
Amazon SimpleDB

Getting Started Guide

API Version 2009-04-15



Amazon SimpleDB: Getting Started Guide

Copyright © 2009 Amazon Web Services LLC or its affiliates. All rights reserved.

Table of Contents

Welcome	1
What's New	4
Introduction to Amazon SimpleDB	5
Getting Set Up	8
Creating an AWS Account	8
Signing Up for Amazon SimpleDB	9
Getting Your AWS Identifiers	9
Getting the Tools You Need	11
Showing Your Preferred Programming Language	12
Verifying Your Setup	12
Working with Amazon SimpleDB	15
Authenticating with Amazon SimpleDB	15
Preparing the Samples	16
Creating a Domain	20
Listing Domains	22
Putting Data into a Domain	24
Getting Data from a Domain	44
Deleting Values from an Attribute	47
Deleting an Attribute	49
Replacing an Attribute	51
Deleting an Item	54
Deleting a Domain	56
Where Do I Go from Here?	58
Amazon SimpleDB Glossary	59
Document Conventions	61

Welcome

Topics

- [Who Should Read This Guide \(p. 1\)](#)
- [Reader Feedback \(p. 2\)](#)
- [How to Use this Guide \(p. 2\)](#)
- [Related Resources \(p. 2\)](#)

This is the *Amazon SimpleDB* Getting Started Guide. This section describes who should read this guide, how the guide is organized, and other resources related to Amazon SimpleDB.

Amazon SimpleDB will occasionally be referred to within this guide as simply "SimpleDB"; all copyrights and legal protections still apply.

Who Should Read This Guide

This guide is intended for developers who are building web-based applications which create and store structured data sets, query the data sets, and quickly return results.

Required Knowledge and Skills

Use of this guide assumes you are familiar with the following:

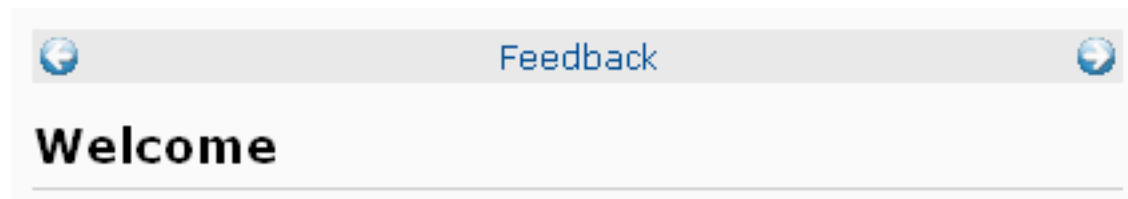
- XML (go to [W3 Schools XML Tutorial](#))
- Basic understanding of web services (go to [W3 Schools Web Services Tutorial](#))

In addition, for this guide, you need to be familiar with one of the following programming languages.

- PHP
- C#
- Java
- Perl
- VB.NET

Reader Feedback

The online version of this guide provides a link at the top of each page that enables you to enter feedback about this guide. We strive to make our guides as complete, error free, and easy to read as possible. You can help by giving us feedback. Thank you in advance!



How to Use this Guide

This guide is organized as a high-level introduction and tutorial. It is divided into several major sections that allow you to practice using Amazon SimpleDB in a simple environment. Each section builds on the previous sections, so that if you read and work through the examples in sequence, you will get a basic understanding of Amazon SimpleDB.

The major sections of this guide are:

- [Getting Set Up \(p. 8\)](#)
- [Working with Amazon SimpleDB \(p. 15\)](#)
- [Where Do I Go from Here? \(p. 58\)](#)
- [Amazon SimpleDB Glossary \(p. 59\)](#)

Related Resources

The following table lists related resources that you will find useful as you work with this service.

Resource	Description
Amazon SimpleDB Developer Guide	The Developer Guide provides a detailed discussion of the service. It includes an architectural overview, programming reference, and API reference.
<i>Amazon SimpleDB Release Notes</i>	The Release Notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
AWS Developer Resource Center	A central starting point find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
AWS Support Center	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and (if you're subscribed to this program) AWS Premium Support.

**Amazon SimpleDB Getting Started Guide
Related Resources**

Resource	Description
Amazon SimpleDB Product Information Page	The primary web page for information about Amazon SimpleDB.
Form for questions related to your AWS account: Contact Us	This form is <i>only</i> for account questions. For technical questions, use the AWS Support Center.
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com.

What's New

This What's New is associated with the 2009-04-15 release of Amazon SimpleDB. This guide was last updated on September 25, 2009.

The following table describes the important changes since the last release of the Amazon SimpleDB Getting Started Guide.

Change	Description	Release Date
Select	Document updated to use <code>Select</code> operation instead of <code>Query</code> .	24 March 2009
Code Sample Fixes	Fixed broken code samples.	24 March 2009
General Improvements	Small changes and fixes were made based on feedback received through the forums and documentation feedback system.	17 July 2008
General Improvements	Small changes and fixes were made based on feedback received through the forums and documentation feedback system.	22 April 2008

Introduction to Amazon SimpleDB

Topics

- [Overview of Amazon SimpleDB \(p. 5\)](#)
- [Key Amazon SimpleDB Concepts \(p. 6\)](#)
- [Overview of Examples \(p. 6\)](#)

This introduction to Amazon SimpleDB is intended to give you a high-level overview of this web service. After reading this section, you should understand the basics you need to work through the examples in this guide.

Overview of Amazon SimpleDB

Amazon SimpleDB is a web service for running queries on structured data in real time. This service works in close conjunction with Amazon Simple Storage Service (Amazon S3) and Amazon Elastic Compute Cloud (Amazon EC2), collectively providing the ability to store, process and query data sets in the cloud. These services are designed to make web-scale computing easier and more cost-effective for developers.

Traditionally, this type of functionality has been accomplished with a clustered relational database that requires a sizable upfront investment, brings more complexity than is typically needed, and often requires a DBA to maintain and administer. In contrast, Amazon SimpleDB is easy to use and provides the core functionality of a database - real-time lookup and simple querying of structured data - without the operational complexity. Amazon SimpleDB requires no schema, automatically indexes your data and provides a simple API for storage and access. This eliminates the administrative burden of data modeling, index maintenance, and performance tuning. Developers gain access to this functionality within Amazon's proven computing environment, are able to scale instantly, and pay only for what they use.

Features

Amazon SimpleDB provides the following major features:

Features

- **Simple to use**—Amazon SimpleDB provides streamlined access to the lookup and query functions that traditionally are achieved using a relational database cluster “ while leaving out other complex, often-unused database operations. The service allows you to quickly add data and easily retrieve or edit that data through a simple set of API calls. Accessing these capabilities through a web service also eliminates the complexity of maintaining and scaling these operations.
- **Flexible**—With Amazon SimpleDB, it is not necessary to pre-define all of the data formats you will need to store; simply add new attributes to your Amazon SimpleDB data set when needed, and the system will automatically index your data accordingly. The ability to store structured data without first defining a schema provides developers with greater flexibility when building applications.
- **Scalable**—Amazon SimpleDB allows you to easily scale your application. You can quickly create new domains as your data grows or your request throughput increases. For the Beta release, a single domain is limited in size to 10 GB and you are limited to a maximum of 100 domains; however, over time these limits may be raised.
- **Fast**—Amazon SimpleDB provides quick, efficient storage and retrieval of your data to support high performance web applications.
- **Reliable**—The service runs within Amazon's high-availability data centers to provide strong and consistent performance. To prevent data from being lost or becoming unavailable, your fully indexed data is stored redundantly across multiple servers and data centers.
- **Designed for use with other Amazon Web Services**—Amazon SimpleDB is designed to integrate easily with other web-scale services such as Amazon EC2 and Amazon S3. For example, developers can run their applications in Amazon EC2 and store their data objects in Amazon S3. Amazon SimpleDB can then be used to query the object metadata from within the application in Amazon EC2 and return pointers to the objects stored in Amazon S3.
- **Inexpensive**—Amazon SimpleDB passes on to you the financial benefits of Amazon's scale. You pay only for resources you actually consume. Compare this with the significant up-front expenditures traditionally required to obtain software licenses and purchase and maintain hardware, either in-house or hosted. This frees you from many of the complexities of capacity planning, transforms large capital expenditures into much smaller operating costs, and eliminates the need to over-buy “safety net” capacity to handle periodic traffic spikes.

Key Amazon SimpleDB Concepts

For information about Amazon SimpleDB concepts, see the *Amazon SimpleDB Developer Guide*.

Overview of Examples

This guide walks you through the process of creating the imaginary clothing/auto parts store “Off the Rack and Pinion.” Through this tutorial, you will:

- Create a domain
- List domains
- Put data into the domain
- List all items
- Delete values from an attribute
- Get the attributes of an item
- Delete an attribute
- Replace an attribute
- Delete an item

- Delete a domain

Getting Set Up

Topics

- [Creating an AWS Account \(p. 8\)](#)
- [Signing Up for Amazon SimpleDB \(p. 9\)](#)
- [Getting Your AWS Identifiers \(p. 9\)](#)
- [Getting the Tools You Need \(p. 11\)](#)
- [Showing Your Preferred Programming Language \(p. 12\)](#)
- [Verifying Your Setup \(p. 12\)](#)

The following sections explain the steps you must take before you can submit a Amazon SimpleDB request.

Creating an AWS Account

To access any web service AWS offers, you must first create an AWS account at <http://aws.amazon.com>. An AWS account is simply an Amazon.com account that is enabled to use AWS products; you can use an existing Amazon.com account login and password when creating the AWS account.



Important

If you have a personal Amazon.com account, you might want to have a separate Amazon.com account just for your AWS activity. You could provide a new e-mail address not already in the Amazon.com system, or provide an e-mail address for an existing Amazon.com account you have but use a different password. You can have multiple Amazon.com accounts that use the same e-mail address, but different passwords.

From your AWS account you can view your AWS account activity, view usage reports, and manage your AWS account access identifiers.

To set up a new account

1. Go to <http://aws.amazon.com>.
2. In the **Sign Up for AWS** box, click **Sign up today**.
The **Sign In** page is displayed.

3. Enter a valid e-mail address, select the button for **No, I am a new customer**, and click **Continue**.
The next page asks for the name you want associated with the account (e.g., Joe Smith) and a password for the account. If you're using an e-mail address already in the Amazon.com system, the page indicates that this is an existing address, but still lets you create a new account with it.
4. Enter the name you want associated with the account and a password and click **Continue**.
The **Account Info** page is displayed.
5. Enter your contact information and select how you learned about AWS. Then read the AWS Customer Agreement, select the check box to indicate that you've read and agree to the terms of the customer agreement, and click **Continue**.
The process is complete and you've created your new AWS account.

At this point, you have an AWS account, but you're not signed up to use Amazon SimpleDB yet. For instructions, see [Signing Up for Amazon SimpleDB \(p. 9\)](#).

Signing Up for Amazon SimpleDB

Before you can use Amazon SimpleDB, you must sign up to use the service. You must already have an AWS account (see [Creating an AWS Account \(p. 8\)](#)).

To sign up for Amazon SimpleDB

1. Go to the [Amazon SimpleDB](#) page, and then click **Sign Up for This Web Service** on the top right corner of the page.

Your next action depends on whether you already have a credit card associated with the account.

2. Choose one of the following:

If you already have a credit card associated with the account:	If you don't have a credit card associated with the account:
<ul style="list-style-type: none">• Review the displayed pricing and credit card information and click Complete Sign Up.	<ol style="list-style-type: none">1. Enter information for a valid credit card and click Continue.2. Enter the billing address to use with the credit card and click Continue.3. Review the displayed pricing and credit card information you provided and click Complete Sign Up.

AWS sends you a confirmation e-mail.

Getting Your AWS Identifiers

For the samples, you will need your *AWS access key identifiers*, which AWS assigned you when you created your AWS account. For example:

- Access Key ID (a 20-character, alphanumeric sequence)
For example: 022QF06E7MXBSH9DHM02
- Secret Access Key (a 40-character sequence)
For example: kWcrIUX5JEDGM/LtmEENI/aVmYvHNif5zB+d9+ct



Caution

Your Secret Access Key is a secret, which only you and AWS should know. It is important to keep it confidential to protect your account. Store it securely in a safe place. Never include it in your requests to AWS, and never e-mail it to anyone. Do not share it outside your organization, even if an inquiry appears to come from AWS or Amazon.com. No one who legitimately represents Amazon will ever ask you for your Secret Access Key.

The Access Key ID is associated with your AWS account. You include it in AWS service requests to identify yourself as the sender of the request.

The Access Key ID is not a secret, and anyone could use your Access Key ID in requests to AWS. To provide proof that you truly are the sender of the request, you also include a digital signature calculated using your Secret Access Key. The sample code handles this for you.

Your Access Key ID and Secret Access Key are displayed to you when you create your AWS account. They are not e-mailed to you. If you need to see them again, you can view them at any time from your AWS account.

To view your AWS access key identifiers

1. Go to the Amazon Web Services web site at <http://aws.amazon.com>.
2. Point to **Your Account** and select **Security Credentials**. If you are not already logged in, you are prompted to do so.
3. Click **Create a New Access Key** and follow the on-screen prompts.
The new Access Key is created and appears in the **Your Access Keys** list. The Access Key ID is displayed, but the Secret Access Key remains hidden.
4. To display the Secret Access Key, click **Show** in the **Secret Access Key** column.

Access Credentials

In order to start using Amazon Web Services you must first identify yourself as the sender of a request to the given service. This is accomplished by sending a digital signature that is derived from a pair of public/private access keys or a valid security certificate.

 **Access Keys**  X.509 Certificates

Access Key ID

Your Access Key ID identifies you as the party responsible for service requests. Use your Access Key ID as the value of the `AWSAccessKeyId` parameter in requests you send to Amazon Web Services (when required).

Secret Access Key

Each Access Key ID has a Secret Access Key associated with it. Use your Secret Access Key to calculate a signature to include in requests to web services that require authenticated requests. Your Secret Access Key is a secret, and should be known only by you and AWS. You should never include your Secret Access Key in your requests to AWS. You should never e-mail your Secret Access Key to anyone. It is important to keep your Secret Access Key confidential to protect your account.

Access Key Rotation

AWS supports multiple concurrent access keys. This allows you to rotate keys without impact to your applications' availability. AWS recommends that you rotate keys on a regular basis. To rotate keys, create a new key below, update your applications to use the new key, and then deactivate/delete the original key.

You are allowed two access keys at any point in time, and the keys may be in the following states:

Active: key can be used to secure requests to AWS.

Inactive: key cannot be used to secure requests to AWS, but can be moved back into the Active state. AWS recommends you use this state for a small window of time before deleting, to ensure that all applications are working well using the new key.

Deleted: key can never again be used to secure requests to AWS. AWS recommends deleting a key when you are sure it is no longer in use by your applications.

Your Access Keys

Created	Access Key ID	Secret Access Key	Status
September 4, 2009	AKIAIDS4LOGTXFG4CKTA	Show	Active (Make Inactive)

[Create a New access Key](#)

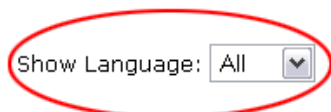
Getting the Tools You Need

You can use most modern programming languages to integrate Amazon SimpleDB requests into applications. In the following table, click the toolkit you want to use to implement Amazon SimpleDB. The link takes you to the web site where you can download and install the toolkit.

Language	API Style	Tools
Java	REST	The Java example requires the following: <ul style="list-style-type: none">• Java 2 Platform Standard Edition 5.0 Development Kit (JDK 5.0) or equivalent
C#	REST	The C# example requires the following: <ul style="list-style-type: none">• Microsoft Visual Studio 2005 C# Express Edition
Perl	REST	The Perl 5 example uses the <code>Digest::SHA1</code> , <code>Bundle::LWP</code> , and <code>XML::Simple</code> modules, which can be downloaded from the CPAN web site .
PHP	REST	The PHP example uses the base installation of PHP5.
VB.NET	REST	The VB.NET example requires the following: <ul style="list-style-type: none">• Microsoft Visual Basic 2005 Express Edition

Showing Your Preferred Programming Language

You can hide the sections of this guide that don't apply to the programming language you are using. There is a language selection menu in the upper-right corner of pages with language-specific text. Select your language to hide all others, or select **All** to show the examples in all available languages.



Verifying Your Setup

- [Java \(p. 13\)](#)
- [C# \(p. 13\)](#)
- [Perl \(p. 13\)](#)
- [PHP \(p. 14\)](#)
- [VB.NET \(p. 14\)](#)

This section helps you confirm that your development environment is set up correctly.

Skip to the section that corresponds to the toolkit you downloaded.

Java

This section ensures that you have the correct versions of JDK installed. These instructions pertain to Linux.

Verifying the JDK Installation

To verify the JDK installation

1. Open a console window.
2. Enter the following command:

```
java -version
```

The command response should be similar to the following.

```
java version "1.5.0_05"  
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_05-b05)  
Java HotSpot(TM) Client VM (build 1.5.0_05-b05, mixed mode, sharing)
```

If you do not get the appropriate response, go to [Sun Microsystems, Inc.](http://www.sun.com) and check the Java documentation to ensure you have installed everything correctly.

C#

This section ensures that you have correctly installed Microsoft Visual Studio.

To confirm the installation

1. Open Visual Studio 2005 C# Express Edition.
2. Select **Microsoft Visual Studio** from the **Help** > menu.
A dialog box opens that lists Microsoft Visual Studio 2005 and version 2.0 of the .NET Framework.

Perl

To confirm your Perl installation

1. Change directories to the Perl installation directory or make sure the installation directory is in your PATH environment variable and run the following command.

```
perl -version
```

The response should be similar to the following.

```
This is perl, v5.6.0 built for Linux-2.4c2.3-i686  
Copyright 1987-2000, Larry Wall
```

2. If the response you receive is dissimilar from the preceding, reinstall Perl.
3. Run the following commands.

```
perl -MDigest::SHA1 -e 1  
perl -MBundle::LWP -e 1  
perl -MXML::Simple -e 1
```

4. If you receive any error messages, reinstall the required Perl modules.

PHP

To ensure that you have correctly installed PHP

1. Use a command-line interface to run the following command.

```
php -version
```



Note

This command assumes you are either in your PHP installation directory or it is in your PATH environment variable.

The response should be similar to the following.

```
PHP 5.1.2 (cli) (built: Jan 11 2006 16:40:00)  
Copyright (c) 1997-2006 The PHP Group  
Zend Engine v2.1.0, Copyright (c) 1998-2006 Zend Technologies
```

2. If the response you receive is dissimilar from the preceding, reinstall PHP.

VB.NET

This section ensures that you have correctly installed Microsoft Visual Basic.

To confirm the installation

1. Open Visual Basic 2005 Express Edition.
2. Select **Microsoft Visual Studio** from the **Help >** menu.
A dialog box opens that lists Microsoft Visual Basic 2005 and version 2.0 of the .NET Framework.

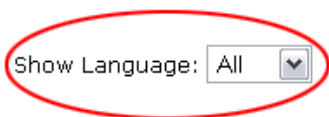
Working with Amazon SimpleDB

This section describes how to authenticate Amazon SimpleDB with AWS, create a domain, store data within a domain, get data from a domain, and execute complex queries.



Note

You can hide the sections of this guide that don't apply to the programming language you are using. There is a language selection menu in the upper-right corner of pages with language-specific text. Select your language to hide all others, or select **All** to show the examples in all available languages.



The following sections are intended to be followed sequentially, similarly to a tutorial.

- [Authenticating with Amazon SimpleDB \(p. 15\)](#)
- [Preparing the Samples \(p. 16\)](#)
- [Creating a Domain \(p. 20\)](#)
- [Listing Domains \(p. 22\)](#)
- [Putting Data into a Domain \(p. 24\)](#)
- [Deleting Values from an Attribute \(p. 47\)](#)
- [Deleting an Attribute \(p. 49\)](#)
- [Replacing an Attribute \(p. 51\)](#)
- [Getting Data from a Domain \(p. 44\)](#)
- [Deleting an Item \(p. 54\)](#)
- [Deleting a Domain \(p. 56\)](#)

Authenticating with Amazon SimpleDB

Amazon SimpleDB authentication is used to validate the identity of the party making a request. Most requests to Amazon SimpleDB require authentication to verify that the subscriber is authorized to perform the requested action. Authentication ensures that you don't get charged for operations you did not authorize and that nobody else sees your private data.

Security always relies on a secret. For Amazon SimpleDB (and other AWS services), your secret is your AWS Secret Access Key. It should not be shared with others outside your organization, even if a request appears to come from Amazon.com. The AWS Secret Access Key is paired with an AWS Access Key ID. The AWS Access Key ID identifies you in your requests to Amazon SimpleDB. When a request is received with an AWS Access Key ID, Amazon SimpleDB verifies that the sender of the request knows the corresponding AWS Secret Access Key by validating a signature that is included with the request. You can find your AWS Access Key ID and Secret Access Key by visiting the [AWS Access Key Identifiers](#) page.

In the following samples, you simply add your AWS Access Key ID and AWS Secret Key to the sample files. For more information about authentication, refer to the *Amazon SimpleDB Developer Guide*.

Preparing the Samples

To complete the following sections, you must make configuration changes to the sample files.

Java

To prepare the sample files

1. Extract the [Amazon SimpleDB sample code](#) into a directory on your machine designated as <sdb home>.
2. Configure your CLASSPATH environment variable to point to the Amazon SimpleDB Samples JAR archive and the included third party JARs. The following is an example in Linux.

```
export CLASSPATH=$CLASSPATH:\
<sdb home>/amazon-simpliedb-2007-11-07-java-library\
/src/com/amazonaws/sdb/samples/\
<sdb home>/lib/amazon-simpliedb-2007-11-07-java-library.jar:\
<sdb home>/third-party/jakarta-commons/commons-codec-1.3.jar:\
<sdb home>/third-party/jakarta-commons/commons-httpclient-3.0.1.jar:\
<sdb home>/third-party/jakarta-commons/commons-logging-1.1.jar:\
<sdb home>/third-party/jaxb-ri-2.1/activation.jar:\
<sdb home>/third-party/jaxb-ri-2.1/jaxb-all-deps.jar:\
<sdb home>/third-party/jaxb-ri-2.1/jaxb-api.jar:\
<sdb home>/third-party/jaxb-ri-2.1/jaxb-impl.jar:\
<sdb home>/third-party/jaxb-ri-2.1/jaxb-xjc.jar:\
<sdb home>/third-party/jaxb-ri-2.1/jsr173_1.0_api.jar:\
<sdb home>/third-party/log4j/log4j-1.2.14.jar
```

3. The following command changes your working directory.

```
<sdb home>/amazon-simpliedb-2007-11-07-java-library/src/com/amazonaws/sdb/
samples/
```

4. Open `src/com/amazonaws/sdb/samples/CreateDomainSample.java`.
5. Determine whether you will run the code against the mock service or your Amazon SimpleDB account:
 - To run the code against the mock service, comment out the following lines:

```
String accessKeyId = "<Your Access Key ID>";
String secretAccessKey = "<Your Secret Access Key>"
AmazonSimpleDB service = new AmazonSimpleDBClient(accessKeyId,
    secretAccessKey)
```

Then, remove comment marks from the following line:

```
AmazonSimpleDB service = new AmazonSimpleDBMock();
```

- To run the code against Amazon SimpleDB, make sure the following lines are not commented out and enter your Access Key ID and Secret Access Key:

```
String accessKeyId = "<Your Access Key ID>";  
String secretAccessKey = "<Your Secret Access Key>";  
AmazonSimpleDB service = new AmazonSimpleDBClient(accessKeyId,  
    secretAccessKey)
```

Make sure the following line is commented out:

```
AmazonSimpleDB service = new AmazonSimpleDBMock();
```

6. Save the file.
7. Repeat this procedure for each file in the samples directory.

C#

To prepare the sample files

1. Extract the [Amazon SimpleDB sample code](#) into a directory on your machine designated as <sdb home>.
2. Open the following sample solution: <sdb home>/amazon-simpledb-2007-11-07-cs-library/src/Amazon.SimpleDB.sln.
3. Open the `AmazonSimpleDBSamples.cs` file.
4. Determine whether you will run the code against the mock service or your Amazon SimpleDB account:
 - To run the code against the mock service, comment out the following lines:

```
String accessKeyId = "<Your Access Key ID>";  
String secretAccessKey = "<Your Secret Access Key>";  
AmazonSimpleDB service = new AmazonSimpleDBClient(accessKeyId,  
    secretAccessKey);
```

Then, remove comment marks from the following line:

```
AmazonSimpleDB service = new AmazonSimpleDBMock();
```

- To run the code against Amazon SimpleDB, make sure the following lines are not commented out and enter your Access Key ID and Secret Access Key:

```
String accessKeyId = "<Your Access Key ID>";  
String secretAccessKey = "<Your Secret Access Key>";  
AmazonSimpleDB service = new AmazonSimpleDBClient(accessKeyId,  
    secretAccessKey);
```

Make sure the following line is commented out:

```
AmazonSimpleDB service = new AmazonSimpleDBMock();
```

5. Save the file.

Perl

To prepare the sample files

1. Extract the [Amazon SimpleDB sample code](#) into a directory on your machine designated as <sdhome>.
2. Open the `src/Amazon/SimpleDB/Samples/CreateDomainSample.pl` sample file.
3. Determine whether you will run the code against the mock service or your Amazon SimpleDB account:

- To run the code against the mock service, comment out the following lines:

```
my $AWS_ACCESS_KEY_ID      = "<Your Access Key ID>";
my $AWS_SECRET_ACCESS_KEY  = "<Your Secret Access Key>";
use Amazon::SimpleDB::Client;
my $service = Amazon::SimpleDB::Client->new($AWS_ACCESS_KEY_ID,
    $AWS_SECRET_ACCESS_KEY);
```

Then, remove comment marks from the following line:

```
use Amazon::SimpleDB::Mock;
my $service = Amazon::SimpleDB::Mock->new;
```

- To run the code against Amazon SimpleDB, make sure the following lines are not commented out and enter your Access Key ID and Secret Access Key:

```
my $AWS_ACCESS_KEY_ID      = "<Your Access Key ID>";
my $AWS_SECRET_ACCESS_KEY  = "<Your Secret Access Key>";
use Amazon::SimpleDB::Client;
my $service = Amazon::SimpleDB::Client->new($AWS_ACCESS_KEY_ID,
    $AWS_SECRET_ACCESS_KEY);
```

Make sure the following lines are commented out:

```
use Amazon::SimpleDB::Mock;
my $service = Amazon::SimpleDB::Mock->new;
```

4. Save the file.
5. Repeat this procedure for each file in the samples directory.

PHP

To prepare the sample files

1. Extract the [Amazon SimpleDB sample code](#) into a directory on your machine designated as <sdhome>.
2. Open a sample file in the `src/Amazon/SimpleDB/Samples/` directory.
3. Determine whether you will run the code against the mock service or your Amazon SimpleDB account:

- To run the code against the mock service, comment out the following lines:

```
$AWS_ACCESS_KEY_ID        = 'Your Access Key ID';
$AWS_SECRET_ACCESS_KEY    = 'Your Secret Access Key';
require_once ('Amazon/SimpleDB/Client.php');
$service = new Amazon_SimpleDB_Client($AWS_ACCESS_KEY_ID,
    $AWS_SECRET_ACCESS_KEY);
```

Then, remove comment marks from the following line:

```
require_once ('Amazon/SimpleDB/Mock.php');  
$service = new Amazon_SimpleDB_Mock();
```

- To run the code against Amazon SimpleDB, make sure the following lines are not commented out and enter your Access Key ID and Secret Access Key:

```
$AWS_ACCESS_KEY_ID      = 'Your Access Key ID';  
$AWS_SECRET_ACCESS_KEY  = 'Your Secret Access Key';  
require_once ('Amazon/SimpleDB/Client.php');  
$service = new Amazon_SimpleDB_Client($AWS_ACCESS_KEY_ID,  
    $AWS_SECRET_ACCESS_KEY);
```

Make sure the following lines are commented out:

```
require_once ('Amazon/SimpleDB/Mock.php');  
$service = new Amazon_SimpleDB_Mock();
```

4. Save the file.
5. Repeat this procedure for each file in the samples directory.

VB.NET

To prepare the sample files

1. Extract the [Amazon SimpleDB sample code](#) into a directory on your machine designated as <sdh home>.
2. Open the following sample: <sdh home>/amazon-simpliedb-2007-11-07-vbnet-library/src/Amazon.SimpleDB.sln.
3. Open the AmazonSimpleDBSamples.vb file.
4. Determine whether you will run the code against the mock service or your Amazon SimpleDB account:

- To run the code against the mock service, comment out the following lines:

```
Dim accessKeyId As String = "<Your Access Key ID>"  
Dim secretAccessKey As String = "<Your Secret Access Key>"  
Dim service As AmazonSimpleDB = New AmazonSimpleDBClient(accessKeyId,  
    secretAccessKey)
```

Then, remove comment marks from the following line:

```
Dim service As AmazonSimpleDB = New AmazonSimpleDBMock()
```

- To run the code against Amazon SimpleDB, make sure the following lines are not commented out and enter your Access Key ID and Secret Access Key:

```
Dim accessKeyId As String = "<Your Access Key ID>"  
Dim secretAccessKey As String = "<Your Secret Access Key>"  
Dim service As AmazonSimpleDB = New AmazonSimpleDBClient(accessKeyId,  
    secretAccessKey)
```

Then, make sure the following line is commented out:

```
Dim service As AmazonSimpleDB = New AmazonSimpleDBMock()
```

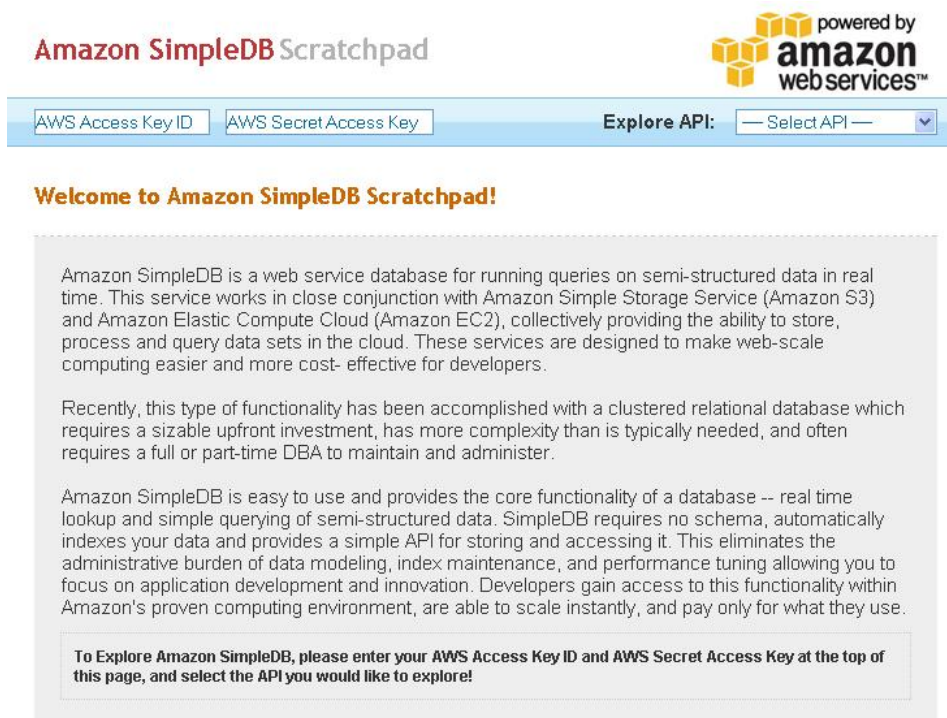
5. Save the file.

Scratchpad

The scratchpad is a sample web application that enables you to test and make calls to Amazon SimpleDB without having to set up a programming environment.

To prepare the scratchpad

1. Copy the scratchpad files to a location on your computer.
2. Open the index.html file with a web browser. The welcome page appears.



3. Enter your AWS Access Key ID and your Secret Access Key.
4. Follow the tutorial described in the remainder of this document.

Creating a Domain

The first step in storing data within Amazon SimpleDB is to create one or more domains. Domains are similar to database tables, except that you cannot perform functions across multiple domains, such as querying multiple domains or using foreign keys. As a result, you should plan a Amazon SimpleDB data architecture that will meet the needs of your project.

All data that will be searched in a single query should be grouped using a single domain. For example, a web commerce application might contain a customer domain that contains all customer data (name, address, phone, and so on) and a products domain that contains all product information (name, description, price, cost, and so on).



Note

Although the Amazon SimpleDB API cannot perform queries across multiple domains, you can design your applications to perform queries across multiple domains.

The following sample code snippets demonstrate creation of the `MyStore` domain.

Java

To run the sample

1. Open a clean copy of `CreateDomainSample.java`.
2. Remove any comment marks from the `invokeCreateDomain(service, request);` line and add the following line after `// @TODO: set request parameters here:`

```
request.withDomainName("MyStore");
```

3. Compile and run the sample.
The `MyStore` domain is created.

C#

To run the sample

1. Open `AmazonSimpleDBSamples.cs`.
2. Replace the Create Domain Action section with the following:

```
CreateDomainRequest createDomainAction = (new  
    CreateDomainRequest()).WithDomainName("MyStore");  
CreateDomainSample.InvokeCreateDomain(service, createDomainAction);
```

3. Run the sample.
The `MyStore` domain is created.

Perl

To run the sample

1. Open a clean copy of `CreateDomainSample.pl`.
2. Remove any comment marks from the `invokeCreateDomain($service, $request);` line and add the following line after the `@TODO: set request line:`

```
my $request = Amazon::SimpleDB::Model::CreateDomainRequest->new({DomainName  
=> "MyStore"});
```

3. Run the sample.
The `MyStore` domain is created.

PHP

To run the sample

1. Open a clean copy of `CreateDomainSample.php`.
2. Remove any comment marks from the `invokeCreateDomain($service, $request);` line and add the following lines after the `// @TODO: set request line:`

```
$request = new Amazon_SimpleDB_Model_CreateDomainRequest();  
$request = new Amazon_SimpleDB_Model_CreateDomain();  
$request->setDomainName('MyStore');
```

3. Run the sample.
The MyStore domain is created.

VB.NET

To run the sample

1. Open `AmazonSimpleDBSamples.vb`.
2. Replace the Create Domain Action section with the following:

```
Dim createAction As New CreateDomainRequest()  
createAction.DomainName = "MyStore"  
  
CreateDomainSample.InvokeCreateDomain(service, createAction)
```

3. Run the sample.
The MyStore domain is created.

Scratchpad

To run the sample

1. Open the scratchpad application with a web browser.
2. Select **Create Domain** from the **Explore API** list box.
3. Enter `MyStore` in the **Create Domain** field.
4. Select from the following:
 - To invoke the request, click **Invoke Request**.
Amazon SimpleDB returns a response.
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
Amazon SimpleDB returns a response.
 - To view the string to sign, click **Display String to Sign**.

Listing Domains

In the previous section, you created a domain. This section provides examples of how to list all domains within your account.

The following sample code snippets demonstrate listing domains.

Java

To run the sample

1. Open a clean copy of `ListDomainsSample.java`.
2. Remove any comment marks from the `invokeListDomains(service, request);` line.
3. Compile and run the sample.
Amazon SimpleDB returns the list of domains within the account, including the newly created MyStore domain.

C#

To run the sample

1. Open `AmazonSimpleDBSamples.cs`.
2. Comment out the code you added in the previous section.
3. Replace the List Domains Action section with the following:

```
ListDomainsRequest ListDomainsAction = new ListDomainsRequest();  
ListDomainsSample.InvokeListDomains(service, ListDomainsAction);
```

4. Run the sample.
Amazon SimpleDB returns the list of domains within the account, including the newly created MyStore domain.

Perl

To run the sample

1. Open a clean copy of `ListDomainsSample.pl`.
2. Remove any comment marks from the `invokeCreateDomain($service, $request);` line and add the following line after the `@TODO: set request` line:

```
my $request = Amazon::SimpleDB::Model::ListDomainsRequest->new();
```

3. Run the sample.
Amazon SimpleDB returns the list of domains within the account, including the newly created MyStore domain.

PHP

To run the sample

1. Open a clean copy of `ListDomainsSample.php`.
2. Remove any comment marks from the `invokeListDomains($service, $request);` line and add the following line after the `@TODO: set request` line:

```
$request = new Amazon_SimpleDB_Model_ListDomainsRequest();
```

3. Run the sample.
Amazon SimpleDB returns the list of domains within the account, including the newly created MyStore domain.

VB.NET

To run the sample

1. Open `AmazonSimpleDBSamples.vb`.
2. Comment out the code you added in the previous section.
3. Replace the List Domains Action section with the following:

```
Dim listDomainsAction As New ListDomainsRequest()  
  
ListDomainsSample.InvokeListDomains(service, listDomainsAction)
```

4. Run the sample.

Amazon SimpleDB returns the list of domains within the account, including the newly created MyStore domain.

Scratchpad

To run the sample

1. Open the scratchpad application with a web browser.
2. Select **ListDomains** from the **Explore API** list box.
3. To limit the number of domains returned, enter a value in the **Max Number Of Domains** field.
4. If there are more domains than you specified in the **Max Number Of Domains** field, Amazon SimpleDB returns a next token.
5. To view the next page of results, enter the next token in the **Next Token** field.
6. Select from the following:
 - To invoke the request, click **Invoke Request**.
Amazon SimpleDB returns the list of domains within the account, including the newly created MyStore domain.
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
 - To view the string to sign, click **Display String to Sign**.

Putting Data into a Domain

After creating a domain, you are ready to start putting data into the domain.

The `PutAttributes` operation creates or replaces attributes within an item. The attributes are specified using the `Attribute.X.Name` and `Attribute.X.Value` parameters. The first attribute is specified by the parameters `Attribute.1.Name` and `Attribute.1.Value`, the second attribute by the parameters `Attribute.2.Name` and `Attribute.2.Value`, and so on.



Note

The `PutAttributes` creates or replaces attributes for one item at a time. To create or replace attributes for multiple items in a single call, use `BatchPutAttributes`. For more information, refer to the *Amazon SimpleDB Developer Guide*.

This section describes how to put the following data into the imaginary clothing/auto parts store "Off the Rack and Pinion."

The following table contains the information that you will be adding, modifying, and performing queries against.

ID	Category	Subcat.	Name	Color	Size	Make	Model	Year
Item_01	Clothes	Sweater	Cathair Sweater	Siamese	Small, Medium, Large			
Item_02	Clothes	Pants	Designer Jeans	Paisley Acid Wash	30x32, 32x32, 32x34			

ID	Category	Subcat.	Name	Color	Size	Make	Model	Year
Item_03	Clothes	Pants	Sweatpants	Blue, Yellow, Pink	Large			2006, 2007
Item_04	Car Parts	Engine	Turbos			Audi	S4	2000, 2001, 2002
Item_05	Car Parts	Emissions	O2 Sensor			Audi	S4	2000, 2001, 2002

Java

To run the sample

1. Open a clean copy of `PutAttributesSample.java`.
2. Add the following lines after `// @TODO: set request parameters here:`

```
String domainName = "MyStore";

/*****
 * Load Item_01:
 *Category = Clothes
 *Subcategory = Sweater
 *Name = Cathair Sweater
 *Color = Siamese
 *Size = Small, Medium, Large
 *****/

String itemNameOne = "Item_01";
java.util.List<ReplaceableAttribute> attributeListOne = new
    ArrayList<ReplaceableAttribute>(7);

attributeListOne.add(new ReplaceableAttribute("Category", "Clothes",
    false));
attributeListOne.add(new ReplaceableAttribute("Subcategory", "Sweater",
    false));
attributeListOne.add(new ReplaceableAttribute("Name", "Cathair Sweater",
    false));
attributeListOne.add(new ReplaceableAttribute("Color", "Siamese", false));
attributeListOne.add(new ReplaceableAttribute("Size", "Small", false));
attributeListOne.add(new ReplaceableAttribute("Size", "Medium", false));
attributeListOne.add(new ReplaceableAttribute("Size", "Large", false));

PutAttributesRequest putAttributesActionOne = new
    PutAttributesRequest(domainName, itemNameOne, attributeListOne);

invokePutAttributes(service, putAttributesActionOne);

/*****
 * Load Item_02:
 *Category = Clothes
 *Subcategory = Pants
 *Name = Designer Jeans
 *****/
```

```
*Color = Paisley Acid Wash
*Size = 30x32, 32x32, 32x34
*****/

String itemNameTwo = "Item_02";
java.util.List<ReplaceableAttribute> attributeListTwo = new
    ArrayList<ReplaceableAttribute>(7);

attributeListTwo.add(new ReplaceableAttribute("Category", "Clothes",
    false));
attributeListTwo.add(new ReplaceableAttribute("Subcategory", "Pants",
    false));
attributeListTwo.add(new ReplaceableAttribute("Name", "Designer Jeans",
    false));
attributeListTwo.add(new ReplaceableAttribute("Color", "Paisley Acid Wash",
    false));
attributeListTwo.add(new ReplaceableAttribute("Size", "30x32", false));
attributeListTwo.add(new ReplaceableAttribute("Size", "32x32", false));
attributeListTwo.add(new ReplaceableAttribute("Size", "32x34", false));

PutAttributesRequest putAttributesActionTwo = new
    PutAttributesRequest(domainName, itemNameTwo, attributeListTwo);

invokePutAttributes(service, putAttributesActionTwo);

/*****
 * Load Item_03:
 *Category = Clothes
 *Subcategory = Pants
 *Name = Sweatpants
 *Color = Blue, Yellow, Pink
 *Size = Large
 *Year = 2006, 2007
 *****/

String itemNameThree = "Item_03";
java.util.List<ReplaceableAttribute> attributeListThree = new
    ArrayList<ReplaceableAttribute>(9);

attributeListThree.add(new ReplaceableAttribute("Category", "Clothes",
    false));
attributeListThree.add(new ReplaceableAttribute("Subcategory", "Pants",
    false));
attributeListThree.add(new ReplaceableAttribute("Name", "Sweatpants",
    false));
attributeListThree.add(new ReplaceableAttribute("Color", "Blue", false));
attributeListThree.add(new ReplaceableAttribute("Color", "Yellow", false));
attributeListThree.add(new ReplaceableAttribute("Color", "Pink", false));
attributeListThree.add(new ReplaceableAttribute("Size", "Large", false));
attributeListThree.add(new ReplaceableAttribute("Year", "2006", false));
attributeListThree.add(new ReplaceableAttribute("Year", "2007", false));

PutAttributesRequest putAttributesActionThree = new
    PutAttributesRequest(domainName, itemNameThree, attributeListThree);

invokePutAttributes(service, putAttributesActionThree);

/*****
 * Load Item_04:
```

```
*Category = Car Parts
*Subcategory = Engine
*Name = Turbos
*Make = Audi
*Model = S4
*Year = 2000, 2001, 2002
*****/

String itemNameFour = "Item_04";
java.util.List<ReplaceableAttribute> attributeListFour = new
    ArrayList<ReplaceableAttribute>(8);

attributeListFour.add(new ReplaceableAttribute("Category", "Car Parts",
    false));
attributeListFour.add(new ReplaceableAttribute("Subcategory", "Engine",
    false));
attributeListFour.add(new ReplaceableAttribute("Name", "Turbos", false));
attributeListFour.add(new ReplaceableAttribute("Make", "Audi", false));
attributeListFour.add(new ReplaceableAttribute("Model", "S4", false));
attributeListFour.add(new ReplaceableAttribute("Year", "2000", false));
attributeListFour.add(new ReplaceableAttribute("Year", "2001", false));
attributeListFour.add(new ReplaceableAttribute("Year", "2002", false));

PutAttributesRequest putAttributesActionFour = new
    PutAttributesRequest(domainName, itemNameFour, attributeListFour);

invokePutAttributes(service, putAttributesActionFour);

/*****
 * Load Item_05:
 *Category = Car Parts
 *Subcategory = Emissions
 *Name = O2 Sensor
 *Make = Audi
 *Model = S4
 *Year = 2000, 2001, 2002
 *****/

String itemNameFive = "Item_05";
java.util.List<ReplaceableAttribute> attributeListFive = new
    ArrayList<ReplaceableAttribute>(8);

attributeListFive.add(new ReplaceableAttribute("Category", "Car Parts",
    false));
attributeListFive.add(new ReplaceableAttribute("Subcategory", "Emissions",
    false));
attributeListFive.add(new ReplaceableAttribute("Name", "O2 Sensor",
    false));
attributeListFive.add(new ReplaceableAttribute("Make", "Audi", false));
attributeListFive.add(new ReplaceableAttribute("Model", "S4", false));
attributeListFive.add(new ReplaceableAttribute("Year", "2000", false));
attributeListFive.add(new ReplaceableAttribute("Year", "2001", false));
attributeListFive.add(new ReplaceableAttribute("Year", "2002", false));

PutAttributesRequest putAttributesActionFive = new
    PutAttributesRequest(domainName, itemNameFive, attributeListFive);

invokePutAttributes(service, putAttributesActionFive);
}
```

3. Compile and run the sample.
The name-value pairs are stored in Amazon SimpleDB.

C#

To run the sample

1. Open `AmazonSimpleDBSamples.cs`.
2. Comment out the code you added in the previous section.
3. Replace the Put Attributes Action section with the following:

```
String domainName = "MyStore";
/*****
 * Load Item_01:
 *     Category = Clothes
 *     Subcategory = Sweater
 *     Name = Cathair Sweater
 *     Color = Siamese
 *     Size = Small, Medium, Large
 *****/
String itemNameOne = "Item_01";
List<ReplaceableAttribute> attributeListOne = new
    List<ReplaceableAttribute>(7);

attributeListOne.Add(new
    ReplaceableAttribute().WithName("Category").WithValue("Clothes"));
attributeListOne.Add(new
    ReplaceableAttribute().WithName("Subcategory").WithValue("Sweater"));
attributeListOne.Add(new
    ReplaceableAttribute().WithName("Name").WithValue("Cathair Sweater"));
attributeListOne.Add(new
    ReplaceableAttribute().WithName("Color").WithValue("Siamese"));
attributeListOne.Add(new
    ReplaceableAttribute().WithName("Size").WithValue("Small"));
attributeListOne.Add(new
    ReplaceableAttribute().WithName("Size").WithValue("Medium"));
attributeListOne.Add(new
    ReplaceableAttribute().WithName("Size").WithValue("Large"));

PutAttributesRequest putAttributesActionOne = new
    PutAttributesRequest().WithDomainName(domainName).WithItemName(itemNameOne);
putAttributesActionOne.Attribute = attributeListOne;

PutAttributesSample.InvokePutAttributes(service, putAttributesActionOne);

/*****
 * Load Item_02:
 *     Category = Clothes
 *     Subcategory = Pants
 *     Name = Designer Jeans
 *     Color = Paisley Acid Wash
 *     Size = 30x32, 32x32, 32x34
 *****/

String itemNameTwo = "Item_02";
List<ReplaceableAttribute> attributeListTwo = new
    List<ReplaceableAttribute>(7);
```

```
attributeListTwo.Add(new
    ReplaceableAttribute().WithName("Category").WithValue("Clothes"));
attributeListTwo.Add(new
    ReplaceableAttribute().WithName("Subcategory").WithValue("Pants"));
attributeListTwo.Add(new
    ReplaceableAttribute().WithName("Name").WithValue("Designer Jeans"));
attributeListTwo.Add(new
    ReplaceableAttribute().WithName("Color").WithValue("Paisley Acid Wash"));
attributeListTwo.Add(new
    ReplaceableAttribute().WithName("Size").WithValue("30x32"));
attributeListTwo.Add(new
    ReplaceableAttribute().WithName("Size").WithValue("32x32"));
attributeListTwo.Add(new
    ReplaceableAttribute().WithName("Size").WithValue("32x34"));

PutAttributesRequest putAttributesActionTwo = new
    PutAttributesRequest().WithDomainName(domainName).WithItemName(itemNameTwo);
putAttributesActionTwo.Attribute = attributeListTwo;

PutAttributesSample.InvokePutAttributes(service, putAttributesActionTwo);

/*****
 * Load Item_03:
 *     Category = Clothes
 *     Subcategory = Pants
 *     Name = Sweatpants
 *     Color = Blue, Yellow, Pink
 *     Size = Large
 *     Year = 2006, 2007
 *****/

String itemNameThree = "Item_03";
List<ReplaceableAttribute> attributeListThree = new
    List<ReplaceableAttribute>(9);

attributeListThree.Add(new
    ReplaceableAttribute().WithName("Category").WithValue("Clothes"));
attributeListThree.Add(new
    ReplaceableAttribute().WithName("Subcategory").WithValue("Pants"));
attributeListThree.Add(new
    ReplaceableAttribute().WithName("Name").WithValue("Sweatpants"));
attributeListThree.Add(new
    ReplaceableAttribute().WithName("Color").WithValue("Blue"));
attributeListThree.Add(new
    ReplaceableAttribute().WithName("Color").WithValue("Yellow"));
attributeListThree.Add(new
    ReplaceableAttribute().WithName("Color").WithValue("Pink"));
attributeListThree.Add(new
    ReplaceableAttribute().WithName("Size").WithValue("Large"));
attributeListThree.Add(new
    ReplaceableAttribute().WithName("Year").WithValue("2006"));
attributeListThree.Add(new
    ReplaceableAttribute().WithName("Year").WithValue("2007"));

PutAttributesRequest putAttributesActionThree = new
    PutAttributesRequest().WithDomainName(domainName).WithItemName(itemNameThree);
putAttributesActionThree.Attribute = attributeListThree;
```

```
PutAttributesSample.InvokePutAttributes(service, putAttributesActionThree);

/*****
 * Load Item_04:
 *     Category = Car Parts
 *     Subcategory = Engine
 *     Name = Turbos
 *     Make = Audi
 *     Model = S4
 *     Year = 2000, 2001, 2002
 *****/

String itemNameFour = "Item_04";
List<ReplaceableAttribute> attributeListFour = new
    List<ReplaceableAttribute>(8);

attributeListFour.Add(new
    ReplaceableAttribute().WithName("Category").WithValue("Car Parts"));
attributeListFour.Add(new
    ReplaceableAttribute().WithName("Subcategory").WithValue("Engine"));
attributeListFour.Add(new
    ReplaceableAttribute().WithName("Name").WithValue("Turbos"));
attributeListFour.Add(new
    ReplaceableAttribute().WithName("Make").WithValue("Audi"));
attributeListFour.Add(new
    ReplaceableAttribute().WithName("Model").WithValue("S4"));
attributeListFour.Add(new
    ReplaceableAttribute().WithName("Year").WithValue("2000"));
attributeListFour.Add(new
    ReplaceableAttribute().WithName("Year").WithValue("2001"));
attributeListFour.Add(new
    ReplaceableAttribute().WithName("Year").WithValue("2002"));

PutAttributesRequest putAttributesActionFour = new
    PutAttributesRequest().WithDomainName(domainName).WithItemName(itemNameFour);
putAttributesActionFour.Attribute = attributeListFour;

PutAttributesSample.InvokePutAttributes(service, putAttributesActionFour);

/*****
 * Load Item_05:
 *     Category = Car Parts
 *     Subcategory = Emissions
 *     Name = O2 Sensor
 *     Make = Audi
 *     Model = S4
 *     Year = 2000, 2001, 2002
 *****/

String itemNameFive = "Item_05";
List<ReplaceableAttribute> attributeListFive = new
    List<ReplaceableAttribute>(8);

attributeListFive.Add(new
    ReplaceableAttribute().WithName("Category").WithValue("Car Parts"));
attributeListFive.Add(new
    ReplaceableAttribute().WithName("Subcategory").WithValue("Emissions"));
attributeListFive.Add(new
    ReplaceableAttribute().WithName("Name").WithValue("O2 Sensor"));
```

```
attributeListFive.Add(new
    ReplaceableAttribute().WithName("Make").WithValue("Audi"));
attributeListFive.Add(new
    ReplaceableAttribute().WithName("Model").WithValue("S4"));
attributeListFive.Add(new
    ReplaceableAttribute().WithName("Year").WithValue("2000"));
attributeListFive.Add(new
    ReplaceableAttribute().WithName("Year").WithValue("2001"));
attributeListFive.Add(new
    ReplaceableAttribute().WithName("Year").WithValue("2002"));

PutAttributesRequest putAttributesActionFive = new
    PutAttributesRequest().WithDomainName(domainName).WithItemName(itemNameFive);
putAttributesActionFive.Attribute = attributeListFive;

PutAttributesSample.InvokePutAttributes(service, putAttributesActionFive);
```

4. Run the sample.

The name-value pairs are stored in Amazon SimpleDB.

Perl

To run the sample

1. Open a clean copy of PutAttributesSample.pl.
2. Add the following lines after the @TODO: set request line:

```
#####
# Load Item_01:
#     Category = Clothes
#     Subcategory = Sweater
#     Name = Cathair Sweater
#     Color = Siamese
#     Size = Small, Medium, Large
#####/

my $request = Amazon::SimpleDB::Model::PutAttributesRequest->new({
    ItemName => "Item_01",
    DomainName=> "MyStore",
    Attribute => [
        {
            Name => "Category",
            Value => "Clothes"
        },
        {
            Name => "Subcategory",
            Value => "Sweater"
        },
        {
            Name => "Name",
            Value => "Cathair Sweater"
        },
        {
            Name => "Color",
            Value => "Siamese"
        },
    ]
});
```

```
        Name => "Size",
        Value => "Small"
    },
    {
        Name => "Size",
        Value => "Medium"
    },
    {
        Name => "Size",
        Value => "Large"
    }
],
});

invokePutAttributes($service, $request);

#*****
# Load Item_02:
#     Category = Clothes
#     Subcategory = Pants
#     Name = Designer Jeans
#     Color = Paisley Acid Wash
#     Size = 30x32, 32x32, 32x34
#*****/

my $request = Amazon::SimpleDB::Model::PutAttributesRequest->new({
    ItemName => "Item_02",
    DomainName=> "MyStore",
    Attribute => [
        {
            Name => "Category",
            Value => "Clothes"
        },
        {
            Name => "Subcategory",
            Value => "Pants"
        },
        {
            Name => "Name",
            Value => "Designer Jeans"
        },
        {
            Name => "Color",
            Value => "Paisley Acid Wash"
        },
        {
            Name => "Size",
            Value => "30x32"
        },
        {
            Name => "Size",
            Value => "32x32"
        },
        {
            Name => "Size",
            Value => "32x34"
        }
    ],
});
```

```
invokePutAttributes($service, $request);

#*****
# Load Item_03:
#     Category = Clothes
#     Subcategory = Pants
#     Name = Sweatpants
#     Color = Blue, Yellow, Pink
#     Size = Large
#     Year = 2006, 2007
#*****/

my $request = Amazon::SimpleDB::Model::PutAttributesRequest->new({
    ItemName => "Item_03",
    DomainName=> "MyStore",
    Attribute => [
        {
            Name => "Category",
            Value => "Clothes"
        },
        {
            Name => "Subcategory",
            Value => "Pants"
        },
        {
            Name => "Name",
            Value => "Sweatpants"
        },
        {
            Name => "Color",
            Value => "Blue"
        },
        {
            Name => "Color",
            Value => "Yellow"
        },
        {
            Name => "Color",
            Value => "Pink"
        },
        {
            Name => "Size",
            Value => "Large"
        },
        {
            Name => "Year",
            Value => "2006"
        },
        {
            Name => "Year",
            Value => "2007"
        }
    ],
});
invokePutAttributes($service, $request);

#*****
# Load Item_04:
```

Amazon SimpleDB Getting Started Guide Perl

```
#      Category = Car Parts
#      Subcategory = Engine
#      Name = Turbos
#      Make = Audi
#      Model = S4
#      Year = 2000, 2001, 2002
# *****/

my $request = Amazon::SimpleDB::Model::PutAttributesRequest->new({
    ItemName => "Item_04",
    DomainName=> "MyStore",
    Attribute => [
        {
            Name => "Category",
            Value => "Car Parts"
        },
        {
            Name => "Subcategory",
            Value => "Engine"
        },
        {
            Name => "Name",
            Value => "Turbos"
        },
        {
            Name => "Make",
            Value => "Audi"
        },
        {
            Name => "Model",
            Value => "S4"
        },
        {
            Name => "Year",
            Value => "2000"
        },
        {
            Name => "Year",
            Value => "2001"
        },
        {
            Name => "Year",
            Value => "2002"
        }
    ],
});
invokePutAttributes($service, $request);

# *****/
# Load Item_05:
#      Category = Car Parts
#      Subcategory = Emissions
#      Name = O2 Sensor
#      Make = Audi
#      Model = S4
#      Year = 2000, 2001, 2002
# *****/

my $request = Amazon::SimpleDB::Model::PutAttributesRequest->new({
```

```
ItemName => "Item_05",
DomainName=> "MyStore",
Attribute => [
    {
        Name => "Category",
        Value => "Car Parts"
    },
    {
        Name => "Subcategory",
        Value => "Emissions"
    },
    {
        Name => "Name",
        Value => "O2 Sensor"
    },
    {
        Name => "Make",
        Value => "Audi"
    },
    {
        Name => "Model",
        Value => "S4"
    },
    {
        Name => "Year",
        Value => "2000"
    },
    {
        Name => "Year",
        Value => "2001"
    },
    {
        Name => "Year",
        Value => "2002"
    }
],
});
invokePutAttributes($service, $request);
```

3. Run the sample.

The name-value pairs are stored in Amazon SimpleDB.

PHP

To run the sample

1. Open a clean copy of `PutAttributesSample.php`.
2. Add the following lines after the `@TODO: set request` line:

```
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();
$domainName = 'MyStore';
/*****
* Load Item_01:
* Category = Clothes
* Subcategory = Sweater
* Name = Cathair Sweater
* Color = Siamese
```

Amazon SimpleDB Getting Started Guide PHP

```
* Size = Small, Medium, Large
*****/
$itemName = 'Item_01';
$attribute1 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute1->withName('Category')->withValue('Clothes');
$attribute2 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute2->withName('Subcategory')->withValue('Sweater');
$attribute3 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute3->withName('Name')->withValue('Cathair Sweater');
$attribute4 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute4->withName('Color')->withValue('Siamese');
$attribute5 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute5->withName('Size')->withValue('Small');
$attribute6 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute6->withName('Size')->withValue('Medium');
$attribute7 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute7->withName('Size')->withValue('Large');
$attributeList = array($attribute1, $attribute2, $attribute3, $attribute4,
$attribute5, $attribute6, $attribute7);
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();
$request->withDomainName($domainName)->withItemName($itemName)-
>setAttribute
($attributeList);
invokePutAttributes($service, $request);
/*****
* Load Item_02:
* Category = Clothes
* Subcategory = Pants
* Name = Designer Jeans
* Color = Paisley Acid Wash
* Size = 30x32, 32x32, 32x34
*****/
$itemName = 'Item_02';
$attribute1 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute1->withName('Category')->withValue('Clothes');
$attribute2 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute2->withName('Subcategory')->withValue('Pants');
$attribute3 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute3->withName('Name')->withValue('Designer Jeans');
$attribute4 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute4->withName('Color')->withValue('Paisley Acid Wash');
$attribute5 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute5->withName('Size')->withValue('30x32');
$attribute6 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute6->withName('Size')->withValue('32x32');
$attribute7 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute7->withName('Size')->withValue('32x34');
$attributeList = array($attribute1, $attribute2, $attribute3, $attribute4,
$attribute5, $attribute6, $attribute7);
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();
$request->withDomainName($domainName)->withItemName($itemName)-
>setAttribute
($attributeList);
invokePutAttributes($service, $request);
/*****
* Load Item_03:
* Category = Clothes
* Subcategory = Pants
* Name = Sweatpants
```

```
* Color = Blue, Yellow, Pink
* Size = Large
* Year = 2006, 2007
*****/
$itemName = 'Item_03';
$attribute1 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute1->withName('Category')->withValue('Clothes');
$attribute2 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute2->withName('Subcategory')->withValue('Pants');
$attribute3 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute3->withName('Name')->withValue('Sweatpants');
$attribute4 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute4->withName('Color')->withValue('Blue');
$attribute5 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute5->withName('Color')->withValue('Yellow');
$attribute6 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute6->withName('Color')->withValue('Pink');
$attribute7 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute7->withName('Size')->withValue('Large');
$attribute8 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute8->withName('Year')->withValue('2006');
$attribute9 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute9->withName('Year')->withValue('2007');
$attributeList = array($attribute1, $attribute2, $attribute3, $attribute4,
$attribute5, $attribute6, $attribute7, $attribute8, $attribute9);
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();
$request->withDomainName($domainName)->withItemName($itemName)-
>setAttribute
($attributeList);
invokePutAttributes($service, $request);
/*****/
* Load Item_04:
* Category = Car Parts
* Subcategory = Engine
* Name = Turbos
* Make = Audi
* Model = S4
* Year = 2000, 2001, 2002
*****/
$itemName = 'Item_04';
$attribute1 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute1->withName('Category')->withValue('Car Parts');
$attribute2 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute2->withName('Subcategory')->withValue('Engine');
$attribute3 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute3->withName('Name')->withValue('Turbos');
$attribute4 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute4->withName('Make')->withValue('Audi');
$attribute5 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute5->withName('Model')->withValue('Yellow');
$attribute6 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute6->withName('Year')->withValue('2000');
$attribute7 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute7->withName('Year')->withValue('2001');
$attribute8 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute8->withName('Year')->withValue('2002');
$attributeList = array($attribute1, $attribute2, $attribute3, $attribute4,
$attribute5, $attribute6, $attribute7, $attribute8);
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();
```

```
$request->withDomainName($domainName)->withItemName($itemName)->setAttribute($attributeList);
invokePutAttributes($service, $request);
/*****
* Load Item_05:
* Category = Car Parts
* Subcategory = Emissions
* Name = O2 Sensor
* Make = Audi
* Model = S4
* Year = 2000, 2001, 2002
*****/
$itemName = 'Item_05';
$attribute1 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute1->withName('Category')->withValue('Car Parts');
$attribute2 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute2->withName('Subcategory')->withValue('Emissions');
$attribute3 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute3->withName('Name')->withValue('O2 Sensor');
$attribute4 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute4->withName('Make')->withValue('Audi');
$attribute5 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute5->withName('Model')->withValue('Yellow');
$attribute6 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute6->withName('Year')->withValue('2000');
$attribute7 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute7->withName('Year')->withValue('2001');
$attribute8 = new Amazon_SimpleDB_Model_ReplaceableAttribute();
$attribute8->withName('Year')->withValue('2002');
$attributeList = array($attribute1, $attribute2, $attribute3, $attribute4,
$attribute5, $attribute6, $attribute7, $attribute8);
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();
$request->withDomainName($domainName)->withItemName($itemName)->setAttribute($attributeList);
```

3. Run the sample.

The name-value pairs are stored in Amazon SimpleDB.

VB.NET

To run the sample

1. Open AmazonSimpleDBSamples.vb.
2. Comment out the code you added in the previous section.
3. Replace the Put Attributes Action section with the following:

```
Dim domainName As String
domainName = "MyStore"
'-----
' Load Item_01:
'     Category = Clothes
'     Subcategory = Sweater
'     Name = Cathair Sweater
'     Color = Siamese
'     Size = Small, Medium, Large
'-----
```

```
Dim itemNameOne As String
itemNameOne = "Item_01"

Dim attributeItemOne_One As New ReplaceableAttribute
attributeItemOne_One.WithName("Category").WithValue("Clothes")
Dim attributeItemOne_Two As New ReplaceableAttribute
attributeItemOne_Two.WithName("Subcategory").WithValue("Sweater")
Dim attributeItemOne_Three As New ReplaceableAttribute
attributeItemOne_Three.WithName("Name").WithValue("Cathair Sweater")
Dim attributeItemOne_Four As New ReplaceableAttribute
attributeItemOne_Four.WithName("Color").WithValue("Siamese")
Dim attributeItemOne_Five As New ReplaceableAttribute
attributeItemOne_Five.WithName("Size").WithValue("Small")
Dim attributeItemOne_Six As New ReplaceableAttribute
attributeItemOne_Six.WithName("Size").WithValue("Medium")
Dim attributeItemOne_Seven As New ReplaceableAttribute
attributeItemOne_Seven.WithName("Size").WithValue("Large")

Dim putAttributesActionOne As New PutAttributesRequest()
putAttributesActionOne.WithDomainName(domainName).WithItemName(itemNameOne)
putAttributesActionOne.WithAttribute(attributeItemOne_One)
putAttributesActionOne.WithAttribute(attributeItemOne_Two)
putAttributesActionOne.WithAttribute(attributeItemOne_Three)
putAttributesActionOne.WithAttribute(attributeItemOne_Four)
putAttributesActionOne.WithAttribute(attributeItemOne_Five)
putAttributesActionOne.WithAttribute(attributeItemOne_Six)
putAttributesActionOne.WithAttribute(attributeItemOne_Seven)

PutAttributesSample.InvokePutAttributes(service, putAttributesActionOne)

'-----
' Load Item_02:
'     Category = Clothes
'     Subcategory = Pants
'     Name = Designer Jeans
'     Color = Paisley Acid Wash
'     Size = 30x32, 32x32, 32x34
'-----

Dim itemNameTwo As String
itemNameTwo = "Item_02"

Dim attributeItemTwo_One As New ReplaceableAttribute
attributeItemTwo_One.WithName("Category").WithValue("Clothes")
Dim attributeItemTwo_Two As New ReplaceableAttribute
attributeItemTwo_Two.WithName("Subcategory").WithValue("Pants")
Dim attributeItemTwo_Three As New ReplaceableAttribute
attributeItemTwo_Three.WithName("Name").WithValue("Designer Jeans")
Dim attributeItemTwo_Four As New ReplaceableAttribute
attributeItemTwo_Four.WithName("Color").WithValue("Paisley Acid Wash")
Dim attributeItemTwo_Five As New ReplaceableAttribute
attributeItemTwo_Five.WithName("Size").WithValue("30x32")
Dim attributeItemTwo_Six As New ReplaceableAttribute
attributeItemTwo_Six.WithName("Size").WithValue("32x32")
Dim attributeItemTwo_Seven As New ReplaceableAttribute
attributeItemTwo_Seven.WithName("Size").WithValue("32x34")

Dim putAttributesActionTwo As New PutAttributesRequest()
putAttributesActionTwo.WithDomainName(domainName).WithItemName(itemNameTwo)
putAttributesActionTwo.WithAttribute(attributeItemTwo_One)
```

```
putAttributesActionTwo.WithAttribute(attributeItemTwo_Two)
putAttributesActionTwo.WithAttribute(attributeItemTwo_Three)
putAttributesActionTwo.WithAttribute(attributeItemTwo_Four)
putAttributesActionTwo.WithAttribute(attributeItemTwo_Five)
putAttributesActionTwo.WithAttribute(attributeItemTwo_Six)
putAttributesActionTwo.WithAttribute(attributeItemTwo_Seven)

PutAttributesSample.InvokePutAttributes(service, putAttributesActionTwo)

'-----
' Load Item_03:
'     Category = Clothes
'     Subcategory = Pants
'     Name = Sweatpants
'     Color = Blue, Yellow, Pink
'     Size = Large
'     Year = 2006, 2007
'-----

Dim itemNameThree As String
itemNameThree = "Item_03"

Dim attributeItemThree_One As New ReplaceableAttribute
attributeItemThree_One.WithName("Category").WithValue("Clothes")
Dim attributeItemThree_Two As New ReplaceableAttribute
attributeItemThree_Two.WithName("Subcategory").WithValue("Pants")
Dim attributeItemThree_Three As New ReplaceableAttribute
attributeItemThree_Three.WithName("Name").WithValue("Sweatpants")
Dim attributeItemThree_Four As New ReplaceableAttribute
attributeItemThree_Four.WithName("Color").WithValue("Blue")
Dim attributeItemThree_Five As New ReplaceableAttribute
attributeItemThree_Five.WithName("Color").WithValue("Yellow")
Dim attributeItemThree_Six As New ReplaceableAttribute
attributeItemThree_Six.WithName("Color").WithValue("Pink")
Dim attributeItemThree_Seven As New ReplaceableAttribute
attributeItemThree_Seven.WithName("Size").WithValue("Large")
Dim attributeItemThree_Eight As New ReplaceableAttribute
attributeItemThree_Eight.WithName("Year").WithValue("2006")
Dim attributeItemThree_Nine As New ReplaceableAttribute
attributeItemThree_Nine.WithName("Year").WithValue("2007")

Dim putAttributesActionThree As New PutAttributesRequest()
putAttributesActionThree.WithDomainName(domainName).WithItemName(itemNameThree)
putAttributesActionThree.WithAttribute(attributeItemThree_One)
putAttributesActionThree.WithAttribute(attributeItemThree_Two)
putAttributesActionThree.WithAttribute(attributeItemThree_Three)
putAttributesActionThree.WithAttribute(attributeItemThree_Four)
putAttributesActionThree.WithAttribute(attributeItemThree_Five)
putAttributesActionThree.WithAttribute(attributeItemThree_Six)
putAttributesActionThree.WithAttribute(attributeItemThree_Seven)
putAttributesActionThree.WithAttribute(attributeItemThree_Eight)
putAttributesActionThree.WithAttribute(attributeItemThree_Nine)

PutAttributesSample.InvokePutAttributes(service, putAttributesActionThree)

'-----
' Load Item_04:
'     Category = Car Parts
'     Subcategory = Engine
'     Name = Turbos
'-----
```

```
'         Make = Audi
'         Model = S4
'         Year = 2000, 2001, 2002
'-----
Dim itemNameFour As String
itemNameFour = "Item_04"

Dim attributeItemFour_One As New ReplaceableAttribute
attributeItemFour_One.WithName("Category").WithValue("Car Parts")
Dim attributeItemFour_Two As New ReplaceableAttribute
attributeItemFour_Two.WithName("Subcategory").WithValue("Engine")
Dim attributeItemFour_Three As New ReplaceableAttribute
attributeItemFour_Three.WithName("Name").WithValue("Turbos")
Dim attributeItemFour_Four As New ReplaceableAttribute
attributeItemFour_Four.WithName("Make").WithValue("Audi")
Dim attributeItemFour_Five As New ReplaceableAttribute
attributeItemFour_Five.WithName("Model").WithValue("S4")
Dim attributeItemFour_Six As New ReplaceableAttribute
attributeItemFour_Six.WithName("Year").WithValue("2000")
Dim attributeItemFour_Seven As New ReplaceableAttribute
attributeItemFour_Seven.WithName("Year").WithValue("2001")
Dim attributeItemFour_Eight As New ReplaceableAttribute
attributeItemFour_Eight.WithName("Year").WithValue("2002")

Dim putAttributesActionFour As New PutAttributesRequest()
putAttributesActionFour.WithDomainName(domainName).WithItemName(itemNameFour)
putAttributesActionFour.WithAttribute(attributeItemFour_One)
putAttributesActionFour.WithAttribute(attributeItemFour_Two)
putAttributesActionFour.WithAttribute(attributeItemFour_Three)
putAttributesActionFour.WithAttribute(attributeItemFour_Four)
putAttributesActionFour.WithAttribute(attributeItemFour_Five)
putAttributesActionFour.WithAttribute(attributeItemFour_Six)
putAttributesActionFour.WithAttribute(attributeItemFour_Seven)
putAttributesActionFour.WithAttribute(attributeItemFour_Eight)

PutAttributesSample.InvokePutAttributes(service, putAttributesActionFour)

'-----
' Load Item_05:
'         Category = Car Parts
'         Subcategory = Emissions
'         Name = O2 Sensor
'         Make = Audi
'         Model = S4
'         Year = 2000, 2001, 2002
'-----
Dim itemNameFive As String
itemNameFive = "Item_05"

Dim attributeItemFive_One As New ReplaceableAttribute
attributeItemFive_One.WithName("Category").WithValue("Car Parts")
Dim attributeItemFive_Two As New ReplaceableAttribute
attributeItemFive_Two.WithName("Subcategory").WithValue("Emissions")
Dim attributeItemFive_Three As New ReplaceableAttribute
attributeItemFive_Three.WithName("Name").WithValue("O2 Sensor")
Dim attributeItemFive_Four As New ReplaceableAttribute
attributeItemFive_Four.WithName("Make").WithValue("Audi")
Dim attributeItemFive_Five As New ReplaceableAttribute
attributeItemFive_Five.WithName("Model").WithValue("S4")
```

```
Dim attributeItemFive_Six As New ReplaceableAttribute
attributeItemFive_Six.WithName("Year").WithValue("2000")
Dim attributeItemFive_Seven As New ReplaceableAttribute
attributeItemFive_Seven.WithName("Year").WithValue("2001")
Dim attributeItemFive_Eight As New ReplaceableAttribute
attributeItemFive_Eight.WithName("Year").WithValue("2002")

Dim putAttributesActionFive As New PutAttributesRequest()
putAttributesActionFive.WithDomainName(domainName).WithItemName(itemNameFive)
putAttributesActionFive.WithAttribute(attributeItemFive_One)
putAttributesActionFive.WithAttribute(attributeItemFive_Two)
putAttributesActionFive.WithAttribute(attributeItemFive_Three)
putAttributesActionFive.WithAttribute(attributeItemFive_Four)
putAttributesActionFive.WithAttribute(attributeItemFive_Five)
putAttributesActionFive.WithAttribute(attributeItemFive_Six)
putAttributesActionFive.WithAttribute(attributeItemFive_Seven)
putAttributesActionFive.WithAttribute(attributeItemFive_Eight)

PutAttributesSample.InvokePutAttributes(service, putAttributesActionFive)
```

4. Run the sample.

The name-value pairs are stored in Amazon SimpleDB.

Scratchpad

To run the sample

1. Open the scratchpad application with a web browser.
2. Add the attribute-value pairs for Item_01:
 - a. Select **PutAttributes** from the **Explore API** list box.
 - b. Enter `MyStore` in the **Domain Name** field.
 - c. Enter `Item_01` in the **Item Name** field.
 - d. For the first name-value pair, enter `Category` in the **Name** field, `clothes` in the **Value** field, and `false` in the **Replace** field.
 - e. Click the plus button (+) to add each of the following additional name-value pairs:
 - Subcategory: Sweater
 - Name: Cathair Sweater
 - Color: Siamese
 - Size: Small
 - Size: Medium
 - Size: Large
 - f. Select from the following:
 - To invoke the request, click **Invoke Request**.
The name-value pairs are stored in Amazon SimpleDB
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
The name-value pairs are stored in Amazon SimpleDB
 - To view the string to sign, click **Display String to Sign**.
3. Add the attribute-value pairs for Item_02:
 - a. Select **PutAttributes** from the **Explore API** list box.
 - b. Enter `MyStore` in the **Domain Name** field.
 - c. Enter `Item_02` in the **Item Name** field.

- d. For the first name-value pair, enter `Category` in the **Name** field, `Clothes` in the **Value** field, and `false` in the **Replace** field.
 - e. Click the plus button (+) to add each of the following additional name-value pairs:
 - Subcategory: Pants
 - Name: Designer Jeans
 - Color: Paisley Acid Wash
 - Size: 30x32
 - Size: 32x32
 - Size: 32x34
 - f. Select from the following:
 - To invoke the request, click **Invoke Request**.
The name-value pairs are stored in Amazon SimpleDB
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
The name-value pairs are stored in Amazon SimpleDB
 - To view the string to sign, click **Display String to Sign**.
4. Add the attribute-value pairs for Item_03:
- a. Select **PutAttributes** from the **Explore API** list box.
 - b. Enter `MyStore` in the **Domain Name** field.
 - c. Enter `Item_03` in the **Item Name** field.
 - d. For the first name-value pair, enter `Category` in the **Name** field, `Clothes` in the **Value** field, and `false` in the **Replace** field.
 - e. Click the plus button (+) to add each of the following additional name-value pairs:
 - Subcategory: Pants
 - Name: Sweatpants
 - Color: Blue
 - Color: Yellow
 - Color: Pink
 - Size: Large
 - Year: 2006
 - Year: 2007
 - f. Select from the following:
 - To invoke the request, click **Invoke Request**.
The name-value pairs are stored in Amazon SimpleDB
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
The name-value pairs are stored in Amazon SimpleDB
 - To view the string to sign, click **Display String to Sign**.
5. Add the attribute-value pairs for Item_04:
- a. Select **PutAttributes** from the **Explore API** list box.
 - b. Enter `MyStore` in the **Domain Name** field.
 - c. Enter `Item_04` in the **Item Name** field.
 - d. For the first name-value pair, enter `Category` in the **Name** field, `Car Parts` in the **Value** field, and `false` in the **Replace** field.
 - e. Click the plus button (+) to add each of the following additional name-value pairs:
 - Subcategory: Engine
 - Name: Turbos

- Make: Audi
 - Model: S4
 - Year: 2000
 - Year: 2001
 - Year: 2002
- f. Select from the following:
- To invoke the request, click **Invoke Request**.
The name-value pairs are stored in Amazon SimpleDB
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
The name-value pairs are stored in Amazon SimpleDB
 - To view the string to sign, click **Display String to Sign**.
6. Add the attribute-value pairs for Item_05:
- a. Select **PutAttributes** from the **Explore API** list box.
 - b. Enter `myStore` in the **Domain Name** field.
 - c. Enter `Item_05` in the **Item Name** field.
 - d. For the first name-value pair, enter `category` in the **Name** field, `car parts` in the **Value** field, and `false` in the **Replace** field.
 - e. Click the plus button (+) to add each of the following additional name-value pairs:
 - Subcategory: Emissions
 - Name: O2 Sensor
 - Make: Audi
 - Model: S4
 - Year: 2000
 - Year: 2001
 - Year: 2002
 - f. Select from the following:
 - To invoke the request, click **Invoke Request**.
The name-value pairs are stored in Amazon SimpleDB
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
The name-value pairs are stored in Amazon SimpleDB
 - To view the string to sign, click **Display String to Sign**.

Getting Data from a Domain

In the previous sections, attributes were added from the sample. This section provides examples of how to write a query using the `Select` operation.



Note

For information about constructing complex `Select` query statements, refer to the *Amazon SimpleDB Developer Guide*.

The following sample code snippets demonstrate querying for items with the attribute `Category` that contain the value `Clothes`.

Java

To run the sample

1. Open a clean copy of `SelectSample.java`.
2. Remove any comment marks from the `invokeSelect(service, request);` line and add the following lines after `// @TODO: set request parameters here:`

```
String selectExpression = "Select * From MyStore Where Category =  
  'Clothes'";  
request.withSelectExpression(selectExpression);
```

3. Compile and run the sample.
Amazon SimpleDB returns items with the attribute *Category* that contain the value *Clothes*.

C#

To run the sample

1. Open `AmazonSimpleDBSamples.cs`.
2. Comment out the code you added in the previous section.
3. Add the following to the Select Action section:

```
String selectExpression = "Select * From MyStore Where Category =  
  'Clothes'";  
SelectRequest selectRequestAction = new  
  SelectRequest().WithSelectExpression(selectExpression);  
SelectSample.InvokeSelect(service, selectRequestAction);
```

4. Run the sample.
Amazon SimpleDB returns items with the attribute *Category* that contain the value *Clothes*.

Perl

To run the sample

1. Open a clean copy of `SelectSample.pl`.
2. Remove any comment marks from the `invokeSelect($service, $request);` line and add the following lines after the `// @TODO: set request parameters here line:`

```
my $request = Amazon::SimpleDB::Model::SelectRequest->new({SelectExpression  
  => "Select * From MyStore Where Category = 'Clothes'"});
```

3. Run the sample.
Amazon SimpleDB returns items with the attribute *Category* that contain the value *Clothes*.

PHP

To run the sample

1. Open a clean copy of `SelectSample.php`.
2. Remove any comment marks from the `invokeSelect($service, $request);` line and add the following lines after the `// @TODO: set request parameters here line:`

```
$request = new Amazon_SimpleDB_Model_SelectRequest();  
$request->setSelectExpression("Select * From MyStore Where Category =  
'Clothes'");
```

3. Run the sample.
Amazon SimpleDB returns items with the attribute *Category* that contain the value *Clothes*.

VB.NET

To run the sample

1. Open `AmazonSimpleDBSamples.vb`.
2. Comment out the code you added in the previous section.
3. Add the following to the Select Action section:

```
Dim selectAction As New SelectRequest()  
selectAction.WithSelectExpression("Select * From MyStore Where Category =  
'Clothes'")  
SelectSample.InvokeSelect(service, selectAction)
```

4. Run the sample.
Amazon SimpleDB returns items with the attribute *Category* that contain the value *Clothes*.

Scratchpad

To run the sample

1. Open the scratchpad application with a web browser.
2. Select **Select** from the **Explore API** list box.
3. Enter `mystore` in the **Domain Name** field.
4. Enter the following in the **Query Expression** field: `['Category' = 'Clothes']`.
5. To limit the number of items returned, enter a value in the **Max Number Of Items** field.
If there are more items than you specified in the **Max Number Of Items** field, Amazon SimpleDB returns a next token.
6. If you need to view the next page of results, enter the next token in the **Next Token** field.



Note

To get the next page of results, you only need to specify the domain name and the next token.

7. Select from the following:
 - To invoke the request, click **Invoke Request**.
Amazon SimpleDB returns items with the attribute *Category* that contain the value *Clothes*.
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
Amazon SimpleDB returns items with the attribute *Category* that contain the value *Clothes*.
 - To view the string to sign, click **Display String to Sign**.

Deleting Values from an Attribute

Previously, the *Color* attribute for *Item_03* was assigned three values (Blue, Yellow, and Pink). This section provides examples of how to delete the value of an attribute for the sample item.

The following sample code snippets demonstrate deletion of the value *Blue* from the *Color* attribute of *Item_03*. The other color values will remain (i.e., Yellow, Pink).

Java

To run the sample

1. Open a clean copy of `DeleteAttributesSample.java`.
2. Remove any comment marks from the `invokeDeleteAttributes(service, request);` line and add the following lines after `// @TODO: set request parameters here:`

```
String domainName = "MyStore";
String itemName = "Item_03";
Attribute attribute = new Attribute("Color", "Blue");

request.withDomainName(domainName).withItemName(itemName).withAttribute(attribute);
```

3. Compile and run the sample.
Amazon SimpleDB deletes the *Blue* value from the *Color* attribute of *Item_03*.

C#

To run the sample

1. Open `AmazonSimpleDBSamples.cs`.
2. Comment out the code you added in the previous section.
3. Replace the Delete Attributes Action section with the following:

```
Amazon.SimpleDB.Model.Attribute deleteValueAttribute = new
    Amazon.SimpleDB.Model.Attribute().WithName("Color").WithValue("Blue");
DeleteAttributesRequest deleteValueAction = new
    DeleteAttributesRequest().WithDomainName("MyStore").WithItemName("Item_03").WithAttribute(deleteValueAttribute);

DeleteAttributesSample.InvokeDeleteAttributes(service, deleteValueAction);
```

4. Run the sample.
Amazon SimpleDB deletes the *Blue* value from the *Color* attribute of *Item_03*.

Perl

To run the sample

1. Open a clean copy of `DeleteAttributesSample.pl`.
2. Remove any comment marks from the `invokeDeleteAttributes($service, $request);` line and add the following lines after the `// @TODO: set request line:`

```
my $request = Amazon::SimpleDB::Model::DeleteAttributesRequest->new({
    ItemName => "Item_03",
    DomainName=> "MyStore",
```

```
Attribute => [  
  {  
    Name => "Color",  
    Value => "Blue"  
  }  
],  
});
```

3. Run the sample.

Amazon SimpleDB deletes the *Blue* value from the *Color* attribute of *Item_03*.

PHP

To run the sample

1. Open a clean copy of `DeleteAttributesSample.php`.
2. Remove any comment marks from the `invokeDeleteAttributes($service, $request);` line and add the following lines after the `// @TODO: set request` line:

```
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();  
  
$attribute = new Amazon_SimpleDB_Model_Attribute();  
$attribute->withName('Color')->withValue('Blue');  
  
$request = new Amazon_SimpleDB_Model_DeleteAttributesRequest();  
$request->withDomainName('MyStore')->withItemName('Item_03')->  
withAttribute($attribute);
```

3. Run the sample.

Amazon SimpleDB deletes the *Blue* value from the *Color* attribute of *Item_03*.

VB.NET

To run the sample

1. Open `AmazonSimpleDBSamples.vb`.
2. Comment out the code you added in the previous section.
3. Replace the Delete Attributes Action section with the following:

```
Dim deleteValueAttribute As New Amazon.SimpleDB.Model.Attribute  
deleteValueAttribute.WithName("Color").WithValue("Blue")  
  
Dim deleteValueAction As New DeleteAttributesRequest()  
deleteValueAction.WithDomainName("MyStore").WithItemName("Item_03").WithAttribute(deleteV  
  
DeleteAttributesSample.InvokeDeleteAttributes(service, deleteValueAction)
```

4. Run the sample.

Amazon SimpleDB deletes the *Blue* value from the *Color* attribute of *Item_03*.

Scratchpad

To run the sample

1. Open the scratchpad application with a web browser.

2. Select **DeleteAttributes** from the **Explore API** list box.
3. Enter `MyStore` in the **Domain Name** field.
4. Enter `Color` in the **Name** field and `Blue` in the **Value** field.
5. Select from the following:
 - To invoke the request, click **Invoke Request**.
Amazon SimpleDB deletes the *Blue* value from the *Color* attribute of *Item_03*.
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
Amazon SimpleDB deletes the *Blue* value from the *Color* attribute of *Item_03*.
 - To view the string to sign, click **Display String to Sign**.

Deleting an Attribute

In the previous section, the *Blue* value was deleted from the *Color* attribute of *Item_03*. This section provides examples of how to delete all the values of an attribute for an item.

The following sample code snippets demonstrate deletion of all values for the *Year* attribute of *Item_03*.

Java

To run the sample

1. Open a clean copy of `DeleteAttributesSample.java`.
2. Remove any comment marks from the `invokeDeleteAttributes(service, request);` line and add the following lines after `// @TODO: set request parameters here:`

```
String domainName = "MyStore";
String itemName = "Item_03";
Attribute attribute = (new Attribute()).withName("Year");

request.withDomainName(domainName).withItemName(itemName).withAttribute(attribute);
```

3. Compile and run the sample.
Amazon SimpleDB deletes the *Year* attribute of *Item_03*.

C#

To run the sample

1. Open `AmazonSimpleDBSamples.cs`.
2. Comment out the code you added in the previous section.
3. Add the following to the Delete Attributes Action section:

```
Amazon.SimpleDB.Model.Attribute deleteAttribute = new
    Amazon.SimpleDB.Model.Attribute().WithName("Year");
DeleteAttributesRequest deleteAttributeAction = new
    DeleteAttributesRequest().WithDomainName("MyStore").WithItemName("Item_03").WithAttribut

DeleteAttributesSample.InvokeDeleteAttributes(service,
    deleteAttributeAction);
```

4. Run the sample.
Amazon SimpleDB deletes the *Year* attribute of *Item_03*.

Perl

To run the sample

1. Open a clean copy of `DeleteAttributesSample.pl`.
2. Remove any comment marks from the `invokeDeleteAttributes($service, $request);` line and add the following lines after the `// @TODO: set request` line:

```
my $request = Amazon::SimpleDB::Model::DeleteAttributesRequest->new({
    ItemName => "Item_03",
    DomainName=> "MyStore",
    Attribute => [
        {
            Name => "Year"
        }
    ],
});
```

3. Run the sample.
Amazon SimpleDB deletes the *Year* attribute of *Item_03*.

PHP

To run the sample

1. Open a clean copy of `DeleteAttributesSample.php`.
2. Remove any comment marks from the `invokeDeleteAttributes($service, $request);` line and add the following lines after the `// @TODO: set request` line:

```
$request = new Amazon_SimpleDB_Model_DeleteAttributesRequest();

$attribute = new Amazon_SimpleDB_Model_Attribute();
$attribute->setName('Year');

$request = new Amazon_SimpleDB_Model_DeleteAttributes();
$request->withDomainName('MyStore')->withItemName('Item_03')->withAttribute($attribute);
```

3. Run the sample.
Amazon SimpleDB deletes the *Year* attribute of *Item_03*.

VB.NET

To run the sample

1. Open `AmazonSimpleDBSamples.vb`.
2. Comment out the code you added in the previous section.
3. Add the following to the Delete Attributes Action section:

```
Dim deleteAttribute As New Amazon.SimpleDB.Model.Attribute
deleteAttribute.WithName("Year")
```

```
Dim deleteAttributeAction As New DeleteAttributesRequest()  
deleteAttributeAction.WithDomainName("MyStore").WithItemName("Item_03").WithAttribute(del  
  
DeleteAttributesSample.InvokeDeleteAttributes(service,  
deleteAttributeAction)
```

4. Run the sample.

Amazon SimpleDB deletes the *Year* attribute of *Item_03*.

Scratchpad

To run the sample

1. Open the scratchpad application with a web browser.
2. Select **DeleteAttributes** from the **Explore API** list box.
3. Enter `MyStore` in the **Domain Name** field.
4. Enter `Item_03` in the **Item Name** field.
5. Enter `Year` in the **Name** field.
6. Select from the following:
 - To invoke the request, click **Invoke Request**.
Amazon SimpleDB deletes the *Year* attribute of *Item_03*.
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
Amazon SimpleDB deletes the *Year* attribute of *Item_03*.
 - To view the string to sign, click **Display String to Sign**.

Replacing an Attribute

In the previous section, all *Year* values were deleted from the *Item_03*. This section provides examples of how to replace the value of an attribute for an item.

The following sample code snippets demonstrate writing the value *Medium* to the *Size* attribute of *Item_03*, overwriting all existing values.

Java

To run the sample

1. Open a clean copy of `PutAttributesSample.java`.
2. Remove any comment marks from the `invokePutAttributes(service, request);` line and add the following lines after `// @TODO: set request parameters here:`

```
String domainName = "MyStore";  
String itemName = "Item_03";  
ReplaceableAttribute attribute = new ReplaceableAttribute("Size", "Medium",  
true);
```

```
request.withDomainName(domainName).withItemName(itemName).withAttribute(attribute);
```

3. Compile and run the sample.

Amazon SimpleDB writes the value *Medium* to the *Size* attribute of *Item_03*, overwriting all existing values.

C#

To run the sample

1. Open `AmazonSimpleDBSamples.cs`.
2. Comment out the code you added in the previous section.
3. Add the following to the Put Attributes Action section:

```
ReplaceableAttribute replaceableAttribute = new
    ReplaceableAttribute().WithName("Size").WithValue("Medium").WithReplace(true);
PutAttributesRequest replaceAction = new
    PutAttributesRequest().WithDomainName("MyStore").WithItemName("Item_03").WithAttribute(r

PutAttributesSample.InvokePutAttributes(service, replaceAction);
```

4. Run the sample.

Amazon SimpleDB writes the value *Medium* to the *Size* attribute of *Item_03*, overwriting all existing values.

Perl

To run the sample

1. Open a clean copy of `PutAttributesSample.pl`.
2. Remove any comment marks from the `invokePutAttributes($service, $request);` line and add the following lines after the `// @TODO: set request` line:

```
my $request = Amazon::SimpleDB::Model::PutAttributesRequest->new({
    ItemName => "Item_03",
    DomainName=> "MyStore",
    Attribute => [
        {
            Name => "Size",
            Value => "Medium",
            Replace => 1
        }
    ],
});
```

3. Run the sample.

Amazon SimpleDB writes the value *Medium* to the *Size* attribute of *Item_03*, overwriting all existing values.

PHP

To run the sample

1. Open a clean copy of `PutAttributesSample.php`.
2. Remove any comment marks from the `invokePutAttributes($service, $request);` line and add the following lines after the `// @TODO: set request` line:

```
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();  
  
$attribute = new Amazon_SimpleDB_Model_ReplaceableAttribute();  
$attribute->withName('Size')->withValue('Medium')->withReplace(true);  
  
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();  
$request->withDomainName('MyStore')->withItemName('Item_03')->  
>withAttribute($attribute);
```

3. Run the sample.

Amazon SimpleDB writes the value *Medium* to the *Size* attribute of *Item_03*, overwriting all existing values.

VB.NET

To run the sample

1. Open `AmazonSimpleDBSamples.vb`.
2. Comment out the code you added in the previous section.
3. Add the following to the Put Attributes Action section:

```
Dim replaceableAttribute As New ReplaceableAttribute  
replaceableAttribute.WithName("Size").WithValue("Medium").WithReplace(True)  
  
Dim replaceAttributeAction As New PutAttributesRequest()  
replaceAttributeAction.WithDomainName("MyStore").WithItemName("Item_03").WithAttribute(re  
  
PutAttributesSample.InvokePutAttributes(service, replaceAttributeAction)
```

4. Run the sample.

Amazon SimpleDB writes the value *Medium* to the *Size* attribute of *Item_03*, overwriting all existing values.

Scratchpad

To run the sample

1. Open the scratchpad application with a web browser.
2. Select **PutAttributes** from the **Explore API** list box.
3. Enter `MyStore` in the **Domain Name** field.
4. Enter `Item_03` in the **Item Name** field.
5. Enter `size` in the **Name** field.
6. Enter `medium` in the **Value** field.
7. Enter `true` in the **Replace** field.
8. Select from the following:
 - To invoke the request, click **Invoke Request**.
Amazon SimpleDB writes the value *Medium* to the *Size* attribute of *Item_03*, overwriting all existing values.
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
Amazon SimpleDB writes the value *Medium* to the *Size* attribute of *Item_03*, overwriting all existing values.

- To view the string to sign, click **Display String to Sign**.

Deleting an Item

In the previous sections, attributes were added and deleted from the sample item and queries were executed against the domain. This section provides examples of how to delete an item by deleting all of its attributes.

The following sample code snippets demonstrate deleting *Item_03*.

Java

To run the sample

1. Open a clean copy of `DeleteAttributesSample.java`.
2. Remove any comment marks from the `invokeDeleteAttributes(service, request);` line and add the following lines after `// @TODO: set request parameters here`:

```
String domainName = "MyStore";  
String itemName = "Item_03";  
request.withDomainName(domainName).withItemName(itemName).withAttribute(attribute);
```

3. Compile and run the sample.

Amazon SimpleDB deletes *Item_03*.

C#

To run the sample

1. Open `AmazonSimpleDBSamples.cs`.
2. Comment out the code you added in the previous section.
3. Add the following to the Delete Attributes Action section:

```
DeleteAttributesRequest deleteItemAction = new  
DeleteAttributesRequest().WithDomainName("MyStore").WithItemName("Item_03");  
  
DeleteAttributesSample.InvokeDeleteAttributes(service, deleteItemAction);
```

4. Run the sample.

Amazon SimpleDB deletes *Item_03*.

Perl

To run the sample

1. Open a clean copy of `DeleteAttributesSample.pl`.
2. Remove any comment marks from the `invokeDeleteAttributes($service, $request);` line and add the following lines after the `// @TODO: set request parameters here` line:

```
my $request = Amazon::SimpleDB::Model::DeleteAttributesRequest->new({  
  ItemName => "Item_03",  
  DomainName=> "MyStore"
```

```
});
```

3. Run the sample.
Amazon SimpleDB deletes *Item_03*.

PHP

To run the sample

1. Open a clean copy of `DeleteAttributesSample.php`.
2. Remove any comment marks from the `invokeDeleteAttributes($service, $request);` line and add the following lines after the `// @TODO: set request parameters here line:`

```
$request = new Amazon_SimpleDB_Model_PutAttributesRequest();  
  
$request = new Amazon_SimpleDB_Model_DeleteAttributesRequest();  
$request->withDomainName('MyStore')->withItemName('Item_03');
```

3. Run the sample.
Amazon SimpleDB deletes *Item_03*.

VB.NET

To run the sample

1. Open `AmazonSimpleDBSamples.vb`.
2. Comment out the code you added in the previous section.
3. Add the following to the Delete Attributes Action section:

```
Dim deleteItemAction As New DeleteAttributesRequest()  
deleteItemAction.WithDomainName("MyStore").WithItemName("Item_03")  
  
DeleteAttributesSample.InvokeDeleteAttributes(service, deleteItemAction)
```

4. Run the sample.
Amazon SimpleDB deletes *Item_03*.

Scratchpad

To run the sample

1. Open the scratchpad application with a web browser.
2. Select **DeleteAttributes** from the **Explore API** list box.
3. Enter `myStore` in the **Domain Name** field.
4. Enter `Item_03` in the **Item Name** field.
5. Select from the following:
 - To invoke the request, click **Invoke Request**.
Amazon SimpleDB deletes *Item_03*.
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
Amazon SimpleDB deletes *Item_03*.
 - To view the string to sign, click **Display String to Sign**.

Deleting a Domain

This section provides examples of how to delete a domain.

The following sample code snippets demonstrate deleting the MyStore domain.

Java

To run the sample

1. Open a clean copy of `DeleteDomainSample.java`.
2. Remove any comment marks from the `invokeDeleteDomain(service, request);` line and add the following line after `// @TODO: set request parameters here:`

```
request.withDomainName("MyStore");
```

3. Compile and run the sample.
Amazon SimpleDB deletes MyStore.

To verify whether the domain is deleted, see [Listing Domains \(p. 22\)](#).

C#

To run the sample

1. Open `AmazonSimpleDBSamples.cs`.
2. Comment out the code you added in the previous section.
3. Add the following to the Delete Domain Action section:

```
DeleteDomainRequest deleteDomainAction = new  
DeleteDomainRequest().WithDomainName("MyStore");  
  
DeleteDomainSample.InvokeDeleteDomain(service, deleteDomainAction);
```

4. Run the sample.
Amazon SimpleDB deletes MyStore.

To verify whether the domain is deleted, see [Listing Domains \(p. 22\)](#).

Perl

To run the sample

1. Open a clean copy of `DeleteDomainSample.pl`.
2. Remove any comment marks from the `invokeDeleteDomain($service, $request);` line and add the following line after the `// @TODO: set request line:`

```
my $request = Amazon::SimpleDB::Model::DeleteDomainRequest->new({DomainName  
=> "MyStore"});
```

3. Run the sample.
Amazon SimpleDB deletes MyStore.

To verify whether the domain is deleted, see [Listing Domains \(p. 22\)](#).

PHP

To run the sample

1. Open a clean copy of `DeleteDomainSample.php`.
2. Remove any comment marks from the `invokeDeleteDomain($service, $request);` line and add the following lines after the `// @TODO: set request` line:

```
$request = new Amazon_SimpleDB_Model_DeleteDomainRequest();  
$request->setDomainName('MyStore');
```

3. Run the sample.
Amazon SimpleDB deletes MyStore.

To verify whether the domain is deleted, see [Listing Domains \(p. 22\)](#).

VB.NET

To run the sample

1. Open `AmazonSimpleDBSamples.vb`.
2. Comment out the code you added in the previous section.
3. Add the following to the Delete Domain Action section:

```
Dim deleteDomainAction As New DeleteDomainRequest()  
deleteDomainAction.WithDomainName("MyStore")
```

```
DeleteDomainSample.InvokeDeleteDomain(service, deleteDomainAction)
```

4. Run the sample.
Amazon SimpleDB deletes MyStore.

To verify whether the domain is deleted, see [Listing Domains \(p. 22\)](#).

Scratchpad

To run the sample

1. Open the scratchpad application with a web browser.
2. Select **DeleteDomain** from the **Explore API** list box.
3. Enter `MyStore` in the **Domain Name** field.
4. Click **Invoke Request**. The scratchpad returns a signed URL
5. Select from the following:
 - To invoke the request, click **Invoke Request**.
Amazon SimpleDB returns a response.
 - To view the signed URL, click **Display Signed URL**. Then, copy and paste the signed URL into a browser.
Amazon SimpleDB returns a response.
 - To view the string to sign, click **Display String to Sign**.

To verify whether the domain is deleted, see [Listing Domains \(p. 22\)](#).

Where Do I Go from Here?

Now that you have completed the basic example presented in this guide, you are ready to start designing your own application. Although, most applications built upon Amazon SimpleDB are not as simple as the example in this guide, the principles used in the example can be readily applied to more complex applications.

The following is a list of common customer requirements and a description of where to go for more information.

- **I need to understand how to execute complex queries**—Refer to the *Select* section of the *Amazon SimpleDB Developer Guide*
- **I need to understand how quickly data is stored data is recorded to Amazon SimpleDB**—Refer to the *Eventual Consistency* section of the *Amazon SimpleDB Developer Guide*
- **Just give me the APIs**—Refer to the *API Reference* section of the *Amazon SimpleDB Developer Guide*
- **Tell me about Amazon SimpleDB data limits**—Refer to the *Amazon SimpleDB Limits* section of the *Amazon SimpleDB Developer Guide*
- **Tell me more about how to authenticate requests**—Refer to the *Request Authentication* section of the *Amazon SimpleDB Developer Guide*
- **I need support**—Join the [Amazon SimpleDB developer forums](#) where you can ask questions and get answers.

Canceling Amazon SimpleDB

If you decide Amazon SimpleDB isn't what you need, you can cancel your registration at any time.

To cancel an AWS product

1. Sign on to AWS.
2. Point to **Your Account** and click **Account Activity**.
3. From the **Account Activity** page, click the **View/Edit Service** link under the service you want to cancel.
4. On the **View/Edit Service** page, click the link to **cancel this service**.

Amazon SimpleDB Glossary

account	AWS account associated with a particular developer.
attribute	Similar to columns on a spreadsheet, attributes represent categories of data that can be assigned to items.
domain	<p>All Amazon SimpleDB information is stored in domains. Domains are similar to tables that contain similar data. You can execute queries against a domain, but cannot execute joins between domains.</p> <p>A customer can have up to 100 domains at any time. The name of the domain must be unique within the customer account.</p>
eventual consistency	<p>Amazon SimpleDB keeps multiple copies of each domain. When data is written or updated (using PutAttributes, DeleteAttributes, CreateDomain or DeleteDomain) and "Success" is returned, all copies of the data are updated. However, it takes time for the data to propagate to all storage locations. The data will eventually be consistent, but an immediate read might not show the change.</p> <p>Consistency is usually reached within seconds, but a high system load or network partition might increase this time. Performing a read after a short period of time should return the updated data.</p>
exponential backoff	A strategy for reducing the load on the system and increasing the likelihood of repeated requests succeeding by incrementally decreasing the rate at which retries are executed. For example, client applications might wait 400 milliseconds before attempting the first retry, 1600 milliseconds before the second, 6400 milliseconds (6.4 seconds) before the third, and so on.
items	Similar to rows on a spreadsheet, items represent individual objects that contain one or more value-attribute pairs
item name	An identifier for an item. The identifier must be unique within the domain.
machine utilization	Charges based on the amount of machine capacity used to complete the particular request (QUERY, GET, PUT, etc.),

	normalized to the hourly capacity of a 1.7 GHz Xeon processor. Machine Utilization is measured in Machine Hour increments.
multi-valued attribute	An attribute with more than one value.
network partition	A rare error condition where some Amazon SimpleDB computers cannot contact each other, but all other components are operating correctly. Normally this is repaired within seconds or minutes.
single-valued attribute	An attribute with one value.
value	Similar to cells on a spreadsheet, values represent instances of attributes for an item. An attribute might have multiple values.

Document Conventions

This section lists the common typographical and symbol use conventions for AWS technical publications.

Typographical Conventions

This section describes common typographical use conventions.

Convention	Description/Example
Call-outs	A call-out is a number in the body text to give you a visual reference. The reference point is for further discussion elsewhere. You can use this resource regularly. 1
Code in text	Inline code samples (including XML) and commands are identified with a special font. You can use the command <code>java -version</code> .
Code blocks	Blocks of sample code are set apart from the body and marked accordingly. <pre># ls -l /var/www/html/index.html -rw-rw-r-- 1 root root 1872 Jun 21 09:33 /var/www/html/ index.html # date Wed Jun 21 09:33:42 EDT 2006</pre>
Emphasis	Unusual or important words and phrases are marked with a special font. You <i>must</i> sign up for an account before you can use the service.
Internal cross references	References to a section in the same document are marked. See Document Conventions (p. 61) .
Logical values, constants, and regular expressions, abstracta	A special font is used for expressions that are important to identify, but are not code. If the value is <code>null</code> , the returned response will be <code>false</code> .

Amazon SimpleDB Getting Started Guide Typographical Conventions

Convention	Description/Example
Product and feature names	Named AWS products and features are identified on first use. Create an <i>Amazon Machine Image</i> (AMI).
Operations	In-text references to operations. Use the <code>GetHITResponse</code> operation.
Parameters	In-text references to parameters. The operation accepts the parameter <i>AccountID</i> .
Response elements	In-text references to responses. A container for one <code>CollectionParent</code> and one or more <code>CollectionItems</code> .
Technical publication references	References to other AWS publications. If the reference is hyperlinked, it is also underscored. For detailed conceptual information, see the <i>Amazon Mechanical Turk Developer Guide</i> .
User entered values	A special font marks text that the user types. At the password prompt, type MyPassword .
User interface controls and labels	Denotes named items on the UI for easy identification. On the File menu, click Properties .
Variables	When you see this style, you must change the value of the content when you copy the text of a sample to a command line. <code>% ec2-register <your-s3-bucket>/image.manifest</code> See Symbol Conventions (p. 63) .

Symbol Conventions

This section describes the common use of symbols.

Convention	Symbol	Description/Example
Mutually exclusive parameters	(Parentheses and vertical bars)	Within a code description, bar separators denote options from which one must be chosen. <code>% data = hdfread (start stride edge)</code>
Optional parameters XML variable text	[square brackets]	Within a code description, square brackets denote completely optional commands or parameters. <code>% sed [-n, -quiet]</code> Use square brackets in XML examples to differentiate them from tags. <code><CustomerId>[ID]</CustomerId></code>
Variables	<arrow brackets>	Within a code sample, arrow brackets denote a variable that must be replaced with a valid value. <code>% ec2-register <your-s3-bucket>/image.manifest</code>