Cezanne OnDemand

OAuth2 flows

Author	Giuseppe Porcelli, Maurizio Romano
Reference	
Revision	
Created	04/11/2013
Updated	26/05/2014

Cezanne Software · 46 Loman Street · London SE1 0EH
Tel +44 (0)20 7202 9300 · Fax +44 (0)20 7202 9321
www.cezannesw.com

Document Control

Revision History

Date	Revision	Author	Comments
26/05/2014		Maurizio Romano	auto_approve parameter Using the flow for single sign-on
26/05/2014		Giuseppe Porcelli	Client credentials flow

Reviewers

Name	Position

Contents

Using OAuth 2.0 to Access Cezanne OnDemand APIs

Cezanne OnDemand APIs use the OAuth 2.0 protocol to manage authentication and authorization. Cezanne OnDemand supports common OAuth 2.0 scenarios such as those for web server, installed, and client-side applications.

OAuth 2.0 is a relatively simple protocol. The first step is to request the registration of your application to the Cezanne OnDemand support. Then your client application requests an access token from the Cezanne OnDemand Authorization Server, reads the token from the response, and sends the token to the API that you want to access.

This section gives an overview of the OAuth 2.0 authorization scenarios that Cezanne OnDemand supports; more detailed content will be provided in the following sections.

Basic steps

1. Ask Cezanne OnDemand support to register your new application.

All applications that access Cezanne OnDemand APIs must be registered by the Cezanne OnDemand support team. The result of this registration process is a set of values (such as a client ID and client secret) that are known to both Cezanne and your application. The set of values varies based on what type of application you are building. For example, a JavaScript application does not require a client secret, but a web server application does.

2. Obtain an access token from the Cezanne OnDemand Authorization Server.

Before your application can access data, it must obtain an access token that grants access to the API. A single access token can grant varying degrees of access to multiple APIs. A variable parameter called scope controls the set of resources and operations that an access token permits. During the access-token request, your application sends one or more values in the scope parameter.

Generally, an application requests an access token using a browser redirect. The request requires an authentication step where the user logs in with their Cezanne OnDemand account. After logging in, the user is asked whether they are willing to grant the permissions that your application is requesting. This process is called user consent.

If the user grants the permission, the authorization server sends your application an access token (or an authorization code that your application can use to obtain an access token). If the user does not grant the permission, the server returns an error.

3. Send the access token to an API.

After an application obtains an access token, it sends the token to the API in an HTTP authorization header. It is possible to send tokens as URI query-string parameters, but we don't recommend it, because URI parameters could be logged by web servers and are not completely secure. Also, it is good REST practice to avoid creating unnecessary URI parameter names.

Access tokens are valid only for the set of operations and resources described in the scope of the token request. For example, if an access token is issued read-only grants, it does not grant access to write operations against an API. You can, however, send the same access token to the API multiple times for operations requiring the same level of privileges.

4. Refresh the access token, if necessary.

Access tokens have limited lifetimes. If your application needs access to an API beyond the lifetime of a single access token, it can obtain a refresh token. A refresh token allows your application to obtain new access tokens.

Refresh tokens should be saved in secure long-term storage as they generally remain valid for long period of times; in certain circumstances when the token storage is not secure by definition (e.g. client JavaScript applications) the refresh token will be not issued at all.

OAuth2 flows

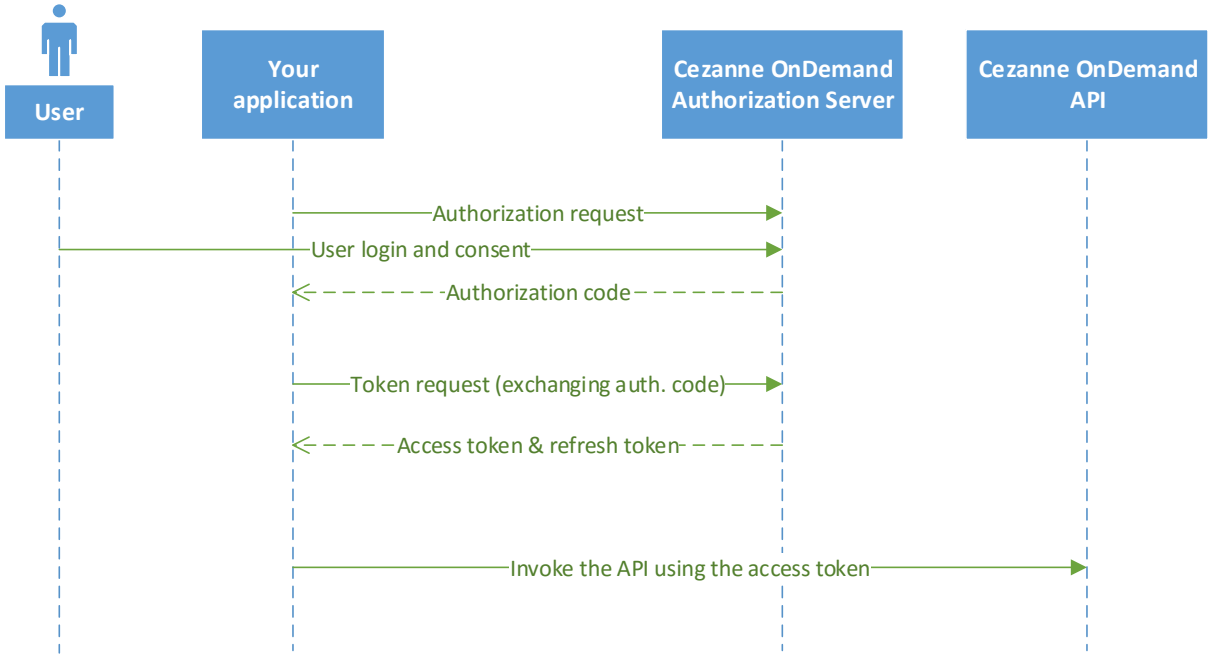
OAuth2 supports different flows, depending on the type of client applications. Cezanne OnDemand currently supports web-server, installed or client-side JavaScript applications.

Web server applications

Web server applications (written with languages and frameworks like ASP.Net, PHP and Java) are fully supported by Cezanne OnDemand OAuth2 endpoints.

The authorization process begins when your application redirects a browser to a specific Cezanne OnDemand authorization URL; the URL includes query parameters that indicate the type of access being requested and the application details. Cezanne OnDemand prompts for the user authentication and consent. The result of this sequence is an authorization code, which the application can exchange later to get an access token and a refresh token.

The application uses the access token to invoke the API and should store the refresh token for future use. Once the access token expires, the application uses the refresh token to obtain a new access token.

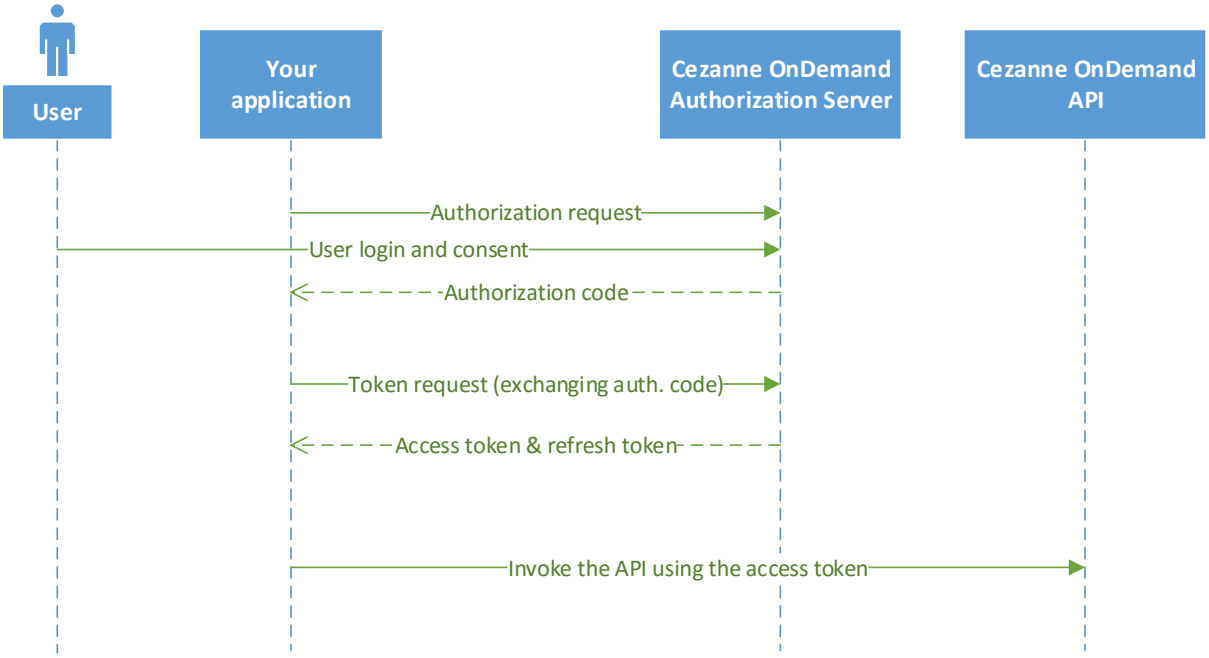


Installed applications

Installed applications are applications installed on devices such as computers, mobile devices and tablets.

Similarly to the web server applications scenario, the authorization process begins when your application redirects a browser to a specific Cezanne OnDemand authorization URL; the URL includes query parameters that indicate the type of access being requested and the application details obtained when asking for application registration. Cezanne OnDemand prompts for the user authentication and consent. The result of this sequence is an authorization code, which the application can exchange later to get an access token and a refresh token.

The application uses the access token to invoke the API and should store the refresh token for future use. Once the access token expires, the application uses the refresh token to obtain a new access token.

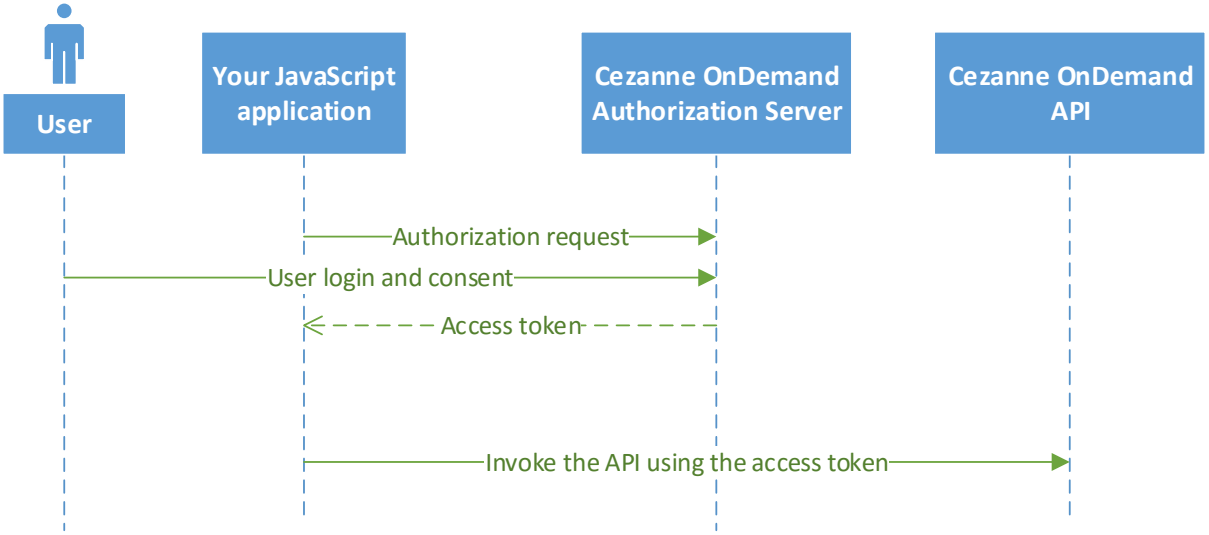


Client-side JavaScript applications

The Cezanne OnDemand OAuth2 authorization server also supports JavaScript applications that run in a browser.

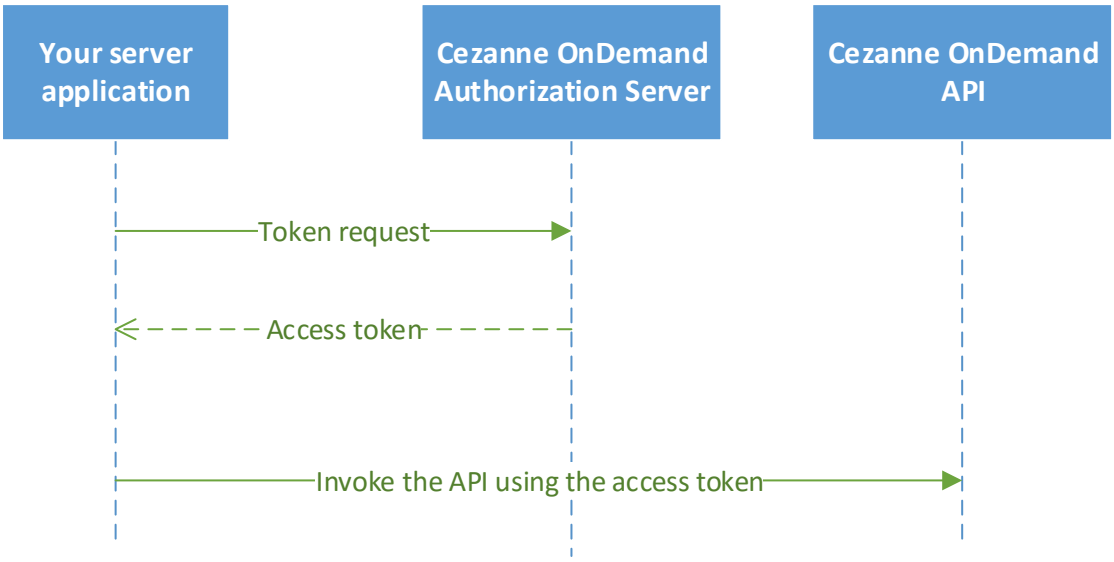
The authorization sequence begins when your application redirects a browser to a specific Cezanne OnDemand authorization URL; the URL includes query parameters that indicate the type of access being requested and the application details obtained when asking for application registration. Cezanne OnDemand prompts for the user authentication and consent.

The result is an access token, which the client application uses to perform the API request. When the token expires, the application repeats the entire process.



Service accounts

Service accounts are specific accounts associated to an OAuth2 client that enable the client to authenticate and act as a specific Cezanne OnDemand user.



The sequence begins with your server application requesting a token to Cezanne OnDemand Authorization Server, providing the client identifier and the client secret as client credentials. The authorization server will authenticate the client, check if the provided client has an enabled service account and return an access token, which the client then uses to perform the API request.

This flow has the following limitations:

- No refresh token is issued
- The client credentials (client identifier and secret) must be kept confidential

OAuth 2.0 for web and installed applications

The Cezanne OAuth 2.0 endpoints support web, mobile and desktop applications.

For web applications, the OAuth Cezanne scenario begins by redirecting a browser (popup, or full page if needed) to a Cezanne URL with a set of query parameters that indicate the type of API access the application requires (scopes).

For desktop and mobile applications, the OAuth Cezanne scenario requires that the application implements embedded browser controls. During application registration, a client secrete is associated to the application, but it not required that this type of application keeps secrets. This scenario begins by redirecting a browser (popup, or full page if needed) to a Cezanne URL with a set of query parameters that indicate the type of access the application requires.

Cezanne handles the user authentication and consent, but the result of the sequence is an authorization code.

The caller receives the authorization code in the query string. After receiving the authorization code, the application can exchange the code for an access token and a refresh token. The application

presents its `client_id` and `client_secret` (obtained during application registration) along with the authorization code when obtaining an access token and refresh token.

The application may access API after it receives the access token. If a refresh token is present in the authorization code exchange, then it may be used to obtain new access tokens at any time.

Authorization request

The authorization request is performed invoking the Cezanne OnDemand authorization endpoint. The endpoint is accessible over SSL only.

Authorization Endpoint
`https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/Authorize`

The endpoint manages the authorization process authenticating the user and handling user consent. The query string parameters supported by the Cezanne OnDemand for these types of applications are:

Parameter name	**Value(s)**	**Description**
`response_type`	`code`	Determines if the Cezanne OAuth 2.0 endpoint returns an authorization code. For installed applications, a value of ***code*** should be used.
`client_id`	The `client_id` value obtained when requesting Cezanne OnDemand support to register a new application.	Indicates the client application performing the request. This value is case sensitive.
`redirect_uri`	One of the `redirect_uri` values registered for the application by Cezanne OnDemand support.	Represent the address where the response is sent by the authorization server. This value is case sensitive and must match exactly the configured one, including the trailing '/'.
`scope`	A space delimited string representing the permissions requested by the application.	The permissions are the grants used to access a specific API. The user has to grant the application to use the requested set of permissions in the user consent page.
`auto_approve`	A boolean value: it can be 'true' or 'false' (without quotes)	This is an optional parameter that specifies if the page that allows to authorize scope must be displayed or not.
`state`	Any string value.	A string that is sent to the server representing the state of the application. This string is then returned in the response. Usages of this parameter can be preventing cross-site-request forgery or preserving some application state. This parameter

		is not mandatory and can be omitted.

An example authorization URL is shown below:

```
https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/Authorize?client_id=oauth2democlient
.app.cezannehr.com &scope=http://www.cezannehr.com/auth-scope/APIRead
http://www.cezannehr.com/auth-scope/APIWrite&redirect_uri=x-cez://oauth-
callback/&response_type=code
```

The auto_approve parameter is an optional parameter which use must be configured when the application is defined by Cezanne team. The possible options for the "**Auto approve mode**" are:

- **Disabled**
  The authorization page is always displayed to the final user and the auto_approve parameter is ignored;
- **Automatic**
  The authorization page is displayed only the first time the user authorizes the application and the auto_approve parameter is ignored;
- **OnDemand**
  The authorization page is displayed at least the first time the user authorizes the application; for the next authorizations the auto_approve parameter specifies if the authorization page should be or not displayed. By default the auto_approve parameter is false.

Choosing a Redirect URI

When you ask Cezanne OnDemand support to register your installed application, you have to communicate your redirect URIs. They are used to return the authorization code to your application.

The authorization code is always returned as a query string parameter on the client. To receive the authorization code on this URL, your application must be listening on the local web server.

For mobile and desktop application you can use:

```
x-cez://oauth-callback/
```

When this value is used, your application may intercept the navigating event of the browser control, disable the current navigation and get the authentication code available on the query string.

Authorization response and exchange for token

After the application receives the authorization code, it may exchange the authorization code for an access token and a refresh token. This request is an HTTP post to the token endpoint. The endpoint is accessible over SSL only.

Token Endpoint
https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/Token

The request can include the following parameters in the body:

Parameter name	**Value(s)**	**Description**

client_id	The client_id value obtained when requesting Cezanne OnDemand support to register a new application.	Indicates the client application performing the request. This value is case sensitive.
client_secret	The client_secret value obtained when requesting Cezanne OnDemand support to register a new application.	A secret for the client ID. Given the nature of the installed applications flow, the client secret cannot be considered confidential.
redirect_uri	One of the redirect_uri values registered for the application by Cezanne OnDemand support.	Represent the address where the response is sent by the authorization server. This value is case sensitive and must match exactly the configured one, including the trailing '/'.
grant_type	authorization_code	As defined in the OAuth 2.0 specification, this field must contain a value of **authorization_code**.
code	A string value.	The authorization code returned from the initial request.

The actual request might look like:

```
POST  https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/Token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: w3.cezanneondemand.com

client_id=oauth2democlient.app.cezannehr.com&
client_secret=A8KbQe6YmLtrhR3WV4Tbz6eqEyWyeaDyMv4c&
redirect_uri=x-cez%3a%2f%2foauth-callback%2f&
grant_type=authorization_code&
code=Pzw5!IAAAAIS6Mcxbxh6W81jE9BCM9YTxstgQPDUmmGSNrtYbCvXwQEAA
```

A successful response to this request contains the following fields:

Parameter name	**Value(s)**	**Description**
access_token	A string.	The access token.
token_type	bearer	The token type. This value is used when invoking an API (see next section).
scope	A space delimited string representing the permissions granted by the user.	The scope string, as specified in the request.
expires_in	An integer value.	The number of seconds the access token is valid. This value can be used to understand in advance that the access token is expiring and refresh the token to get a new one.

`refresh_token`	A string.	A token that may be used to obtain a new access token, and are included by default for installed applications. Refresh tokens are valid until the user revokes access.

A successful response is returned as a JSON array, similar to the following:

```
{
   "access_token":"gAAAAEzcUusQbANDjf1bLK-P5PD6L7RV7PbBtXelFvyq_PdIWSYEtw3MnQO",
   "token_type":"bearer",
   "expires_in":600,
   "refresh_token":"qZR6!IAAAAHvezRU3n1Cd7AxX5nDD7ocM19CO2KJgEUcA9ZPwxmutgQEAA",
}
```

Notes:

- Other fields may be included in the response. Your application should allow additional fields to be returned in the response. The set shown above is the minimum set.
- The actual values of `code`, `access_token` and `refresh_token` parameters have been truncated for readability; real codes are quite longer.

Invoking Cezanne OnDemand APIs

After your application has obtained an access token, your application can access a Cezanne OnDemand API by including it in either an `access_token` query parameter or an `Authorization: Bearer` HTTP header.

For example, a call to the OData API using the Authorization HTTP header looks like the following:

```
GET https://w3.cezanneondemand.com/CezanneOnDemand/DataService.svc/People?$format=json
HTTP/1.1
Authorization: Bearer
gAAAAAddm7cVRPlfGVBCtZsQzxMjwsbki_EX8U5SXcM9RnJGvaRE9DSTvPAMlpiwyMSQTcXPqjfpgDOu-
JZxhqzEEBXPhvlGAGnt4jHWQ8dhJ8
Host: w3.cezanneondemand.com
```

For security reasons, we strongly suggest to use the Authorization header rather than specifying query parameters.

Using a Refresh Token

Obtaining a new access token is simple. To obtain a new access token, make a HTTPs POST to the token endpoint.

Token Endpoint
`https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/Token`

These requests must include the following parameters in the body:

Parameter name	Value(s)	Description
client_id	The client_id value obtained when requesting Cezanne OnDemand support to register a new application.	Indicates the client application performing the request. This value is case sensitive.
client_secret	The client_secret value obtained when requesting Cezanne OnDemand support to register a new application.	A secret for the client ID. Given the nature of the installed applications flow, the client secret cannot be considered confidential.
refresh_token	A string value.	The refresh token returned from the authorization code exchange.
grant_type	refresh_token	As defined in the OAuth 2.0 specification, this field must contain a value of **refresh_token**.

Such a request will look similar to the following:

```
POST https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/Token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: w3.cezanneondemand.com

client_id=oauth2democlient.app.cezannehr.com&
client_secret=A8KbQe6YmLtrhR3WV4Tbz6eqEyWyeaDyMv4c&
grant_type=refresh_token&
refresh_token=qZR6!IAAAANgCRnxuzvvVXyWA98E43oqcceBgsgwpM9a986CGc
```

As long as the user has not revoked the access granted to your application, the response includes a new access token. A response from such a request is shown below:

```
{
    "access_token":"gAAAAJYAHiYAsO062l6mQW7kwe4qBmL5ZL4IYi3s8W7Nx",
    "token_type":"bearer",
    "expires_in":600,
    "refresh_token":"qZR6!IAAAANj4-b4q4XBe5gE6Oe9hlaNrmKWH2jDiSNJBfuXj0OD7GtasLUuAIzA",
}
```

Notes:

- Other fields may be included in the response. Your application should allow additional fields to be returned in the response. The set shown above is the minimum set.
- You should save refresh tokens in long-term storage and continue to use them as long as they remain valid.

Using the flow for Single Sign On (SSO)

What we've illustrated in the previous paragraph is a typical use case of OAuth2 that allows to invoke Cezanne OnDemand API to access data like reports, pictures or to update application entities.

A specific Cezanne API named **TokenInfo** can be used to identify users and link them to previously registered accounts without having to manage their identities. The **TokenInfo** endpoint provides base information related the logged user like first name, last name, email address and a unique **UserGuid** code that identifies uniquely the account. So a company may create in advance their own user account and assign privileges in their own application. Later they can create a relationship between their user and one Cezanne OnDemand account storing the **UserGuid** in the user profile. So if a user has already logged against Cezanne OnDemand, the application invoking the TokenInfo endpoint can discover the user identity and grants him the access.

For example, a call to the **TokenInfo** API using the Authorization HTTP header looks like the following:

```
GET https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/TokenInfo
HTTP/1.1
Authorization: Bearer
gAAAAAddm7cVRPlfGVBCtZsQzxMjwsbki_EX8U5SXcM9RnJGvaRE9DSTvPAMlpiwyMSQTcXPqjfpgDOu-
JZxhqzEEBXPhvlGAGnt4jHWQ8dhJ8
Host: w3.cezanneondemand.com
```

and it will provide a json result that includes the following information:


```
{
        "PersonGuid" : "7ab6ef7a-43cd-472f-852a-bbee3992ad19",
        "TenantGuid" : "00000000-0000-0000-0000-000000000000",
        "RoleGuid" : "00000000-0000-0000-0000-000000000111",
        "UserGuid" : "85feff6b-4d94-47c3-b429-6713ac346196",
        "Username" : "hrprof@cezannesw.com",
        "CountryId" : 1,
        "CountryName" : "United Kingdom",
        "EmailAddress" : "hrprof@cezannesw.com",
        "FirstName" : "HR Professional",
        "FormattedName" : "Cheryl Abrahams",
        "LastName" : "HR Professional",
        "LocaleId" : 2057,
        "LocaleName" : "en-GB",
        "RoleHierarchy" : ["00000000-0000-0000-0000-000000000111"],
        "TimeZone" : "Greenwich Standard Time"
}
```


- PersonGuid, the identifier of the physical person in the application database (if any);
- TenantGuid, the unique identifier of the tenant (company);
- RoleGuid, the identifier of the user role in Cezanne OnDemand;
- **UserGuid**, **the unique identifier of the account**;
- UserName, the user name;
- CountryId, the country code;
- CountryName, the country name;
- FirstName, the first name;
- FormattedName, the formatted (full) name;
- LastName, the last name;
- LocaleId, the locale code;
- LocaleName, the locale name
- RoleHiearchy, the role hierarchy in Cezanne OnDemand which the role guid inherits grants
- TimeZone, the time zone.

OAuth2 for client-side JavaScript applications

This scenario begins by redirecting browser page or a popup to a Cezanne OnDemand authorization URL with a set of query parameters that indicate the type of access the application requires. Like other scenarios, Cezanne OnDemand handles the user logon and consent, and the result is an access token. The access token is returned on the fragment of the response, and a client side script has to extract the access token from the response.

The application can then access an API after it receives and parses the access token. It is strongly suggested to enable HTTPS in your application when implementing this flow.

Authorization request

The authorization request is performed invoking the Cezanne OnDemand authorization endpoint. The endpoint is accessible over SSL only.

Authorization Endpoint
`https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/Authorize`

The endpoint manages the authorization process authenticating the user and handling user consent. The query string parameters supported by the Cezanne OnDemand for client-side application flow are:

Parameter name	**Value(s)**	**Description**
`response_type`	`token`	Indicates that the response type of the authorization request must be a token directly.
`client_id`	The `client_id` value obtained when requesting Cezanne OnDemand support to register a new application.	Indicates the client application performing the request. This value is case sensitive.
`redirect_uri`	One of the `redirect_uri` values registered for the application by Cezanne OnDemand support.	Represent the address where the response is sent by the authorization server. This value is case sensitive and must match exactly the configured one, including the trailing '/'.
`scope`	A space delimited string representing the permissions requested by the application.	The permissions are the grants used to access a specific API. The user has to grant the application to use the requested set of permissions in the user consent page.
`state`	Any string value.	A string that is sent to the server representing the state of the application. This string is then returned in the response. Usages of this parameter can be preventing cross-site-request forgery or preserving some

		application state. This parameter is not mandatory and can be omitted.

An example of authorization URL is the following:

```
https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/Authorize?client_id=oauth2jsclient.app.cezannehr.com&response_type=token&redirect_uri=http://localhost/OAuth2JSClient/&scope=http://www.cezannehr.com/auth-scope/APIRead
```

Authorization response

If the user accepts to grant the requested scope to the application, Cezanne OnDemand authorization server redirects to the configured redirect URI providing the access token as a fragment parameter. The client JavaScript application has to parse the fragment to extract the value of the access token.

The fragment contains also additional parameters like the lifetime of the token and the state, if it was passed in the authorization request. The parameters that can be returned are:

Parameter name	**Value(s)**	**Description**
`access_token`	A string.	The access token.
`token_type`	`bearer`	The token type. This value is used when invoking an API (see next section).
`scope`	A space delimited string representing the permissions granted by the user.	The scope string, as specified in the request.
`expires_in`	An integer value.	The number of seconds the access token is valid. This value can be used to understand in advance that the access token is expiring and re-start the authorization flow to get a new one.
`State`	Any string value.	The state string that is returned in the response.

An example response is show below:

```
http://localhost/OAuth2JSClient/#
token_type=bearer&access_token=gAAAAH5ycGdXFRZKSf1dsQzkwU83xCDli11-
RznDVkCD9DN0hLxYRLdCBsDbhKJNJgT3tG6eb0nzxMS849VJzBBi2Ec1AknqBpE07HhWjKVWwvtN2mBkHrBdko2TX
xlUdTnRN8upSR06nqpeAFTK_0N03JnrcXh0Xz0R9MlZuQc6z8n-
pAEAAIAAAACmzDPBfcrXh8XeFMGsBP5imLcXdB9ejXyAPBK9PtJ4_PPdSq2yZbl6Lp-
XJK9U2_e9TcmInfkIFNZbgWMPgCTytYROPftW78gomdyYSwBC0a9k4HdJUWDKQLwKiLd3GKDlaVgYPAIm4-
qAIASH-tc5MLPkkS-Q7fmc7gz0fOp-skZw-MCPSrwcztZzeMTgk1be9ZnBmv1zqgvxKXJ2B1-
AdMwRvCsz6tQc_RZswfv7Ynns6YIAyjgxAp0JfY_3wLG5HBqjCEszWlp2Um0ZKh7gFuMPugvwgw7kRjtEmOk-e-
xH6SyefMmUeqtZr-
1eXPHN5xsYNCe2VTVezhzfS_Sp6PAhIqqj4W1dWDxaYNoUooFXOEEsYYEiDIoObx0BGoziL5bXOqBuUMeqhFU2mSo
0gZpGqB3KD96PGxSN6L02E70fEYf7XWaPKSG5gwR5XTsHdFdfyVRR45JSGVyRaJ4d2vuRHkVOKuTbCoUJpamrDUeK
Er_6ThzHXvFfEuwwIZPKN-
```

```
rI3tAXOKnrKLjw3kJFYJFT9sauhxZXijix5WtIAQ&scope=http://www.cezannehr.com/auth-
scope/APIRead&expires_in=600&state=myState
```

Please note that our response could contain new parameters in the future; as such your application should be designed to support a variable number of parameters.

Invoking Cezanne OnDemand APIs

After your application has obtained an access token, your application can access a Cezanne OnDemand API by including it in either an `access_token` query parameter or an `Authorization: Bearer` HTTP header.

For example, a call to the OData API using the Authorization HTTP header looks like the following:

```
GET https://w3.cezanneondemand.com/CezanneOnDemand/DataService.svc/People?$format=json
HTTP/1.1
Authorization: Bearer gAAAAH5ycGdXFRZKSf1dsQzkwU83xCDli11-
RznDVkCD9DN0hLxYRLdCBsDbhKJNJgT3tG6eb0nzxMS849VJzBBi2Ec1AknqBpE07HhWjKVWwvtN2mBkHrBdko2TX
xlUdTnRN8upSR06nqpeAFTK_0N03JnrcXh0Xz0R9MlZuQc6z8n-
pAEAAIAAAACmzDPBfcrXh8XeFMGsBP5imLcXdB9ejXyAPBK9PtJ4_PPdSq2yZbl6Lp-
XJK9U2_e9TcmInfkIFNZbgWMPgCTytYROPftW78gomdyYSwBC0a9k4HdJUWDKQLwKiLd3GKDlaVgYPAIm4-
qAIASH-tc5MLPkkS-Q7fmc7gz0fOp-skZw-MCPSrwcztZzeMTgk1be9ZnBmv1zqgvxKXJ2B1-
AdMwRvCsz6tQc_RZswfv7Ynns6YIAyjgxAp0JfY_3wLG5HBqjCEszWlp2Um0ZKh7gFuMPugvwgw7kRjtEmOk-e-
xH6SyefMmUeqtZr-
1eXPHN5xsYNCe2VTVezhzfS_Sp6PAhIqqj4W1dWDxaYNoUooFXOEEsYYEiDIoObx0BGoziL5bXOqBuUMeqhFU2mSo
0gZpGqB3KD96PGxSN6L02E70fEYf7XWaPKSG5gwR5XTsHdFdfyVRR45JSGVyRaJ4d2vuRHkVOKuTbCoUJpamrDUeK
Er_6ThzHXvFfEuwwIZPKN-rI3tAXOKnrKLjw3kJFYJFT9sauhxZXijix5WtIAQ
Host: w3.cezanneondemand.com
```

For security reasons, we strongly suggest to use the Authorization header rather than specifying query parameters.

OAuth2 for server to server applications

Cezanne OnDemand endpoints also support server-to-server interactions like the ones needed when a third party application needs to connect to Cezanne OnDemand to synchronize data on a scheduled basis.
For this scenario you will need to identify an account that will be considered the *service account*: this account must be configured by Cezanne OnDemand support and all service operations will be executed with its security context.

Once the service account is configured, the `client_id` and `client_secret` parameters will act as client credentials used to obtain a valid `access_token`.

Token request

This request is an HTTP post to the token endpoint. The endpoint is accessible over SSL only.

Token Endpoint
`https://w3.cezanneondemand.com/CezanneOnDemand/OAuth/Token`

The request must include the following parameters in the body:

Parameter name	**Value(s)**	**Description**
`grant_type`	`client_credentials`	As defined in the OAuth 2.0 specification, this field must contain a value of **client_credentials**.
`scope`	A space delimited string representing the permissions requested by the application.	The permissions are the grants used to access a specific API.

Also, the request must include an Authorization header, whose value must be set to:

`Basic [the base64 encoded` *`header_value`*`]`

where *header_value* is obtained as:

`client_id:client_secret`

Header name	**Value(s)**	**Description**
`Authorization`	`Basic base64(client_id:client_secret)`	The header must contain the word Basic followed by the Base64 encoded string of the concatenation of client_id and client_secret, colon separated.

The server will return the access token in the response body as in JSON format. An example response is shown below:

```
{
	"access_token":"gAAAACy33Ngzk6OcpYyOALnwEwAzcQFymsrRZLTNpKuky8hdZ8IlpdwyBGzKABu",
	"token_type":"bearer",
	"expires_in":600,
	"scope":"http:\/\/www.cezannehr.com\/auth-scope\/APIRead"
}
```


No refresh token will be generated by this flow, assuming the flow can be re-instantiated to obtain a new access token when the current access token expires.

Using the flow for data synchronization

The server-to-server flow is one of the best options to perform data synchronization for integrations with Cezanne OnDemand.
Data synchronization is executed invoking the OData API whose metadata document is available at:

Dataservice Endpoint
https://w3.cezanneondemand.com/CezanneOnDemand/DataService.svc/$metadata

The following steps are needed to implement the flow correctly:

- Ask Cezanne OnDemand support to create an OAuth2 client enabled for server-to-server flow
- Cezanne OnDemand support will provide you with the client identifier and the client secret plus the granted scopes used to limit the access to read-only or read/write.
- Use the client credentials (`client_id` and `client_secret`) to get an access token.
- Invoke the required APIs to perform data integration including the access token in the Authorization header.

An example of an API call is:


```
GET https://w3.cezanneondemand.com/CezanneOnDemand/DataService.svc/People?$format=json
HTTP/1.1
Authorization: Bearer gAAAAH5ycGdXFRZKSf1dsQzkwU83xCDli11-
RznDVkCD9DN0hLxYRLdCBsDbhKJNJgT3tG6eb0nzxMS849VJzBBi2Ec1AknqBpE07HhWjKVWwvtN2mBkHrBdko2TX
xlUdTnRN8upSR06nqpeAFTK_0N03JnrcGG0Xz0R9MlZuQc6z8n-
pAEAAIAAAACmzDPBfcrXh8XeFMGsBP5imLcXdB9ejXyAPBK9PtJ4_PPdSq2yZbl6Lp-
XJK9U2_e9TcmInfkIFNZbgWMPgCTytYROPftW78gomdyYSwBC0a9k4HdJUWDKQLwKiLd3GKDlaVgYPAIm4-
qAIASH-tc5MLPkkS-Q7fmc7gz0fOp-skZw-MCPSrwcztZzeMTgk1be9ZnBmv1zqgvxKXJ2B1-
AdMwRvCsz6tQc_RZswfv7Ynns6YIAyjgxAp0JfY_3wLG5HBqjCEszWlp2Um0ZKh7gFuMPugvwgw7kRjtEmOk-e-
xH6SyefMmUeqtZr-
1eXPHN5xsYNCe2VTVezhzfS_Sp6PAhIqqj4W1dWDxaYNoUooFXOEEsYYEiDIoObx0BGoziL5bXOqBuUMeqhFU2mSo
0gZpGqB3KD96PGxSN6L02E70fEYf7XWaPKSG5gwR5XTsHdFdfyVRR45JSGVyRaJ4d2vuRHkVOKuTbCoUJpamrDUeK
Er_6ThzHXvFfEuwwIZPKN-rI3tAXOKnrKLjw3kJFYJFT9sauhxZXijix5WtIAQ
Host: w3.cezanneondemand.com
```