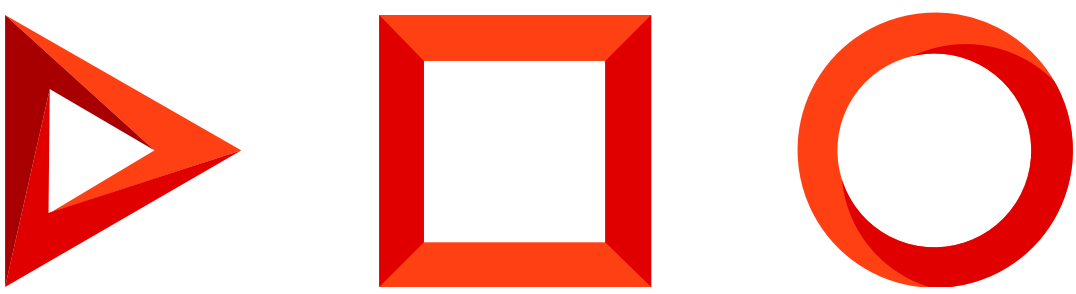
Creatio Academy

Интеграция и внешний API

Аутентификация

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

Аутентификация

Средний

Аутентификация — проверка подлинности предъявленного пользователем идентификатора. Положительным **результатом аутентификации** является авторизация пользователя, то есть предоставление ему прав доступа к ресурсам, определенным для выполнения его задач. Подробнее об аутентификации читайте в Википедии.

Все внешние запросы к веб-сервисам Creatio должны быть аутентифицированы.

Виды аутентификации

Виды аутентификации, которые поддерживаются в Creatio:

- Анонимная аутентификация (Anonymous).
- Базовая аутентификация (Basic-аутентификация).
- Аутентификация на основе cookies (Forms-аутентификация).
- Аутентификация на основе открытого протокола авторизации OAuth 2.0. Настройка подробно описана в статье Настроить авторизацию интегрированных приложений по протоколу OAuth 2.0.

Рекомендуемым способом аутентификации для интеграции с приложением является **Forms-аутентификация**, которая реализована с помощью веб-сервиса `AuthService.svc`. Cookie, полученные в ответ от веб-сервиса `AuthService.svc`, необходимо использовать в следующих запросах к Creatio.

Пример использования аутентификационного cookie приведен в описании сервисов работы с данными OData и DataService.

Защита от CSRF-атак

CSRF (англ. Cross Site Request Forgery — "межсайтовая подделка запроса") — вид атак на посетителей веб-сайтов, использующий недостатки протокола HTTP. По умолчанию защита включена, но ее можно отключить.

> **Важно.** Отключение защиты от CSRF-атак рекомендуется использовать только в случае применения **basic-аутентификации**.

Защиту от CSRF-атак можно **отключить** для:

- Всех сервисов приложения.
- Одного сервиса приложения.
- Нескольких методов разных сервисов.

Чтобы отключить защиту от CSRF-атак для **всех сервисов приложения**, отключите настройку

`UseCsrfToken` в файлах `Web.Config` и `...\Terrasoft.WebApp\Web.Config`.

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

Чтобы отключить защиту от CSRF-атак для **одного сервиса приложения**, задайте имя сервиса в настройке `DisableCsrfTokenValidationForPaths` файла `Web.Config`.

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

Чтобы отключить защиту от CSRF-атак для **нескольких методов разных сервисов**, задайте методы в настройке `DisableCsrfTokenValidationForPaths` файла `Web.Config`.

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

Реализовать аутентификацию на C#

Средний

Пример. Реализовать аутентификацию на C#.

Реализовать аутентификацию

1. Создайте обычное консольное приложение C#, назвав его, например, `RequestAuthentification`.
2. Реализуйте аутентификацию.

Пример программной реализации аутентификации

```
// Отправляет запрос сервису аутентификации и обрабатывает ответ.
public void TryLogin() {
    var authData = @"{
        ""UserName"":""" + _userName + @""",
        ""UserPassword"":""" + _userPassword + @"""
    }";
    var request = CreateRequest(_authServiceUrl, authData);
    _authCookie = new CookieContainer();
    request.CookieContainer = _authCookie;
    // При успешной аутентификации сохраняем аутентификационные куки для
    // дальнейшего использования в запросах к Creatio. В случае неудачной
    // аутентификации в консоль приложения выводится сообщение о причине
    // ошибки.
```

```
    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 {_ap
                }
            }
            string authName = ".ASPXAUTH";
            string authCookeValue = response.Cookies[authName].Value;
            _authCookie.Add(new Uri(_appUrl), new Cookie(authName, authCookeValue));
        }
    }
}
// Создает запрос к сервису аутентификации.
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;
        }
// Метод реализует защиту от CSRF-атак: копирует cookie, содержащий CSRF-токен,
// и помещает его в заголовок следующего запроса.
private void AddCsrfToken(HttpWebRequest request) {
    var cookie = request.CookieContainer.GetCookies(new Uri(_appUrl))["BPMCSRF"];
    if (cookie != null) {
        request.Headers.Add("BPMCSRF", cookie.Value);
    }
}
```

Веб-сервис AuthService.svc API

Средний

Структура запроса

```
// Строка запроса.
POST Creatio_application_address/ServiceModel/AuthService.svc/Login

// Заголовки запроса.
Accept: application/json
ForceUseSession: true

// Тело запроса.
{
    "UserName":"Name of user",
    "UserPassword":"Password of user"
}
```

Структура ответа

```
// Код состояния.
Status: code

// Заголовки ответа.
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

// Тело ответа.
{
    "Code": 0,
    "Message": "",
    "Exception": null,
    "PasswordChangeUrl": null,
    "RedirectUrl": null
}
```

Строка запроса

`POST` **required**

Метод запроса на аутентификацию. Неизменяемая часть запроса.

`Creatio_application_address` **required**

Адрес приложения Creatio.

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

Метод веб-сервиса аутентификации, который необходимо вызвать для выполнения аутентификации. Неизменяемая часть запроса.

Заголовки запроса

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

Кодировка и тип ресурса, который передается в теле запроса.

`ForceUseSession` `true` **required**

Заголовок `ForceUseSession` отвечает за принудительное использование уже существующей сессии.

Тело запроса

`UserName` `string` **required**

Имя пользователя Creatio.

`UserPassword` `string` **required**

Пароль пользователя Creatio.

Код состояния ответа

`code`

Код состояния ответа на запрос.

Возможные коды состояния

200 OK	Запрос успешно завершен и значение ресурса не равно нулю. В этом случае тело ответа должно содержать код состояния аутентификации. Если содержит значение 0, то аутентификация успешна. В случае неудачной аутентификации код состояния будет равен 1 и тело запроса будет содержать сообщение о причине неудачной аутентификации.
403 Forbidden	Сервер не может предоставить доступ к ресурсу, указанному в запросе (например, в случае ошибки в имени метода). Дополнительная информация может содержаться в теле ответа.

Заголовки ответа

Ответ на POST запрос содержит аутентификационные cookie, которые необходимо сохранить на стороне клиента или на клиентском компьютере, чтобы использовать их в дальнейших запросах веб-службы Creatio.

Тело ответа

`Code`

Код состояния аутентификации. Если содержит значение 0, то аутентификация успешна. Иначе — неуспешна.

`Message`

Сообщение о причине неуспешной аутентификации.

`Exception`

Объект, содержащий детальное описание исключения, связанного с неуспешной аутентификацией.