# Web Services


versión: 4.x

# Tabla de Contenidos

1

# Módulo de webservices

# Módulo de webservices

El módulo de WebServices de CS-Time permite el intercambio de datos con otros sistemas a través de la tecnología de los Web Services. En este módulo se utiliza la tecnología WCF (*Windows Communication Foundation*) de Microsoft para servir Web Services desde el servidor Http de CS-Time. (Puede encontrar gran cantidad de información y recursos en la página web de Microsoft, http://msdn.microsoft.com/en-us/netframework/aa663324.aspx).

**Requisitos**

Para usar este módulo es necesario cambiar la configuración por defecto de CS-Time modificando el archivo CS-Time.ini para añadir la siguiente configuración:

core.HTTPServer=C:\Archivos de programa\SPEC\CS-TimeV4\bin\HTTPServers.WCFWebServer.dll

Además será necesario contar con **.NET Framework 3.5** con el "**Service Pack 1**" instalado tanto en la máquina servidora como en la que contiene el programa cliente.

## Configuración

La configuración del servidor y del cliente se realiza a través de los archivos .config adecuados, en el caso del servidor se incluirá en el archivo CS-Time.exe.config.

## Configuración para funcionar sin seguridad

Es la manera más simple de usar los web services sin necesidad de usar certificados.

En el archivo CS-Time.exe.config se deberá añadir:

```
<system.serviceModel>
	<bindings>
		<wsHttpBinding>
                                        <binding  name="WSHttpBinding_IWebServiceContract"
maxReceivedMessageSize="100000000">
                        <readerQuotas maxStringContentLength="100000000" />
                        <security mode = "None"/>
                    </binding>
		</wsHttpBinding>
	</bindings>
	<services>
		<service   behaviorConfiguration   =   "MEX"   name="HTTPServers.WCFWebServer.
WebServices.WebService">
                <host>
                        <baseAddresses />
                    </host>
                    <endpoint
                                address  = "http://<SERVER>:<PORT>/WebService"
                                bindingConfiguration = "WSHttpBinding_IWebServiceContract"
                                binding="wsHttpBinding"
                                         contract  =  "HTTPServers.WCFWebServer.WebServices.
IWebServiceContract"
                    />
		</service>
	</services>
```

```
	<behaviors>
		<serviceBehaviors>
			<behavior name="MEX">
				<dataContractSerializer maxItemsInObjectGraph="655360000" />
			  </behavior>
		</serviceBehaviors>
	</behaviors>
</system.serviceModel>
```

Donde <SERVER>:<PORT> corresponde al nombre del servidor donde está instalado CS-Time y al puerto configurado para los clientes del mismo, por defecto el 8091 TCP.

En el archivo de configuración del programa cliente se deberá añadir el siguiente código:

```
<system.serviceModel>
	<bindings>
		<wsHttpBinding>
			<binding                          name="WSHttpBinding_IWebServiceContract"
maxReceivedMessageSize="1000000000" closeTimeout="00:01:00">
				<readerQuotas maxDepth="32" maxStringContentLength="100000000" />
				<security mode = "None"/>
			</binding>
		</wsHttpBinding>
	</bindings>
	 <client>
		<endpoint address="http://<SERVER>:<PORT>/WebService" binding="wsHttpBinding"
                                  bindingConfiguration="WSHttpBinding_IWebServiceContract"
contract="ServiceReference1.IWebServiceContract"
                                                  name="WSHttpBinding_IWebServiceContract"
behaviorConfiguration="SimpleBehavior">
		</endpoint>
	  </client>
	   <behaviors>
		<endpointBehaviors>
			<behavior name="SimpleBehavior">
				<dataContractSerializer maxItemsInObjectGraph="655360000" />
			</behavior>
		  </endpointBehaviors>
	</behaviors>
</system.serviceModel>
```

Donde <SERVER>:<PORT> corresponde al nombre del servidor donde está instalado CS-Time y al puerto configurado para los clientes del mismo, por defecto el 8091 TCP.

## Configuración para funcionar con certificado de seguridad

Por defecto los webservices no tienen seguridad integrada pero modificando el fichero de configuración de la aplicación es posible añadir seguridad. Nuestros webservices implementan una validación del usuario de CS-Time en el servidor. Para ellos es necesario obtener un certificado de confianza de una "Autoridad Certificadora".

Para hacer pruebas es posible crear un certificado en el servidor desde la consola de .Net utilizando los siguientes comandos:

makecert.exe -sr LocalMachine -ss MY -a sha1 -n CN=<SERVERNAME> -sky exchange -pe

certmgr.exe -add -r LocalMachine -s My -c -n <SERVERNAME> -r CurrentUser -s TrustedPeople

Una vez tenga el certificado debe añadir los siguientes elementos en el archivo de aplicación del servidor:

```
<?xml version="1.0" encoding="utf-8" ?>

<configuration>

.....................

<system.serviceModel>

  <bindings>

        <wsHttpBinding>

                                    <binding           name="WSHttpBinding_IWebServiceContract"
maxReceivedMessageSize="100000000">

                      <readerQuotas maxStringContentLength="100000000" />

                      <security mode="Message">

                            <message clientCredentialType="UserName" />

                      </security>

               </binding>

        </wsHttpBinding>

  </bindings>

  <services>

         <service behaviorConfiguration = "MEX" name="HTTPServers.WCFWebServer.WebServices.
WebService">

               <host>

                      <baseAddresses />

               </host>

                   <endpoint   address      =   "http://__<SERVERNAME>:<PORT>__/WebService"
bindingConfiguration = "WSHttpBinding_IWebServiceContract" binding="wsHttpBinding" contract =
"HTTPServers.WCFWebServer.WebServices.IWebServiceContract"  />

        </service>

  </services>

  <behaviors>

        <serviceBehaviors>

               <behavior name = "MEX">

                        <serviceMetadata  httpGetEnabled  =  "true"  httpGetUrl  =  "http://
__<SERVERNAME>:<PORT>__/WebService"/>

                      <serviceCredentials>
```

```
                                <userNameAuthentication  userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="HTTPServers.WCFWebServer.WebServices.
CustomUserNamePasswordValidator, HTTPServers.WCFWebServer" />

                                            <serviceCertificate        findValue="<SERVERNAME>"
storeLocation="LocalMachine" storeName="My" x509FindType="FindBySubjectName" />

                        </serviceCredentials>

                </behavior>

        </serviceBehaviors>

  </behaviors>

</system.serviceModel>

.....................

</configuration>
```

En caso de usar un certificado de prueba no emitido por una Entidad Certificadora, en la aplicación cliente también deberá añadir en el archivo de configuración el atributo behaviorConfiguration="ClientCertificateBehavior" al endpoint y las siguientes lineas:

```
<behaviors>

  <endpointBehaviors>

        <behavior name="ClientCertificateBehavior">

                <clientCredentials>

                        <serviceCertificate>

                                <!--    PeerOrChainTrust usar sólo para certificados no firmados
por una autoridad certificadora            -->

                                <authentication certificateValidationMode="PeerOrChainTrust" />

                        </serviceCertificate>

                </clientCredentials>

        </behavior>

  </endpointBehaviors>

</behaviors>
```

## Obtención del wsdl

Una vez el sistema esté correctamente configurado el wsdl se obtendrá en la url:

http://server:8091/webservice?

donde

*server* es le nombre del servidor donde está instalado CS-Time

*8091* es el puerto configurado en CS-Time para accesos de los clientes de la aplicación.

## Métodos

El wsdl se obtendrá en la url: http://server:8091/webservice?

Los diferentes métodos disponibles utilizan dos conceptos:

**WSElement**: Es un elemento genérico que se utiliza para obtener y enviar datos de los diferentes métodos. Contiene un diccionario donde cada clave es el nombre de un campo de un elemento de la aplicación.

**Container**: Un container es una colección de elementos del mismo tipo de la aplicación, existen containers de empleados, de calendarios, de jornadas, etc. La enumeración WSContainer devuelve todos los containers disponibles.

Los métodos disponibles son:

- ***WSElement List(Container container)***: Obtiene la lista de elementos del **container**. Sólo obtiene el id de cada elemento y el nombre descriptivo.

**Ejemplo**

```
WSElement employees = webservice.List(WSContainer.Employee);
	foreach (KeyValuePair<string, object> pair in employees.Data)
	{
		WSElement employee = pair.Value as WSElement;
		if (employee != null)
			Console.Write( employee.Data["name"]);
	}
```

- ***WSElement ListFields(Container container, string[] fields, string filter)***: Obtiene la lista de elementos del **container**. Obtiene los campos que son indicados en **fields**. La opción **filter** permite una expresión de filtro para obtener sólo los elementos que cumplen la condición.

**Ejemplo**

```
//Fields to return
string[] fields = new string[] { "id" };

//Employees with enroll pending
WSElement employeesIds = webservice.ListFields(WSContainer.Employee, fields, "this.
enrollActive = true");

//Employees with no fingerprint and no cards
employeesIds = webservice.ListFields(WSContainer.Employee, fields, "IsNull(this.Finger)
&& Count(this.Cards) == 0");
```

- ***WSElement Get(Container container, int id)***: Obtiene el elemento con identificador numérico **id** del **container**.
- ***WSElement GetFields(Container container, int id, string[] fields)***: Obtiene el elemento con identificador numérico **id** del **container**. Obtiene sólo los campos del elemento que se pasa en **fields**.
- ***bool Set(Container container, WSElement element)***: Modifica un elemento del **container**. Si el campo **id** de **element** no existe será una inserción si no existe será una modificación. Los campos que no existan en **element** no serán modificados.

- ***bool Delete(Container container, int id)***: Elimina el elemento con identificador **id** del **container**.
- ***Clocking[] Clockings(int employeeId, DateTime firstDate, DateTime lastDate, ClockingType type)***: Obtiene los marcajes del empleado employeeId desde la fecha firstDate a la fecha lastDate. ClockingType nos indica el tipo de marcaje que queremos retornar: Attendance para horario, Access para accesos y Visit para visitas.
- ***WSElement GetResults(int employeeId, DateTime firstDate, DateTime lastDate***): Obtiene todos los resultados del empleado employeeId desde la fecha firsDate hasta la fecha lastDate.
- ***WSElement GetTimeSlot(int employeeId, int readerId, DateTime day)***: Obtiene las franjas de acceso permitido actuales para una persona y lector.
- ***bool SetTimeSlot(int employeeId, DateTime startDay, DateTime endDay, int readerId, WSElement timeSlots)***: Modifica las franjas de acceso de una persona para un lector en un intervalo de días.
- ***bool AddPlanning(int employeeId, string planningName, bool allDay, int startMinute, int endMinute, int timeTypeId, DateTime[] dateInterval)***: Crea una planificación para un empleado
- ***WSElement GetDateRange(DateTime[] dateList)***: Crea un DataRange a partir de un listado de fechas

#### Métodos obsoletos

- ***Result[] Results(int employeeId, string type, DateTime firstDate, DateTime lastDate, string[] resultsNames)***: Obtiene los resultados de un empleado employeeId especificados en resultsNames desde la fecha firsDate a la fecha lastDate. Restricciones de uso: en una misma consulta no es posible incluir resultados aritméticos y de otro tipo; sólo se pueden obtener resultados en formato horario.

"string type" puede ser "total", "daily" o "monthly"

Es possible obtener los valores "accumulated", "available", "start value", etc: **A.Accumulated**.Calculation_Name, **A.Available**.Calculation_Name, **A.Initial**.Calculation_Name

#### Recomendación para crear elementos

Para crear nuevos elementos se recomienda hacer una llamada al método Get pasando el id -1, esto devuelve un WSElement con la estructura de datos necesaria para crear el elemento. Ejemplo de creación de un empleado:

```
//New employee

WSElement employee = webservice.Get(WSContainer.Employee, -1);

employee.Data[“name”] = “ETEST1”;

employee.Data [“nameEmployee”] = “John”;

employee.Data [“LastName”] = “Smith”;

employee.Data [“NoAttendance”] = true;

webservice.Set(WSContainer.Employee, employee);
```

## Ejemplos

#### Añadir un empleado (simplificado)

```
//New employee

WSElement employee = webservice.Get(WSContainer.Employee, -1);

employee.Data[“name”] = “ETEST1”;

employee.Data [“nameEmployee”] = “John”;

employee.Data [“LastName”] = “Smith”;

employee.Data [“NoAttendance”] = true;

webservice.Set(WSContainer.Employee, employee);
```

### Añadir un empleado (extendido)

```
//New employee

WSElement employee = ws.Get(WSContainer.Employee, -1);

employee.Data["name"] = "ETEST1";

employee.Data ["nameEmployee"] = "John";

employee.Data ["LastName"] = "Smith";

employee.Data["sex"] = 1;  //Gender(0:Unknow,1:Male,2:Female)

employee.Data["birthdate"] = new DateTime(1978, 1, 1); //birthdate

employee.Data["email"] = "jsmith@specsa.com";  //Email

employee.Data["Mobile"] = "+34676384932";  //Work mobile

employee.Data["PersonalMobile"] = "+34676848293";  //Personal mobile

employee.Data["ss"] = "33424676848234";  //Personal CNSS

employee.Data["Town"] = "Barcelona";  //Town

employee.Data["Province"] = "Barcelona";  //Province (not have country)

employee.Data["employeeCode"] = "000001"; //employee code

employee.Data["TimeTypesEmployee"] = ws.ListFields(WSContainer.TimeType, new string[] { "id"
}, null); //time types

WSElement readers = ws.ListFields(WSContainer.Reader, new string[] { "id" }, "this.Horario !=
false"); //time & attendance readers

foreach (KeyValuePair<string, object> reader in readers.Data)

{

       WSElement wsreader = reader.Value as WSElement;

       (employee.Data["Readers"]     as     WSElement).Data.Add(wsreader.Data["id"].ToString(),
wsreader.Data["id"]); //reader id

}

employee.Data["Portal.UsaPortal"] = true; //user portal
```

```
employee.Data["loginName"] = "SPEC\\jsmith"; //login name


WSElement calendar = employee.Data["Calendar"] as WSElement; //schedule
WSElement subcalendars = calendar.Data["Calendars"] as WSElement;
WSElement years = new WSElement();
years.Data = new Dictionary<string, object>();
WSElement year = new WSElement();
year.Data = new Dictionary<string, object>();
WSElement days = new WSElement();
days.Data = new Dictionary<string, object>();
WSElement shifts = new WSElement();
shifts.Data = new Dictionary<string, object>();
shifts.Data.Add("0", 2);
WSElement day = new WSElement();
day.Data = new Dictionary<string, object>();
day.Data.Add("shifts", shifts);//Add personal shift to day
days.Data.Add("1", day);
year.Data.Add("Year", 2012);
year.Data.Add("days", days);
years.Data.Add("0", year);
calendar.Data.Add("years", years);
WSElement subcalendar = new WSElement();
subcalendar.Data = new Dictionary<string, object>();
subcalendar.Data.Add("id", 9);
subcalendars.Data.Add("0", subcalendar); //Add schedule
ws.Set(WSContainer.Employee, employee);
```

**Buscar un empleado por DNI y obtener su ID**

*Nota: En la aplicación el campo DNI recibe el nombre interno de name.*

```
WSElement emplist = ws.ListFields(WSContainer.Employee, null, "this.name = \"" +
DNIorMyEmployeeId + "\"");
if (emplist == null || emplist.Data.Count == 0)
{
```

```
        MessageBox.Show("Employee not exists");
        return;
}
//Get the emp internal id
int empId = -1;
foreach (KeyValuePair<string, object> employee in emplist.Data)
{
        WSElement wsEmp = employee.Value as WSElement;
        empId = (int) wsEmp.Data["id"];
        break;
}
```

### Obtener las tarjetas asociadas a un empleado

```
//Fill cards
WSElement employee = ws.Get(WSContainer.Employee, idEmployee);
WSElement wsCards = employee.Data[@"Cards"] as WSElement;
foreach (KeyValuePair<string, object> card in wsCards.Data)
{
        WSElement wsCard = ws.Get(WSContainer.Card, (int)card.Value);
        cards.Items.Add(new    ListItem((int)wsCard.Data[@"id"],    wsCard.Data[@"Number"]    as
string));
}
```

### Buscar un empleado por su número de tarjeta

```
WebServiceContractClient ws = new WebServiceContractClient();
//Fields to return
string[] fields = new string[] { "id" };
//Employees with card number 1153432423
WSElement employeesIds = webservice.ListFields(WSContainer.Employee, fields, "Contains(this.
Cards, Get(\"Tarjeta\", \"1153432423\"))");
```

### Crear una tarjeta y asociarla a un empleado

```
WSElement card = ws.Get(WSContainer.Card, -1);

card.Data[@"Number"] = "000003453454334";

card.Data[@"employee"] = idEmployee; //Need to be assigned

ws.Set(WSContainer.Card, card);
```

**Añadir una planificación a un empleado**

```
WebServiceContractClient ws = new WebServiceContractClient();

ws.AddPlanning(idEmployee, planningName, bAllDay, minuteIni, minuteEnd, idTimeType, new DateTime[] { planningDateIni, planningDayEnd });
```

**Obtener las planificaciones en un intervalo temporal**

```
WebServiceContractClient ws = new WebServiceContractClient();

//Get plannings in the date range from selected employee

string[] fields = new string[] { @"id", @"name", @"stateDescription", @"firstDay" };

WSElement plannings = ws.ListFields(WSContainer.Planning, fields, string.Format(@"this.employee.id == {0} && this.firstDay >= ""{1}"" && this.firstDay < ""{2}""", idEmployee, dateIni.ToString(@"yyyy-MM-dd"), dateEnd.ToString(@"yyyy-MM-dd")));
```

**Eliminar una planificación**

```
WebServiceContractClient ws = new WebServiceContractClient();

ws.Delete(WSContainer.Planning, idPlanning);
```

**Crear o modificar valores iniciales** (válido desde la v.4.1.12.10620)

```
private void InitialValues()

{

        WSElement employee = GetFirstEmployee(new string[] { "id" }, null);

        //Get all employee properties

        employee = ws.Get(WSContainer.Employee, (int)employee.Data["id"]);


        //Get a arithmetic result(only need the id)

        WSElement arithmetics = ws.ListFields(WSContainer.Arithmetic, new string[] { "id" }, null);
```

```
        int idArithmetic = -1;
        foreach (KeyValuePair<string, object> item in arithmetics.Data)
                idArithmetic = (int)(item.Value as WSElement).Data["id"];

        //Create the structure
        WSElement exception = new WSElement();
        exception.Data = new Dictionary<string, object>();
        exception.Data.Add("iniVal", 10); //The value
        exception.Data.Add("hasIniVal", true);
        exception.Data.Add("date", DateTime.Today); //need a closing date

        WSElement exceptions = new WSElement();
        exceptions.Data = new Dictionary<string, object>();
        exceptions.Data.Add("0", exception);

        WSElement initial = new WSElement();
        initial.Data = new Dictionary<string, object>();
        initial.Data.Add("aritmetico", idArithmetic);
        initial.Data.Add("exceptions", exceptions);

        WSElement itemValue = new WSElement();
        itemValue.Data = new Dictionary<string, object>();
        itemValue.Data.Add("id", idArithmetic);
        itemValue.Data.Add("initial", initial);

        //Add the initial value
        WSElement  initialValuesList = employee.Data[@"initialValuesList"] as WSElement;
        initialValuesList.Data.Add((initialValuesList.Data.Count).ToString(), itemValue);
        //SaveEmployee
        ws.Set(WSContainer.Employee, employee);
}
```

**Obtener resultados de un aritmético para un empleado el día actual**

```
//Get all results for today for an employee
result = ws.GetResults(idEmployee, DateTime.Today, DateTime.Today.AddDays(1));
//Internal name of the arithmetic calculation
const string myArithmeticName = "Vacaciones";
//One array element for every day
foreach (KeyValuePair<string, object> date in result.Data)
{
	WSElement calcs = date.Value as WSElement;
	//All arithmetics calculations
	WSElement arit = calcs.Data["Arithmetic"] as WSElement;
	//Selected arithmetic calculation
	WSElement vacation = arit.Data[myArithmeticName] as WSElement;
	//Values for this day
	float vacationConsumedTodayValue = (float)vacation.Data["Value"];
	float vacationStartValue = (float)vacation.Data["Initial"];
	float vacationAvailableValue = (float)vacation.Data["Available"];
	float vacationAccumulatedValue = (float)vacation.Data["Accumulated"];
	float vacationDiscardTodayValue = (float)vacation.Data["Discard"];
	//This element will have a value only if the current day has a transfer
	float vacationTransferredTodayValue = (float)vacation.Data["Transfered"];
}
```