



Manual técnico del servicio Web API de transmisión automática de AutoControles

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

Ver. 1.0 octubre 2023



Contenido

Introducción	2
Objeto del dominio	3
Descripción de los servicios.....	4
Sección autenticación	4
DameToken	4
Sección Autocontroles	5
AutoControl.....	5
Sección consulta de autocontroles	7
Consulta.....	7
Generación automática de clientes	9
Microsoft Visual Studio	9
Swagger codegen	9
Ejemplos de código consumiendo los servicios	11



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 las depuradoras de la Epsar de la Confederación Hidrográfica del Júcar.

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.



Objeto del dominio

La Confederación hidrográfica del Júcar tienen entre sus funciones la tramitación y el seguimiento de los expedientes administrativos sobre los usos de agua.

Cada expediente administrativo está relacionado con un aprovechamiento con sus correspondientes titulares y usos de agua correspondiente.

Al objeto de la automatización del envío de los autocontroles nos interesa definir los conceptos de expediente e identificador.

Expediente: Conjunto de actuaciones encaminadas a la resolución de autorización de vertido, directo o indirecto de aguas residuales susceptibles de contaminar las aguas continentales o cualquier otro elemento del Dominio Público Hidráulico. Este documento vendrá identificado mediante una *referencia*. (ejemplos: 2022C-VS-00113 o 1999C-VS-00125).

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

Punto de Vertido: Lugar del DPH donde se realiza el vertido de las aguas residuales depuradas, procedentes de la estación de depuración (ejemplo: PV1 → 1).

Punto de Control: Instalación o lugar de fácil acceso y localización, donde se realiza el seguimiento y muestreo de las aguas depuradas antes de ser vertidas a DPH (ejemplo: PC1 → 1).

Parámetro (PARACOD): Es el código por el que se representa cada resultado analítico. Por ejemplo:

- El parámetro 'Caudal instantáneo' se corresponderá con el PARACOD → 'CAUDAL'
- El parámetro 'Conductividad a 20°C instantáneo' se corresponderá con el PARACOD → 'COND 20°'
- 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 propia resolución.



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.

Sección autenticación

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](https://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:



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"  
}
```

Sección consulta de autocontroles

Consulta

Este método de la API permite consultar los controles introducidos mediante una serie de parámetros.

Ruta relativa: /api/Consulta/consulta

Tipo de llamada: GET

Parámetros:

- numPuntoVertido (int): Número del Punto del vertido
- numPuntoControl (int): Número del Punto del control
- fechaControl (datetime): fecha de cuando se introdujeron los controles. Formato de fecha (yyyy-MM-dd)

Ejemplo de llamada a consulta:

```
https://app.chj.es/api/Consulta/consulta?NumPuntoVertido=1&NumPuntoControl=1&FechaControl=2023-02-16
```

Resultados desde el servidor:

- Respuesta 400 (BAD_REQUEST):
Significa que se ha producido un error no controlado en la petición
- Respuesta 401 (UNAUTHORIZED):
Significa que se ha producido un error en la autenticación mediante un JWT incorrecto

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 error:

```
{  
  "type": "https://AutoControlWebApi/probs/RefIdentNotFound",  
  "title": "Error en la consulta de datos de AutoControl",  
  "status": 404,  
  "detail": "",  
  "instance": "/api/Consulta/consulta"  
}
```

Formato respuesta correcta

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

Ejemplo de respuesta correcta con datos:

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

Ejemplo de respuesta correcta sin resultados:

```
{  
  "listControles": []  
}
```



Generación automática de clientes

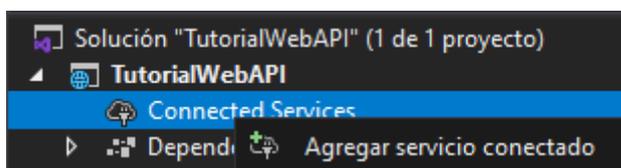
El estándar OpenAPI facilita la creación de clientes mediante un fichero JSON que define la funcionalidad de los servicios. Este fichero JSON puede ser descargado desde la siguiente URL

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

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.

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#.



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 término "Bearer ", (importante el espacio en blanco entre Bearer y el token)

El contenido de la parcial 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 = " XXXXXXXXXXX";
    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; }
}
```



Ejemplo de código para consultar los controles introducidos a través de un cliente http:

```
async void bPruebaConsulta_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);

    int numPuntoVertido = 1;
    int numPuntoControl = 1;
    DateTime fechaControl = new DateTime(2023, 2, 16);

    HttpResponseMessage responseConsulta = await
http.GetAsync($"{BaseWebApi}Consulta/consulta?NumPuntoVertido={numPuntoVertido}&NumPuntoControl
={numPuntoControl}&FechaControl={fechaControl.ToString("yyyy-MM-dd")}");

    jsonString = await responseConsulta.Content.ReadAsStringAsync();
    if (responseConsulta.IsSuccessStatusCode == false)
    {
        //Error
        return;
    }
}
```

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; }
}
```