Creatio Academy

Integrations & API

Authentication

Version 8.0

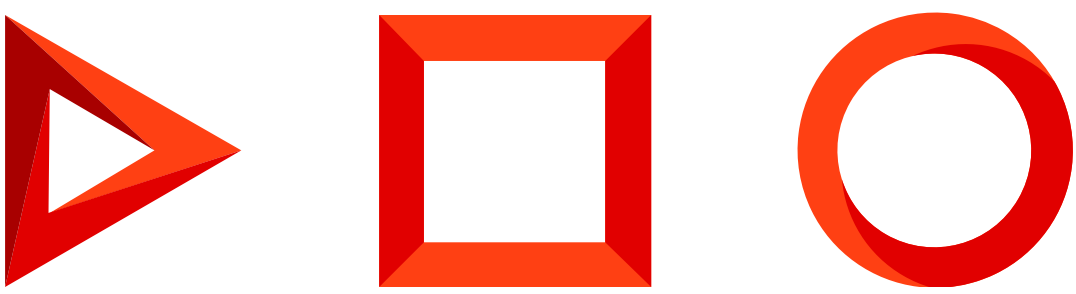

Table of Contents

Authentication

Medium

Authentication verifies the authenticity of the ID specified by a user. The positive **authentication result** is user authorization, i. e., Creatio grants the user permissions to resources they can use to solve their problems.

Attention. Restrict user permissions to objects, records, and columns required for the corresponding integration to work. We recommend creating a specific user for integration and giving the required permissions. If you give unnecessary permissions to the integration user, you put Creatio at security risk. We do not recommend giving the integration user permission to execute the [*Can manage configuration elements*] (`CanManageSolution` code) system operation or Supervisor user permissions.

Learn more about authentication on Wikipedia.

Authenticate all external requests to Creatio web services.

Authentication types

Creatio supports the following **authentication types**:

- Anonymous authentication
- Basic authentication
- Cookie-based authentication (Forms authentication)
- Authentication based on the OAuth 2.0 open authorization protocol. Learn more in a separate article: Set up OAuth 2.0 authorization for integrated applications.

We recommend using **Forms authentication** to integrate external services with Creatio. Forms authentication is implemented via the `AuthService.svc` web service. Use the cookies received in response from the `AuthService.svc` web service in the subsequent requests to Creatio.

View examples that use authentication cookies in separate articles: OData, DataService.

Disable protection against CSRF attacks

CSRF (Cross Site Request Forgery) is a type of an attack on website visitors. CSRF attacks are based on HTTP protocol disadvantages. Protection is enabled by default, but you can disable it.

Attention. We recommend disabling protection against CSRF attacks only when you use **basic authentication**. If needed, you can disable CSRF protection for a single service or several methods of different services. If you disable CSRF protection for all services, you put Creatio at security risk.

You can disable protection against CSRF attacks for the following **service types**:

- all Creatio services
- single Creatio service
- several methods of different Creatio services

Disable protection against CSRF attacks for all services

1. Open the `Web.Config` file in the Creatio root directory.
2. Set the `UseCsrfToken` setting to `false`.

`Web.Config` file

```
<add value="false" key="UseCsrfToken" />
```

3. Repeat the setup in the `...\Terrasoft.WebApp\Web.Config` file.

Disable protection against CSRF attacks for a single service

1. Open the `Web.Config` file in the Creatio root directory.
2. Set the service name to the `DisableCsrfTokenValidationForPaths` setting.

`Web.Config` file

```
<add key="DisableCsrfTokenValidationForPaths" value="/ServiceModel/ MsgUtilService.svc" />
```

Disable protection against CSRF attacks for several methods of different services

1. Open the `Web.Config` file in the Creatio root directory.
2. List the method names in the `DisableCsrfTokenValidationForPaths` setting.

`Web.Config` file

```
<add key="DisableCsrfTokenValidationForPaths" value="/MsgUtilService.svc/Ping,/AuthService.sv
```

Implement authentication using C#

Medium

Example. Implement authentication using C#.

Example implementation algorithm

Create a C# console application in Visual Studio and give it a name, e.g., `RequestAuthentification`.

Example of a software implementation of the authentication

```
// Sends a request to the authentication service and processes the response.
public void TryLogin() {
    var authData = @"{
        ""UserName"":""" + _userName + @""",
        ""UserPassword"":""" + _userPassword + @"""
    }";
    var request = CreateRequest(_authServiceUrl, authData);
    _authCookie = new CookieContainer();
    request.CookieContainer = _authCookie;
    // Upon successful authentication, we save authentication cookies for
    // further use in requests to Creatio. In case of failure
    // authentication application console displays a message about the reason
    // of the mistake.
    using (var response = (HttpWebResponse)request.GetResponse())
    {
        if (response.StatusCode == HttpStatusCode.OK)
        {
            using (var reader = new StreamReader(response.GetResponseStream()))
            {
                var responseMessage = reader.ReadToEnd();
                Console.WriteLine(responseMessage);
                if (responseMessage.Contains("\"Code\":1"))
                {
                    throw new UnauthorizedAccessException($"Unauthorized {_userName} for {_appUr
                }
            }
            string authName = ".ASPXAUTH";
            string authCookeValue = response.Cookies[authName].Value;
            _authCookie.Add(new Uri(_appUrl), new Cookie(authName, authCookeValue));
        }
    }
}
// Create request to the authentication service.
private HttpWebRequest CreateRequest(string url, string requestData = null)
        {
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
            request.ContentType = "application/json";
            request.Method = "POST";
            request.KeepAlive = true;
            if (!string.IsNullOrEmpty(requestData))
```

```
            {
                using (var requestStream = request.GetRequestStream())
                {
                    using (var writer = new StreamWriter(requestStream))
                    {
                        writer.Write(requestData);
                    }
                }
            }
            return request;
        }
// Method realizes protection from CSRF attacks: copies cookie, which contents CSRF-token
// and pass it to the header of the next request.
private void AddCsrfToken(HttpWebRequest request) {
    var cookie = request.CookieContainer.GetCookies(new Uri(_appUrl))["BPMCSRF"];
    if (cookie != null) {
        request.Headers.Add("BPMCSRF", cookie.Value);
    }
}
```

AuthService.svc web-service API

Medium

Request structure

```
// Request string.
POST Creatio_application_address/ServiceModel/AuthService.svc/Login

// Request headers.
Content-Type: application/json; charset=utf-8
ForceUseSession: true

// Request body.
{
    "UserName":"User name",
    "UserPassword":"User password"
}
```

Response structure

```
// HTTP status code.
```

```
Status: code
// Response headers.

Set-Cookie: BPMLOADER=cookie_value; path=/Creatio_application_address; HttpOnly
Set-Cookie:  .ASPXAUTH=cookie_value; path=/Creatio_application_address; HttpOnly
Set-Cookie: BPMCSRF=cookie_value; path=/
Set-Cookie: UserName=cookie_value; expires=date_expire_to; path=/; HttpOnly

// Response body example.
{
     "Code": 0,
     "Message": "",
     "Exception": null,
     "PasswordChangeUrl": null,
     "RedirectUrl": null

}
```

Request string

`method` **required**

Authentication service supports POST HTTP method.

`Creatio_application_address` **required**

Creatio application address.

`ServiceModel/AuthService.svc/Login` **required**

To perform authentication, call the `Login` `AuthService.svc` method.

Request headers

`Content-Type` `application/json` **required**

Encoding and resource type passed in the request body.

`ForceUseSession` `true` **required**

The `ForceUseSession` header accounts for using the existing session.

Request body

The request body must pass the Creatio user credentials. The credentials are passed as a JSON object.

`UserName` `string` **required**

The user name of a Creatio user.

`UserPassword` `string` **required**

The password of a Creatio user.

HTTP status code

`code`

HTTP status code.

Status codes

200 OK	The request has been completed sucessfully and the resource value is not equal to zero. In this case, the request body should contain the authentication status code. If it contains 0, the authentication is successful. In case of unsuccessful authentication, the authentication status code will equal 1 and the request body will contain a message about the cause of the unsuccessful authentication.
403 Forbidden	The server cannot provide access to the resource specified in the request (for example, if a method name is spelled incorrectly). Request body can contain additional information.

Response headers

The response to a POST request contains authentication cookies. You need to save these cookies on the side of the client or on the client computer to use them in your further Creatio web service queries.

Response body

`Code` **required**

If the code contains a “0” value, the authentication is successful. Othewise, it is failed.

`Message` **required**

The message notifying of an unsuccessful authentication.

`Exception` **required**

The object that contains a detailed description of the exception connected with the unsuccessful authentication.