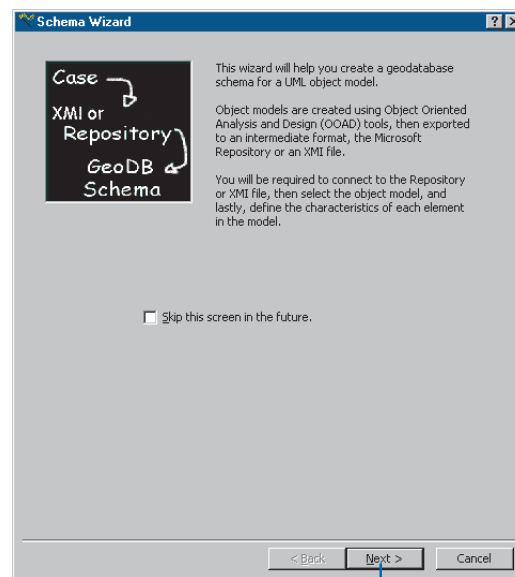
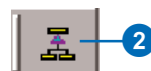
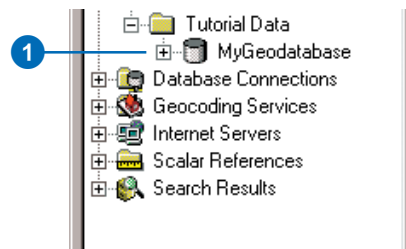
See Also

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

Connecting to an XMI file or repository

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

Clicking the check box in the introduction dialog box will allow you to skip the introduction dialog box the next time you use the wizard.

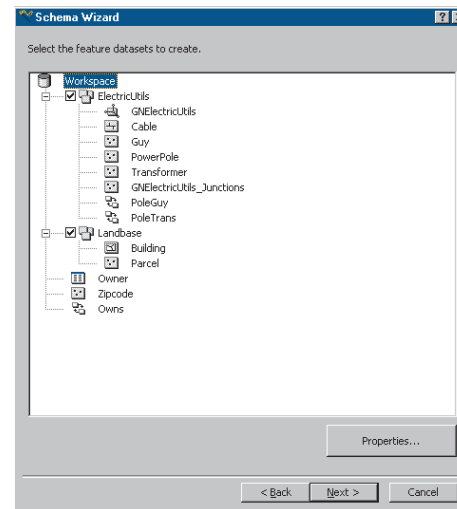
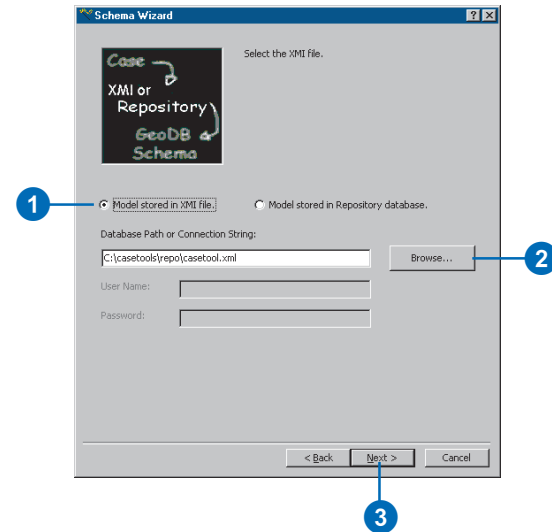


Selecting an XMI file

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.



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

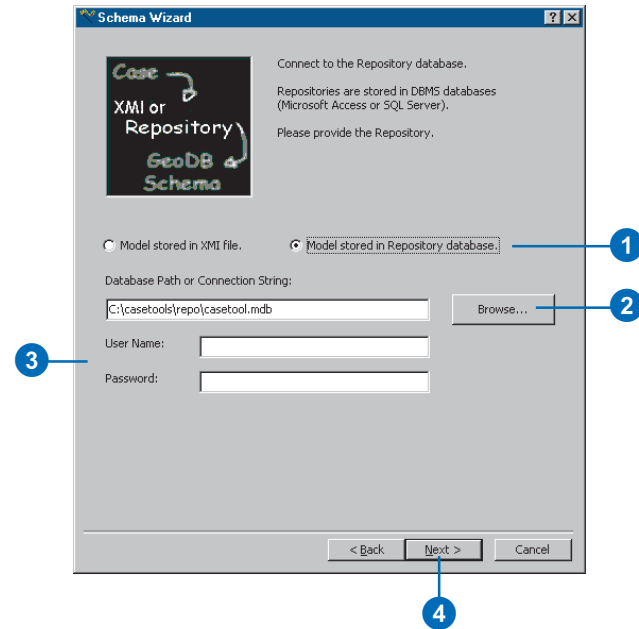
Tip

Microsoft Repository

If you are connecting to a repository stored in Microsoft SQL Server through the Schema Wizard, 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.

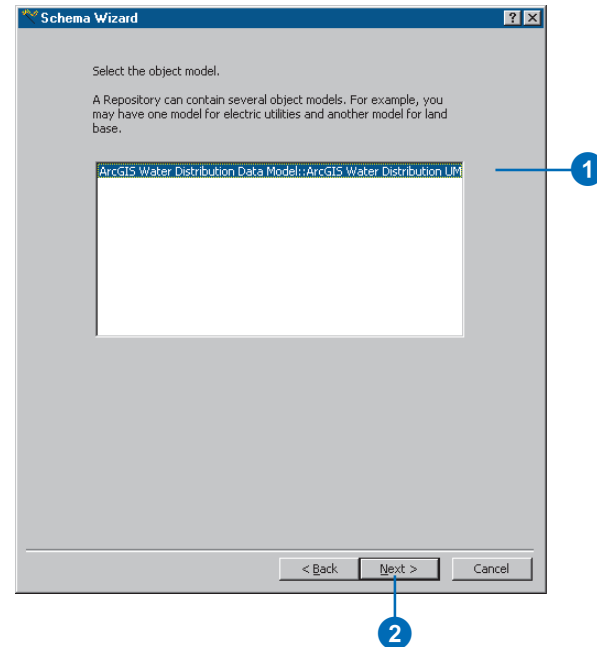


Selecting an object model

1. If you are using a Repository database, the Schema Wizard will let you select the model you want to create the schema for. Select the model.

2. Click Next.

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



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

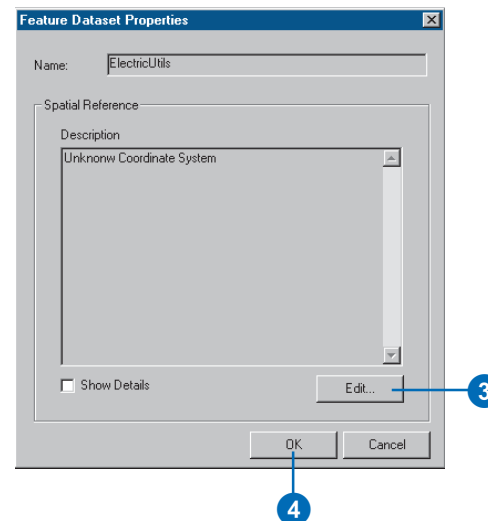
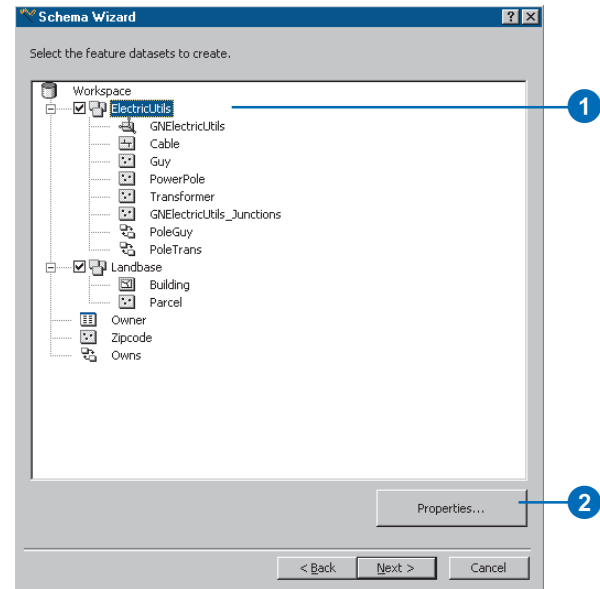
When you select a feature dataset and assign it a spatial reference, all of the feature classes within that feature dataset share the same spatial reference. The exception is the M-domain. Feature classes in the same feature dataset may have different M-domains.

See Also

For a detailed explanation of spatial references, see the chapter 'Creating new items in a geodatabase' in this book.

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



Setting properties for object classes or tables

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

Tip

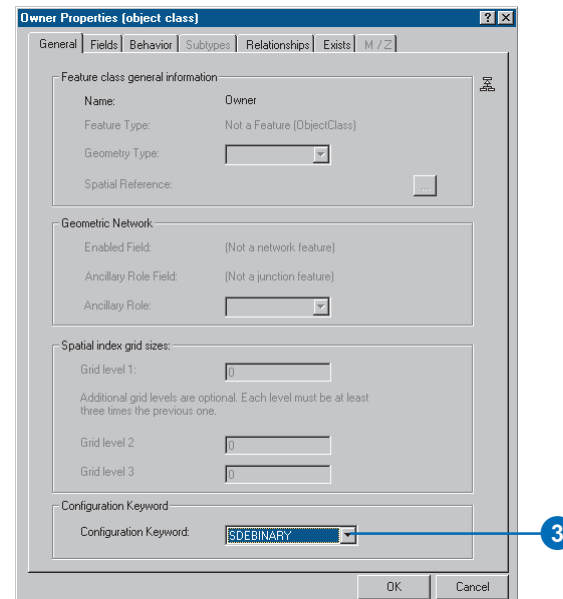
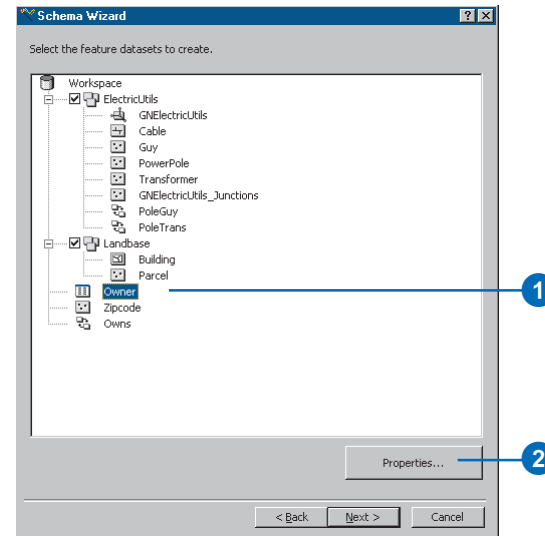
Classes that exist in a geodatabase

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. Click the dropdown arrow and click a storage configuration keyword if you are generating schema for an ArcSDE geodatabase. If you do not select a keyword, DEFAULTS will be used. ►



Tip

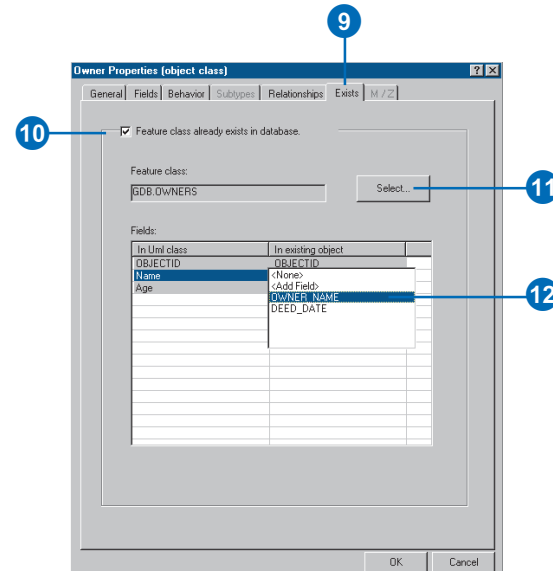
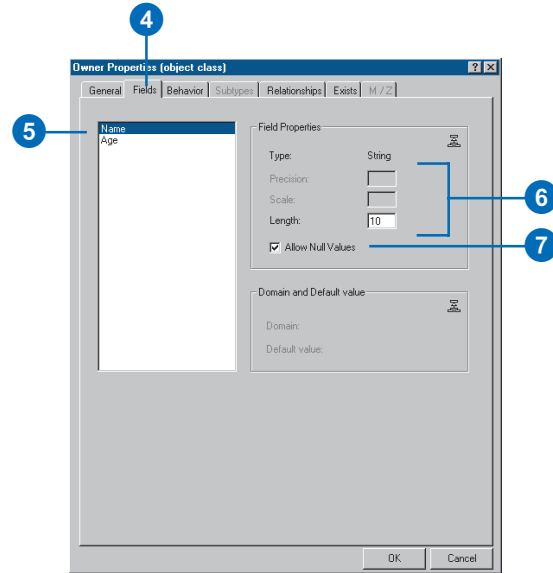
Related objects

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

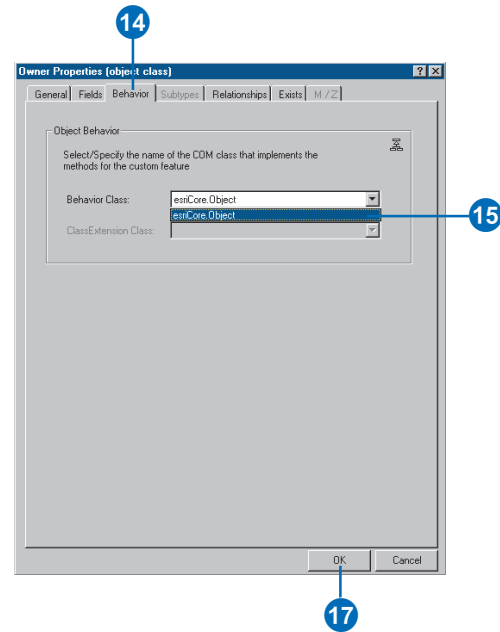
See Also

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

- Click the Fields tab if the table in which you want to store this class does not already exist in the database. If the table does exist, skip to step 9.
- 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.
- 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.
- Check Allow Null Values if you want *null values* to be permitted in this field.
- Repeat steps 5 through 7 for each field in the table, then skip to step 13.
- Click the Exists tab if the table in which you want to store this class already exists in the database.
- Check the check box to indicate that the table already exists.
- Click Select to select the table from the list of tables in the database.
- For each property in the UML class, click the field in the existing table that will store that property. ►



13. Skip to step 17 if you chose to use ESRI-provided components for all of your classes.
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. Examine the properties of the subtypes or relationship classes associated with this table, if there are any, by clicking the Relationships and Subtypes tabs.
17. Click OK.
18. Repeat steps 2 through 17 for the rest of your tables.



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.

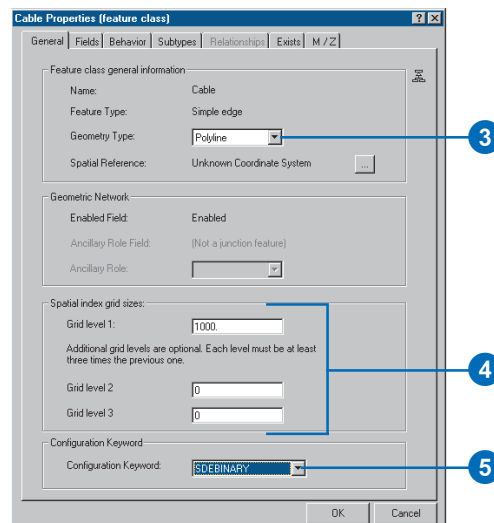
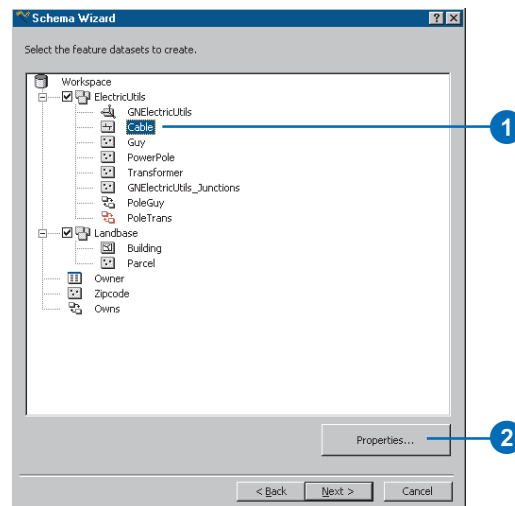
Standalone feature classes have their own spatial reference.

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 Properties 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. Click the dropdown arrow and click a storage configuration keyword if generating schema for an ArcSDE geodatabase. If you do not select a keyword, DEFAULTS will be used.
6. Follow steps 5 through 18 of 'Setting properties for object classes or tables' in this chapter.
7. Repeat steps 1 through 6 for the rest of the feature classes in your feature dataset.

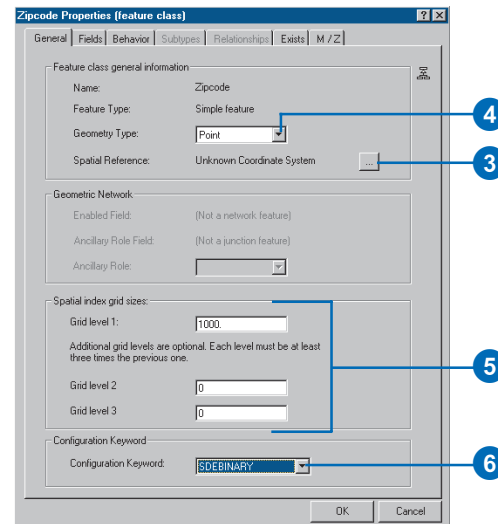
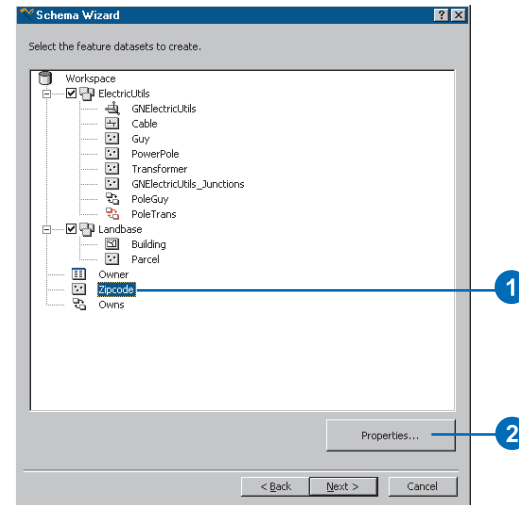


See Also

For a detailed explanation of spatial references and how to set them for objects in a geodatabase, see the chapter 'Creating new items in a geodatabase' in this book.

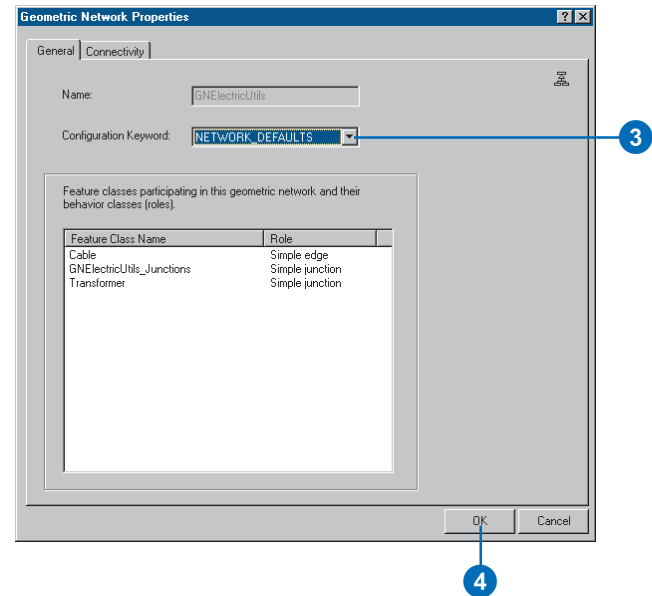
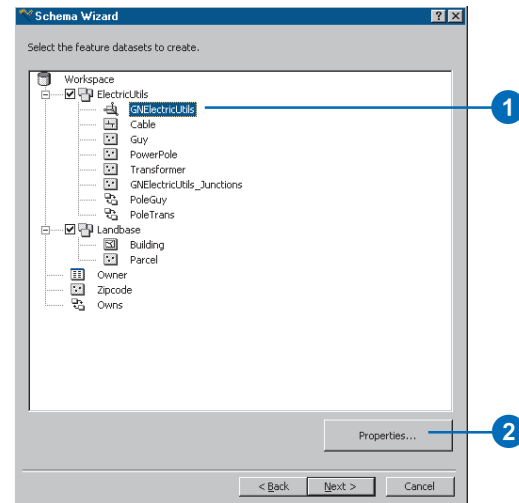
Setting properties for standalone feature classes

1. Click the feature class whose settings you want to modify.
2. Click Properties to set the properties for this class.
3. Click the Browse 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. Click the Configuration Keyword dropdown arrow and click a storage configuration keyword if generating schema for an ArcSDE geodatabase. If you do not select a keyword, DEFAULTS will be used.
7. Follow steps 5 through 18 of 'Setting properties for object classes or tables' in this chapter.
8. Repeat steps 1 through 6 for the rest of the standalone feature classes in your model.



Setting properties for a geometric network

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



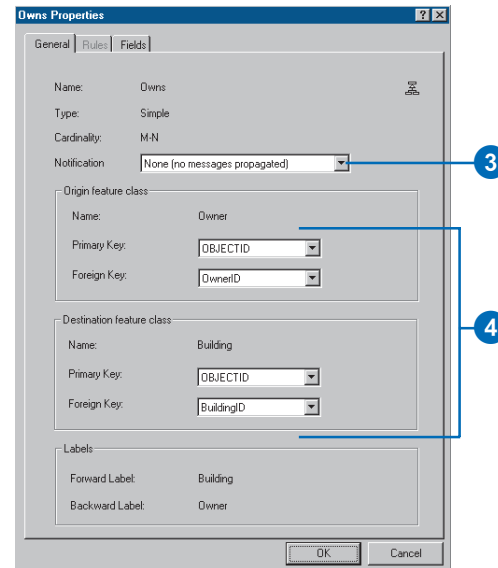
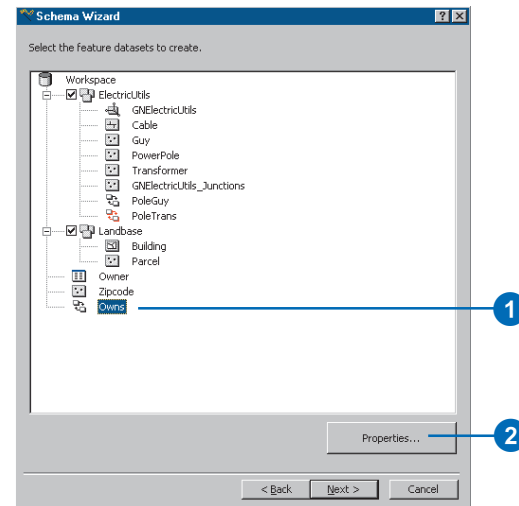
Setting properties for relationship classes

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.

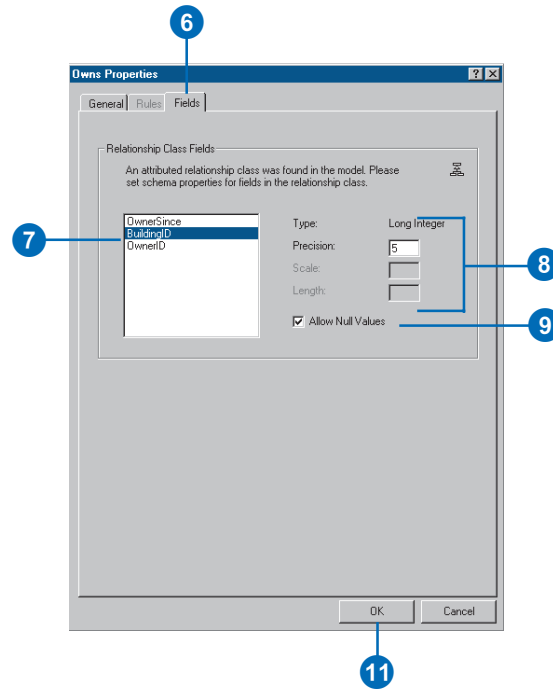
See Also

For a detailed discussion of primary and foreign keys and relationship messaging, see the chapter 'Defining relationship classes' in this book.

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.



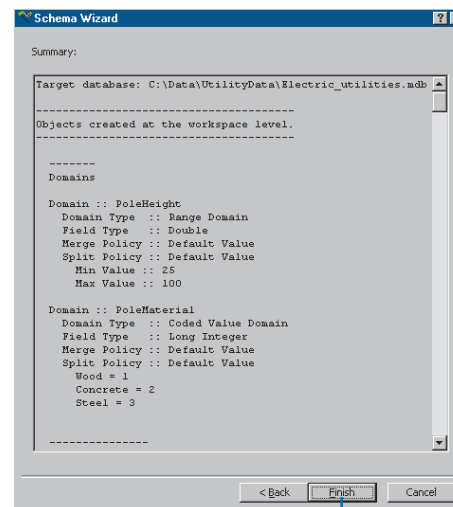
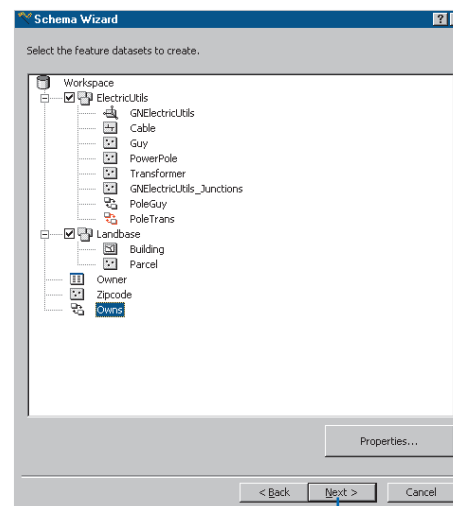
5. Skip to step 11 if your relationship class is not modeled as a class with attributes.
6. Click the Fields tab.
7. Click a field in the list of fields. If you used tagged values in your UML model to set the field properties, skip to step 11.
8. Type its precision if it is an integer field, its precision 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.



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.



Glossary

active data frame

The data frame in a map that is currently being worked on—for example, the data frame to which layers are being added. The active data frame is shown in bold text in the ArcMap table of contents.

alias

Another name for a field in a table.

aligned dimension

A dimension that runs parallel to the baseline and represents the true distance between the beginning and end dimension points.

annotation

1. Descriptive text used to label features. It is used for display, not for analysis.
2. A feature class used to label other features. Information stored for annotation includes a text string, the location at which it is displayed, and a text symbol (color, font, size, and so on) for display.
3. The process of automating text placement or the text associated with a feature or an area on a map.

arc-node topology

Arcs represent linear features and the borders of area features in a coverage. Every arc has a from-node, which is the first vertex in the arc, and a to-node, which is the last vertex. Nodes indicate the endpoints and intersections of arcs. They do not exist as independent features. Together they define the direction of the arc. Arc-node topology defines connectivity in coverages—arcs are connected to each other if they share a common node.

ArcInfo workspace

A file-based collection of coverages, grids, TINs, or shapefiles stored as a directory of folders in the file system.

ArcSDE

A gateway to a multiuser commercial RDBMS—for example, Oracle, Microsoft SQL Server, Informix, and IBM® DB2. ArcSDE is an open, high-performance spatial data server that employs client/server architecture to perform efficient spatial operations and manage large, shared geographic data. Was known as a Spatial Database Engine (SDE) before 1999.

asynchronous

Operations or events that do not happen at the same time. In disconnected editing, modifying the properties of a check-out is an asynchronous operation; changes made to the check-out in a master geodatabase do not affect the associated check-out in a check-out geodatabase.

attribute

A characteristic of a map feature. Attributes of a river might include its name, length, average depth, and so on.

attribute domain

A named constraint in the database. An attribute constraint can be applied to a field of a subtype of a feature class or object class to make an attribute rule. Types of attribute domains include range and coded value domains.

attribute table

A database management system (DBMS) or other tabular file containing rows and columns. In ArcInfo, attribute tables are associated with a class of geographic features such as wells or roads. Each row represents a geographic feature. Each column represents one attribute of a feature, with the same column representing the same attribute in each row. See also feature attribute table.

Attributes dialog box

A dialog box that lets you view and edit attributes of features you've selected in ArcMap.

azimuth

An angle measured from north. Often used to define an oblique cylindrical projection or the angle of a geodesic between two points.

behavior

Properties of an object in a geodatabase that describe how it can be edited and drawn. Behavior includes, but is not limited to, validation rules, subtypes, default values, and relationships.

buffer

A zone of a specified distance around features. Both constant- and variable-width buffers can be generated on each feature's attribute values. The resulting buffer zones form polygons that are either inside or outside the specified buffer distance from each feature. Buffers are useful for proximity analysis (for example, to find all stream segments within 300 feet of a proposed logging area).

CAD

See computer-aided design.

CAD feature class

A feature class in a CAD dataset. A CAD feature dataset is composed of feature classes representing all the points, lines, polygons, or annotation in the CAD drawing. For example, a CAD drawing may contain two line layers representing roads and parcel boundaries, respectively. The CAD dataset's line feature class represents all features in both the road and parcel boundary layers.

centroid

The mathematical or geographical center point of a polygon or the midpoint of a line.

check-out

A geodatabase entity that represents data that has been checked out to or from another geodatabase.

check-out geodatabase

A personal or ArcSDE geodatabase that contains data checked out from a master geodatabase.

check-out version

A version created in a check-out ArcSDE geodatabase when data was checked out. This version is created as a child of the synchronization version. Only the edits made to this check-out version can be checked back in to the master geodatabase.

checking in

The process of transferring changes to a master geodatabase.

checking out

The process of copying data to a check-out geodatabase.

circle

A geometric shape for which the distance from the center to any point on the edge is equal.

circular arc

When creating features, it is often necessary to create a circular arc. Instead of being made of numerous vertices, a circular arc has only two vertices as endpoints. ArcMap offers four ways to create a segment that is a circular arc. These include the Arc tool, the Endpoint Arc tool, the Tangent Curve tool, and the Tangent Curve command.

cluster tolerance

The minimum distance between vertices in the topology. Vertices that fall within the cluster tolerance will be snapped together during the validate topology process. The default cluster tolerance is the minimum possible cluster tolerance, based on the precision and extent defined for the spatial reference of the dataset. See also fuzzy tolerance, used analogously in coverage data model.

clustering

A part of the topology validation process in which vertices that fall within the cluster tolerance are snapped together.

coincident

Vertices or boundaries are coincident when they are within the cluster tolerance of one another. See also cluster tolerance.

column

The vertical dimension of a table. A column has a name and a data type applied to all values in the column. See also item, field, and attribute.

compress

The process of shrinking the size of a database or file. Improves database performance by removing redundant rows shared by multiple versions. The process can only be run by the ArcSDE administrator.

computer-aided design

An automated system for the design, drafting, and display of graphically oriented information.

conflict

In the versioning reconciliation process, if the same feature in the edit version and reconciliation version have both been edited, the feature is said to be in conflict. Resolving the conflict requires you to make the decision as to the feature's correct representation using the conflict resolution dialog box.

connectivity

1. In a geodatabase, the state of edges and junctions in a logical network that controls flow, tracing, and pathfinding.
2. The topological identification in a coverage of connected arcs by recording the from- and to-node for each arc. Arcs that share a common node are connected. See also arc-node topology.

connectivity rules

Network rules that constrain the type of network features that may be connected to one another and the number of features of any particular type that can be connected to features of another type. In most networks, not all edge types can logically connect to all junction types. Similarly, not all edge types can logically connect to all other edge types through all junction types. There are two types of connectivity rules: edge-junction and edge-edge.

constraints

In real-world databases, an object's attributes can't have any particular value based solely on what data types and ranges a particular field type in the database allows. In reality, the permissible values are a range or list of values.

construct features

The process of taking selected features from one or more feature classes and creating new features in a target feature class in an edit session. The Construct Features tool uses the input geometries of the selected features to construct polygons or lines following polygon boundaries, depending on the geometry of the target feature class.

contiguity

In coverages, the topological identification of adjacent polygons by recording the left and right polygons of each arc. See also polygon-arc topology.

control points

Points you establish on a paper map to represent known ground points or specific locations. Control points are used to register a paper map before you begin digitizing features on it with a digitizer.

coordinate

A set of numbers that designates location in a given reference system such as x,y in a planar coordinate system or x,y,z in a three-dimensional coordinate system. Coordinates represent locations on the earth's surface relative to other locations.

coordinate system

1. A reference system used to measure horizontal and vertical distances on a planimetric map. A coordinate system is usually defined by a map projection; a spheroid of reference; a datum; one or more standard parallels; a central meridian; and possible shifts in the x- and y-directions to locate x,y positions of point, line, and area features.
2. In ArcInfo, a system with units and characteristics defined by a map projection. A common coordinate system is used to spatially register geographic data for the same area.

coverage

A file-based vector data storage format for storing the location, shape, and attributes of geographic features. A coverage usually represents a single theme such as soils, streams, roads, or land use. It is one of the primary vector data storage formats for ArcInfo.

A coverage stores geographic features as primary features (such as arcs, nodes, polygons, and label points) and secondary features (such as tics, map extent, links, and annotation). Associated feature attribute tables describe and store attributes of the geographic features.

cracking

A part of the topology validation process in which vertices are created at the intersection of feature edges.

current task

During editing in ArcMap, a setting in the Current Task dropdown list that determines with which task the sketch construction tools (Sketch, Arc, Distance–Distance, and Intersection) will work. The current task is set by clicking a task in the Current Task dropdown list. All tasks in the Current Task dropdown list work with a sketch that you create. For example, the Create New Feature task uses a sketch you create to make a new feature. The Extend/Trim Feature task uses a sketch you create to determine where the selected feature will be extended or trimmed. The Cut Polygon Feature task uses a sketch you create to determine where the polygon will be cut.

custom behavior

Behavior is the implementation of an object class method. ESRI-provided objects have a set of methods associated with them. A developer can choose to override one of these methods or create additional methods. In this instance, the object is said to have custom behavior.

custom feature

A feature with specialized behavior instantiated in a class by a developer.

custom object

Objects that have custom behavior provided by a developer.

dangle tolerance

The minimum length allowed for dangling arcs in coverages in the ArcInfo Clean process. Clean removes dangling arcs that are shorter than the dangle length. Also known as the dangle length.

dangling arc

In coverages, an arc having the same polygon on both its left and right sides and having at least one node that does not connect to any other arc. It often identifies where a polygon does not close properly (for example, undershoot), where arcs don't connect properly, or where an arc was digitized past its intersection with another arc (for example, overshoot). A dangling arc is not always an error. For example, dangling arcs can represent cul-de-sacs in street centerline maps.

data

A collection of related facts usually arranged in a particular format and gathered for a particular purpose.

data frame

In ArcMap, a frame on the map that displays layers occupying the same geographic area. You may have one or more data frames on your map depending upon how you want to organize your data. For instance, one data frame might highlight a study area, and another might provide an overview of where the study area is located.

data integrity

Maintenance of data values according to data model and data type. For example, to maintain integrity, numeric columns will not accept character data.

data source

Any geographic data, such as a coverage, shapefile, raster, or feature class, in a geodatabase.

data type

The characteristic of columns and variables that defines what types of data values they can store. Examples include character, floating point, and integer.

data view

An all-purpose view in ArcMap for exploring, displaying, and querying geographic data. This view hides all map elements such as titles, North arrows, and scalebars. See also layout view.

database

A collection of related files organized for efficient retrieval of information. In the context of an ArcSDE geodatabase, some relational databases group data together in discretely named databases (for example, SQL Server, Sybase®), while others do not (for example, Oracle).

dataset

1. Any feature class, table, or collection of feature classes or tables in the geodatabase.
2. A named collection of logically related data items arranged in a prescribed manner.

decimal degrees

Degrees of latitude and longitude expressed as a decimal rather than in degrees, minutes, and seconds.

default junction type

Two edge types may be connectable through more than one junction type. You can establish which of those junction types is the default for connecting the two edge types. This junction type is the default junction type.

digitizing

1. To encode geographic features in digital form as x,y coordinates.
2. The process of converting the features on a paper map into digital format. When you digitize a map, you use a digitizing tablet, or digitizer, connected to your computer and trace over features with a digitizer puck, which is similar to a mouse. The x,y coordinates of these features are automatically recorded and stored as spatial data.

digitizing mode

Also called absolute mode, digitizing mode is one of the ways in which a digitizing tablet operates. In digitizing mode, the location of the tablet is mapped to a specific location on the screen. Moving the digitizer puck on the tablet surface causes the screen pointer to move to precisely the same position.

dimension construction methods

Dimension construction methods dictate what type of dimension feature is created and the number of points required to complete the feature's geometry. Construction methods include simple aligned, aligned, linear, rotated linear, free aligned, and free linear.

dimension feature

Dimension features are a special kind of map annotation that show specific lengths or distances on a map. A dimension feature may indicate the length of a side of a building or land parcel or it may indicate the distance between two features such as a fire hydrant and the corner of a building. Dimension features are stored in a dimension feature class.

dimension feature class

In the geodatabase, dimension features are stored in dimension feature classes. Like other feature classes in the geodatabase, all features in a dimension feature class have a geographic location and attributes and can either be inside or outside of a feature dataset.

dimension style

A dimension feature's style describes its symbology, what parts of it are drawn, and how it is drawn. Every time you create a new dimension feature, it is assigned a particular style. A collection of dimension styles is associated with a dimension feature class.

Dimensioning toolbar

A toolbar in ArcMap that facilitates the creation of dimension features.

direct connect

A two-tiered architecture for connecting to spatial databases. Direct connect does not require the ArcSDE application server to connect to a spatial database.

dirty areas

Areas that have been edited after the initial topology validation process. Dirty areas represent regions surrounding features that have been altered and that require an additional topology validation to be performed in order to discover any topology errors that may be present.

disconnected editing

The process of copying data to another geodatabase, editing that data, then merging the changes with the data in the source or master geodatabase.

distance units

The units—for example, feet, miles, meters, or kilometers—ArcMap uses to report measurements, dimensions of shapes, distance tolerances, and offsets.

double precision

Refers to a high level of coordinate accuracy based on the possible number of significant digits that can be stored for each coordinate. ArcInfo datasets can be stored in either single- or double-precision coordinates. Double-precision coverages store up to 15 significant digits per coordinate (typically, 13 to 14 significant digits), retaining the accuracy of much less than one meter at a global extent. See also single precision.

edge (topology)

A line segment in a topology that defines lines or polygon boundaries. Multiple features in one or more feature classes may share topology edges.

edge element

See logical network.

edge–edge rule

A connectivity rule that establishes that an edge of type A may connect to an edge of type B through a junction of type C. Edge–edge rules always involve a junction type.

edge–junction cardinality

A rule may exist that allows an edge of type A to connect to a junction of type B. By default, any number of edges of type A can connect to a single junction of type B. You may want to restrict this. You can specify that between two and five edges of type A can connect to a junction of type B; if there are less than two edges, or more than five edges, the connectivity rule is being violated. Similarly, you can restrict the number of junctions of type C that can connect to any junction of type D. This range of permissible connections is edge–junction cardinality.

edge–junction rule

A connectivity rule that establishes that an edge of type A may connect to a junction of type B.

edit cache

A setting used in spatial data editing in ArcMap that causes the features visible in the current map extent to be held in memory on your local machine. Designed to be used when working with large amounts of data, an edit cache results in faster editing because ArcMap doesn’t have to retrieve the data from the server.

edit session

In ArcMap editing takes place within an edit session. An edit session begins when you choose Start Editing from the Editor menu and ends when you choose Stop Editing.

Editor toolbar

A toolbar that lets you create and modify features and their attributes in ArcMap.

ellipse

A geometric shape equivalent to a circle that is viewed obliquely; a flattened circle.

error

Violations of a topology rule detected during the topology validation process.

extent

The coordinates defining the minimum bounding rectangle (that is, xmin, ymin and xmax, ymax) of a data source. All coordinates for the data source fall within this boundary.

feature

- 1. An object class in a geodatabase that has a field of type geometry. Features are stored in feature classes.
- 2. A representation of a real-world object.
- 3. A point, line, or polygon in a coverage or shapefile.
- 4. A representation of a real-world object in a layer on a map.

feature attribute table

A table used to store attribute information for a specific coverage feature class. ArcInfo maintains the first several items of these tables. Feature attribute tables supported for coverages include the following:

<cover>.PAT	for polygons or points
<cover>.AAT	for arcs
<cover>.NAT	for nodes
<cover>.RAT	for routes
<cover>.SEC	for sections
<cover>.PAT	for regions
<cover>.TAT	for annotation (text)

where <cover> is the coverage name.

feature class

1. The conceptual representation of a category of geographic features. When referring to geographic features, feature classes include point, line, area, and annotation. In a geodatabase, an object class that stores features and has a geometry field type.
2. A classification describing the format of geographic features and supporting data in a coverage. Coverage feature classes for representing geographic features include point, arc, node, route-system, route, section, polygon, and region. One or more coverage features are used to model geographic features (for example, arcs and nodes can be used to model linear features such as street centerlines). The tic, annotation, link, and boundary feature classes provide supporting data for coverage data management and viewing.
3. The collection of all the point, line, or polygon features or annotation in a CAD dataset.
4. In a geodatabase, an object class that stores features and has a field of type geometry.

feature dataset

In geodatabases, a collection of feature classes that share the same spatial reference. Because the feature classes share the same spatial reference, they can participate in topological relationships with each other such as in a geometric network, linear network, or topology. Several feature classes with the same geometry may be stored in the same feature dataset. Object classes and relationship classes can also be stored in a feature dataset.

field

A column in a table. Each field contains the values for a single attribute.

fuzzy tolerance

An extremely small distance used to resolve inexact intersection locations due to the limited arithmetic precision of computers. It defines the resolution of a coverage resulting from the Clean operation or a topological overlay operation such as Union, Intersect, or Clip.

In geodatabase feature classes, this concept is replaced by cluster tolerance.

geocoding

The process of creating geometric representations for locations (such as point features) from descriptions of locations (such as addresses).

geocoding index

An index on geocoding reference data used by geocoding services.

geocoding reference data

Data that a geocoding service uses to determine the geometric representations for locations.

geocoding service

An object that defines a process for creating geometric representations for locations (such as point features) from descriptions of locations (such as addresses).

geodatabase

A geographic database that is hosted inside a relational database management system that provides services for managing geographic data. These services include validation rules, relationships, and topological associations.

geodatabase data model

Geographic data model that represents geographic features as objects in an object-relational database. Features are stored as rows in a table; geometry is stored in a shape field. Supports sophisticated modeling of real-world features. Objects may have custom behavior. Compare to the georelational data model.

geometric network

A geometric network can be thought of as a one-dimensional nonplanar graph, or logical network, that is composed of features. These features are constrained to exist within the network and can, therefore, be considered network features. ArcInfo will automatically maintain the explicit topological relationships between network features in a geometric network.

georelational data model

A geographic data model that represents geographic features as an interrelated set of spatial and descriptive data. The georelational model is the fundamental data model used in coverages. Compare to the geodatabase data model.

index

A data structure used in a database to improve query performance. Feature classes also have spatial indexes that improve spatial query performance.

instance

The name of the process running on the ArcSDE server that allows connections and access to spatial data.

intersect

The topological integration of two spatial datasets that preserves features that fall within the area common to both input datasets. See also union.

IP address

The server's address on the network. The address consists of four numbers, each separated by a ".".

item

1. A column of information in an INFO table.
2. An element in the Catalog tree. The Catalog tree can contain both geographic data sources and nongeographic elements such as folders, folder connections, and file types.

junction element

See logical network.

label point

A feature class in a coverage used to represent point features and identify polygons.

layer

1. A collection of similar geographic features—such as rivers, lakes, counties, or cities—of a particular area or place for display on a map. A layer references geographic data stored in a data source, such as a coverage, and defines how to display it. You can create and manage layers as you would any other type of data in your database.
2. A feature class in a shared geodatabase managed with SDE 3.

layout view

The view for laying out your map in ArcMap. Layout view shows the virtual page upon which you place and arrange geographic data and map elements—such as titles, legends, and scalebars—for printing. See also data view.

left–right topology

A topological data structure used to represent contiguity between polygons in the coverage data model. Left–right topology supports analysis functions, such as adjacency. See also topology.

linear dimension

A dimension whose length doesn't represent the true distance between the beginning and end dimension points. Linear dimensions can be vertical, horizontal, or rotated. A vertical dimension's line represents the vertical distance between the beginning and end dimension points. A horizontal linear dimension's line represents the horizontal distance between the begin and end dimension points. A rotated linear dimension is a dimension whose line is at some angle to the baseline and whose length represents the length of the dimension line itself, not the baseline.

logical network

A logical network is an abstract representation of a network. A logical network consists of edge, junction, and turn elements and the connectivity between them. You can ask a logical network which elements are connected but you cannot ask it for the geometry of these elements. A logical network does not contain any coordinate data, so you cannot ask it for the location of its elements. For this, you need a geometric network. In a logical network, an edge element is connected to two junction elements (a from-junction and a to-junction), and a junction can have zero or more edges connected to it. A turn has a from-edge, a junction, and a to-edge. Each element can also have many weights associated with it. Weights are typically used to describe the cost to traverse an edge or turn or the cost to pass through a junction.

map

1. A graphical presentation of geographic information. It contains geographic data and other elements, such as a title, North arrow, legend, and scalebar. You can interactively display and query the geographic data on a map and also prepare a printable map by arranging the map elements around the data in a visually pleasing manner.
2. The document used in ArcMap that lets you display and work with geographic data. A map contains one or more layers of geographic data and various supporting map elements, such as scalebars. Layers on a map are contained in data frames. A data frame has properties, such as scale, projection, and extent, and also graphic properties such as where it is located on a map's page. Some maps have one data frame while other more advanced maps may have several data frames.

map document

In ArcMap, the disk-based representation of a map. Map documents can be printed or embedded into other documents. Map documents have an .mxd file extension.

map topology

A temporary set of topological relationships between coincident parts of simple features on a map, used to edit shared parts of multiple features.

map units

The units—for example, feet, miles, meters, or kilometers—in which the coordinates of spatial data are stored.

master check-out version

A version in the master geodatabase, created when data is checked out, that represents the state of the data at the time it was checked out.

master geodatabase

An ArcSDE geodatabase from which data has been checked out.

merge policy

In geodatabases, all attribute domains have a merge policy associated with them. When two features are merged into a single feature in ArcMap, the merge policies dictate what happens to the value of the attribute to which the domain is associated. Standard merge policies are default value, sum, and weighted average.

minimum bounding rectangle

A rectangle, oriented to the x- and y-axes, that bounds a geographic feature or a geographic dataset. It is specified by two coordinates: xmin, ymin and xmax, ymax. For example, the extent defines a minimum bounding rectangle for a coverage.

multipart feature

A feature that is composed of more than one physical part but only references one set of attributes in the database. For example, in a layer of states, the state of Hawaii could be considered a multipart feature. Although composed of many islands, it would be recorded in the database as one feature.

multipoint feature

A feature that consists of more than one point but only references one set of attributes in the database. For example, a system of oil wells might be considered a multipoint feature, as there is a single set of attributes for the main well and multiple well holes.

multiuser geodatabase

A geodatabase in an RDBMS served to client applications (for example, ArcMap) by ArcSDE. Multiuser geodatabases can be very large and support multiple concurrent editors. They are supported on a variety of commercial RDBMSs, including Oracle, Microsoft SQL Server, IBM DB2, and Informix.

network trace

In the most generic sense, a network trace means to navigate through the network following the connectivity of the network for some purpose.

node

An endpoint of a topology edge. Topology nodes may also be introduced along an edge during editing.

null value

The absence of a value. A geographic feature for which there is no associated attribute information.

object

The representation of a real-world entity stored in a geodatabase. An object has properties and behavior.

object class

A collection of objects in the geodatabase that have the same behavior and the same set of attributes. All objects in the geodatabase are stored in object classes.

overshoot

That portion of an arc digitized past its intersection with another arc. See also dangling arc.

pan

To move the viewing window up, down, or sideways to display areas in a geographic dataset that, at the current viewing scale, lie outside the viewing window.

password

The password used for authentication when you log on to an ArcSDE geodatabase.

personal geodatabase

A geodatabase, usually on the same network as the client application (for example, ArcMap), that supports one editor at a time. Personal geodatabases are managed in a Microsoft Jet Engine database. Personal geodatabases can contain geometric networks, linear networks, and topologies and can be used as checkout databases for disconnected editing of geographic data in the field.

planarize

The process of creating multiple line features by splitting longer features at the places where they intersect other line features. This process can be useful when you have nontopological linework that has been “spaghetti” digitized or imported from a computer-aided design (CAD) drawing.

point

A single x,y coordinate that represents a single geographic feature such as a telephone pole.

point mode digitizing

One of two methods of digitizing features using the ArcMap Editor’s Sketch tool or from a paper map using a digitizer. With point mode digitizing, you can create or edit features by digitizing a series of precise points, or vertices. Point mode digitizing is effective when precise digitizing is required—for example, when digitizing a perfectly straight line. See also stream mode digitizing.

polygon

A two-dimensional feature representing an area such as a state or county.

polygon–arc topology

PAT. A coverage polygon is made up of arcs, which define the boundary, and a label point, which links the polygon feature to an

attribute record in the coverage PAT. In the coverage data model, polygons are represented topologically as a list of arcs and a label that make up each polygon. In the geodatabase data model, polygons are stored as simple features, which may participate in topological relationships with other features.

port number

The TCP/IP port number that an ArcSDE geodatabase instance is communicating on.

post

Posting is the process of applying the current edit session to the reconciled target version.

precision (dataset)

The number of system units per one unit of measure in the X,Y domain of the coordinate system dataset. Precision defines the smallest storable distance between coordinates in the dataset. A spatial reference with a precision of one will store integer values, while a precision of 1,000 will store three decimal places.

preliminary topology

In coverages, refers to incomplete region or polygon topology. Region topology defines region–arc and region–polygon relationships. A topological region has both the region–arc relationship and the region–polygon relationship. A preliminary region has the region–arc relationship but not the region–polygon relationship. In other words, preliminary regions have no polygon topology. Polygon topology defines polygon–arc–label point relationships. A preliminary polygon has the polygon–label point relationship but not the polygon–arc relationship. Coverages with preliminary topology have red in their icons in the Catalog.

projection

A mathematical formula that transforms feature locations from the earth's curved surface to a map's flat surface. A projected coordinate system employs a projection to transform locations expressed as latitude and longitude values to x,y coordinates. Projections cause distortions in one or more of these spatial properties: distance, area, shape, and direction.

property

An attribute of an object defining one of its characteristics or an aspect of its behavior. For example, the Visible property affects whether a control can be seen at run time. You can set a data source's properties using its Properties dialog box.

pseudonode

A node connecting only two edges or a logical split defined in the topology cache while editing. Pseudonodes of the latter sort become a vertex after editing.

pull check-in

A check-in operation initiated from a master geodatabase.

push check-in

A check-in operation initiated from a check-out geodatabase.

pyramids

In raster datasets, reduced resolution layers, or pyramids, record the original data in decreasing levels of resolution. The coarsest level of resolution is used to quickly draw the entire dataset. As you zoom in, layers with finer resolutions are drawn; performance is maintained because you're drawing successively smaller areas.

query

A question or request used for selecting features. A query often appears in the form of a statement or logical expression. In ArcMap, a query contains a field, an operator, and a value.

radius

The distance from the center to the outer edge of a circle or circular curve.

rank

A method of assigning an accuracy value to feature classes to avoid having vertices from a feature class collected with a high level of accuracy being snapped to vertices from a less accurate feature class. Vertices from higher ranking feature classes will not be moved when snapping with vertices with lower ranked feature classes. The highest rank is 1 and you can assign up to 50 different ranks.

raster

Represents any data source that uses a grid structure to store geographic information.

RDBMS

Relational database management system. A database management system with the ability to access data organized in tabular files that can be related to each other by a common field (item). An RDBMS has the capability to recombine the data items from different files, providing powerful tools for data usage. ArcSDE supports several commercial RDBMSs.

reconcile

Reconciling is the process of merging all modified datasets, feature classes, and tables in the current edit session and a second target version. All features and rows that do not conflict are merged into the edit session, replacing the current features or rows. Features that are modified in each version are conflicts and require further resolution via the conflict resolution dialog box.

record

1. In an attribute table, a single “row” of thematic descriptors. In SQL terms, a record is analogous to a tuple.
2. A logical unit of data in a file. For example, there is one record in the ARC file for each arc in a coverage.

reference data

Tables or feature classes containing address information that geocoding services use to find the locations of addresses.

relate

An operation that establishes a temporary connection between corresponding records in two tables using an item common to both (for example, key attributes). Each record in one table is connected to those records in the other table that share the same value for the common item. See also relational join.

relational join

The operation of relating and physically merging two attribute tables using their common item.

relationship

An association or link between two or more objects in a geodatabase. Relationships can exist between spatial objects (features in feature classes) or nonspatial objects (rows in a table), or between spatial and nonspatial objects.

relationship class

Objects in a real-world system often have particular associations with other objects in the database. These kinds of associations between objects in the geodatabase are called relationships. Relationships can exist between spatial objects (features in feature classes), nonspatial objects (rows in a table), or between spatial and nonspatial objects. While spatial objects are stored in the geodatabase in feature classes, and nonspatial objects are

stored in object classes, relationships are stored in relationship classes.

row

1. A record in an attribute table. The horizontal dimension of a table composed of a set of columns containing one data item each.
2. A horizontal group of cells in a raster.

rule (connectivity)

A constraint on the type of network features that may be connected to one another and the number of features of any particular type that can be connected to features of another type. There are two types of connectivity rules: edge–junction and edge–edge rules.

rule (topology)

An instruction to the geodatabase defining the permissible relationships of features within a given feature class or between features in two different feature classes. Topology rules are compared against the features in the feature class during the topology validation process and violations are marked as errors. After the topology validation process, the errors can be corrected by editing the feature class; occasionally, violations of a topology rule may represent acceptable conditions. In these cases, the errors can be marked as exceptions.

scanning

The process of capturing data in raster format with a device called a scanner. Some scanners also use software to convert raster data to vector data.

schema

The structure or design of a database.

schema-only check-out

A type of check-out that creates the schema of the data being checked out in the check-out geodatabase but does not copy any data.

segment

A line that connects vertices. For example, in a sketch of a building, a segment would represent one wall.

select

To choose from a number or group of features or records; to create a separate set or subset.

Selectable layers list

A list on the Selection toolbar that lets you choose from which layers you can select.

For example, suppose you wanted to select a large number of buildings by drawing a box around them but selected a parcel by mistake as you drew the selection box. To avoid this, you might uncheck the Parcels layer in the Selectable layers list so that parcels cannot be selected.

selected set

A subset of the features in a layer or records in a table. ArcMap provides several ways to select features and records graphically or according to their attribute values.

selection anchor

When editing in ArcMap, a small “x” located in the center of selected features. The selection anchor is used when you move features using snapping. It is the point on the feature or group of features that will be snapped to the snapping location. This is also the point around which your selection will rotate when you use the Rotate tool and around which your feature will scale when you use the Scale tool. You can reposition the selection anchor.

server

The computer where the ArcSDE geodatabase you want to access is located.

shape

The characteristic appearance or visible form of a geographic object. Geographic objects can be represented on a map using one of three basic shapes: points, lines, or polygons.

shapefile

A vector data storage format for storing the location, shape, and attributes of geographic features. A shapefile is stored in a set of related files and contains one feature class.

shared boundary

A segment or boundary common to two features. For example, in a parcel database, adjacent parcels will share a boundary. Another example might be a parcel that shares a boundary on one side with a river. The segment of the river that coincides with the parcel boundary would share the same coordinates as the parcel boundary.

shared vertex

A vertex common to multiple features. For example, in a parcel database, adjacent parcels will share a vertex at the common corner.

simple feature

A feature that implements ESRI Simple Feature. Simple features with shared geometry can be edited with a map topology using ArcView and higher licensed seats of ArcMap.

single precision

Refers to a level of coordinate accuracy based on the number of significant digits that can be stored for each coordinate. Single-precision numbers store up to seven significant digits for each coordinate, retaining a precision of ± 5 meters in an extent of 1,000,000 meters. ArcInfo datasets can be stored as either single- or double-precision coordinates. See also double precision.

sketch

When editing in ArcMap, a shape that represents a feature's geometry. Every existing feature on a map has an alternate form, a sketch. A sketch lets you see exactly how a feature is composed, with all vertices and segments of the feature visible. To modify a feature, you must modify its sketch. To create a feature, you must first create a sketch. You can only create line and polygon sketches, as points have neither vertices nor segments.

Sketches help complete the current task. For example, the Create New Feature task uses a sketch you create to make a new feature. The Extend/Trim Feature task uses a sketch you create to determine where the selected feature will be extended or trimmed. The Cut Polygon Feature task uses a sketch you create to determine where the polygon will be cut into two features.

sketch constraints

In ArcMap editing, the angle or length limitations you can place on segments you're creating. These commands are available on the Sketch tool context menu. For example, you can set a length constraint that specifies that the length of the segment you're creating will be 50 map units. At whatever angle you create that segment, its length will be constrained to 50 map units.

Angle constraints work in the same way. For example, you can set an angle constraint that specifies that the angle of the segment you're creating will be 45 degrees measured from another feature that already exists. At whatever length you create that segment, its angle will be constrained to 45 degrees.

sketch operations

In ArcMap, editing operations that are performed on an existing sketch. Examples are Insert Vertex, Delete Vertex, Flip, Trim, Delete Sketch, Finish Sketch, and Finish Part. All of these operations are available from the Sketch context menu, which is available when you right-click any part of a sketch using any editing tool.

snapping

The process of moving a feature to coincide exactly with coordinates of another feature within a specified snapping distance or tolerance.

snapping environment

Settings in ArcMap Editor's Snapping Environment window and Editing Options dialog box that help you establish exact locations in relation to other features. You determine the snapping environment by setting a snapping tolerance, snapping properties, and a snapping priority.

snapping priority

During ArcMap editing, the order in which snapping will occur by layer. You can set the snapping priority by dragging the layer names in the Snapping Environment window to new locations.

snapping properties

In ArcMap editing, a combination of a shape to snap to and a method for what part of the shape you will snap to. You can set your snapping properties to have a feature snap to a vertex, edge, or endpoint of features in a specific layer. For example, a layer snapping property might let you snap to the vertices of buildings. A more generic, sketch-specific snapping property might let you snap to the vertices of a sketch you're creating.

snapping tolerance

The distance within which the pointer or a feature will snap to another location during ArcMap editing.

If the location being snapped to (vertex, boundary, midpoint, or connection) is within the distance you set, the pointer will automatically snap. For example, if you want to snap a power line to a utility pole and the snapping tolerance is set to 25 pixels, whenever the power line comes within a 25-pixel range of the pole, it will automatically snap to it. Snapping tolerance can be measured using either map units or pixels.

spatial database

Any DBMS that contains spatial data.

spatial domain

Describes the range and precision of x,y coordinates and z- and m-values that can be stored in a feature dataset or feature class in a geodatabase.

spatial join

A type of spatial analysis in which the attributes of features in two different layers are joined together based on the relative locations of the features.

spatial reference

Describes both the projection and spatial domain extent for a feature dataset or feature class in a geodatabase.

split policy

All attribute domains have a split policy associated with them. When a feature is split into two new features in the ArcMap Editor, the split policies dictate what happens to the value of the attribute to which the domain is associated. Standard split policies are duplicate, default value, and geometry ratio.

SQL

Structured Query Language. A syntax for defining and manipulating data from a relational database. Developed by IBM in the 1970s, it has become an industry standard for query languages in most relational database management systems.

stream mode digitizing

One of the two methods of digitizing features from a paper map. Also known as streaming, stream mode digitizing provides an easy way to capture features when you don't require much precision—for example, to digitize rivers, streams, and contour lines. With stream mode, you create the first vertex of the feature and trace over the rest of the feature with the digitizer puck. You can also use digitize in stream mode with the ArcMap Sketch tool when editing “freehand”. See also point mode digitizing.

stream tolerance

The minimum distance the pointer must be moved from the last vertex before the next vertex will be created when using the Sketch tool in stream mode.

When streaming, vertices are automatically created at a defined interval as you move the mouse. For example, if the stream tolerance is set to 10 map units, you must move the pointer at least 10 map units before the next vertex will be created. If you move the pointer more than 10 map units, there may be more space between vertices, but there will always be a minimum interval of 10 map units. Stream tolerance is measured in map units.

subtypes

Although all objects in a feature class or object class must have the same behavior and attributes, not all objects have to share the same default values and validation rules. You can group features and objects into subtypes. Subtypes differentiate objects based on their rules.

symbol

A graphic pattern used to represent a feature. For example, line symbols represent arc features; marker symbols, points; shade symbols, polygons; and text symbols, annotation. Many characteristics define symbols including color, size, angle, and pattern.

symbolology

The criteria used to determine symbols for the features in a layer. A characteristic of a feature may influence the size, color, and shape of the symbol used.

synchronization version

A version created in a check-out ArcSDE geodatabase when a check-out is made to that geodatabase. This version is created as a child of the DEFAULT version and represents the state of the data at the time the check-out was created.

table

Information formatted in rows and columns. A set of data elements that has a horizontal dimension (rows) and a vertical dimension (columns) in an RDBMS. A table has a specified number of columns but can have any number of rows. See also attribute table.

table of contents

In ArcMap, lists all the data frames and layers on the map and shows what the features in each layer represent.

tabular data

Descriptive information that is stored in rows and columns and can be linked to map features.

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.

target layer

Used in ArcMap editing, a setting in the Target layer dropdown list that determines to which layer new features will be added. The target layer is set by clicking a layer in the Target layer dropdown list. For instance, if you set the target layer to Buildings, any features you create will be part of the Buildings layer. You must set the target layer whenever you're creating new features—whether you're creating them with the Sketch tool, by copying and pasting, or by buffering another feature.

tic

Registration of geographic control points for a coverage representing known locations on the earth's surface. Tics allow all coverage features to be recorded in a common coordinate system such as Universal Transverse Mercator (UTM). Tics are used to register map sheets when they are mounted on a digitizer and to transform the coordinates of a coverage, for example, from digitizer units (inches) to the appropriate values for a coordinate system (which are measured in meters for UTM).

tolerances

A coverage uses many processing tolerances (fuzzy, tic match, dangle length) and editing tolerances (weed, grain, edit distance, snap distance, and nodesnap distance). Stored in a TOL file, ArcInfo uses the values as defaults in many automation, editing, and processing operations. You can edit a coverage's tolerances using its Properties dialog box in ArcCatalog.

topological association

The spatial relationship between features that share geometry, such as boundaries and vertices. When you edit a boundary or vertex shared by two or more features using the topology tools in the ArcMap Editor, the shape of each of those features is updated.

topological feature

A feature that supports network connectivity that is established and maintained based on geometric coincidence.

topology

1. In geodatabases, relationships between connected features in a geometric network or shared borders between features in a topology.
2. In coverages, the spatial relationships between connecting or adjacent features (for example, arcs, nodes, polygons, and points). The topology of an arc includes its from- and to-nodes and its left and right polygons. Topological relationships are built from simple elements into complex elements: points (simplest elements), arcs (sets of connected points), areas (sets of connected arcs), and routes (sets of sections, which are arcs or portions of arcs). Redundant data (coordinates) is eliminated because an arc may represent a linear feature, part of the boundary of an area feature, or both.

topology rules

An instruction to the geodatabase defining the permissible relationships of features within a given feature class or between features in two different feature classes. Topology rules are compared against the features in the feature class during the topology validation process and violations are marked as errors. After the topology validation process, the errors can be corrected by editing the feature class; occasionally, violations of a topology rule may represent acceptable conditions. In these cases, the errors can be marked as exceptions.

tracing

The building of a set of network elements according to some procedure.

transaction

1. A group of atomic data operations that comprise a complete operational task such as inserting a row into a table.
2. A logical unit of work as defined by a user. Transactions can be data definition (create an object), data manipulation (update an object), or data read (select from an object).

true curve

See circular arc.

undershoot

An arc that does not extend far enough to intersect another arc. See also dangling arc.

union

A topological overlay of two polygonal spatial datasets that preserves features that fall within the spatial extent of either input dataset; that is, all features from both coverages are retained. See also intersect.

username

The identification used for authentication when you log in to an ArcSDE geodatabase.

validate (topology)

The process of comparing the topology rules against the features in the dataset. When you validate a topology, features that violate the rules are marked as error features. Topology validation is typically performed after the initial topology rules have been defined, after the feature classes have been modified, or if additional feature classes or rules have been added to the map topology.

validation rule

Validation rules can be applied to objects in the geodatabase to ensure that their state is consistent with the system that the database is modeling. The geodatabase supports attribute, connectivity, relationship, and custom validation rules.

version

A version is an alternative representation of the database that has an owner, a description, a permission (private, protected, or public), and a parent version. Versions are not affected by changes occurring in other versions of the database.

vertex

1. One of a set of ordered x,y coordinates that defines a line or polygon feature.
2. A point that joins two segments of a feature. For instance, a square building would have four vertices, one at each corner.

virtual page

The map page, as seen in layout view.

wizard

A tool that leads a user step by step through an unusually long, difficult, or complex task.

work flow

An organization's established processes for design, construction, and maintenance of facilities.

work order

One specific task that proceeds through each stage of an organization's work flow processes such as design, acceptance, and construction in the field.

workspace

A container of geographic data. This can be a folder that contains shapefiles, an ArcInfo workspace that contains coverages, a personal geodatabase, or an ArcSDE database connection.

Index

A

- Abstract classes 371
- Active data frame
 - defined 423
- Address data. *See* Geocoding: address data
- Address matching. *See* Geocoding
- Aggregation. *See* Relationship class: composite
- Alias
 - defined 423
 - described 20–22
 - feature class 20, 37, 228
 - field 19, 20, 29, 39
 - table 20, 30
- Aligned dimension
 - defined 423
- Alternate street names. *See* Geocoding services:
 - reference data: alternate street names
- AM/FM (Automated mapping/Facilities management) 205
- Analyze command 54, 99
- Angular unit 35
- Annotation
 - and ArcCatalog 239
 - and ArcMap 240
 - and disconnected editing 326. *See also* Disconnected Editing: annotation
 - class
 - coverage annotation feature class 247
 - creating 240–241
 - described 236
 - in ArcMap 245
 - managing 238
 - populating 240
 - standalone 236
 - storage 246
 - converting
 - and geometric networks 238
 - and versioning 238
 - Convert Coverage Annotation command 247
 - coverage 237, 247
 - coverage pseudo items 237
- Annotation (continued)
 - converting (continued)
 - labels 237, 245
 - creating 237–238, 240, 241, 245
 - defined 423
 - described 236–238
 - editing 241
 - feature-linked
 - converting from coverage 237
 - creating 242, 244
 - described 236
 - in ArcCatalog 239
 - in ArcMap 244
 - features 54, 236, 245
 - nonfeature-linked 236, 239, 240–241
 - performance 238
 - placement 239
 - symbolology 237
 - versioning 309–310
- Annotation feature class. *See* Annotation: class
- Appending data 58–60
- Application programming interface (API) 335
- Arc-node topology 115
 - defined 423
- ArcCatalog
 - creating schema 5
 - Customize dialog box 408
 - customizing 301, 408
 - disconnected editing 343, 344
 - getting help 15
 - schema locking 22
 - tree 10, 117
 - versioning 295, 296, 297
- ArcInfo UML Model
 - described 360
 - ESRI classes 360
 - ESRI interfaces 360
 - ESRI network 360
 - logical view 360
 - user features 360, 370
 - using 365

- ArcInfo workspace
 - defined 423
- ArcMap
 - annotation 240
 - commands 101
 - converting annotation 237, 245
 - Customize dialog box 247
 - customizing 101, 150, 247
 - default values 19
 - disconnected editing 322, 338, 343, 344
 - document 245
 - drawing tools 237
 - overflow window 244
 - relationships 181
 - Selectable layers list 438
 - table of contents 244, 245
 - versioning 295, 304–306, 307
- ArcSDE. *See also* SDE: connection
 - administration tools 60
 - analyzing data 99
 - and disconnected editing 321, 330, 345
 - connections 8
 - adding 11
 - testing 11
 - defined 423
 - described 1
 - for coverages 56
 - migrating data 60
 - registering data with the geodatabase 98
- ArcStorm 5, 56
- ArcToolbox
 - building geometric networks 211
 - data importing tools 49, 50
 - data management tools 226
- Area 21
- Area feature
 - in geodatabase
 - attributes 120
 - described 120
 - geometry 120
 - topology 120

- Association 181, 182
- Asynchronous
 - defined 424
- Attribute
 - annotation 247
 - creating 18
 - defined 424
 - fields and coverages 118
 - importing data 50
 - in UML 375
 - mentioned 5, 293, 309, 316
 - relationship classes
 - 182, 184, 190, 195, 236
 - rules 164
 - validation rule 175
 - validation rule mentioned 164
- Attribute domains
 - and topology 109
 - associating with a field 29, 39, 175
 - associating with a subtype
 - 176, 179, 358, 384
 - coded value domain
 - code description 165, 168, 173
 - codes 165, 168, 173, 380
 - creating 173
 - described 165
 - in ArcCatalog 168
 - in UML 358, 380
 - creating 171
 - defined 424
 - deleting 174
 - described 164–167
 - field type 168, 171
 - in UML 358, 370, 408, 409
 - mentioned 6, 19, 24, 98, 163, 229
 - merge policy
 - defining 172, 173
 - in UML 358, 379
 - modifying 174
 - properties 168, 174
 - range domain

- Attribute domains (continued)
 - range domain (continued)
 - creating 171–172
 - described 165–167
 - in UML 358
 - minimum and maximum value
 - 168, 172, 379
 - split policy
 - defining 172, 173
 - in UML 358, 379
 - mentioned 168
 - valid values 358, 379
- Attributes
 - dialog box
 - defined 424
 - tables
 - defined 424
- Automated mapping/Facilities management (AM/FM) 205
- Azimuth
 - defined 424

B

- Barrier 208
- Behavior 1, 164, 236, 310, 416
 - and topology 109
 - defined 424
- Binary numbers
 - described 23
- BLOB (binary large object)
 - described 25
- Buffers
 - defined 424

C

- CAD (computer-aided design)
 - CAD feature class
 - defined 424
 - field mapping (table) 53

- CAD (computer-aided design) (continued)
 - CAD feature class (continued)
 - loading 80
 - defined 424
- Cardinality 182–185
- CASE tools
 - Code Generation Wizard 360
 - data loading 55
 - described 353
 - in ArcCatalog 364
 - in ArcGIS 360
 - locked database icon 362
 - mentioned 2, 6
 - Model Explorer tree 366
 - UML Static Structure stencil 369
 - workspace diagram 369
 - model icon 362
 - modeling database structure 355
 - Schema Wizard
 - adding command to toolbar 408
 - attributed relationship classes 364
 - connecting to a repository 409, 411
 - connecting to an XMI file 409
 - deleted relationship classes 364
 - described 354
 - domains 363
 - existing geodatabase 361
 - feature class 362, 417
 - feature dataset 362, 417
 - field properties 363
 - generating schema 422
 - geometric network 364, 419
 - mentioned 360, 408
 - model name 362
 - object class properties 414
 - object model, selecting 412
 - relationship class 364, 420
 - setting feature dataset properties 413
 - spatial reference 369
 - subtypes 363
 - table properties 414
 - XMI file, selecting 410

- CASE tools (continued)
 - semantics checker 361, 406
 - report 407
 - running 406
 - spatial reference 417
 - tagged values
 - in Visio diagram 367
 - UML elements
 - matching 362
 - read only 362
 - Visio diagram 367
- Centroid
 - defined 424
- Check-out
 - defined 424
 - geodatabase
 - defined 425
 - version
 - defined 425
- Checking in
 - defined 425
- Checking out
 - defined 425
- Circle
 - defined 425
- Circular arc
 - defined 425
- Cluster tolerance
 - defined 425
- Clustering
 - defined 425
 - topology validation 111
- Coincident
 - defined 425
- Column
 - defined 425
- COM (Component Object Model)
 - 6, 22, 50, 60, 98
 - class 416
- Complex edge 206, 222

- Complex junction 207, 358
- Component Object Model (COM)
 - 6, 22, 50, 60, 98
- Composite relationships 240
 - and UML 377
- Compress 298
 - defined 425
- Computer-aided design (CAD)
 - defined 425
- Configuration keyword 28, 38, 75–76, 79, 414, 417, 418, 419
- Conflict 293, 307, 308–309, 312–313
 - and topology features
 - versioned geodatabase 160
 - defined 426
- Connection file 8
- Connectivity
 - and disconnected editing 333
 - defined 426
- Connectivity rules
 - and CASE tools 394
 - creating 232
 - default junctions 231
 - defined 426
 - described 231
 - edge–edge rule
 - creating 232
 - default junction 394, 428
 - described 231
 - in UML 360, 394
 - edge–junction rule
 - cardinality 233, 394
 - creating 233
 - defined 429
 - described 231
 - in UML 360, 394
 - in UML 360
 - mentioned 7, 164, 211, 234
- Constraints. *See also* Attribute domains
 - defined 426

- Construct features
 - defined 426
- Contiguity
 - defined 426
- Control points
 - defined 426
- Converting data 5, 100
- Coordinate
 - defined 426
- Coordinate system
 - and CASE tools 357
 - and feature datasets 31
 - and importing data 65
 - custom 18, 35
 - defined 426
 - defining 31, 35
 - described 18
 - geographic 35
 - projected 36
 - saving 32
- Copy/Paste geodatabase data 54, 87, 89, 91
- Coverage
 - annotation 54, 235
 - data mapping 51
 - data model 1
 - and point features 119
 - defined 427
 - items
 - mapping (table) 52
 - type 51
 - loading data 100
 - mentioned 2, 5–6, 49
 - migrating to a geodatabase 118
 - polygon features
 - topological relationships 120
 - tics 51
- Cracking. *See also* Topology: validation
 - defined 427
 - topology validation 111
- Current task
 - defined 427

- Custom
 - behavior
 - defined 427
 - in UML 399
 - feature 44
 - defined 427
 - object 3, 4, 6, 27, 30. *See also* Behavior
 - creating 6
 - defined 427

D

- Dangle
 - described 119
 - tolerance
 - defined 427
- Dangling arc
 - defined 427
- Data 293
 - and versioned database 294
 - converters 101
 - defined 427
 - dictionary 20
 - integrity
 - defined 428
 - loading 56–58
 - migration
 - to create topology 118
 - model
 - mentioned 322, 326
 - quality and topology 109
 - source
 - defined 428
 - type 23. *See also* Field: properties: data
 - type
 - (table) 24
 - BLOB 25
 - date 25
 - double 23
 - float 23
 - long integer 23

- Data (continued)
 - type (continued)
 - short integer 23
 - stored in a DBMS 25
 - text 24
- Data frames
 - defined 427
- Data types
 - defined 428
- Data view
 - defined 428
- Database 293
 - and versioning 294
 - defined 428
- Dataset
 - defined 428
- Date data type
 - described 25
- Datum 35
- DB2 1
- DBMS (database management system)
 - mentioned 2. *See also* ArcSDE
- Decimal degrees
 - defined 428
- Default junctions
 - junction type
 - defined 428
- Default values
 - and CASE tools 408
 - and importing data 53, 77
 - and topology 109
 - associating with a field 39, 175
 - associating with a subtype 176, 358, 384
 - described 19, 164
 - in UML 373
 - mentioned 98
- Destination class
 - and disconnected editing 335
- Digitizing
 - defined 428
 - mode
 - defined 428

- Dimension feature class
 - and ArcCatalog 253
 - creating 254, 254–255
 - by importing a style 257
 - with a custom style 256
 - defined 251, 429
 - Properties dialog box 253
- Dimension features
 - baseline 251
 - construction methods
 - defined 428
 - creating 249, 252
 - defined 429
 - discussed 250
 - Feature Class Properties dialog box 253
 - mentioned 249, 254
 - performance 252
 - types
 - aligned 250
 - horizontal linear 250
 - rotated linear 250
 - vertical linear 250
- Dimension styles
 - ArcCatalog 253
 - arrow
 - and text fit 252
 - display 252
 - begin symbol 252
 - creating 263–264
 - and managing 259
 - default style
 - mentioned 254
 - setting 267
 - defined 251, 429
 - deleting 268
 - dimension line
 - display 252
 - symbol 252
 - end symbol 252
 - extension line display 252
 - importing 257, 265

- Dimension styles (continued)
 - mentioned 249, 254
 - offset and overshoot 252
 - overriding 259
 - properties
 - lines and arrows 260
 - text 261
 - text and arrow fit 262
 - renaming 266
 - style ID 263
 - text display 252
- Dimensioning toolbar
 - defined 429
- Direct connect 8, 12, 13
 - defined 429
- Dirty areas
 - and topology 113
 - versioned geodatabase 152
 - defined 429
- Disconnected editing
 - and ArcMap 322
 - annotation 327
 - check in 343
 - pull model 332, 343
 - push model 332, 344
 - check in command 343
 - check in procedure
 - deletion of rows 335
 - described 335
 - geometric networks 333
 - nulling of foreign keys 335
 - related data 335
 - topology 334
- Check-In wizard 343
 - About Checking In Data 343
 - progress dialog box 343
 - Reconcile and Post box 343
- check-out
 - customizing 340
 - default behavior 326
 - limitations 321
 - manager 346

- Disconnected editing (continued)
 - check-out (continued)
 - preparing 322
 - renaming 346
 - reviewing properties 346
 - schema-only 339, 340
 - spatial extent 338, 340
 - template 339
 - unregistering 346
 - check-out procedure 321
 - amount to check out 322
 - and ArcMap 338
 - automated processes 322, 326
 - behavior 338
 - dependent datasets 338
 - described 325, 330
 - included information 330
 - related data 326, 330
 - saving new map documents 342
 - topologies 325
 - check-out version 330
 - Check-Out wizard 338
 - About Checking Out Data button 339
 - advanced check-out options 339, 340
 - post check-out option 341
 - related data check box 340
 - reusing schema 339
 - summary report 341
 - data filters
 - definition queries 322
 - intersection of data filters 322
 - selections 322
 - spatial extent 322, 326, 340
 - defined 429
 - delta databases
 - described 332
 - described 320
 - destination object 326
 - dirty areas 162
 - extracting data. *See* Extracting data
 - functionality 320
 - restrictions 331

- Disconnected editing (continued)
 - geodatabase property dialog box 346
 - managing check-outs 331, 346
 - exporting changes to a delta geodatabase 350
 - in a check-out geodatabase 349
 - in a master geodatabase 346
- master geodatabase 320
 - check in models 332
 - independence from check-out geodatabase 331, 332
 - mentioned 330
- mentioned 90
- reconcile. *See* Reconcile: and disconnected editing
- synchronization version 330
- topology errors 162
- topology validation 162

- Distance units
 - defined 429

- Domain
 - and CASE tools 363
 - creating in UML 379
 - domain description 171
 - Domain properties dialog box 169
 - name 171
 - type 168, 172, 173

- Double precision
 - defined 429
 - described 23

- Drawing exchange format (DXF) 49
- DXF (drawing exchange format) 49

E

- Edge. *See also* Topology: edge
 - described 115
- Edge element 429. *See also* Logical network
 - defined 429
- Edge feature class
 - creating 228–229
 - mentioned 232

- Edge features. *See* Network features

- Edge–Edge rule
 - defined 429
 - in UML 396

- Edge–Junction rule
 - in UML 394

- Edge–Junction cardinality
 - defined 430

- Edge–Junction rule
 - defined 430

- Edit cache
 - and geometric networks 209
 - defined 430

- Edit session. *See also* Editing; Editing in ArcMap
 - and versioning 306, 307, 308, 312, 316
 - defined 430
 - mentioned 55, 94, 100

- Edit sessions
 - and disconnected editing 339

- Editing 305

- Editing in ArcMap
 - and subtypes 164
 - and versioning 307
 - editing relationships
 - described 203
 - loading objects 102
 - snapping environment 105
 - Target layer 102

- Editor toolbar
 - defined 430

- Ellipse
 - defined 430

- Embedded foreign key 182. *See also* Key field

- Error
 - defined 430

- Error features
 - and topology
 - versioned geodatabase 155

- Error inspector
 - mentioned 114

- Errors and exceptions
 - in topology 114
 - report 114

- ESRI Annotation Feature. *See* Feature type

- ESRI Complex Edge Feature 228

- ESRI Dimension Feature 254

- ESRI Simple Edge Feature 228

- ESRI Simple Junction Feature 230

- ESRI simple row 50

- Exceptions
 - in topology

- and versioned geodatabase 155

- Exclusive lock. *See* Schema locking: exclusive lock

- Extent 338, 340
 - defined 430

- Extracting data 89
 - described 331
 - filters applied 89
 - selected features 91
 - wizard 91

F

- Feature. *See also* Feature class; Feature type; Object

- attribute table
 - defined 430

- data loading 94
 - defined 430

- described 1
 - mentioned

- 181, 235, 240, 293, 307, 316, 408

- selecting 187

- simple 234, 438

- splitting and merging 166

- Feature class

- adding to topology 140

- as a template 38

- attribute domains 175

- CASE tools 362, 408

- converting data 50

- Feature class (continued)
 - creating
 - described 18, 37–38
 - custom features 44
 - defined 431
 - described 18
 - geometric networks 206–209
 - importing data 68, 77, 80
 - in UML 357, 374
 - creating 370
 - labeling 245
 - loading data 94, 100
 - mentioned
 - 1, 5, 17, 103, 169, 174, 179, 304, 308
 - participating in topology 117
 - registering with the geodatabase 60, 98
 - relationship classes 181
 - removing from topology 140
 - simple 18, 53, 77, 208, 220
 - spatial reference 18, 41–43
 - standalone 18, 37, 50, 62, 77, 80
 - creating 41–43
 - subtypes 164
 - versioned geodatabase 296
- Feature dataset
 - annotation classes 236, 239
 - CASE tools 362
 - creating 369
 - converting data 50
 - creating 31
 - defined 431
 - described 18, 21
 - geometric networks 210, 211
 - importing data 62, 68, 77, 80
 - mentioned 5, 17, 37, 121, 216, 220, 228–229, 241, 255, 256, 258
 - modeling 21
 - relationship classes 186
 - topology 110, 117
 - versioned geodatabase 296

- Feature layers
 - maintained by a topology 113
- Feature type 228
 - custom 37, 228
 - ESRI Annotation Feature 236, 240, 242
 - ESRI Complex Edge Feature 228
 - ESRI Dimension Feature 254
 - ESRI Simple Edge Feature 228
 - ESRI Simple Feature 50, 54, 236
 - ESRI Simple Junction Feature 230
 - ESRI Simple Row 50
 - simple 18, 37, 220, 228
 - topological 18
- Feature-linked annotation. *See also* Annotation
 - disconnected editing 336
- Field
 - annotation 236, 240, 243
 - attribute domains 29, 174, 175
 - CASE tools 363
 - corrected names 65
 - creating 27
 - in UML 372
 - defined 431
 - deleting 28, 65, 74, 76, 79
 - described 19
 - importing data 65, 74, 75–76, 76, 79
 - in UML 415, 421
 - indexes 45
 - loading data 95
 - mapping 51
 - mentioned 17, 309
 - properties
 - and CASE tools 363
 - data type 27, 51, 164, 195, 370
 - default value 29
 - length 357, 415, 421
 - name 195
 - precision 19, 357, 415, 421
 - scale 19, 357, 415, 421
 - related 187, 203
 - required 20, 27, 38, 208, 229, 357
 - for annotation 241

- Field (continued)
 - required for versioning 296
 - type
 - choosing 19, 23
- Float
 - described 23
- Foreign key 195, 196, 359
- Fuzzy tolerance
 - defined 431

G

- Geocoded feature classes
 - attributes
 - score 271
 - standardized address 271
 - status 271
 - automatically maintaining 271
- Geocoding
 - address data
 - Single Field 290
 - StreetMap 290
 - U.S. addresses 290–291
 - ZIP 291
 - ZIP+4 291
 - address tables 270
 - required format 290
 - defined 431
 - output feature class 271
 - required reference data 270
- Geocoding index
 - automatically maintaining 282–283, 287–289
 - defined 431
 - described 282
 - manually maintaining 282, 284–286
- Geocoding reference data
 - defined 431
- Geocoding service
 - defined 431
- Geocoding Services
 - folder 279

- Geocoding services
 - client-side 270
 - creating 279–281
 - described 270
 - in ArcCatalog 272
 - in ArcMap 272–273
 - mentioned 269
 - Properties dialog box 272
 - reference data
 - alternate street names 270, 276–277, 280
 - place name aliases 270, 278, 280
 - preparing 274–278
 - server-side 270
 - styles
 - Single Field 274
 - StreetMap 274
 - US One Address 275
 - US One Range 275–276
 - US Streets 276
 - ZIP 277
 - ZIP+4 277
 - ZIP+4 Range 278
- Geocoding Services folder 272
- Geodatabase
 - and versions 293
 - creating 10
 - three strategies 354
 - data model 1, 163
 - defined 431
 - disconnected
 - editing 9
 - elements
 - matching with CASE tools 362
 - in ArcCatalog 8, 9
 - three ways to create 4–5
 - Upgrading 26
- Geodatabase data model
 - defined 432
- Geographic database 163. *See also*
 - Geodatabase

- Geometric network
 - adding feature classes 228
 - appending data 59–60
 - Build Geometric Network Wizard 220, 226
 - building 211, 214, 220
 - in ArcToolbox 226
 - CASE tools 357, 364, 392
 - connectivity rules 232
 - copying data 87
 - creating 211, 216
 - defined 21, 432
 - deleting 137, 234
 - described 206
 - editing 9
 - flow direction 207
 - in ArcCatalog 210
 - in UML 360
 - loading data 57–58
 - managing 137, 234
 - mentioned 3, 18–22, 98, 205
 - modeling 209
 - performance 209
 - properties 231, 232
 - renaming 137, 234
 - snapping models 212
 - sources and sinks 207–209, 223
 - mentioned 229
 - versioning 309, 311
- Geometric relationships 109
- Geometry
 - area and length 20
 - field 37, 40
 - importing data 50
 - mentioned 18, 235, 309
 - tracking properties 20
 - type
 - and CASE tools 417, 418
 - in UML 370
 - mapping (table) 51
 - setting 40

- Georelational data model
 - defined 432
- Graphics 245

H

- Help
 - finding answers to questions 15

I

- Importing data
 - CAD feature classes 80
 - coverages 50–61, 68
 - data mapping 51–53
 - described 50–61
 - geodatabase feature classes 53, 77
 - Import menu 63
 - in batch 63, 75, 77, 80
 - mentioned 5, 49
 - multiple feature classes 70
 - raster 81
 - shapefiles 50–61, 62
 - tables 50–61, 75

Index

- attribute
 - ascending 45
 - creating 45
 - deleting 46
 - described 45–46
 - unique 45
- defined 432
- Instance
 - defined 432
- Intersect
 - defined 432
- Invalid features 231
- IP address 11, 13
 - defined 432

Item
defined 432

J

Junction element
defined 432

Junction feature class. *See also* Geometric network
and connectivity rules 232, 233
and disconnected editing 333
and versioning 310
creating 228, 230
described 207

Junction features. *See* Network features

K

Key field 182, 190, 194, 311

L

Label point
defined 432
migrating into a geodatabase 118

Labels
converting to annotation 245
creating 237
expression 243
rules 237
using related objects 203

Layers
defined 432

Layout view
defined 432

Left–Right topology
defined 433

Line 20

Line feature
attributes
in geodatabase 119

Line feature (continued)
geometry
in geodatabase 119
topological relationships 119
topology
in geodatabase 119

Linear
unit 36

Linear dimension
defined 433

Load Objects command
adding to ArcMap 101
loading data 102

Loading data
described 55
example 55–61
Extract Data wizard 91
mentioned 5, 49
Object Loader 7, 55, 102
relationships 96
Simple Data Loader 7, 55, 94–95, 100

Local Area Network (LAN)
mentioned 319

Locked database icon
mentioned 362

Logical network 206. *See also* Geometric network
defined 433

Long integer
described 23

M

Map 244, 304. *See also* ArcMap
defined 433
document
and disconnected editing 342
LIBRARIAN 56

Map document
defined 433

Map topology
defined 433

Map units
defined 433

Master check-out version
defined 433

Master geodatabase
and topology 162
defined 434

Merge policy
default value 167
defined 434
described 166–167
geometry weighted 167
sum values 167

Microsoft Repository 414

Microsoft Visio
Drawing templates 365
creating a new diagram 366
mentioned 353, 361, 364
opening a file 365

Migrating data
to create topology 118

Minimum bounding rectangle
defined 434

Mobile users. *See* Disconnected editing

Model icon
mentioned 362

Multipart features
defined 434

Multipoint 21

Multipoint features
defined 434

Multiuser
database 22

Multiuser geodatabase
defined 434

Multiversions. *See* Version

N

Network classes

disconnected editing 325

Network connectivity 55, 205, 206, 220

- Network elements 208
- Network feature class
 - creating 220, 228
 - in UML 357, 360, 394
 - managing 234
- Network features. *See also* Geometric network
 - ancillary role 207
 - CASE tools 408
 - complex 1, 206
 - creating 21
 - described 206, 206–209
 - edges 206, 211, 220, 231
 - editing in ArcMap 100, 105
 - enabled and disabled 208
 - feature class. *See* Edge feature class
 - junctions 206, 211, 231, 358
 - mentioned 228, 234
 - simple 206
 - sources and sinks 228, 230
 - versioning 310
- Network trace
 - defined 434
- Network weights
 - associating with a field 229
 - creating 218, 223
 - described 208–209
 - mentioned 211
- Node 51
 - defined 434
 - described 115
 - topology 116
- Null value
 - defined 434
- Null values
 - and CASE tools 415, 421
 - in attribute domains 165
 - mentioned 19, 39

O

- Object
 - and relationships 18, 182, 188, 197
 - defined 434
 - described 1
 - invalid 164
 - mentioned 3, 5, 181
 - simple 1, 27
 - valid 164
- Object class 3, 18
 - defined 434
 - described 18
- Object Loader. *See also* Loading data
 - wizard
 - described 100
- Object Management Group (OMG)
 - mentioned 354
- Object model
 - designing 365
- Object-oriented 1, 6
- ObjectID 19, 28, 29, 38, 98
 - and UML 376
- Oracle 1, 8, 12
- Overshoot
 - defined 434

P

- Pan
 - defined 434
- Password
 - defined 434
- Perimeter 21
- Permissions
 - and disconnected editing 332
- Personal geodatabase
 - 10, 49, 64, 73, 76, 78, 169, 417, 418
 - defined 435
- Place name aliases. *See* Geocoding services:
 - reference data: place name aliases

- Planarize
 - defined 435
- Point 21, 51
 - defined 435
- Point events
 - and shared geometry 116
- Point feature
 - attributes
 - in geodatabase 119
 - used when creating polygons 118
 - geometry
 - in geodatabase 119
 - topology
 - in geodatabase 119
- Point mode digitizing
 - defined 435
- Polygon 20
 - arc topology
 - defined 435
 - creating from lines 148
 - defined 435
 - Feature Class From Lines tool 148
 - topology 115
- Port number
 - defined 435
- Post 307, 312–313, 316
 - and disconnected editing 332
 - and versioned geodatabase 294
 - defined 435
- Precision. *See* Field: properties: precision
 - defined 435
 - described 18
 - spatial domain 33
- Preliminary topology
 - defined 435
- Primary key 190, 194, 196, 359
- Prime meridian 35
- Privileges
 - and versioning
 - described 298, 307, 317
 - private 298

- Privileges (continued)
 - and versioning (continued)
 - protected 298
 - public 298
 - delete 48
 - described 48
 - granting 17, 48
 - insert 48
 - mentioned 165
 - revoking 17, 48
 - select 48
 - update 48
- Projecting data 50
- Projection 3, 36
 - defined 436
- Property
 - defined 436
- Pseudonode
 - defined 436
 - described 115, 119
- Pull check-in
 - defined 436
- Pull model. *See* Disconnected Editing: master geodatabase: check in models
- Push check-in
 - defined 436
- Push model. *See* Disconnected Editing: master geodatabase: check in models
- Pyramids
 - defined 436

Q

- Query 95, 104
 - defined 436
- Query Builder 96, 104

R

- Radius
 - defined 436

- Rank
 - defined 436
- Ranks
 - topology
 - described 110, 113
- Raster
 - data compression 61
 - defined 436
 - importing 60
 - mosaic 61
 - pyramids 60
 - tile size 61
- Rational Rose
 - mentioned 354
- RDBMS (relational database management system) 1, 20, 45, 51, 54. *See also* DBMS (database management system): mentioned. *See also* ArcSDE
- Reconcile 307, 312–313, 316, 436
 - and disconnected editing 332, 343
 - defined 436
- Record
 - defined 437
- Reference
 - scale 243, 255
 - for annotation 241
- Reference data
 - defined 437
- Region topology 116
- Registering data 50, 98
- Relate
 - defined 437
- Related objects. *See also* Relationship class and disconnected editing 335, 337, 342
- Relational database management system (RDBMS) 1, 20, 45, 51
 - defined 436
- Relational join
 - defined 437
- Relationship
 - defined 437

- Relationship class
 - annotation 239, 243, 245
 - attributed
 - and CASE tools 421
 - described 183–185
 - in UML 359
 - cardinality 195, 197, 359
 - CASE tools 364, 409, 416
 - composite 183–185, 195, 236, 359
 - creating
 - attributed 195
 - composite 192
 - in UML 374
 - simple 188
 - defined 437
 - deleting 199
 - described 22, 181, 182–185
 - destination class
 - 182, 184, 188, 192, 195, 196, 197, 236
 - disconnected editing 335
 - foreign key. *See* Foreign key
 - in ArcCatalog 186, 188
 - in ArcMap 187, 200, 203
 - in UML 359, 390
 - managing 199
 - mentioned 18, 22, 163
 - messaging 183, 189, 192, 359
 - modeling 184
 - origin class
 - 184, 188, 192, 195, 196, 197, 236, 359, 390
 - path labels
 - backward path label 189, 192
 - described 183–185
 - editing in ArcMap 200
 - forward path label 189, 192
 - performance 184
 - renaming 199
 - simple 183–185, 192, 195
 - versioning 308, 309–310
 - versus join and relate 187

- Relationship rules
 - cardinality 184, 197, 198
 - creating 197
 - creating in UML 390
 - described 184–185
 - in UML 359
 - mentioned 164, 186
- Relationships
 - and disconnected editing 326
 - and topology 109
 - and versioning 310
 - composite 240
 - creating 187
 - deleting 187
 - described 182–185
 - destination object 198, 236, 390
 - in UML 359
 - mentioned 1, 3, 7, 50–61, 98, 107, 181, 205
 - origin object 198, 236
 - related object 310
- Remote geodatabase. *See* Disconnected Editing; SDE: geodatabase
- Route topology 115
- Row 27, 94, 181, 182, 408
 - defined 437
- Rule
 - connectivity
 - defined 437
 - topology
 - defined 437
- Rules
 - and topology 112, 131
 - topology
 - mentioned 109

S

- Scale. *See* Field: properties: scale
- Scanning
 - defined 437

- Schema
 - creation 5
 - defined 437
 - design
 - using CASE tools 354
 - designing 3, 5, 15, 17, 165
 - geometric networks 221
 - in UML
 - described 354
 - generating 405
 - loading data 95
 - mentioned 2
 - modifying 22
 - updating 361
- Schema locking
 - and dimension styles 259
 - and geometric networks 137, 215, 234
 - and relationship classes 185
 - and subtypes 167
 - and topology 111
 - described 22
 - exclusive lock
 - 111, 137, 185, 215, 234, 259
 - described 22
 - shared lock 22, 111, 137, 185, 215, 234
- Schema Wizard. *See* CASE tools: Schema Wizard
- geodatabase elements 362
- Schema-only check-out
 - defined 438
- SDE. *See* ArcSDE
 - connection 8, 11, 12, 13, 62, 169
 - geodatabase
 - 8, 10, 28, 38, 49, 64, 73, 78, 414, 417, 418, 419
 - layers 60
 - server 11, 13
 - service 11
- SDE for coverages. *See* ArcSDE
- SDE server 438
- Segments
 - defined 438

- Select
 - defined 438
- Selectable layers list
 - defined 438
- Selected set
 - defined 438
- Selection 244
- Selection anchor
 - defined 438
- Server
 - defined 438
- Shape
 - defined 438
- Shape_Area 21
- Shape_Length 21
- Shapefile
 - as a template 18
 - data loading 94, 100
 - data mapping 52
 - defined 438
 - field mapping (table) 52, 53
 - mentioned 2, 5–6, 49, 137, 234
 - polygons
 - and topology rules 120
- Shared boundary
 - defined 438
- Shared vertex
 - defined 438
- Short integer
 - described 23
- Simple edge. *See also* Network features
 - and junction 206
- Simple feature
 - defined 438
- Single precision
 - defined 439
- Sketch
 - constraints
 - defined 439
 - defined 439
 - operations
 - defined 439

- Snapping
 - agent 440
 - defined 439
 - environment
 - defined 439
 - features 222
 - priority
 - defined 439
 - properties
 - defined 439
 - snap tolerance 222
 - tolerance
 - defined 440
- Snapping environment
 - defined 439
- Spatial database
 - defined 440
 - engine (SDE) 332
- Spatial domain
 - defined 440
 - described 18
- Spatial extent 338, 340. *See also* extent
- Spatial index
 - creating 45–46, 47
 - deleting 47
 - described 18–19
 - grid size
 - 40, 47, 50, 64, 73, 78, 357, 417, 418
- Spatial join
 - defined 440
- Spatial reference
 - and feature classes 37
 - and feature datasets 31
 - and importing data 50, 65
 - defined 440
 - defining 66, 72
 - described 18
 - importing 31–32
 - precision 18, 33
 - selecting 31–32
 - spatial domain 31, 32, 440

- Spatial relationships. *See* Topology: rules
- Split policy
 - default value 166
 - defined 440
 - described 166–167
 - duplicate 166
 - geometry ratio 166
- Splitting and merging features. *See* Merge policy; Split policy
- SQL (Structured Query Language)
 - defined 440
 - Server 8, 13
- Standardized address 271
- Stereotyped package
 - and UML 369
- Stream mode digitizing
 - defined 440
- Stream tolerance
 - defined 440
- Structured Query Language (SQL)
 - defined 440
- Subtypes
 - attribute domains 174, 175
 - CASE tools 363, 408, 416
 - connectivity rules 231, 232, 394
 - creating 176
 - in UML 384
 - data copying 87
 - data importing 53, 77
 - data loading 95
 - default subtype 177
 - defined 440
 - deleting 179
 - described 164–167
 - field
 - creating in UML 384
 - in UML 358, 360
 - mentioned 3, 6, 18
 - modifying 176, 179
 - relationship rules 184, 197

- Subtypes (continued)
 - subtype
 - code 164, 177, 384
 - description 177, 179
 - field 95, 102, 164, 176, 384
 - topology 109
 - rules 136
- Symbol
 - defined 441
- Symbology 187, 236, 239
 - defined 441
 - using related objects 203
- Synchronization version
 - defined 441

T

- Table
 - attribute domains 174, 175
 - CASE tools 408
 - creating 27–28
 - custom objects 30
 - from template 28
 - simple objects 27
 - dBASE 5–6, 49, 53, 75, 94, 100
 - defined 441
 - designer 27
 - in UML 137, 234, 357
 - INFO 5–6, 49, 75, 94, 100
 - mentioned 5, 17, 18, 181, 304
 - relationship classes 182, 182–185, 186, 188, 189, 195, 374
 - subtypes 164, 169, 179
 - versioned geodatabase 296
 - wizard
 - mentioned 28
- Table of Contents
 - disconnected editing 339
- Table of contents
 - defined 441

- Tabular data
 - defined 441
- Tagged values
 - defined 441
- Target geodatabase
 - mentioned 361
- Target layer
 - defined 441
- TCP/IP 11
- Text data type
 - described 24
- Tic
 - defined 441
- Tolerances
 - defined 441
- Topological association
 - defined 442
- Topological feature
 - defined 442
- Topology
 - adding feature classes 121
 - described 126
 - and ArcCatalog 117
 - and ArcSDE 125
 - arc-node 115
 - basics 112
 - building 110
 - resources needed 110
 - cluster tolerance
 - changing 122, 139
 - described 110, 112, 122
 - ranks 110, 113, 123, 141
 - Z ranks 123
 - conflicts
 - and versioned geodatabase 160
 - creating 110, 121
 - assigning a name 122
 - in a versioned geodatabase 150
 - mentioned 109
 - setting cluster tolerance 122
 - dangles
 - described 119

- Topology (continued)
 - defined 442
 - described 21, 107, 109
 - dirty areas
 - and versioned geodatabase 152
 - disconnected editing 162, 325, 334
 - edge
 - defined 429
 - error features
 - and versioned geodatabase 155
 - error summary 147
 - exceptions
 - and versioned geodatabase 155
 - feature dataset 21
 - feature geometry 115
 - geometries involved 115
 - geometry
 - edge 115
 - node 115
 - pseudo-node 115, 119
 - how to use 109
 - integrated features 307
 - maintained feature layers 113
 - managing 117
 - mentioned 18, 107
 - migrating data 118
 - area features 120
 - coverages 118
 - line features 119
 - point features 118
 - modifying 138
 - adding a feature class 140
 - adding a rule 142
 - changing cluster tolerance 139
 - changing number of ranks 141
 - changing rank of feature class 141
 - getting properties 138
 - removing a feature class 140
 - removing a rule 143
 - renaming 138
 - network topology 137, 206, 211, 234. *See also* Network connectivity

- Topology (continued)
 - New Topology wizard 121
 - node 116
 - permissions 150
 - point and line features 116
 - point events 116
 - polygon 115
 - region 116
 - route 115
 - rules 112, 123, 131
 - assigning a feature class 124
 - coverage arc features 119
 - errors and exceptions 114
 - exceptions 135
 - getting a description 143
 - line rules 132
 - loading a rule set 145
 - mentioned 5
 - point rules 134
 - polygon rules 131
 - polyline features 119
 - Rule Description panel 124
 - saving a rule set 124, 144
 - setting 123
 - sharing geometry 115–116
 - storage 107
 - subtypes 136
 - validation
 - clustering features 111
 - cracking features 111
 - dirty areas 113
 - mentioned 109, 117
 - process 130
 - validating a new topology 125
 - versioned databases 150
 - theory 152
 - view 117
 - wizard
 - mentioned 110, 117
- Topology rules. *See* Rule: topology
 - defined 442

- Tracing
 - defined 442
- Transaction 307, 316–317
 - defined 442
- Transportation networks 205
- True curve
 - defined 442

U

UML

- applying to existing databases 361
- associations
 - described 374
- attributed relationship classes
 - creating 377
- class
 - adding codes 383
 - attributes 367
 - creating an extension 402
 - creating an interface 399
 - mentioned 367
- elements
 - fields 363
 - mentioned 355
- feature datasets 369
- mentioned 2
- nonattributed relationship classes
 - creating 374
 - mentioned 364
- packages 360
 - creating 366
 - examples 366
- relationship class
 - creating 374
- static structure diagrams
 - creating 366
- tagged values 357, 367, 370, 415, 420
 - (table) 355
 - described 355, 367
 - domain 356
 - feature class/object class 355

UML (continued)

- tagged values (continued)
 - fields 355
 - geometric network 356
 - recognized elements 367
 - relationship class 356
 - setting 367
- using existing data 361
- UML model
 - abstract class 371
 - and CASE tools 409, 414
 - attribute domains
 - associating with a subtype 386
 - code value domain 382
 - codes 383
 - creating 379
 - description 383
 - field type 380, 382
 - maximum value 381
 - merge policy 380, 382
 - minimum value 381
 - range domain 379
 - split policy 380, 382
 - TemplateCodedValueDomain 382
 - TemplateRangeDomain 379
 - binary association 374, 390
 - checking for errors 406. *See also* CASE
 - tools: semantics checker
 - class 414
 - diagrams 369
 - extensions 399
 - connectivity rules
 - creating 394
 - default junction 397
 - junction subtypes 396
 - custom behavior 399
 - default values 373, 386
 - exporting to the repository
 - Microsoft Access 405
 - feature class 370
 - feature dataset 356
 - and spatial reference 357

UML model (continued)

- fields
 - creating 372
 - data type 373
 - described 370
- generalization 371
- generating schema 408
- in Visio Enterprise 366
- interfaces 399
- relationship class
 - attributed 375
 - cardinality 375
 - destination class 374
 - foreign key 374
 - origin class 374
 - primary key 374
 - stereotype 377
 - tagged values 374, 377
- relationship rules 390, 391
- setting tagged values 355, 367, 441
- subtypes
 - and relationship rules 390
 - associating with parent class 389
 - creating 384, 386
 - creating subtype field 384
 - default subtype 385
 - including a field 386
 - setting initial value 389
 - subtype field 387
- Undershoot
 - defined 442
- Unified Modeling Language (UML)
 - described 353
- Union
 - defined 442
- Upgrading
 - geodatabase 26
- Username
 - defined 442

- V**
- Validate
 - topology
 - defined 442
 - Validation rule
 - defined 443
 - Validation rules. *See also* Attribute domains; Connectivity rules; Relationship rules; Topology: validation
 - and importing data 50
 - and loading data 105
 - and topology 109
 - described 164
 - mentioned 3
 - Vector data 1
 - Version
 - administering 297
 - changing properties 300
 - compressing database 303, 317
 - deleting 299
 - renaming 299
 - changing in ArcMap 305
 - conflict. *See also* Conflict
 - displaying 314
 - feature-linked annotation 310
 - geometric networks 310
 - resolving 307–311, 315
 - connecting to 14
 - creating 297
 - DEFAULT 10
 - defined 443
 - described 293, 294–295
 - descriptions 299
 - disconnected editing 330
 - editing 307–311, 312
 - autoreconciliation 307–308, 312
 - post 308, 313
 - reconcile 307, 313
 - in ArcCatalog 296
 - in ArcMap 304–306
 - Version (continued)
 - permissions 298, 317
 - post. *See* Post
 - preserving edits 296
 - purpose 9
 - reconcile. *See also* Reconcile
 - and post a topology 150
 - refresh 300, 305, 306
 - registering data 296
 - relationships
 - conflicts 310
 - scenarios 316–317
 - topology 150
 - conflicts 311, 315
 - theory 152
 - transaction. *See* Transaction
 - unregistered database 150
 - Versioned data 55, 94, 100, 165, 199, 221
 - loading 102
 - Vertex
 - creating new
 - topology validation 111
 - defined 443
 - moving
 - topology validation 111
 - Virtual page
 - defined 443
 - Visio. *See* Microsoft Visio
- X**
- Workspace 304
 - defined 443
 - XML Metadata Interchange (XMI)
 - described 354
- W**
- Wide Area Network (WAN)
 - mentioned 319
 - Wizard
 - defined 443
 - Work flow. *See also* Version
 - common stages 294
 - defined 443
 - disconnected editing 319
 - process 294, 316
 - Work order. *See also* Work flow
 - defined 443