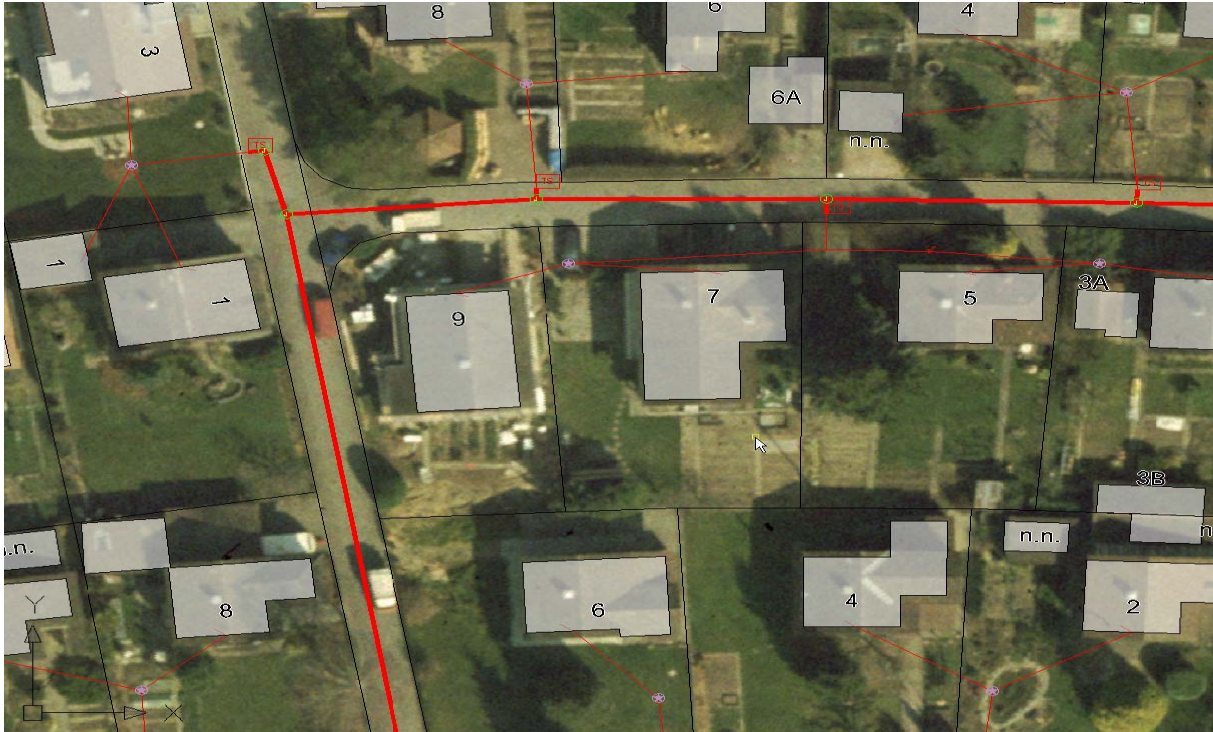


Taking a Closer Look at the Autodesk Topobase API



Autodesk® Topobase™ software includes powerful application programming interfaces (APIs) that can be used to build custom applications and application extensions and add them to the Topobase framework. This white paper covers a few standard steps that one might take in creating a custom Topobase application.

Contents

Introduction	3
Connecting to the Database	3
Creating a New Database User	4
Creating a Workspace.....	5
Creating Topics and Feature Classes.....	5
Reading and Writing Features.....	10
Summary	12
Resources	12
Autodesk Developer Network	12
Learn More	12

Introduction

Autodesk® Topobase™ software has two API components. The first is a Server API, written in Java® and PL/SQL™, which runs in Oracle® 10g database. This API supports data model editing. The second is the Framework API. Because the functionality of the Server API is exposed in the Framework API, the latter is regarded as the main Topobase API. The Server API is not normally used for application development.

The Framework API provides a wide range of functionality, including access to the database, full Oracle Spatial support, reading and writing of Topobase features, job handling, feature rules, and creation of forms and plug-ins. This paper examines the use of the API for connecting to the database, creating a new user, creating feature classes, and reading and writing features.

Connecting to the Database

Before you can use Topobase, you have to establish a connection to the database. The class `Topobase.Data.TBConnection` is used for establishing this connection. Figure 1 shows the inheritance hierarchy of `TBConnection`.



Figure 1. Inheritance hierarchy of the `TBConnection` class

To connect to the database, initialize a new instance of the `TBConnection` class via its constructor and invoke its `Open` method to open the connection. After successfully opening the connection, you can then proceed to carry out other tasks, such as creating feature classes and features. After completing all required tasks, close the connection by calling `TBConnection.Close`. Note that the connection should also be disposed of when it is no longer used. The following code snippet shows the procedure for connecting to Topobase:

```

// Declare a connection object.
Using(Topobase.Data.TBConnection myConnection =
    new Topobase.Data.TBConnection(
        Properties.Settings.Default.Oracle_User,
        Properties.Settings.Default.Oracle_Password,
        this.Application.Connection.DataSource,
        true,
        this.Application.Connection));
{
    // Open the connection to the database.
    // If the DB has no Topobase structure it will be created now.
    myConnection.ShowForms = true;
    myConnection.Open();

    MessageBox.Show("Connection State: " + myConnection.State.ToString());

    // Close connection.
    myConnection.Close();
}

```

Creating a New Database User

To create a new database user, use the `TBConnection.CreateTopobaseUser` method. This method takes as one of its parameters a `Connection` object that corresponds to the current user name and password.

This code snippet shows how to open a connection and create a database user:

```
private void btn_CreateTbUser_Click(object sender, EventArgs e)
{
    // Create a new empty Oracle user.
    if
(!this.Application.Connection.ExistsUser(Properties.Settings.Default.Oracle_User))
    {

        // The name of the new database is "TBDE401" with password "sys".
        // To change it, edit the project settings.
        Using (Topobase.Data.TBConnection myConnection =
            new Topobase.Data.TBConnection(
                Properties.Settings.Default.Oracle_User,
                Properties.Settings.Default.Oracle_Password,
                this.Application.Connection.DataSource,
                true,
                this.Application.Connection));{

            try
            {
                //(this.Application.Connection, "USERS", "TEMP", "INDX");
                myConnection.CreateTopobaseUser(
                    this.Application.Connection,
                    Properties.Settings.Default.Default_Tablespace,
                    Properties.Settings.Default.Temporary_Tablespace,
                    Properties.Settings.Default.Index_Tablespace);
            }
            catch (Topobase.Exception.TBException ex)
            {
                if (ex.Caller == Topobase.Exception.TBExceptionCaller.TB3Server)
                {
                    MessageBox.Show("Unable to create user:" + Environment.NewLine +
                        ex.Message);
                }
                else
                {
                    throw;
                }
            }
            finally
            {
                myConnection.Close();
            }
        }
    }
    else
    {
        MessageBox.Show("User already exists.");
    }
}
```

Creating a Workspace

A document must be assigned to a workspace to be accessible by Topobase Client or Topobase Web. It is important to create workspaces for later use. In this paper, you will create a workspace, TBDE401, manually.

1. Start Topobase Administrator.
2. Click Workspace>Create.
3. Enter **TBDE401** for the workspace name, and click Create.
4. Click Import.
5. Click Import From Existing Oracle schema.
6. Check the TBDE401 schema.

The Workspace Manager will look like Figure 2.

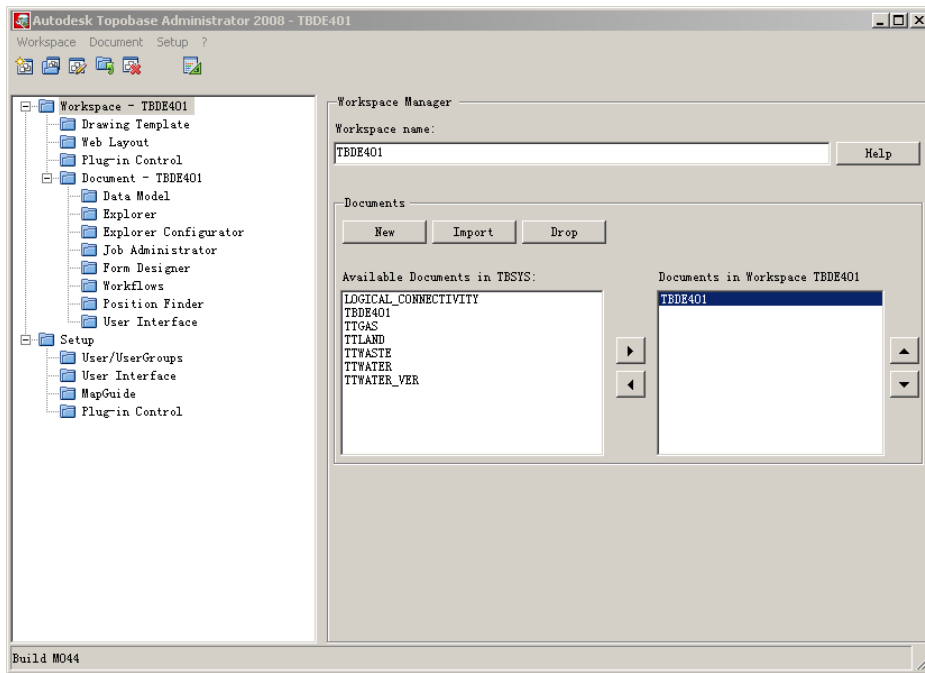


Figure 2. Workspace TBDE401

Creating Topics and Feature Classes

A *topic* is a collection of feature class tables. Think of topics as containers used to organize feature classes. Each topic may have subtopics.

To build a clear and transparent data structure, you can group feature classes into topics, group several topics into main topics, and define feature classes with subfeature classes.

A topic is represented in the API by Topobase.Data.Topic. Topobase.Data.Topic represents a collection of topic objects, each representing a TB_Topic row. To create a topic object, use the Topobase.Data.Topics.Add() method. The following code snippet shows how to add a new topic to Topobase:

TAKING A CLOSER LOOK AT THE AUTODESK TOPOBASE API

```
Topobase.Data.Topic MyTopic;  
MyTopic = myConnection.Topics.Add("MYTOPIC", "My Topic Caption");
```

To delete a topic, use the `Topobase.Data.Topics.Remove()` method, as shown here:

In Topobase, a feature class is the basic class for objects. For example, a parcel is a feature class. In a database, each feature class corresponds to one Oracle table.

```
if (myConnection.Topics.Contains("MYTOPIC"))  
{  
    myConnection.Topics.Remove("MYTOPIC");  
}
```

You can group several feature classes for each topic. Each feature class contains many entities, instances, or records, which are called *features*.

In the Topobase API, a feature class is represented by `Topobase.Data.FeatureClass`. Figure 3 shows the inheritance hierarchy for `Topobase.Data.FeatureClass`.

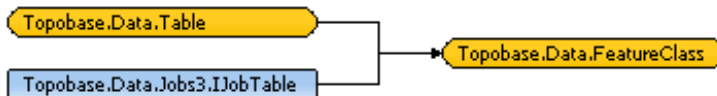


Figure 3. Inheritance hierarchy for `Topobase.Data.FeatureClass`

`Topobase.Data.FeatureClasses.Add()` is used for creating a `FeatureClass` instance, and `Data.FeatureClasses.Remove()`, for deleting a feature class, as shown here:

```
//Add a feature class  
Topobase.Data.FeatureClass MyPointFeatureClass;  
MyPointFeatureClass = myConnection.FeatureClasses.Add(MyTopic,  
    "MYPOINT", "My Point Caption",  
    Topobase.Data.FeatureClassType.Point);
```

```
//Remove a feature class  
if (myConnection.FeatureClasses.Contains("MYPOINT"))  
{  
    myConnection.FeatureClasses.Remove("MYPOINT");  
}
```

Each feature class consists of one feature table. This table has a fixed basic structure, but you can create additional columns (attributes). The basic structure is that every feature has an identifier FID (feature identifier) that is unique for each database schema (document).

A feature class can have any number of attributes (Oracle columns), one of which is the type "geometry." The following table shows the general types of feature classes.

Feature Class Type	Description
Attribute	For attribute tables (features without geometry).
Line String	For polylines.

TAKING A CLOSER LOOK AT THE AUTODESK TOPOBASE API

Polygon	For (closed) polygons; consists of arcs as well as lines or polylines.
Centroid	For centroids, a special point associated with a polygon that is defined by line strings. A centroid is surrounded by the features of a line string feature class that builds the polygon. Examples: parcels, land use.
Label	For text; Is related to a parent feature class and therefore can only be created from feature class level.
Compound Polygon	For areas; built as a group of a line string and a polygon feature class.
Compound Linestring	For polylines; a group of two line string feature classes.
Point	For points.

The `Topobase.Data.FeatureClass.Attributes` property returns all attributes from the database. The type is `Table.Attributes`. With this property, you can add some attributes to the feature class.

In this section, you create a topic named `MYTOPIC` and a point feature class named `MYPOINT`. The feature class is designed as shown in Figure 4.

Name	Caption	Unit	Data Type	Length/Prec.	Scale	Optional	Default
FID	FID		Number	10		False	
GEOM	GEOM		Geometry	1		True	
ORIENTATION	Orientation	gon clockwise	Number	6	3	False	100.0
QUALITY	Quality		Number	10		True	
Z	Z	meter	Number	20	8	True	
NAME	NAME		Varchar2	255		True	
ACTIVE	ACTIVE		Number	1		True	
CREATED	CREATED		Date	7		True	
MODIFIED	MODIFIED		Date	7		True	
COORDINATES	COORDINATES		Varchar2	255		True	

Figure 4. MyPoint feature class design

The code snippet for creating the topic and feature class is as follows:

```
private void btn_CreateTopicAndFeature_Click(object sender, EventArgs e)
{
    // Connect to a Topobase.
    Topobase.Data.TBConnection myConnection = new Topobase.Data.TBConnection(
        Properties.Settings.Default.Oracle_User,
        Properties.Settings.Default.Oracle_Password,
        this.Application.Connection.DataSource,
        true,
        this.Application.Connection);
}
```

TAKING A CLOSER LOOK AT THE AUTODESK TOPOBASE API

```
myConnection.ShowForms = true;
myConnection.Open();

// Add a new topic to the Topobase.
Topobase.Data.Topic MyTopic;
MyTopic = myConnection.Topics.Add("MYTOPIC", "MyTopic");

// Create a point feature class.
Topobase.Data.FeatureClass MyPointFeatureClass;
try
{
    MyPointFeatureClass = myConnection.FeatureClasses.Add(MyTopic, "MYPOINT",
" MyPoint ", Topobase.Data.FeatureClassType.Point);
    // Add some attributes to the feature class.
    if (!MyPointFeatureClass.Attributes.Contains("Name"))
    {
        MyPointFeatureClass.Attributes.Add("Name",
Topobase.Data.DataType.VarChar2,
        255);
    }
    if (!MyPointFeatureClass.Attributes.Contains("Active"))
    {
        MyPointFeatureClass.Attributes.Add("Active",
Topobase.Data.DataType.Number,
        1);
    }

    if (!MyPointFeatureClass.Attributes.Contains("Created"))
    {
        MyPointFeatureClass.Attributes.Add("Created",
Topobase.Data.DataType.Date);
    }

    if (!MyPointFeatureClass.Attributes.Contains("Modified"))
    {
        MyPointFeatureClass.Attributes.Add("Modified",
Topobase.Data.DataType.Date);
    }

    if (!MyPointFeatureClass.Attributes.Contains("Coordinates"))
    {
        MyPointFeatureClass.Attributes.Add("Coordinates",
        Topobase.Data.DataType.VarChar2, 255);
    }
}
catch (Topobase.Exception.TBException ex)
{
    MessageBox.Show("Unable to create feature class:" + Environment.NewLine +
ex.Message);
}

// Close connection.
myConnection.Close();
}
```

Now Open Workspace TBDE401 in the Topobase Administrator, select Data Model in the left pane, you will find the feature class MyPoint listed in the right pane.

TAKING A CLOSER LOOK AT THE AUTODESK TOPOBASE API

Select Explorer in the left pane, check MyPoint, and click Save Tree. This will set the features to be shown in the Feature Explorer, as shown in Figure 5.

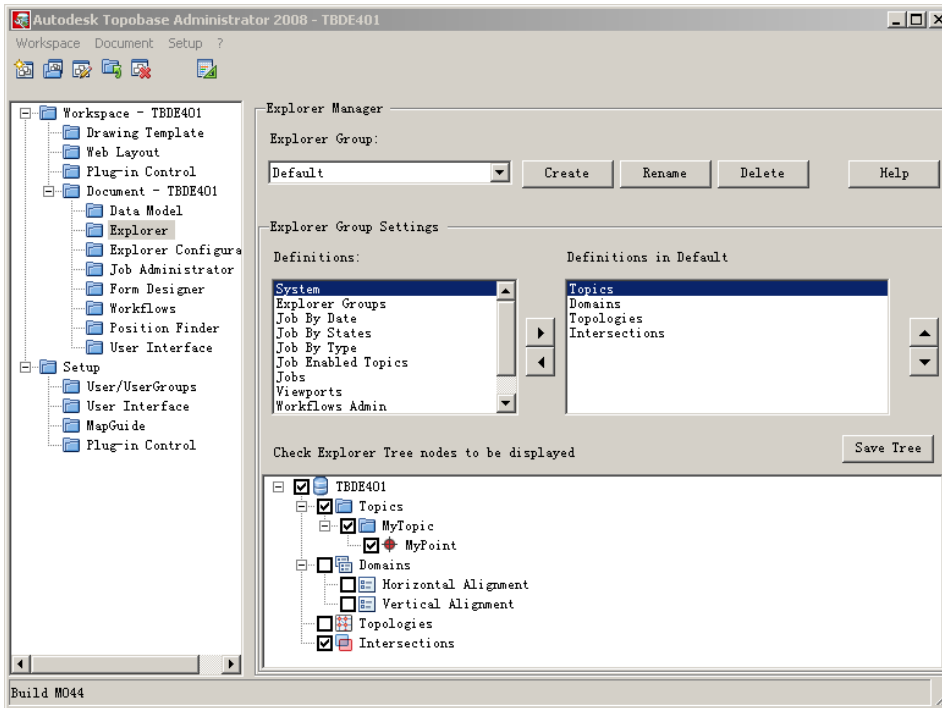


Figure 5. Setting the features to be shown in the Feature Explorer

Open Workspace TBDE401 in the Topobase Client. You will see MyPoint listed under MyTopic in the task pane, as shown in Figure 6.

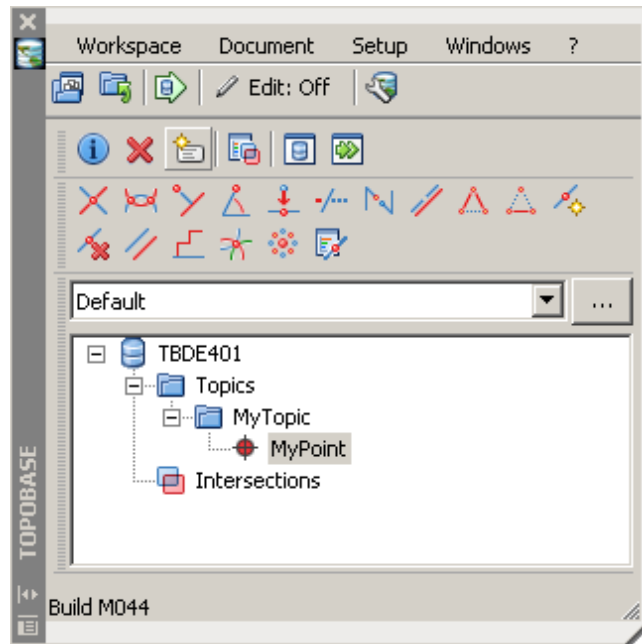


Figure 6. MyPoint in Topobase Client

Reading and Writing Features

A feature is an entity of a feature class. Each feature in a feature class represents a row or record in the feature class table.

A feature is represented by `Topobase.Data.Feature` in the API. The inheritance hierarchy is shown in Figure 7.

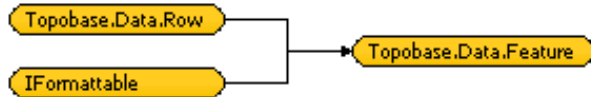


Figure 7. Inheritance hierarchy for feature

Call the `FeatureClass.CreateFeature` method to create an instance of the feature as shown here. This reserves a unique key (FID) for the feature.

```
//Get the FeatureClass
Topobase.Data.FeatureClass myPointClass = myConnection.FeatureClasses["MYPOINT"];
//Create a new Feature
Topobase.Data.Feature myPointFeature = myPointClass.CreateFeature();
```

Once the feature instance is created, set its properties. For example, you can add geometry to the feature through the `Feature.Geometry` property, as shown here:

```
myPointFeature.Geometry = new Topobase.Graphic.Point(100, 100);
```

Use the `FeatureClass.InsertFeature` method to insert the new feature into Topobase; in other words, create a new record in the table.

```
myPointClass.InsertFeature(ref myPointFeature);
```

Use `FeatureClass.UpdateFeature()` to update an existing feature in Topobase.

```
myPointClass.UpdateFeature(ref myPointFeature);
```

To delete an existing feature, use `FeatureClass.DeleteFeature()`. This method requires the feature identifier (FID).

```
bool success = myPointClass.DeleteFeature(fid);
```

The following function writes a point feature to Topobase:

```
private void btn_WritePointFeature_Click(object sender, EventArgs e)
{
    //Connect to a Topobase
    Using(Topobase.Data.TBConnection myConnection =
        new Topobase.Data.TBConnection(
            Properties.Settings.Default.Oracle_User,
            Properties.Settings.Default.Oracle_Password,
            this.Application.Connection.DataSource,
            true,
            this.Application.Connection));
}
```

TAKING A CLOSER LOOK AT THE AUTODESK TOPOBASE API

```
myConnection.ShowForms = true;
myConnection.Open();

//Get the FeatureClass
Topobase.Data.FeatureClass myPointClass =
myConnection.FeatureClasses["MYPOINT"];

//Create a new Feature
Topobase.Data.Feature myPointFeature = myPointClass.CreateFeature();

// Set the X,Y coordinates
int x = 1000.0;
int y = 1000.0;

//Add Geometry to the Feature
myPointFeature.Geometry = new Topobase.Graphic.Point(x, y);

//Insert the new Feature into Topobase
myPointClass.InsertFeature(ref myPointFeature);

// show FID of inserted feature
MessageBox.Show("Feature ID = " + myPointFeature.FID);
//Close Connection
myConnection.Close();
// Update tree.
ShowStructure();
}
}
```

To retrieve a feature, use `FeatureClass.GetFeature()`. The following code snippet shows how to retrieve a feature that has a specified FID:

```
private void btn_ReadPointFeature_Click(object sender, EventArgs e)
{
    //Connect to a Topobase
    using(Topobase.Data.TBConnection myConnection =
        new Topobase.Data.TBConnection(
            Properties.Settings.Default.Oracle_User,
            Properties.Settings.Default.Oracle_Password,
            this.Application.Connection.DataSource,
            true,
            this.Application.Connection));
    {
        myConnection.ShowForms = true;
        myConnection.Open();
        //Get the FeatureClass
        Topobase.Data.FeatureClass myPointClass =
myConnection.FeatureClasses["MYPOINT"];

        //Read a feature with a specified FID
        Topobase.Data.Feature myPointFeature = myPointClass.GetFeature(1);
        //Check if found
        if (myPointFeature == null)
        {
            MessageBox.Show("Feature with this FID not found");
            return;
        }
        //Read Geometry
        Topobase.Graphic.Point myPoint = myPointFeature.Geometry as
Topobase.Graphic.Point;
        MessageBox.Show("X = " + myPoint.X + " Y = " + myPoint.Y);
        //Close Connection
        myConnection.Close();
    }
}
```

Summary

The Autodesk Topobase API enables developers to extend the basic functionality of Topobase for their own needs. The Framework API, which is the main API, provides a rich set of functionality, that can be used to customize Topobase and build custom modules to suit specific requirements.

Resources

Autodesk Developer Network

Whether you are a commercial or individual software developer, Autodesk Developer Network (ADN) membership provides the business, software, support, and training you need to meet your customers' needs. ADN provides valuable resources, whether you are developing boxed solutions, custom applications, or tools for internal use; providing consulting and systems integration services; performing research; or writing or publishing learning materials.

The Autodesk Developer Network includes API training programs on Autodesk Topobase. For more information visit www.autodesk.com/adn.

Learn More

For additional information about Topobase, visit the Topobase product center at www.autodesk.com/topobase.

To learn more about enhancing the performance of Topobase, visit the Topobase Insiders Blog at topobaseinsiders.typepad.com.

Additional information about tuning Oracle can be found at www.oracle.com.

Autodesk and Topobase are registered trademarks or trademarks of Autodesk, Inc., in the USA and/or other countries. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product offerings and specifications at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2008 Autodesk, Inc. All rights reserved.