Manual técnico del servicio Web API de transmisión automática de AutoControles

Confederación Hidrográfica del Júcar, O.A. – Comisaría de Aguas

Ver. 1.0 febrero 2023

Contenido

1 Introducción 2
2 Objeto del dominio 2
3 Descripción de los servicios 3
3.1 Sección autenticación 3
3.1.1 DameToken 3
3.2 Sección Autocontroles 4
3.2.1 AutoControl 4
4 Generación automática de clientes 6
4.1 Microsoft Visual Studio 6
4.2 Swagger codegen 6
5 Ejemplos de código consumiendo los servicios 7

1 Introducción

El presente documento es una guía para la utilización remota de los servicios web diseñados para la carga automática de autocontroles en continuo de vertidos de aguas residuales

Estos servicios deberán ser invocados por aplicaciones informáticas desarrolladas para tal fin y con el objetivo de facilitar esta labor se han elaborados siguiendo los estándares de la especificación OpenAPI.

Swagger es un framework para documentar APIs Rest y el principal beneficio para un tercero que quiera consumir estas APIs es la posibilidad de automatizar la generación de los clientes que va a hacer uso de las mismas. Todo ello a través de la descripción de dichas APIs mediante ficheros JSON.

Swagger dio lugar a la especificación OpenAPI para describir APIs REST de manera uniforme e independiente del lenguaje de programación que se quiera utilizar para su consumo.

A continuación, se describirán las funcionalidades y características de esta WebAPI, así como unas directrices para su utilización desde aplicaciones que deseen hacer uso de los servicios descritos.

2 Objeto del dominio

La Confederación Hidrográfica del Júcar tiene atribuidas por el art. 24 del RD Legislativo 1/2001, de 20 de julio, por el que se aprueba el texto refundido de la ley de aguas, entre otras, las siguientes funciones:

a) El otorgamiento de autorizaciones y concesiones referentes al dominio público hidráulico.

b) La inspección y vigilancia del cumplimiento de las condiciones de concesiones y autorizaciones relativas al dominio público hidráulico.

Enmarcado en el PERTE de digitalización del ciclo del agua, resulta imprescindible un cambio de paradigma en el control de la calidad y cantidad del agua vertida con una transmisión en continuo.

A los efectos de este manual, se definen los siguientes conceptos:

Expediente: Conjunto ordenado de documentos y actuaciones que sirven de antecedente y fundamento a la resolución administrativa de autorización de vertido a Dominio Público Hidráulico (DPH). Se identifica con una *referencia*. (YYYYC-VS-nnnnn o YYYYC-VI-nnnnn).

Identificador: Es un código o contraseña alfanumérica asignada a cada expediente (ejemplo: 16B2F59D-5FD0-4A3D-8378-F149097514F6)

Punto de Vertido: Lugar geográfico donde se vierten las aguas residuales depuradas a DPH (ejemplo: PVn → n).

Punto de Control: Emplazamiento de fácil acceso y localización, donde se realiza el control efectivo de las características cuantitativas y cualitativas de las aguas depuradas antes de ser vertidas a DPH (ejemplo: PCn → n).

Parámetro y PARACOD: Es el código por el que se representa cada contaminante analizado.:

- El parámetro 'Caudal instantáneo' se corresponderá con el PARACOD → 'CAUDAL'
- El parámetro 'Conductividad a 20ºC instantáneo' se corresponderá con el PARACOD → 'COND 20º'
- El parámetro 'Turbidez instantáneo' se corresponderá con el PARACOD → 'TRURBISITU'
- El parámetro 'Oxígeno disuelto instantáneo' se corresponderá con el PARACOD → 'O2 DTO'
- El parámetro 'ph instantáneo' se corresponderá con el PARACOD → 'PH'

Toda esta información se le proporcionará al titular del expediente de autorización de vertido en el apartado 6º (Medición en continuo) de la Resolución.

3 Descripción de los servicios

Todos los servicios que define la API se encuentran accesibles a través de la siguiente ruta:

https://app.chj.es/AutoControlWebApi

Para realizar pruebas puede utilizar el servicio disponible en:

https://app.chj.es/AutoControlWebApiTest

A continuación, se van a enumerar describiendo los parámetros de entrada, valores de salida y resultados devueltos por el servidor.

3.1 Sección autenticación

3.1.1 DameToken

Previo a la utilización de cualquiera del resto de servicios es necesario una autenticación. Desde el Área de Calidad del agua se les habrá previamente suministrado un identificador para cada uno de los expedientes de los que sea titular y mediante la referencia del expediente y el identificador suministrado se realiza la autenticación.

El método DameToken de la API proporciona un token JWT (JSON Web Token) que, una vez obtenido con una autenticación básica, será requerido para completar el resto de operaciones definidas en la WebAPI.

Puede consultar la documentación JWT en JSON Web Tokens - jwt.io

Ruta relativa: /api/Autenticacion/dametoken

Tipo de llamada: GET

Parámetros:

- Referencia (string): Referencia de un expediente de concesión de aguas
- Identificador (string): cadena previamente proporcionada de un modo privado y seguro al tenedor de derechos del expediente de referencia

Ejemplo de llamada a DameToken:

```
https://app.chj.es/api/Autenticacion/dametoken?Referencia=2007VI9911&Identificador=91879878-d01e-4994-a31c-487675abaccc
```

Resultados desde el servidor:

- Respuesta 200 (SUCCESS):
- Significa que toda ha ido correctamente y que en el cuerpo de la respuesta está en formato JSON el JWT. A continuación, se muestra un ejemplo de respuesta:


```
{
"jwt":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiI4OGY3ZDAyMy0zNGIxLTQ4MjUtYmYwOS01YTg3NGI4
ZmYyNzAiLCJyZWYiOiIyMDEzQ0MwMDMxIiwiaWQiOiIxYWNhMWNlMy1lYTM1LTQxYjItYTM0MC01NzA5
Yjk3NzA1YzkiLCJleHAiOjE2MjYzMzUxMzgsImlzcyI6Imh0dHBzOi8vd3d3LmNoai5lcyIsImF1ZCI6Imh0dHBzOi8v
d3d3LmNoai5lcyJ9.Bm3RFWIVxMHRhR6o_8JSwvQX1l446Rsc0zDCJLvKTrA"
}
```


- Respuesta 404 (NOT FOUND): Significa que no se ha podido encontrar una relación entre la referencia de expediente y el identificador enviados como parámetros. A continuación se muestra un ejemplo de respuesta:


```
{
  "type": "https://AutoControlWebApi/probs/RefIdentNotFound",
  "title": "Referencia o Identificador no válido",
  "status": 404,
  "detail": "No se ha especificado una Referencia o Identificador válido",
  "instance":  "/api/Autenticacion/dametoken?Referencia=XXXXXXXXXX&Identificador=XXXXXXXX-XXXX-
XXXX-XXXX-XXXXXXXXXXXX"
}
```


3.2 Sección Autocontroles

3.2.1 AutoControl

Este método de la API permite enviar la información de un autocontrol mediante una estructura JSON en el cuerpo de la petición.

Ruta relativa: /api/AutoControl/autocontrol

Tipo de llamada: POST

Cuerpo de la llamada: Encapsulado mediante JSON en el “body” del mensaje que responde a la siguiente estructura


```
{
  "numPuntoVertido": 0,
  "numPuntoControl": 0,
  "listControles": [
    {
      "paracod": "string",
      "valor": 0
    }
  ]
}
```

Ejemplo de envío:

```
{
  "numPuntoVertido": 1,
  "numPuntoControl": 1,
  "listControles": [
    {
      "paracod": "TRURBISITU",
      "valor": 12
    },
    {
      "paracod": "PH",
      "valor": 7.7
    }
  ]
}
```

- **numPuntoVertido**: Identificador numérico del punto de vertido. Si el punto de vertido proporcionado es PV1 se enviará el valor numérico 1 (Proporcionado por CHJ)
- **numPuntoControl**: Identificador numérico del punto de control. Si el punto de control proporcionado es PC1 se enviará el valor numérico 1 (Proporcionado por CHJ)
- **listControles:** Lista de los distintos parámetros tomados, cada uno con los siguientes datos:
  - **paracod**: Código del parámetro (Proporcionado por CHJ de los posibles valores) Ejemplos: PH, THURBISITU, … y estos deberán venir en las unidades indicadas en la autorización.
  - **valor**: Valor numérico del resultado de la determinación (No se admitirán valores negativos)

Resultados desde el servidor:

- Respuesta 200 (SUCCESS): Significa que toda ha ido correctamente y que se ha incorporado el autocontrol
- Respuesta 400 (BAD_REQUEST): Significa que se ha producido un error no controlado en la petición y que no se ha incorporado el autocontrol
- Respuesta 401 (UNAUTHORIZED): Significa que se ha producido un error en la autenticación mediante un JWT incorrecto
- Respuesta 403 (FORBIDDEN): Significa que alguno de los parámetros enviados del autocontrol es incorrecto
- Respuesta 404 (NOT_FOUND): Significa que los valores proporcionado producen problemas para su inserción: Referencia no válida para el JWT, parámetro PARACOD no existe, etc.

El cuerpo de la respuesta siempre, en caso de problemas, tendrá formato JSON con la estructura siguiente:

```
{
  "type":                                                    string,
  "title":                                                   string,
  "status":                                                  integer($int32),
  "detail":                                                  string,
  "instance":                                                string
}
```

Ejemplo:

```
{
  "type":                                    "https://AutoControlWebApi/probs/RefIdentNotFound",
  "title":        "El      PARACOD      XXXXX      no      existe      o      no      es      correcto",
  "status":                                                  404,
  "detail":                                                  "",
  "instance":                                                "/api/AutoControl/autocontrol"
}
```

4 Generación automática de clientes

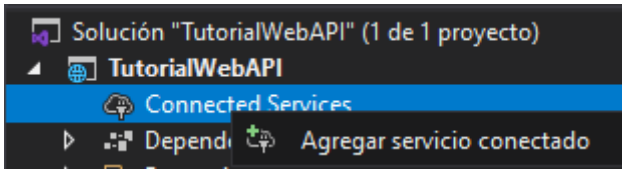
El estándar OpenAPI facilita la creación de clientes mediante un fichero JSON que define la funcionalidad de los servicios. Este fichero JSON puede ser descargado desde la siguiente URL

https://app.chj.es/AutoControlWebApi/swagger/v1/swagger.json

4.1 Microsoft Visual Studio

Se va a describir el proceso de generación de este cliente para una solución web de MS Visual Studio 2022. El procedimiento deber ser similar para otras plataformas o lenguajes de programación.

Lo primero es agregar un servicio conectado desde el explorador de soluciones de VS



Una vez seleccionada la opción de agregar servicio, procederemos a seleccionar como referencia de servicios OpenAPI el fichero swagger.json descargado desde la URL y Visual Studio generará los fuentes necesarios del cliente que consumirá los servicios.

Concretamente, se habrá generado en la siguiente ruta del proyecto:

obj\swaggerClient.cs

Todas las clases, atributos y métodos presentes en el cliente estarán accesibles para la aplicación.

4.2 Swagger codegen

Como ya se ha mencionado en la sección anterior, existe la posibilidad de generar automáticamente clientes para gran cantidad de lenguajes de programación. Para ello han desarrollado la herramienta java Swagger codegen que permitirá generar el código fuente de los clientes en el lenguaje que se elija.

A continuación, se describe un ejemplo de generación de cliente con esta herramienta:

Se requiere descargar la aplicación java empaquetada jar desde la URL

https://mvnrepository.com/artifact/io.swagger.codegen.v3/swagger-codegen-cli/3.0.27

Se puede crear un fichero config.json de configuración definiendo los espacios de nombre de la aplicación que va a utilizar el cliente.

```
{
  "modelPackage": "WAPIAutoControl",
  "apiPackage": "WAPIAutoControl"
}
```

Finalmente ejecutando el comando

```
java -jar .\swagger-codegen-cli.jar generate -i .\swagger.json -l csharp -o c# -c .\config.json
```

En la carpeta c# (previamente creada) se generará una solución que dará lugar a una dll IO.Swagger.dll con todas sus dependencias tras la compilación.

El parámetro -l indica el lenguaje para el que se quiere crear el cliente, en este caso c#.

5 Ejemplos de código consumiendo los servicios

Consumir los servicios una vez generada la clase cliente es simple. La única complejidad reside en proporcionar el token JWT a las llamadas a la API.

El asistente de Visual Studio crea las clases necesarias para invocar los servicios web definidos en el archivo swagger.json, pero no tiene en cuenta que el servicio está securizado mediante JWT.

Por tanto, tendremos que añadir el código necesario para poder pasar el token JWT, para ello crearemos una clase parcial de la misma clase generada por el asistente de VS.

En esta clase parcial crearemos un método con la firma

```
PrepareRequest(System.Net.Http.HttpClient client,System.Net.Http.HttpRequestMessage request, string url)
```

Este método lo llama el cliente de swagger cada vez que tiene que hacer una llamada al web api. En el método lo que haremos es añadir el Header “Authorization” al request con el contenido del token anteponiéndole el temino “Bearer “, (importante el espacio en blanco entre Bearer y el token)

El contenido de la partial class podría ser el siguiente:

```
  public partial class swaggerClient
   {
     public ResultadoDameToken token { get; set; }

     partial           void           PrepareRequest(System.Net.Http.HttpClient           client,
System.Net.Http.HttpRequestMessage request, string url)
     {
       if (token != null && client.DefaultRequestHeaders.Authorization == null)
         {
           client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token.Jwt);
         }
     }
```

```
    }
```

Y este podría ser un ejemplo de cómo subir unos valores desde una aplicación

```
ResultadoDameToken Token = null;
swaggerClient cliente;

try
{
  var http = new System.Net.Http.HttpClient();
  cliente = new swaggerClient(http);

  cliente.BaseUrl = "https://app.chj.es/AutoControlWebApi";
  Token = await cliente.DametokenAsync("XXXXXXXXXX", "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX");
  cliente.token = Token;

  DatosAutocontrol objDatos = new DatosAutocontrol();
  objDatos.NumPuntoControl = 1;
  objDatos.NumPuntoVertido = 1;
  objDatos.ListControles = new List<Controles>();
     objDatos.ListControles.Add(new Controles() { Paracod = "TRURBISITU", Valor = 12 });
     objDatos.ListControles.Add(new Controles() { Paracod = "PH", Valor = 7.7 });

     await cliente.AutocontrolAsync(objDatos);
}
catch (ApiException<ProblemDetails> api)
{
     Console.WriteLine(api.Result.Title + "\n" + api.Result.Detail);
}
catch (Exception ex)
{
     Console.WriteLine(ex.Message);
}
Console.WriteLine("Success!");
```

Otro ejemplo de código podría ser a través de un cliente http:

```
async void bPrueba_Click(object sender, EventArgs e)
{
    string BaseWebApi = " https://app.chj.es/AutoControlWebApi/api/";
    HttpClient http = new HttpClient();

    // ejemplo
    string referencia = " XXXXXXXXXX";
    string identificador = " XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX ";

    HttpResponseMessage                    response                    =                    await
http.GetAsync($"{BaseWebApi}Autenticacion/dametoken?Referencia={referencia}&Identificador={ident
ificador}");
    string jsonString = await response.Content.ReadAsStringAsync();
    if (response.IsSuccessStatusCode == false)
    {
        //Error
        return;
    }

    ResultadoDameToken? token = JsonConvert.DeserializeObject<ResultadoDameToken>(jsonString);


    http.DefaultRequestHeaders.Add("Authorization", "Bearer " + token?.Jwt);


    DatosAutocontrol objDatos = new DatosAutocontrol();
    objDatos.NumPuntoControl = 1;
    objDatos.NumPuntoVertido = 1;
    objDatos.ListControles.Add(new Controles(){ PARACOD = " TRURBISITU", Valor = 12 });
    objDatos.ListControles.Add(new Controles() { PARACOD = "PH", Valor = 7.7 });

    string json = JsonConvert.SerializeObject(datosLectura);
```

```
    HttpResponseMessage resultado = await http.PostAsync($"{BaseWebApi}AutoControl/autocontrol",
new StringContent(json, Encoding.UTF8, "application/json"));
    if (resultado.IsSuccessStatusCode)
    {
        // ok
    }
    else
    {
                //error
    }
}
```

Y declarando la clase:

```
public class ResultadoDameToken
        {
            [Newtonsoft.Json.JsonProperty("jwt",  Required  =  Newtonsoft.Json.Required.Default,
NullValueHandling = Newtonsoft.Json.NullValueHandling.Ignore)]
            public string Jwt { get; set; }
        }
```