namespace DeveloperApiCSharpSample

{

 using System;

 using System.Globalization;

 using System.Net.Http;

 using System.Threading.Tasks;

 /// <summary>

 /// Sample code for how to update add-on submissions

 /// </summary>

 public class InAppProductSubmissionUpdateSample

 {

 private ClientConfiguration ClientConfig;

 /// <summary>

 /// Constructor

 /// </summary>

 /// <param name="c">An instance of ClientConfiguration that contains all parameters populated</param>

 public InAppProductSubmissionUpdateSample(ClientConfiguration c)

 {

 this.ClientConfig = c;

 }

 public void RunInAppProductSubmissionUpdateSample()

 {

 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 // SETTINGS

 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 var iapId = this.ClientConfig.InAppProductId;

 var clientId = this.ClientConfig.ClientId;

 var clientSecret = this.ClientConfig.ClientSecret;

 var serviceEndpoint = this.ClientConfig.ServiceUrl;

 var tokenEndpoint = this.ClientConfig.TokenEndpoint;

 // Get authorization token

 Console.WriteLine("Getting authorization token ");

 var accessToken = IngestionClient.GetClientCredentialAccessToken(

 tokenEndpoint,

 clientId,

 clientSecret).Result;

 Console.WriteLine("Getting the add-on");

 var client = new IngestionClient(accessToken, serviceEndpoint);

 dynamic iap = client.Invoke<dynamic>(

 HttpMethod.Get,

 relativeUrl: string.Format(

 CultureInfo.InvariantCulture,

 IngestionClient.GetInAppUrlTemplate,

 IngestionClient.Version,

 IngestionClient.Tenant,

 iapId),

 requestContent: null).Result;

 Console.WriteLine(iap.ToString());

 // Let's see if there is a pending submission. Warning! If it was created through the API,

 // it will be deleted so that we could create a new one in its stead.

 if (iap.pendingInAppProductSubmission != null)

 {

 var submissionId = iap.pendingInAppProductSubmission.id.Value as string;

 // Let's try deleting it. If it was NOT created via the API, then you need to manually

 // delete it from the dashboard. This is a safety measure to make sure that a human user and

 // an automated system don't make conflicting edits.

 Console.WriteLine("Deleting the pending submission");

 client.Invoke<dynamic>(

 HttpMethod.Delete,

 relativeUrl: string.Format(

 CultureInfo.InvariantCulture,

 IngestionClient.InAppSubmissionUrlTemplate,

 IngestionClient.Version,

 IngestionClient.Tenant,

 iapId,

 submissionId),

 requestContent: null).Wait();

 }

 // Create a new submission, which will be an exact copy of the last published submission.

 Console.WriteLine("Creating a new submission");

 dynamic clonedSubmission = client.Invoke<dynamic>(

 HttpMethod.Post,

 relativeUrl: string.Format(

 CultureInfo.InvariantCulture,

 IngestionClient.InAppSubmissionUrl,

 IngestionClient.Version,

 IngestionClient.Tenant,

 iapId),

 requestContent: null).Result;

 var clonedSubmissionId = clonedSubmission.id.Value as string;

 Console.WriteLine(clonedSubmission.ToString());

 // Update the add-on price and keep the rest unchanged.

 clonedSubmission.pricing.priceId = "Tier2"; // $0.99

 // Because we are not uploading any new images, we don't need to upload the zip file.

 // Update the submission.

 Console.WriteLine("Updating the submission");

 client.Invoke<dynamic>(

 HttpMethod.Put,

 relativeUrl: string.Format(

 CultureInfo.InvariantCulture,

 IngestionClient.InAppSubmissionUrlTemplate,

 IngestionClient.Version,

 IngestionClient.Tenant,

 iapId,

 clonedSubmissionId),

 requestContent: clonedSubmission).Wait();

 // Tell the system that we are done updating the submission.

 Console.WriteLine("Committing the submission");

 client.Invoke<dynamic>(

 HttpMethod.Post,

 relativeUrl: string.Format(

 CultureInfo.InvariantCulture,

 IngestionClient.InAppProductCommitSubmissionUrlTemplate,

 IngestionClient.Version,

 IngestionClient.Tenant,

 iapId,

 clonedSubmissionId),

 requestContent: null).Wait();

 // Periodically check the status until it changes from "CommitsStarted" to either

 // successful status or a failure.

 Console.WriteLine("Waiting for the submission commit processing to complete. This may take a couple of minutes.");

 string submissionStatus = null;

 do

 {

 Task.Delay(TimeSpan.FromSeconds(5)).Wait();

 dynamic statusResource = client.Invoke<dynamic>(

 HttpMethod.Get,

 relativeUrl: string.Format(

 CultureInfo.InvariantCulture,

 IngestionClient.InAppSubmissionStatusUrlTemplate,

 IngestionClient.Version,

 IngestionClient.Tenant,

 iapId,

 clonedSubmissionId),

 requestContent: null).Result;

 submissionStatus = statusResource.status.Value as string;

 Console.WriteLine("Current status: " + submissionStatus);

 }

 while ("CommitStarted".Equals(submissionStatus));

 if ("CommitFailed".Equals(submissionStatus))

 {

 Console.WriteLine("Submission has failed. Please check the Errors collection of the submissionResource response.");

 return;

 }

 else

 {

 Console.WriteLine("Submission commit success! Here is the new price:");

 dynamic sub = client.Invoke<dynamic>(

 HttpMethod.Get,

 relativeUrl: string.Format(

 CultureInfo.InvariantCulture,

 IngestionClient.InAppSubmissionUrlTemplate,

 IngestionClient.Version,

 IngestionClient.Tenant,

 iapId,

 clonedSubmissionId),

 requestContent: null).Result;

 Console.WriteLine(sub.pricing.priceId.Value as string);

 }

 }