Product Documentation

Team Server

Developer Guide

Version 2016 (16.1.4)

Embarcadero Technologies, Inc. is a leading provider of award-winning tools for application developers and database professionals so they can design systems right, build them faster and run them better, regardless of their platform or programming language. Ninety of the Fortune 100 and an active community of more than three million users worldwide rely on Embarcadero products to increase productivity, reduce costs, simplify change management and compliance, and accelerate innovation. The company's flagship tools include: Embarcadero® Change Manager™, Embarcadero® RAD Studio, DBArtisan®, Delphi®, ER/Studio®, JBuilder®, and Rapid SQL®. Founded in 1993, Embarcadero is headquartered in Austin, with offices located around the world. Embarcadero is online at www.embarcadero.com.

May, 2016

CONTENTS

Developer Guide

Team Server provides an API that you can use to create software (applications, plugins, and more) that interacts with Team Server in real time. You can use the API to add, edit, and remove resources, perform searches, and much more.

You can access the API from http://<domain>:<port>/api/v1/<resource> , where <domain> and <port> are those of a running **Team Server** installation, and <resource> is one of the resources that the API supports.

There is a small Java application available that gives a graphical example of the API connection. It is available for download here. A video containing instructions is available here: http://www.embarcadero.com/data-modeling/use-cases/API-demo

Topics

See Also

Obtaining Authorization to Use the API

To use the **Team Server** API, you need to follow these steps:

1. Register your client application with Embarcadero to ensure that the client ID of your client application is not in conflict with the client ID of any other **Team Server** API client application.
2. Register your client application in the target Team Server installation, so that **Team Server** administrators can keep track of your client application.
3. Obtain an API access token that you can provide in your API requests to get access to certain parts of the API that you would be unable to access otherwise. **Team Server** provides three different flows to obtain an access token. Choose the flow that fits your client application best:

Note: All workflows require the credentials of a social user account; non-social users cannot access the Team Server API.

- Web Server Authorization Flow. This authorization flow is for client applications that are accessible via HTTP requests, such as web server applications.
- Embedded Web Browser Authorization Flow. This authorization flow is for client applications that can have an embedded web browser control (web view).
- Password Authorization Flow. This authorization flow is for client applications which users trust with their credentials. Your client application asks users for their credentials, and uses those user credentials to request an access token.

Once you have an access token, include it in all your API requests. For example: http://connect.example.com/api/v1/people?**access_token=1/fFAGRNJru1FTz70BzhT3Zg**

Along with an access token, you get a refresh token. You can use your refresh token to obtain a new access token, for example, when your current access token expires. If you obtain a new access token before your previous access token expires, consider revoking your previous access token.

See Also

- Browsing Resources Using the API
- Managing Resources Using the API
- Registry API

Registering Your API Client Application in the Embarcadero Developer Network

You must register your **Team Server** API client applications in the Embarcadero Developer Network (EDN) to ensure that the client ID of your client applications are not in conflict with the client IDs of any other Team Server API client application.

You only need to register your application in the EDN once, providing information such as the name and a description of your client application.

To register your application in the EDN you need to perform an API request with the following parameters:

Item	Value
URL	http://teamserver.example.com/api/v1/registry/client
Method	POST
Headers	`Content-Type` `application/json`
Body	

```
{
    // Mandatory Fields

    "ednUser": "username", // Your EDN username.
    "ednpwd": "password",  // Your EDN password.

    "name": "My App",      // The name of your client application.
    "description":         // A description of your client
application.
        "My custom API client application",

    "clientType": "1",     // The type of your client application,
which may be any of the following:
                           // • Web Application ("0").
                           // • Native Application ("1").

    // Optional Fields

    "nativeType": "4",     // If you choose "clientType": "1", use
this property to specify which platform your client
                           // application supports. You may use any of
the following values:
                           // • Windows ("0").
                           // • Mac OS X ("1").
                           // • Android ("2").
                           // • iOS ("3").
                           // • Other ("4").

    "home":                // Your client application home page.
        "http://myapp.example.com",

    "image":               // A direct link to an image that
represents your client application.
        "http://myapp.example.com/logo.png",

    "redirectURL":         // The OAuth 2.0 callback URL of your
client application.
        "http://myapp.example.com/oauthcallback"
}
```

If your request is successful, you get the following response:

```
{
    "secret":               // The secret of your client application.
                           // WARNING: Do not share this value publicly.
        "a28e0ca4-27cb-4361-bf97-3b26c612d66a",

    "clientId":             // A UUID that uniquely identifies your client
application.
        "6a2a39ba-9688-493d-b348-187468f599ae",

    "clientType":           // A string that identifies the type of your client
application.
        "installApplication"
}
```

You must store the data from the response, as your application needs it later in the authorization process.

See Also

- Registering Your API Client Application in Team Server
- Registry API

Registering Your API Client Application in Team Server

tem	Value
URL	http://teamserver.example.com/api/v1/clients
Method	POST
Headers	`Content-Type` `application/json`
Body	

```
{
    // Mandatory Fields

    "clientId":            // An UUID that uniquely identifies your client
application.
        "6a2a39ba-9688-493d-b348-187468f599ae",

    "secret":              // The secret of your client application.
        "a28e0ca4-27cb-4361-bf97-3b26c612d66a",

    "name": "My App",      // The name of your client application.
    "description":         // A description of your client application.
        "My custom API client application",

    "clientType": "1",     // The type of your client application, which may
be any of the following:
                           // • Web Application ("0").
                           // • Native Application ("1").

    // Optional Fields

    "nativeType": "4",     // If you choose "clientType": "1", use this
property to specify which platform your client
                           // application supports. You may use any of the
following values:
                           // • Windows ("0").
                           // • Mac OS X ("1").
                           // • Android ("2").
                           // • iOS ("3").
                           // • Other ("4").

    "home":                // Your client application home page.
        "http://myapp.example.com",

    "image":               // A direct link to an image that represents your
client application.
        "http://myapp.example.com/logo.png",

    "redirectURL":         // The OAuth callback URL of your client
application.
        "http://myapp.example.com/oauthcallback",

    "stewards": [          // A list with the username of EDN users that are
stewards of your client application.
```

```
        { "name": "Jane" },
        { "name": "John" }
    ]
}
```

In order to obtain authorization to access the Team Server API you must register your client application in the running instance of **Team Server** that you want to access.

To register your application in the target instance of **Team Server**, perform an API request with the following parameters:

Note: The specified "clientId" and "secret" must be those that you previously registered with Embarcadero.

If your request is successful, you get the following response:

```
{
    "client": {

        // Here go the fields in your request, with the exception of the
"secret" of your client application.
        // Two additional fields are returned:

        "type": "client",

        "url":              // URL that you can use to update, delete and get
your client application data from the API.
            "http://localhost:80/api/v1/clients/6a2a39ba-9688-493d-b348-
187468f599ae"
    }
}
```

Once your client is registered in the target instance of **Team Server**, you can obtain an API authorization token to obtain further access to the API.

See Also

- Registering Your API Client Application with Embarcadero
- Using the Web Server Authorization Flow
- Using the Embedded Web Browser Authorization Flow
- Using the Password Authorization Flow
- Client API

Using the Web Server Authorization Flow

The **Web Server Authorization Flow** allows client applications that are accessible via HTTP requests, such as web server applications, to obtain an API access token.

This authorization flow consists of the following steps:

1. Obtain an authorization code:
    1. Your client application opens **Team Server** in a web browser, using an URL with a set of query parameters that indicate the type of access that your client application requires, as well as a callback URL where Team Server can reach your client application.
    2. **Team Server** handles the user authentication and consent. If the user agrees, Team Server performs a request against the callback URL that your client application provided as a query parameter in the URL (in the previous step). This request includes a query parameter with an authorization code.
2. Exchange the authorization code for an access token:
    1. Your client application, after it receives the authorization code, performs a request against **Team Server** providing the authorization code.
    2. **Team Server** responds with an access token.

You obtain a refresh token as well as an access token. You can use that refresh token to obtain a new access token when your current access token expires.

Obtaining an Authorization Code

You application must open in a web browser an URL that points to the target **Team Server** installation. In this URL, you must include query parameters indicating the type of API access that your client application requires. Team Server authenticates the user, informs the user of the access request, and allows the user to choose whether to grant your client application the requested access permissions or not.

The URL that you must open is http://teamserver.example.com/api/oauth/authorize . You must append the following query parameters to this URL:

Item	Example	Description
response_type	code	The type of response that Team Server must return in the request to your callback URL. Provide the value code here to request that Team Server returns an authorization code in the request to your callback URL.
client_id	6a2a39ba-9688-493d-b348-187468f599ae	The ID of your client application, as registered in the target Team Server installation.
redirect_uri	http://myapp.example.com/oauthcallback	The callback URL of your client application. The value of this parameter must exactly match one of the values used when registering the app.
scope (optional)	read+write	Space-delimited list of access scopes. Possible values are: o read o write o read+write The default value is read+write .
state (optional)	Custom string	Custom string that you want your client application to receive in the request to your callback URL. You can use this parameter for anything you want. For example, you can use it to redirect users to the correct resource in your site.

The following is an example URL:

http://teamserver.example.com/api/oauth/authorize?response_type=code&client_id=6a2a39ba-9688-493d-b348-187468f599ae&redirect_uri=http://myapp.example.com/oauthcallback

After your user logs into **Team Server** and grants your client application access to the API, Team Server performs a request against your callback URL providing a code query parameter. For example:

http://myapp.example.com/oauthcallback?code=6U4MUI

Store that code.

Exchanging an Authorization Code for an Access Token

Once you have an authorization code, you must perform a request against http://teamserver.example.com/api/oauth/token . You must append the following query parameters to this URL:

Item	Example	Description
code	6fGVc2	The authorization code that your client application previously obtains.
client_id	6a2a39ba-9688-493d-b348-187468f599ae	The ID of your client application, as registered in the target Team Server installation.
client_secret	a28e0ca4-27cb-4361-bf97-3b26c612d66a	The secret of your client application, as registered in the target Team Server installation.
grant_type	authorization_code	The type of your access token request. Provide the value authorization_code here, as your request to Team Server is for an access token in exchange for your authorization code.
redirect_uri	http://myapp.example.com/oauthcallback	The callback URL of your client application.

The following is an example URL:

http://teamserver.example.com/api/oauth/token?code=6fGVc2&client_id=6a2a39ba-9688-493d-b348-187468f599ae&client_secret=a28e0ca4-27cb-4361-bf97-3b26c612d66a&grant_type=authorization_code&redirect_uri=http://myapp.example.com/oauthcallback

Team Server responds in JSON format. The server response includes the following information:

```
{
    // Token to include in every API request to get access.
    "access_token": "d4ac0c07-0013-4939-b9ee-0112fdbb7d64",

    // Type of token. This is always "bearer".
    "token_type": "bearer",

    // Token that you can use to get a brand-new access token without further
user interaction.
    "refresh_token": "bcd5a78c-9f0a-4ba6-9baa-5872e5acf7bb",

    // Number of seconds before the access token expires. Default value is
86400 (7 days).
    "expires_in": 86399,
```

```
    // Granted scope.
    "scope": "read write"
}
```

You can now start using the **Team Server** API, including the provided access token in every API request.

See Also

- Obtaining an Access Token Using a Refresh Token
- Using the Embedded Web Browser Authorization Flow
- Using the Password Authorization Flow
- OAuth 2.0 API

Using the Embedded Web Browser Authorization Flow

The **Embedded Web Browser Authorization Flow** allows client applications to have an embedded web browser control (web view).

This authorization flow consists of the following steps:

1. Obtain an authorization code:
    1. Your client application opens **Team Server** in an embedded web browser, using an URL with a set of query parameters that indicate the type of access that your client application requires, as well as a callback URL.
    2. **Team Server** handles the user authentication and consent. If the user agrees, Team Server performs a request against the callback URL that your client application provided as a query parameter in the URL (in the previous step). This request includes a query parameter with an authorization code.
2. Exchange the authorization code for an access token:
    1. Your client application, after it receives the authorization code, performs a request against **Team Server**, providing the authorization code.
    2. **Team Server** responds with an access token.

You obtain a refresh token as well as an access token. You can use that refresh token to obtain a new access token when your current access token expires.

Obtaining an Authorization Code

You application must open in an embedded web browser an URL that points to the target **Team Server** installation. In this URL, you must include query parameters indicating the type of API access that your client application requires. Team Server authenticates the user, informs the user of the access request, and allows the user to choose whether to grant or not your client application the requested access permissions.

The URL that you must open is http://teamserver.example.com/api/oauth/authorize . You must append the following query parameters to this URL:

Item	Example	Description
response_type	code	The type of response that Team Server must return in the request to your callback URL. Provide the value code here to request that Team Server returns an authorization code in the request to your callback URL.
client_id	6a2a39ba-9688-493d-b348-187468f599ae	The ID of your client application, as registered in the target Team Server installation.
redirect_uri	http://myapp.example.com/oauthcallback	The callback URL of your client application. The value of this parameter must exactly match one of the values used when registering the app. You may also choose urn:ietf:wg:oauth:2.0:oob or an http://localhost port. This will determine how the authorization code is returned to you.
scope (optional)	read+write	Space-delimited list of access scopes. Possible values are: o read o write o read+write The default value is read+write .
state (optional)	Custom string	Custom string that you want your client application to receive in the request to your callback URL. You may use this parameter for anything you want. For example, you can use it to redirect users to the correct resource in your site.

The following is an example URL:

http://teamserver.example.com/api/oauth/authorize?response_type=code&client_id=6a2a39ba-9688-493d-b348-187468f599ae&redirect_uri=http://myapp.example.com/oauthcallback

After your user logs into **Team Server** and grants your client application access to the API, Team Server redirects the embedded web browser to your callback URL, providing a code query parameter. For example:

http://myapp.example.com/oauthcallback?code=6U4MUI

Store that code.

Exchanging an Authorization Code for an Access Token

Once you have an authorization code, you must perform a request against http://teamserver.example.com/api/oauth/token . You must append the following query parameters to this URL:

Item	Example	Description
code	6fGVc2	The authorization code that your client application previously obtains.
client_id	6a2a39ba-9688-493d-b348-187468f599ae	The ID of your client application, as registered in the target Team Server installation.
client_secret	a28e0ca4-27cb-4361-bf97-3b26c612d66a	The secret of your client application, as registered in the target Team Server installation.
grant_type	authorization_code	The type of your access token request. Provide the value authorization_code here, as your request to Team Server is for an access token in exchange for your authorization code.
redirect_uri	http://myapp.example.com/oauthcallback	The callback URL of your client application.

The following is an example URL:

http://teamserver.example.com/api/oauth/token?code=6fGVc2&client_id=6a2a39ba-9688-493d-b348-187468f599ae&client_secret=a28e0ca4-27cb-4361-bf97-3b26c612d66a&grant_type=authorization_code&redirect_uri=http://myapp.example.com/oauthcallback

Team Server responds in JSON format. The server response includes the following information:

```
{
    // Token to include in every API request to get access.
    "access_token": "d4ac0c07-0013-4939-b9ee-0112fdbb7d64",

    // Type of token. This is always "bearer".
    "token_type": "bearer",

    // Token that you can use to get a brand-new access token without further
user interaction.
    "refresh_token": "bcd5a78c-9f0a-4ba6-9baa-5872e5acf7bb",
```

```
    // Number of seconds before the access token expires. Default value is
86400 (7 days).
    "expires_in": 86399,

    // Granted scope.
    "scope": "read write"
}
```

You can now start using the **Team Server** API, including the provided access token in every API request.

See Also

- Obtaining an Access Token Using a Refresh Token
- Using the Web Server Authorization Flow
- Using the Password Authorization Flow
- OAuth 2.0 API

Using the Password Authorization Flow

The **Password Authorization Flow** allows client applications to use user credentials in exchange for an access token.

You must perform a GET request against http://teamserver.example.com/api/oauth/token providing the credentials of a **Team Server** user with permission to use the API. In your request, you must append the following query parameters to this URL:

Item	Example	Description
username	username	The username of the user, encoded in UTF-8.
password	password	The password of the user, encoded in UTF-8.
client_id	6a2a39ba-9688-493d-b348-187468f599ae	The ID of your client application, as registered in the target Team Server installation.
client_secret	a28e0ca4-27cb-4361-bf97-3b26c612d66a	The secret of your client application, as registered in the target Team Server installation.
grant_type	password	The type of your access token request. Provide the value password here, as your request to Team Server is for an access token in exchange for user credentials.

The following is an example URL:

http://teamserver.example.com/api/oauth/token?username=username&password=password&client_id=6a2a39ba-9688-493d-b348-187468f599ae&client_secret=a28e0ca4-27cb-4361-bf97-3b26c612d66a&grant_type=password

Team Server responds in JSON format. The server response includes the following information:

```
{
    // Token to include in every API request to get access.
    "access_token": "d4ac0c07-0013-4939-b9ee-0112fdbb7d64",

    // Type of token. This is always "bearer".
    "token_type": "bearer",

    // Token that you can use to get a brand-new access token without further
user interaction.
    "refresh_token": "bcd5a78c-9f0a-4ba6-9baa-5872e5acf7bb",

    // Number of seconds before the access token expires. Default value is
86400 (7 days).
    "expires_in": 86399,
```

```
    // Granted scope.
    "scope": "read write"
}
```

Note: If you get an error instead, check the OAuth 2.0 API troubleshooting information.

You can now start using the **Team Server** API, including the provided access token in every API request.

You obtain a refresh token as well as an access token. You can use that refresh token to obtain a new access token when your current access token expires.

See Also

- Obtaining an Access Token Using a Refresh Token
- Using the Web Server Authorization Flow
- Using the Embedded Web Browser Authorization Flow
- OAuth 2.0 API

Obtaining an Access Token Using a Refresh Token

You can use a refresh token to obtain a new access token without having users go again through the authorization flow.

To obtain a new access token using a refresh token, you must perform a request against http://teamserver.example.com/api/oauth/token providing your refresh token. In your request, you must append the following query parameters to this URL:

Item	Example	Description
refresh_token	bcd5a78c-9f0a-4ba6-9baa-5872e5acf7bb	Your refresh token.
client_id	6a2a39ba-9688-493d-b348-187468f599ae	The ID of your client application, as registered in the target Team Server installation.
client_secret	a28e0ca4-27cb-4361-bf97-3b26c612d66a	The secret of your client application, as registered in the target Team Server installation.
grant_type	refresh_token	The type of your access token request. Provide the value refresh_token here, as your request to Team Server is for an access token in exchange for a refresh token.

The following is an example URL:

http://teamserver.example.com/api/oauth/token?refresh_token=bcd5a78c-9f0a-4ba6-9baa-5872e5acf7bb&client_id=6a2a39ba-9688-493d-b348-187468f599ae&client_secret=a28e0ca4-27cb-4361-bf97-3b26c612d66a&grant_type=refresh_token

Team Server responds in JSON format. The server response includes the following information:

```
{
    // Token to include in every API request to get access.
    "access_token": "d4ac0c07-0013-4939-b9ee-0112fdbb7d64",

    // Type of token. This is always "bearer".
    "token_type": "bearer",

    // Token that you can use to get a brand-new access token without further
user interaction.
    "refresh_token": "bcd5a78c-9f0a-4ba6-9baa-5872e5acf7bb",

    // Number of seconds before the access token expires. Default value is
86400 (7 days).
    "expires_in": 86399,

    // Granted scope.
    "scope": "read write"
```

```
}
```

You can now continue using the **Team Server** API, including the provided access token in every API request.

See Also

- Using the Web Server Authorization Flow
- Using the Embedded Web Browser Authorization Flow
- Using the Password Authorization Flow
- OAuth 2.0 API

Revoking an Access Token

After you request a new access token using a refresh token, if your previous access token has not expired, you should revoke it so that it cannot be used anymore to access the **Team Server** API.

To revoke an access token, perform an API request with the following parameters:

Item	Value
URL	http://teamserver.example.com/api/**revoketoken/<token>**
Method	DELETE

On success, **Team Server** responds "revoke", and the target access token cannot be used anymore to access the **Team Server** API.

See Also

- Obtaining an Access Token Using a Refresh Token
- OAuth 2.0 API

Browsing Resources Using the API

Team Server provides APIs to load and browse information about resources of many types.

Loading Information About a Single Resource

Several APIs let you load the information of a single resource given its ID. If successful, requests to these APIs provide a key with the name of the type of resource (for example, "datasource"). The value of this key is a JSON object (a series of key-value pairs within braces) with detailed information about the target resource, including other resources that are associated with the loaded resource.

For example, performing a GET v1/businessglossaries/15?access_token=<access token> request returns any information about the glossary with ID 15:


```
{
    "businessglossary": {
        "id": 15,
        "status": "",
        "description": "",
        "link": "/glossary/view.spg?glossaryKey=15",
        "name": "asdf",
        "type": "Glossary",
        "stewards": [
            {
                "id": "1",
                "name": "admin",
                "type": "person",
                "url": "Not available"
            }
        ],
        "url": "/v1/businessglossaries/15"
    }
}
```


Retrieving a List of Resources

Most APIs let you retrieve a list of resources of a specific type as well, with requests such as GET v1/diagrams?access_token=<access token>. This is an example response:


```
{
    "metadata_": {
        "limit": 25,
        "totalCount": 3,
        "offset": 0
    },
    "diagrams": [
        {
            "models": [
                {
                    "id": "5",
                    "link": "/object/view.spg?key=5",
                    "name": "Adventure Works",
```

```
                    "type": "Physical",
                    "url": "/v1/models/5"
                },
                {
                    "id": "4",
                    "link": "/object/view.spg?key=4",
                    "name": "Adventure Works DW",
                    "type": "Physical",
                    "url": "/v1/models/4"
                },
                {
                    "id": "3",
                    "link": "/object/view.spg?key=3",
                    "name": "Logical",
                    "type": "Logical",
                    "url": "/v1/models/3"
                }
            ],
            "id": 2,
            "author": "Product Management",
            "createdAt": 1369928964,
            "company": "My Company",
            "link": "/object/view.spg?key=2",
            "name": "Adventure Works.DM1",
            "fileName": "Adventure Works.DM1",
            "type": "Diagram",
            "url": "/v1/diagrams/2",
            "version": "1.1"
        },
        {
            "models": [
                {
                    "id": "5800",
                    "link": "/object/view.spg?key=5800",
                    "name": "Logical",
                    "type": "Logical",
                    "url": "/v1/models/5800"
                }
            ],
            "id": 5799,
            "author": "Product Management",
            "createdAt": 1369928437,
            "company": "My Company",
            "link": "/object/view.spg?key=5799",
            "name": "e-Commerce Order Shopping Cart",
            "fileName": "e-Commerce Order Shopping Cart.dm1",
            "type": "Diagram",
            "url": "/v1/diagrams/5799",
            "version": "1.0"
        },
        {
            "models": [
                {
                    "id": "6106",
                    "link": "/object/view.spg?key=6106",
                    "name": "Logical",
                    "type": "Logical",
                    "url": "/v1/models/6106"
                },
                {
                    "id": "6108",
                    "link": "/object/view.spg?key=6108",
                    "name": "Migration Project (Oracle 8i)",
```

```
                    "type": "Physical",
                    "url": "/v1/models/6108"
                },
                {
                    "id": "6107",
                    "link": "/object/view.spg?key=6107",
                    "name": "Northwind Production (SQL Server 2000)",
                    "type": "Physical",
                    "url": "/v1/models/6107"
                }
            ],
            "id": 6105,
            "author": "Product Management",
            "createdAt": 1369929045,
            "company": "My Company",
            "link": "/object/view.spg?key=6105",
            "name": "Northwind Sample",
            "fileName": "Northwind.dm1",
            "type": "Diagram",
            "url": "/v1/diagrams/6105",
            "version": "1.0"
        }
    ]
}
```


Filtering Lists of Resources

You can filter lists of resources by the first letter of the names of its resources, or using search terms.

To filter by the first letter of the name, use the alphaFilter parameter in the query string of your request. For example: `GET v1/entities/5804/attributes?alphaFilter=Z&access_token=<access token>`

To filter using search terms, use the q parameter in the query strings of your request. For example: `GET v1/tables?q=account&access_token=<access token>`

Paginating Results

To paginate a list of resources, you can include the following parameters in the query string of your request:

- "limit" lets you define the maximum number of resources to retrieve.
- "offset" lets you define a number of resources to skip.

For example, the following request would return a list of entities, filtered so that you only get the results from the 26th to the 35th: `GET v1/entities?limit=10&offset=25&access_token=<access token>`

When you retrieve a list of resources, in addition to the key that contains the list of resources, there is a special key, "metadata_", which includes filtering information. This key provides a JSON object with information about any used filtering and pagination options, including the total number of resources available. You can use this information in the pagination logic of your client application.

Managing Resources Using the API

Team Server provides APIs to create, modify and delete resources such as glossaries, terms, and data sources.

Creating a Resource

To create a resource, you need to perform a `POST` request with the `Content-Type HTTP` header defined with the value `application/json`.

The body of your request must be a JSON object that defines the properties of your new resource, including at least the mandatory fields for the type of the target resource (glossaries, terms, data sources). For example, the following JSON object defines a glossary:

```
{
    "name": "My Glossary",
    "description": "My custom glossary.",
    "status": "In progress",
    "stewards": [
        { "name": "jane" }
    ]
}
```

The URL of the request, relative to the API root (for example, http://teamserver.example.com/api), can be one of the following:

- v1/businessglossaries
- v1/businessterms
- v1/datasources

Modifying a Resource

To modify the data of a resource, you need to perform a `PUT` request with the `Content-Type HTTP` header defined with the value `application/json`.

The body of your request must be a JSON object that defines the properties of your resource that you want to change. For example, the following JSON object changes the status of a glossary:

```
{
    "status": "Finished"
}
```

The URL of the request, relative to the API root (for example, http://teamserver.example.com/api), must include the ID of the target resource. This URL can be one of the following:

- v1/businessglossaries/<id>

- v1/businessterms/<id>
- v1/datasources/<id>

Deleting a Resource

To create a resource, you need to perform a `DELETE` request.

The URL of the request, relative to the API root (for example, http://teamserver.example.com/api), must include the ID of the target resource. This URL can be one of the following:

- v1/businessglossaries/<id>
- v1/businessterms/<id>
- v1/datasources/<id>

See Also

- Obtaining Authorization to Use the API
- Browsing Resources Using the API
- Glossary API
- Term API
- Data Source API

Enabling Secure API Requests

Secure HTTP connections to the **Team Server** API are enabled if secure HTTP is enabled in the Team Server Configuration Manager.

To enable them, you need to enable secure HTTP connections with Team Server.

See Also

- Obtaining Authorization to Use the API
- OAuth 2.0 API