Introduction to

Appercept AWS SDK for Delphi

Presented by Richard Hatherall

Appercept AWS SDK for Delphi

General Features

- AWS HTTP client layer built on `System.Net.HttpClient`
- Compatible with all Delphi-supported target platforms
- Direct mapping of all features for implemented services
- Honours AWS configuration:
  - Shared files
  - Environment variables
  - Platform services
- Automatic credential resolution
- Automatic credential refreshing
- Scoped exceptions
- Automatic retries (based on policy)
- Convenient utility classes help you to:
  - perform complex or asynchronous actions
  - build complex data structures
- IDE-integrated AWS configuration manager
- Components for FMX and VCL

Appercept AWS SDK for Delphi

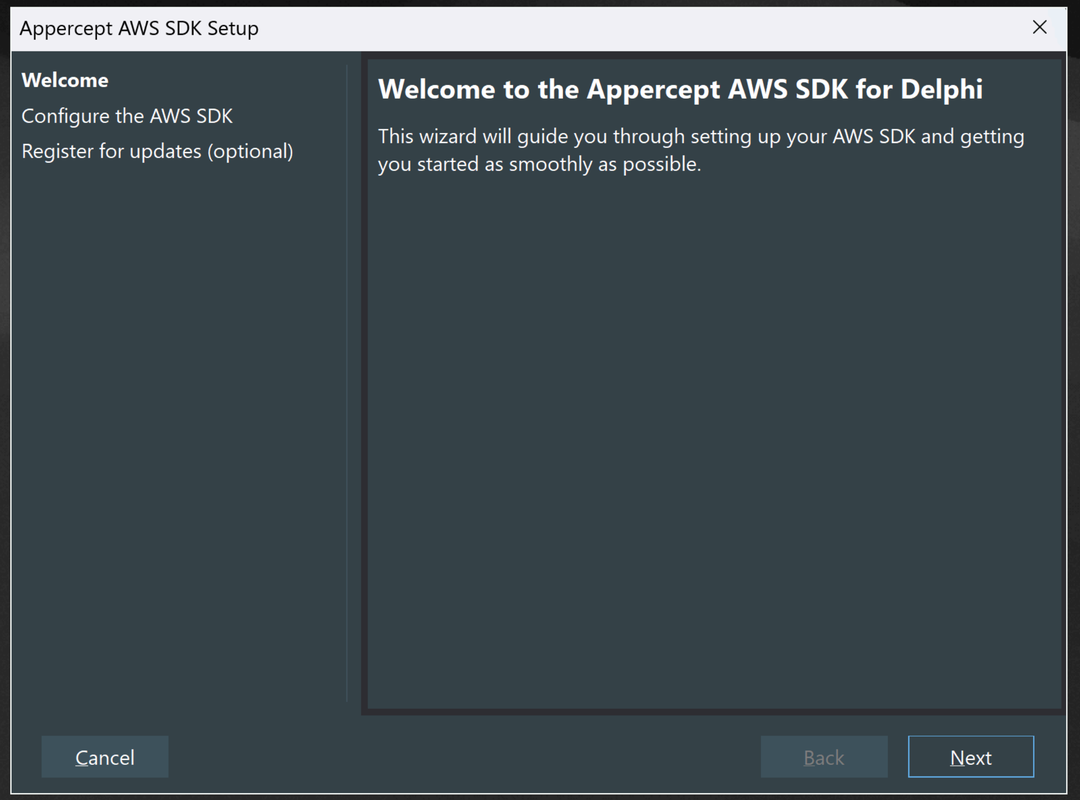
Supported Services

- Application Integration
  - Amazon Simple Notification Service (SNS)
  - Amazon Simple Queue Service (SQS)
- Business Applications
  - Amazon Simple Email Service (SES)
- Cryptography & PKI
  - AWS Key Management Service (KMS)
- Machine Learning
  - Amazon Polly
  - Amazon Textract
  - Amazon Transcribe
  - Amazon Translate
- Security, Identity & Compliance
  - Amazon Cognito
  - AWS Secrets Manager
  - AWS Security Token Service (STS)
- Storage
  - Amazon Simple Storage Service (S3)

AWS SDK Configuration

First Launch

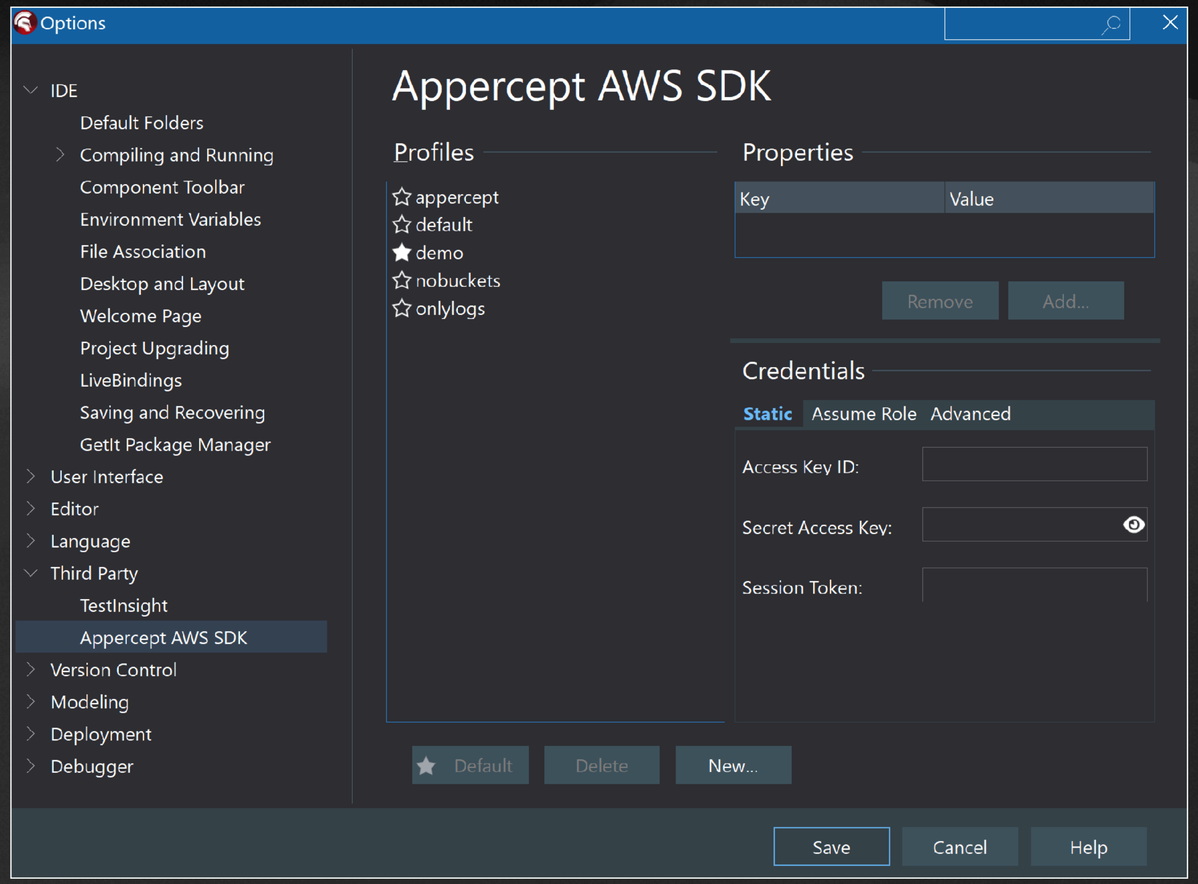
- Configuration wizard on first launch
- Configure the essentials to work with AWS



AWS SDK Configuration

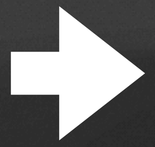
IDE Integrated Configuration Manager

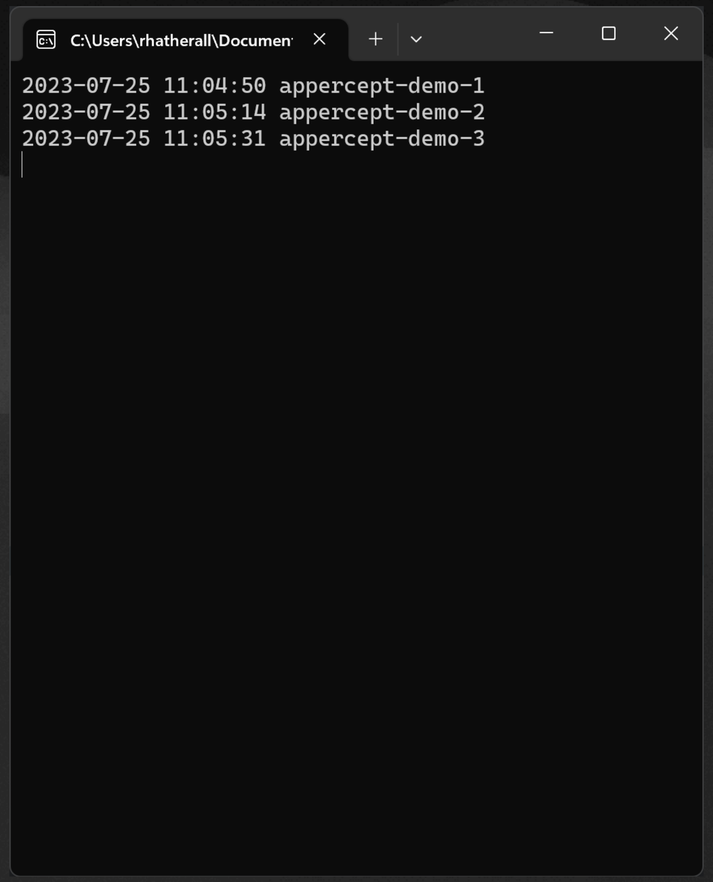
- Edit your AWS configuration profiles
- Choose an “IDE profile” to be used for debugging
- Stores configuration in your user profile `.aws` folder – where the official AWS SDKs load their configuration



Basic Service Usage Example

```
1  program ListS3Buckets;
2
3  {$APPTYPE CONSOLE}
4
5  {$R *.res}
6
7  uses
8    // 1. Add service unit to "uses" clause.
9    AWS.S3,
10   System.SysUtils;
11
12 var
13   // 2. Define a reference to the service client interface.
14   S3: IS3Client;
15
16 begin
17   try
18     // 3. Create a service client.
19     S3 := TS3Client.Create;
20
21     // 4. Make a request and store the response.
22     var Response := S3.ListBuckets;
23
24     // 5. Inspect the response.
25     for var BucketName in Response.Buckets do
26       Writeln(Format('%s %s', [
27         FormatDateTime('yyyy-mm-dd hh:nn:ss', BucketName.CreationDate),
28         BucketName.Name
29       ]));
30
31     Readln;
32   except
33
34     // 6. Don't forget to handle service exceptions.
35     on E: ES3Exception do
36       Writeln(E.ClassName, ': ', E.Message);
37
38     on E: Exception do
39       Writeln(E.ClassName, ': ', E.Message);
40   end;
41 end.
```





Requests

Each action accepts arguments via a request object.

```
program Requests;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  AWS.S3,
  System.SysUtils;

var
  S3Client: IS3Client;

begin
  try
    // Create an S3 client.
    S3Client := TS3Client.Create;

    // Create a request object.
    var Request: IS3GetObjectRequest := TS3GetObjectRequest.Create(
     'a-bucket',
     'my-object'
    );
    // Call action with request object.
    S3Client.GetObject(Request);

    // Or use a convenient overload.
    S3Client.GetObject('a-bucket', 'my-object');

  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```

Responses

Each action returns a response.

```
program Responses;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  AWS.S3,
  System.SysUtils;

var
  S3Client: IS3Client;

begin
  try
    // Create an S3 client.
    S3Client := TS3Client.Create;

    // Store the response.
    var Response := S3Client.GetObject('a-bucket', 'my-object');

    // Check if the response is successful.
    if Response.IsSuccessful then
    begin
      // Do something with the response.
    end;

  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```

Exceptions

On errors, actions raise exceptions scoped to the service client.

```
program Exceptions;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  AWS.S3,
  System.SysUtils;

var
  S3Client: IS3Client;

begin
  try
    // Create an S3 client.
    S3Client := TS3Client.Create;

    try
      var Response := S3Client.GetObject('a-bucket', 'my-object');
    except
      // A specific S3 exception.
      on E: ES3NoSuchKey do
        Writeln('Object not found!');

      // Any S3 excepion.
      on E: ES3Exception do
        Writeln(E.ClassName, ': ', E.Message);
    end;

  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```

Retries

Retries are automatic for “retryable” errors.

The behaviour can be altered through options.

```
program Retries;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  AWS.S3,
  System.SysUtils;

var
  Options: IS3Options;
  Client: IS3Client;

begin
  try
    Options := TS3Options.Create;

    // Specify a retry limit.
    // Defaults to 3.
    Options.RetryLimit := 10;

    // Specify a duration in milliseconds as the base delay.
    // Defaults to 300.
    Options.RetryBaseDelay := 500; // Half a second.

    // Enable "jitter" with either "equal" or "full".
    /// Defaults to "none".
    Options.RetryJitter := 'full';

    Client := TS3Client.Create(Options);
    try
      var Response := Client.ListBuckets;
    except
      on E: ES3SlowDown do
        Writeln('Too many slow down responses!');
    end;

  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```

Utilities

Many utilities to assist you…

```
program BuildIAMPolicy;

{$APPTYPE CONSOLE}

{$R *.res}

uses
  AWS.IAM.PolicyDocument,
  System.SysUtils;

begin
  try
    Writeln(
      TIAMPolicyDocument.Build
        .BeginStatement
          .Sid('AllowListingContents')
          .Action([
            's3:ListBucketMultipartUploads',
            's3:ListBucketVersions',
            's3:ListBucket',
            's3:ListMultipartUploadParts'
          ])
          .Resource('arn:aws:s3:::*')
        .EndStatement
        .BeginStatement
          .Sid('AllowListingBuckets')
          .Action('s3:ListAllMyBucket')
          .Resource('*')
        .EndStatement
        .Document
        .ToString
    );

    Readln;
  except
    on E: Exception do
      Writeln(E.ClassName, ': ', E.Message);
  end;
end.
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListingContents",
      "Action": [
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:ListMultipartUploadParts"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    },
    {
      "Sid": "AllowListingBuckets",
      "Action": [
        "s3:ListAllMyBucket"
      ],
      "Resource": "*"
    }
  ]
}
```

Demos

appercept

Introduction to

Appercept AWS SDK for Delphi

https://github.com/appercept/aws-sdk-delphi-samples
https://www.appercept.com
support@appercept.com
@ApperceptHQ

Presented by Richard Hatherall

embarcadero®
TECHNOLOGY PARTNER