MINISTERIO
PARA LA TRANSICIÓN ECOLÓGICA
Y EL RETO DEMOGRÁFICO

CONFEDERACIÓN
HIDROGRÁFICA
DEL JÚCAR, O. A.

Manual técnico del servicio Web API de transmisión automática de lecturas de contadores

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

Ver. 1.3 Enero 2024

Contenido

Versiones del documento

VERSION	FECHA	AUTOR	APROBADO POR	CAMBIOS
1.1	Noviembre 2022	Emilio Luis Sanchis Pastor	Emilio Luis Sanchis Pastor	Versión inicial
1.2	Diciembre 2023	Pablo Pozo Ruiz	Emilio Luis Sanchis Pastor	• Se añade sección ComprobarLectura, • Se elimina sección CambioContador
1.3	Enero 2024	Pablo Pozo Ruiz	Emilio Luis Sanchis Pastor	• Se añade método Test

Introducción

El presente documento es una guía para la utilización remota de los servicios web diseñados para la carga automática de lecturas de contadores de consumo de agua relativos a concesiones administrativas otorgadas por 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 de las lecturas de los contadores nos interesa definir los conceptos de expediente, captación, contador e identificador.

Expediente: Es el conjunto de actuaciones para la resolución administrativa de la concesión del uso de agua de un determinado aprovechamiento. En este documento vendrá identificado mediante una *referencia*.

Captación: Es cada uno de los puntos en los que se extrae agua para el uso correspondiente, para un agua subterránea se corresponde con el pozo. En el documento nos referiremos a la captación por el código *idCaptacion.*

Contador: Es el dispositivo de medida del agua extraída de la captación. Las medidas realizadas por el contador son las que se deben transmitir a la CHJ mediante este Web API. El contador viene definido por un código denominado *idContador*. En el caso que hubiera que sustituir el contador por otro por el motivo que sea la CHJ proporcionará otro *idContador* para identificar de manera univoca el contador.

Identificador: Es un código o contraseña asignada a cada referencia de expediente.

Todos estos códigos se le proporcionan al titular del aprovechamiento mediante un oficio.

Descripción de los servicios

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

https://app.chj.es/ContadoresWebApi

Para realizar pruebas puede utilizar el servicio disponible en:

https://app.chj.es/ContadoresWebApiTest

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 dominio público 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://comisaria.chj.es/ContadoresWebApi/api/Autenticacion/dametoken?Referencia=2007IP9911&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.eyJqdGkiOiI4OGY3ZDAyMy0zNGIxLTQ4MjUtYmYwOS01Y
Tg3NGI4ZmYyNzAiLCJyZWYiOiIyMDEzQ0MwMDMxIiwiaWQiOiIxYWNhMWNlMy1lYTM1LTQxYjItYTM0MC01
NzA5Yjk3NzA1YzkiLCJleHAiOjE2MjYzMzUxMzgsImlzcyI6Imh0dHBzOi8vd3d3LmNoai5lcyIsImF1ZCI
6Imh0dHBzOi8vd3d3LmNoai5lcyJ9.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://ContadoresWebApi/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"
}
```

Sección Lecturas

EnviarLectura

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

Asimismo, nos permite enviar la foto correspondiente del contador tomada en el momento de la lectura. La foto no deberá tener un tamaño superior a 2Mb y se deberá enviar codificada como un string en Base64. Los formatos permitidos para la foto son jpg y png.

Ruta relativa: /api/Lecturas/enviarlectura

Tipo de llamada: POST

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

```
{
        idCaptacion     integer($int32)
        idContador      integer($int32)
        fecha           string($date-time)
        valor           number($double)
        tipoIncidencia eIncidenciainteger($int32) Enum:[ 0, 1 ]
        formatoFoto     string nullable: true
        fotoBase64      string nullable: true
}
```

Ejemplo de envío sin foto:

```
{
  "idCaptacion": 6456,
  "idContador": 456,
  "fecha": "2021-07-20T10:33:38.352Z",
  "valor": 567567,
  "tipoIncidencia": 0,
  "formatoFoto": "",
  "fotoBase64": ""
}
```

Ejemplo de envío con foto:

```
{
  "idCaptacion": 6456,
  "idContador": 456,
  "fecha": "2021-07-20T10:33:38.352Z",
  "valor": 567567,
  "tipoIncidencia": 0,
  "formatoFoto": "jpg",
  "fotoBase64":
"iVBORw0KGgoAAAANSUhEUgAAAt4AAAPTCAIAAAAIH5liAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQ
UAAAAJcEhZcwAADsQAAA7EAZUrDhsAAP+lSURBVHhe7J0FmCzHea7nLM7Ozs7OMjMz89ndg3uYmUEsWQaZ4
jiGwA3dJA7ayQ3ZgRsmJw7HceIbsmM7phhlW7JYOsykvW/… (recortado por razones de
legibilidad)"
}
```


- `idCaptacion`: Identificador de la captación correspondiente al contador (Proporcionado por CHJ)
- `idContador`: Identificador del contador (Proporcionado por CHJ)
- `fecha`: Fecha de la lectura del contador
- `valor`: Valor de la lectura
- **`tipoInicidencia`:** Los valores permitidos son: 0 – lectura normal. 1 – Vuelta de contador, o sea, que el contador está funcionando correctamente pero ha llegado a su máximo y ha vuelto a contar empezando por 0
- `formatoFoto`: Formato de la foto. Posibles valores “jpg”, “png”, “”. Se dejará en blanco (vacío) para indicar que no se envía foto de la lectura.
- `fotoBase64`: Foto codificada en Base64 para su envío como string. La foto no puede tener un tamaño superior a 2Mb.

Resultados desde el servidor:

- Respuesta 200 (SUCCESS):
  Significa que toda ha ido correctamente y que se ha incorporado la lectura
- Respuesta 400 (BAD_REQUEST):
  Significa que se ha producido un error no controlado en la petición y que no se ha incorporado la lectura
- 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 de la lecturas es incorrecto o incompatible con lecturas anteriores: volumen inferior al último registrado, fecha anterior a la última registrada, etc.
- Respuesta 404 (NOT_FOUND):
  Significa que los parámetros identificando el contador producen problemas para su inserción: Referencia no válida para el JWT, Identificador de contador inexistente, Contador dado de baja, No hay captación asociada al contador, Contador no relacionado con el expediente de referencia, 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://ContadoresWebApi/probs/RefIdentNotFound",
  "title": "El contador con identificador 0 no existe",
  "status": 404,
  "detail": "",
  "instance": "/api/Lecturas/enviarlectura"
}
```

Sección Comprobar Lecturas

ComprobarLectura

Este método de la API permite la comprobación de la correcta carga de las lecturas en la base de datos.

Ruta relativa: /api/ComprobarLectura/

Tipo de llamada: GET

Parámetros:

- Id: El identificador del contador.
- fechaInicio: La fecha inicial de período a comprobar, en formato dd/MM/yyyy
- fechaFin: La fecha final del período a comprobar, en formato dd/MM/yyyy

Ejemplo de llamada al método:

```
https://comisaria.chj.es/ContadoresWebApi/api/ComprobarLectura?id=881&fechaInicio=17%2F01%2F2011&fechaFin=13%2F03%2F2012
```

Resultados desde el servidor:

- Respuesta 200 (SUCCESS):
  Todo ha ido correctamente en la petición de datos al servidor. Se obtiene una respuesta en formato JSON con los datos solicitados, por ejemplo:

  ·Para el id 881, y el rango de búsqueda entre 10/12/2003 y 09/02/2004 obtenemos

```
{
  "id": 881,
  "fecha": ”10/12/2003” ,
  "formattedValor": “2561114,00”
},
{
  "id": 881,
  "fecha": ”09/01/2004” ,
  "formattedValor": “2625210,00”
},
{
  "id": 881,
  "fecha": ”09/02/2004” ,
  "formattedValor": “2682630,00”
}
```

- Respuesta 400 (BAD REQUEST):
  Esta respuesta se produce cuando no se introducen los criterios de búsqueda, o se introducen incorrectamente. Se producirá en los siguientes casos:
  - No se introduce ningún valor de fechaInicio o fechaFin.

- Los valores de fechaInicio o fechaFin no se ajustan al formato dd/MM/yyyy.
- El valor de fechaFin es anterior al de fechaInicio.

- Respuesta 404 (NOT FOUND)
  No se ha podido encontrar ningún resultado que satisface los criterios de búsqueda. El cuerpo de la respuesta tendrá formato JSON con la estructura siguiente:

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

Ejemplo:

```
{
  "type": "https://ContadoresWebApi/probs/RefIdentNotFound",
  "title": "No se encontraron datos con los criterios proporcionados.",
  "status": 404,
  "detail": "",
  "instance": "/api/ComprobarLectura/"
}
```

Sección Test

Test

Este método de la API permite comprobar, antes de comenzar a enviar datos, si la conexión con la base de datos es correcta. Para ello, intenta obtener el primer registro de una tabla de ejemplo y, si no hay ningún error, devuelve mensaje de Ok. En caso contrario, devuelve mensaje de error.

Este método no necesita autenticación, por lo que podremos utilizarlo sin el token proporcionado por DameToken.

Ruta relativa: /api/Test

Tipo de llamada: GET

Parámetros: No aplicable.

Ejemplo de llamada a Test:

```
https://comisaria.chj.es/ContadoresWebApi/api/Test
```

Resultados desde el servidor:

- Respuesta 200 (SUCCESS)
  La conexión con la base de datos es correcta. Obtenemos un mensaje de OK que confirma la conexión

- Respuesta 404
  Obtenemos como respuesta un mensaje 404 “Not Found”, que nos indica que no se ha podido conectar con el servidor.

Generación automática de clientes

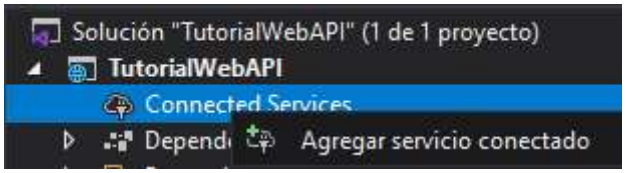
El estándar OpenAPI facilita la creación de clientes mediante un fichero JSON que define la funcionalidad de los servicios. Este fichero JSON puede ser descargado desde la siguiente URL

https://app.chj.es/ContadoresWebApi/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 2019. 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": "WAPIContadores",
  "apiPackage": "WAPIContadores"
}
```

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 temino “Bearer “, (importante el espacio en blanco entre Bearer y el token)

El contenido de la partial class podría ser el siguiente:

```
namespace TutorialWebAPI
{
  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 una lectura de un contador desde la aplicación

```
ResultadoDameToken Token = null;
swaggerClient cliente;
ProblemDetails dameTokenError = null;
ProblemDetails enviarLecturaError = null;
ProblemDetails cambioContadorError = null;

dameTokenError = null;

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

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

  DatosLectura dl = new DatosLectura();
```

```
    dl.Fecha = System.DateTimeOffset.Now;
    dl.IdContador = 37;
    dl.IdCaptacion=4564;
    dl.Valor = 5000000;
    dl.formatoFoto = "";
    dl.fotoBase64 = "";
    await cliente.EnviarlecturaAsync(dl);
}
catch (ApiException<ProblemDetails> api)
{
    await context.Response.WriteAsync(api.Result.Title + "\n" + api.Result.Detail);
}
catch (Exception ex)
{
    await context.Response.WriteAsync(ex.Message);
}

await context.Response.WriteAsync("Success!");
```