
# NG ConnectionPack Guide

# NG ConnectionPack Guide

## The ability to access to Web based services from Delphi

***by LMD Innovative***

*LMD NG Connection Pack is a part of Next Generation (NG) package suite. All these packages are based on new IDE and language features of latest Delphi IDE versions. NG ConnectionPack provides the ability to access to Web based services (REST services), such as cloud storage services.*

# NG ConnectionPack Guide



Printed: 2020 in Germany.

# Table of Contents

# What's new

# Part I

# 1 What's new

## 2020

- New authentication handlers:
- New task dialog based UI with "Authorize..." button
- System browser authentication with redirect to localhost URLs. System browser usage, as opposed to built-in web-view (`TWebBrowser`), is now required by Google.
- Local HTTP server support to accept authentication localhost redirects with automatic selection of available port.
- Nice response HTML with the message and check-mark icon.
- Special support for localhost redirect URLs allowing to specify auto-port, like "http://localhost:[port]" or explicit port - "http://localhost:[8000]". "127.0.0.1" is also supported.
- API-Explorer:
- Colors improvement. Especially for dark IDE theme (Delphi 10.3 and higher).
- Search/filtering in services and operations lists.
- Minor bugs fixing in API-Explorer.

## 2018

- Package has been fully re-designed. More formal way of implementing REST services has been implemented.
- New advanced API Explorer tool has been provided, which can be used directly inside Delphi IDE at design-tim, and allows to test and learn REST services quickly and intuitively, without writing code.
- Supported service list has been changed. Please read here about currently supported services.
- Package now tries to provide most complete REST service wrappers. REST services are implemented "as is", much closer to original services structure. Service operation list, requests parameter list, resulting data format - all are quite similar to the original services structure, provided by respective vendors.
- New formal type system for data types and requests parameters: supports primitive, list, map and objects types. As well, special types for uploading and downloading stream data is provided.
- All types are nullable, which allows to support dynamic nature of the corresponding REST services. Developer can always specify only required request parameters, and as well, can detect, which data properties has been really returned from the server.
- New `EHttpError` exception class is used to report HTTP exceptions, which provides access to `StatusCode`, `StatusText` and extended `ErrorInfo` text, returned as a `HTTP` body.

Please read here about new package architecture.

# Overview

# Part II

# 2 Overview

LMD NG ConnectionPack is a part of Next Generation (NG) package suite. All these packages are based on new IDE and language features of latest Delphi IDE versions. NG ConnectionPack provides the ability to access to Web based services (REST services), such as cloud storage services.

The first thing required to connect REST services is to register your application. This is usually done by creating an account in the developer's zone of the corresponding service provider web site, and then filling the application description data, such as the Name, Logo, and some other parameters. As a result you'll get application's `ClientID` and `Secret` codes. Provided `ClientID` and `Secret` values should be copied into corresponding properties of the NG ConnectionPack components. More precisely the registration process is described in Application Registration section.

After you've got your `ClientID` and `Secret` codes NG ConnectionPack components will be able to connect to the corresponding REST services. However, application's end-users are also required to authenticate themselves in the corresponding services. This process is called authentication, and performed using modal dialog, where the corresponding REST service login page is shown to the user along with your registered application name, logo and required access rights. After successful user login, the service provides so called `access_token` (and some other info), which is used subsequently to sign Http requests.

Its important to note that returned authentication info can be stored and loaded using `SaveState` and `LoadState` methods, and its strongly recommended to use there methods to persist authentication info between your application runs. First, this will prevent the user from logging in each time your application run. Second, for some services there is an explicit limit of tokens, which can be returned, and thus the requirement of storing authentication info between application runs are stated explicitly in the service's documentation. Current access token can be reset using `ResetToken` method; the user will be forced to re-authenticate after token reset.

NG ConnectionPack shows authentication dialog as required on demand; there no way to show login dialog explicitly, but you can always check that the authentication info is Ok, using `RefreshAccess` public method. Look at Authentication section for further description.

After successful user authentication, real features of the services become available to the application. Here its important to note that NG ConnectionPack components implement *state-less* wrappers for corresponding REST services. Components do not store internally any returned data, such as file objects or calendar events. On each request new set of objects will be returned. Most of such objects are really implemented as Delphi records (thanks to Delphi new language features, which allow the records to have properties and methods); so, the memory management is mostly automatic. Look at State-less and Memory Management section for more info.

NG ConnectionPack ships with visual tool, called API Explorer, which accessible at design-time via Delphi's Tools menu sub-item or as a component editor for NG ConnectionPack components. API Explorer provides an easy and intuitive way for REST operations execution, including uploading and downloading data without code writing. Its a great tool for test purposes and services understanding. Please read here.

Note: Starting from 2018 release NG ConnectionPack has been almost fully re-designed. The packages no longer provide unified interface for cloud storage components. Instead we've concentrated on the formal way of implementing REST services, which allow to provide most complete functionality for each service, with API structure, closely related to corresponding services API and documentation. For more information about package architecture please look here.

# Components

NG ConnectionPack include many service components, accessible from the Delphi's component palette at design-time. The list of supported services is big, and following only some examples are represented:

- `TNGDropbox`
- `TNGBoxNet`
- `TNGGCalendars, TNGGDrive, TNGGMail, TNGGPeople, TNGGSheets, TNGGTasks,` ect.
- Full service list along with descriptions and documentation links is provided in Service List topic.

Each component is a descendant of `TNGRestService` base class, which offers application settings management, authentication/token management and access scopes management.

Otherwise, each component have its own public interface, which is closely related to the corresponding REST service API. Its a good idea to read corresponding REST services API documentation to understand its methods and related data structure; we cannot provide the documentation for each service, because of big amount of services and its methods/data.

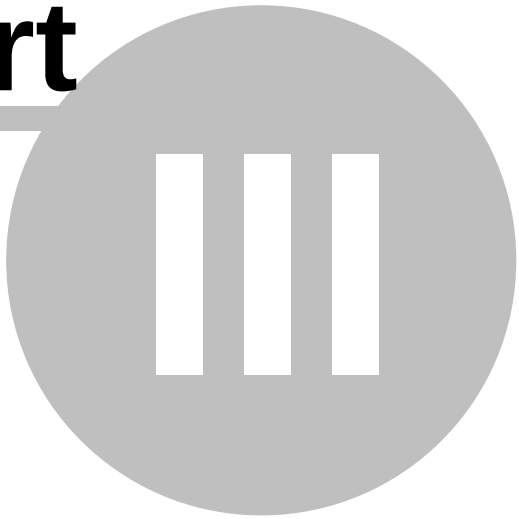
# Features

Following is a short feature list of NG ConnectionPack package:

- Provides most complete state-less support of various REST service APIs.
- Provides OAuth 2.0 authentication dialog, along with a custom auth-UI interface for building application specific authentication dialogs.
- Provides API Key authentication method for Google services.
- Allows to store/load/reset authentication state to prevent explicit user authentication each time application starts.
- Provides a smart memory management model, simplifying the package usage.
- Provides grouping capability, which allow to implement a single authentication into a group of services of the same vendor (specifically - for Google services).

# Package Architecture

# Part



# 3 Package Architecture

# Components

NG ConnectionPack installs components into Delphi Component Palette, where each component represents a single REST service. All components are descendants of `TNGRestService` base class. Base class provides common public interface for application setup, authentication and token handling.

## TNGRestService Base Class

The following properties are used to make initial application setup:

| | |
|---|---|
| `ClientID` | This property should be set to the corresponding client id of the registered application. |
| `Secret` | This property should be set to the corresponding client secret code of the registered application. |
| `RedirectUrl` | This property has a default value for some services, which should not be changed in most cases. But, unfortunately, some service vendors does not provide a default redirect page, so its user's responsibility to provide one. For example, for TNGBoxNet tests and demos we've used https://blank.org publicly available site. |
| `Scopes` | This property is can be used to setup access scopes. `Scopes` property has initial default value, so specifying scopes is optional. Also `SetDefaultScopes` method can be used at run-time to reset scopes list to its default value. Changing `Scopes` property at run-time can reset current access token. |

The following properties and methods can be used to handle authentication token:

| | |
|---|---|
| `RefreshAccess` | Checks, that the current access token is exist and is not expired. Access tokens have limited live time, so they should be refreshed from time to time. NG ConnectionPack do this automatically, so, usually, there no need to call this method explicitly. Also, note, that this method does not guarantee that the subsequent REST API call will succeed, but, usually it will. Read more here. |
| `ResetToken` | Clears existing access token. The method can be used to "log out" current user, and subsequently, the user will be forced to re-authenticate. |
| `SaveToken` | Save current access token and related information to stream or file. |
| `LoadToken` | Load access token information from stream or file. |
| `Token` | Provides read-only access to current access token value. |
| `TokenScopes` | Provides read-only access to scopes with which current access token is compatible. |
| `Refresh` | Provides read-only access to current refresh token value. |
| `Expires` | Provides read-only access to the date-time point, when current access token will expire. |
| `OnTokenChanged` | The event is fired when the user is re-authenticated or access token refreshed using `RefreshAccess` or `ResetToken` methods or, implicitly, as a result of calling REST API function. |

## REST Operations

Rest operations are implemented as component public methods, which return corresponding request objects. For example:

```
var
  rq:  TNGGDrive.TListFiles;
  lst: TNGFileList;
begin
  rq := NGGDrive1.Files_List;
  rq.Q('trashed=false');
  lst := rq.Execute;
end;
```

Here `ListFiles` is a REST operation, which correspond to Google Drive API `Files.List`. The method is a public method of the corresponding `TNGGDrive` component class, which returns `TListFiles` request object. `TListFiles` request type is declared as a nested type inside the parent REST service component class, and so should be referenced in custom code as `TNGGDrive.TListFiles`.

Request object contains builder methods, which correspond to REST API parameters, and allow to setup REST request. Please note, that some parameters can be optional, while other are mandatory. Request object parameters are then converted to parts of the HTTP request URL, URL query parameters, HTTP request headers, or HTTP body. For example, `Q` method has been used above to specify file search query. Builder methods has been specially designed to shorten request configuration code, and always returns `Self` (request). So, the example can be re-written in a more compact way:

```
var
  lst: TNGFileList;
begin
  lst := NGGDrive1.Files_List
                  .Q('trashed=false')
                  .Execute;
end;
```

Builder methods only allow to set request parameters, there no ways to retrieve values of specified request parameter back.

To send request to REST server and execute it, each request object declare `Execute` method. The method returns operation results data, which is of type `TNGFileList` in above example.

# Data Types

As described above REST operation methods return request objects, and, as well, request `Execute` methods return resulting data. NG ConnectionPack has its own formal way of describing structure of request object builder parameters and returning resulting data. Usually, a lot of data types are declared in addition to a service component class. Above example uses `TNGFileList` data type, which is a collection of file objects returned from `ListFiles` operation. The following type kinds are supported by NG ConnectionPack data type system:

- `NGString`
- `NGInteger`
- `NGInt64`
- `NGBoolean`
- `NGFloat`
- `NGDateTime`
- `NGDate`
- `NGUpload`
- `NGDownload`
- `NGList<T>`
- `NGMap<T>`

- NGObject

Primitive types, such as NGString and NGInteger represents individual primitive values, such as object property or array element values. The main difference between corresponding Delphi primitive types is that NG ConnectionPack values are nullable, which is described below in more details. NGList<T> template type allow to declare collection like type of any primitive, map or object elements. NGMap<T> template type allow to declare a dictionary of values of type T with string keys, which is used in some services to decode JSON object values; for example, a collection of colors in Google Calendars API. NGObject type is a base class for object like values, which have predefined typed set of properties. NGUpload and NGDownload types are special types for performing stream based data uploading and downloading.

## Nullable Types

Primitive types are nullable; they can store special "null" value, which should be treated as undefined or unspecified. The rationale of having nullable types is:

Starting from 2018 release NG ConnectionPack introduce REST services "as is". Component set of operation methods are closely related to the corresponding REST service API operations, as well, as operations parameters and resulting data structure. This way we provide most complete implementation of REST services. However, we cannot test, even theoretically, all possible combinations of requests parameters and returned data. So, its impossible to decide, which parameter can be omitted in request or which property can be unspecified in operation result data.

So, nullable types allow to omit request parameter or a property in an object request parameter, which is very important in REST services, due to its very dynamic nature. As well nullable types allow to know for the user, which properties was really returned in a resulting data object.

Most types, except NGObject, are really implemented as smart records in Delphi, which are records types with properties, methods and overloaded operators. NG ConnectionPack types support the following interface, which relates to nullablity concept:

| | |
|---|---|
| HasValue | The function allows to determine whether the variable or property has assigned a value of its null (undefined). Trying to read value from null (undefined) variable, will raise an Exception. |
| SetNull | The function allow to clear value from variable or property; after that HasValue function will return False. |
| Implicit | Implicit cast operator from special TNGNullType type. Works just like SetNull function. TNGNullType type is simple an enumeration with a single declared value NGNull. |

Examples:

```
var
  v: NGInteger;
```

Check, whether the variable v has value inside:

```
if not v.HasValue then
  ShowMessage('v is null');
```

Clear variable v value using implicit operator and special value NGNull (which is of TNGNullType):

```
v := NGNull;
```

## Primitive types

Primitive types, like `NGString` or `NGInteger` can store simple values of corresponding Delphi's primitive types. Primitive types provide declare the following public interface:

| | |
|---|---|
| `HasValue` | `HasValue` and Implicit cast operator from `TNGNullType` - nullablity handling function, as described above. |
| `Value` | Read-only property which allows to read the corresponding value stored in variable or property. If the variable has no value, exception will be raised. |
| `ValueOrDefault` | The function returns either stored in a variable value, or, if variable stores no value, default value. |
| `Implicit` | Two implicit operators for converting from/to corresponding Delphi primitive type. Read operator will raise exception, if a variable has no value, just like `Value` function. Write operator always succeeds. |

Examples:

```
var
  v: NGInteger;
```

Convert some integer number to `NGInteger` type using implicit operator and assign the resulting value to `v`:

```
v := 7;
```

Read variable `v` value using `Value` property or implicit cast operator:

```
x := v.Value;
x := v; // Implicit conversion.
```

Read variable `v` value using `ValueOrDefault` method:

```
x := v.ValueOrDefault(7);
```

In some cases Delphi compiler will not deduct that implicit conversion to base primitive type is required. This can happens while overloaded methods resolution or in some build-in binary operators. To workaround such cases, in addition to `Value` property shown above, explicit type conversion can be used:

```
SomeMyProc(Integer(v));
```

## Type "any"

Type `NGAny` can represent a values of any supported in JSON type, like string, number, boolean, array or object. The support of `any` type in NG ConnectionPack is limited, and its value can be read/written as pure JSON string only. This JSON string is injected into requests "as is" without any validation.

| | |
|---|---|
| `HasValue` | `HasValue` and Implicit cast operator from `TNGNullType` - nullablity handling function, as described above. |
| `Json` | Read-only string property which allows to read the corresponding JSON value stored in variable or property. If the variable has no value, empty string is returned. |
| `FromJson` | Static function, which allow to create `NGAny` values from pure JSON string or from `TNGJson.TBuilder` values. `'null'` JSON string value is convcerted to undefined |

| | |
|---|---|
| | `NGAny` value. |
| `Implicit` | Several implicit cast operators are provided for converting from Delphi primitive type, like strings, numbers, booleans, dates or date-time values. |

Examples:

```
var
  v: NGAny;
```

Convert some values to `NGAny` type using implicit operator and assign the resulting value to `v`:

```
v := 7;
v := 'my string value';
v := True;
```

Read variable `v` value using `Json` property:

```
json := v.Json;
```

## Lists

Lists, which are collections of objects or primitive values are represented by `NGList<T>` template type. Some nullable underlying type, such as `NGString` or `NGFloat` should be used as a `T` parameter value. For simplicity lists are not nullable. They can only be empty - contains no items. List instances are auto-initialized, including object properties and list/map items of `NGList<T>` type; so lists usage is very simple:

```
var
  obj: TNGSomeDataObject;
begin
  obj.ListProp.Add(7);
  obj.ListProp.Add(9);
  ...
end;
```

In case of list of objects there no need to destroy object item values, because the values are owned by the parent list instance and will be automatically destroyed with it.

List declare the following public method and properties:

| | |
|---|---|
| `Add(Value: T)` | Method adds new value to the list. |
| `Delete(AIndex: Integer)` | Method deletes list item at `AIndex` index. Since memory management is mostly automatic in NG ConnectionPack, there no need to dispose deleted item value |
| `Clear` | Method removes all existing items from the list. |
| `Count` | Read-only property, which provide access to list items count. |
| `High` | Read-only property, which is equal to `Count - 1`, and works similar to Delphi's standard `High` function. |
| `Items[AIndex: Integer]: T` | Property, which provides access to list items. |

Examples:

```
var
  a: NGListy<NGInteger>;
```

Add new item to the list:

```
a.Add(7);
```

Iterate item elements:

```
sum := 0;
for i := 0 to x.High do
  Inc(sum, a[i]);
```

Clear items variable:

```
a.Clear;
```

## Maps

Maps implements a dictionary concept with string keys. Maps are similar to objects, however, unlike objects, which provides predefined set of properties of different types, maps allows to have dynamic set of properties of the same type. Maps are useful in some cases to represent JSON data that need to be sent or received from REST service. One example of maps usage is a collection of colors in Google Calendars service, where element keys represent color names, and element values represent HEX color values.

Map are represented by `NGMap<T>` template type. Some nullable underlying type, such as `NGString` or `NGFloat` should be used as a `T` parameter value. For simplicity maps are not nullable like lists. They can only be empty - contains no items. Map instances are auto-initialized, including object properties and list/map items of `NGMap<T>` type; so maps usage is very simple:

```
var
  obj: TNGSomeDataObject;
begin
  obj.MapProp['MyKeyA'] := 7;
  obj.MapProp['MyKeyB'] := 9;
  ...
end;
```

In case of map of objects there no need to destroy object item values, because the values are owned by the parent map instance and will be automatically destroyed with it.

Maps declare the following public method and properties:

| | |
|---|---|
| `Remove(AKey: string)` | Procedure deletes map element with `AKey` string key. Since memory management is automatic in NG ConnectionPack, there no need to dispose deleted element. |
| `ContainsKey(AKey: string)` | Function determined, whether specified element with specified `AKey` key is resided inside map. |
| `Keys: TArray<string>` | Function returns Delphi's array of keys, contained in map. |
| `Clear` | Method removes all existing elements from the map. |
| `Count` | Read-only property, which provide access to maps's element count. |

| | |
|---|---|
| `Items[AKey: string]: T` | Property, which provides access to map elements. |

Examples:

```
var
  m: NGMap<NGInteger>;
```

Add new or replace existing element in map:

```
m['MyKey'] := 7;
```

Iterate map elements:

```
keys := m.Keys;
for i := 0 to High(keys) do
begin
  elem := m[keys[i]];
  ...
end;
```

Clear map variable:

```
m.Clear;
```

## Objects

Objects provides predefined set of properties of different types. Unlike maps and arrays, which are template types, objects are pre-declared types in NG ConnectionPack. NG ConnectionPack contains a lot of pre-defined data objects types, since each REST service require quite various data for different operations. Unlike other types, objects are implemented in Delphi using Delphi object types, so, explicit destruction is required for object instances in some cases. The nullability of objects is supported via standard Delphi `nil` concept. Data object types declares parameter-less constructor, which should be used to create object instances:

```
var
  o: TNGSomeDataObject;
begin
  o := TNGSomeDataObject.Create;
  try
    ...
  finally
    o.Free;
  end;
end;
```

Object properties of object types (child objects) are owned by the parent object instance and will be automatically destroyed with it:

```
var
  o: TNGSomeDataObject;
begin
  o := TNGSomeDataObject.Create;
  try
    o.ObjProp := TNGOtherDataObject.Create;
    o.ObjProp.X := 7;
    ...
  finally
    o.Free; // o.ObjProp instance is automatically destroyed here.
  end;
end;
```

Objects declare various read-write data specific properties; for example, `TNGFile` object type declares a lot of supported by Google Drive API properties, such as `Id`, `Name`, `Size`, ect.

Object properties are undefined (null) by default. Internal implementation does not store underfined properties, and this is another implementation aspect, which allow NG ConnectionPack to provide most complete REST services implementation, because, usually, rest data is quite big and represents a lot of properties. For example, `TNGFile` data type declares 51 property.

Examples:

```
var
  o: TNGFile;
```

Create new object instance using default constructor:

```
o := TNGFile.Create;
```

Assign values to object properties:

```
o.id   := '234234234';
o.Name := 'MyFile.txt';
o.Size := 1024;
```

Destroy object variable:

```
o.Free;
```

## Upload and Download

When REST service operation is intended for data uploading, its request object have a builder method with a parameter of `NGUpload` type. For example, request object of `TNGGDrive.CreateFileUpload` operation have `Data(Value: NGUpload)` method, which allows to setup uploading data:

```
var
  upl: NGUpload;
  fl:  TNGFile;
begin
  upl.Stream := MyReadDataStream;
  fl := NGGDrive1.Files_CreateUpload
                 .Data(upl)
                 .Execute;
end;
```

`NGUpload` type declares the following properties, which allows to setup data stream:

| | |
|---|---|
| `Stream` | Property allows to specify data to be uploaded as a standard Delphi's `TStream` object. The content of this associated stream will be read during REST operation execution. To read whole stream NG ConnectionPack will first reset stream's position to point to the beginning of the stream. Non nil value for `Stream` property is required to be specified. Otherwise an exception will be raised during operation execution. |
| `MimeType` | Property allows to specify data mime type. This property is optional for some operations, but required for others. As well, operation can use some predefined default MimeType, if its not specified in `NGUpload` object. |

For downloading data `NGDownload` type can be used. Its generally similar to `NGUpload` type. And its also provide `Stream` property, which in this case should be set to reference of `TStream` object, which

will receive data during operation execution. As well, for download operations, request Execute method takes `NGDownload` object as a parameter:

```
var
  dnl: NGDownload;
begin
  dnl.Stream := MyWriteDataStream;
  NGGDrive1.Files_GetDownload
           .Execute(dnl);
end;
```

Please note, that big data download is performed incrementally, using several underlying network messages. So, a network failure error can occure, somewhere in the middle of downloading process, when some downloaded data has been already written into the stream.

Associated `TStream` object is not owned by `NGUpload` or `NGDownload` objects, and so, need to be destroyed eventually after operation execution.

The rationale of using distinct `NGUpload` and `NGDownload` types instead of simple Delphi's `TStream` type is:

- First, `NGUpload` and `NGDownload` types are smart records, just like any other data types in NG ConnectionPack.
- Second, NG ConnectionPack types provide some additional interface, such as `MimeType` property in `NGUpload` type; and more properties can be added in future.

# Exceptions

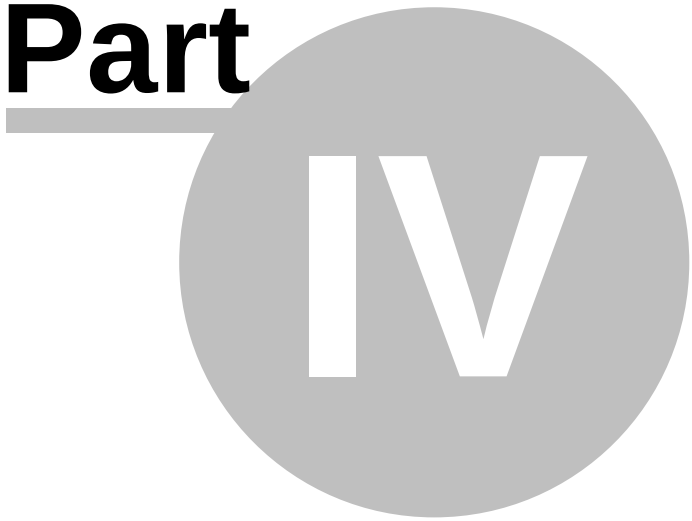
Among other possible exceptions, NG ConnectionPack declare two additional exception classes:

| | |
|---|---|
| `EHttpError` | Exception is raised when HTTP requests return error status codes, such as 4XX codes. Exceptions of this type provide extended information about the error, such as `StatusCode`, `StatusText` and `ErrorInfo` text. |
| `EHttpClient` | Exception is raised when some system error occures while executing HTTP request, such as WinAPI error, ect. In such situations HTTP request may even not be executed yet, so, no extended information from the server is available. |

`EHttpError` public interface:

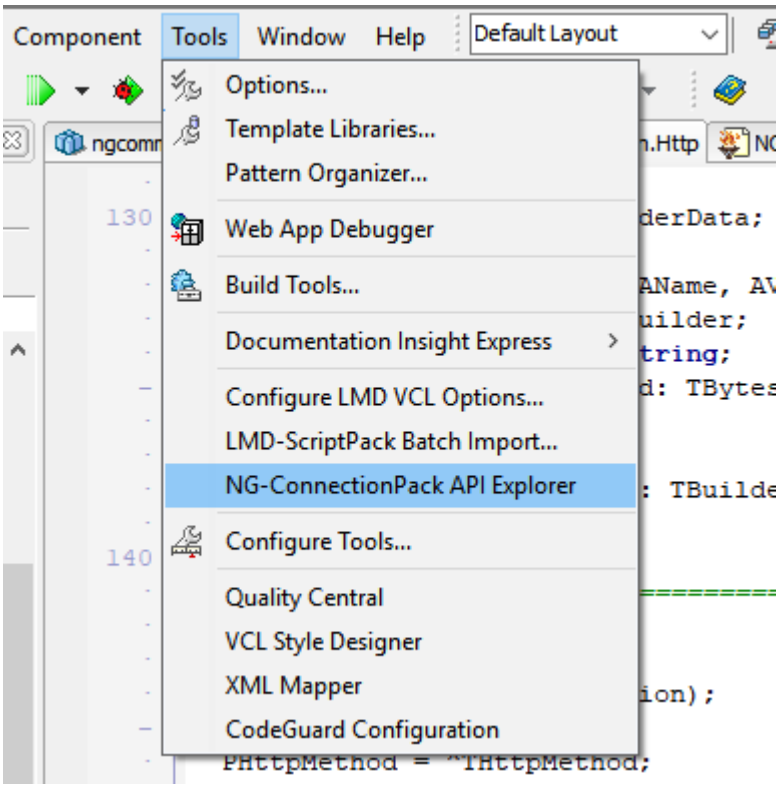
| | |
|---|---|
| `Message` | Standard Delphi property, inherited from `TException` class. Message property value contains `StatusText`; as well, if `EHttpError.ShortFormat` class propertry is set to `False`, extended `ErrorInfo` will be added to `Message` property value. |
| `ShortFormat` | Class level (static) property, which determines, whether extended `ErrorInfo` should be added to `Message` property value. |
| `StatusCode` | Property provides access to HTTP status code. |
| `StatusText` | Property provides access to short HTTP status text. |
| `ErrorInfo` | Property contains extended information, returned as a HTTP body. Usually, this information is a JSON object with error description. The format of JSON information vary in different services, so we provide the text as is, without any attempt to parse it. |

# API Explorer



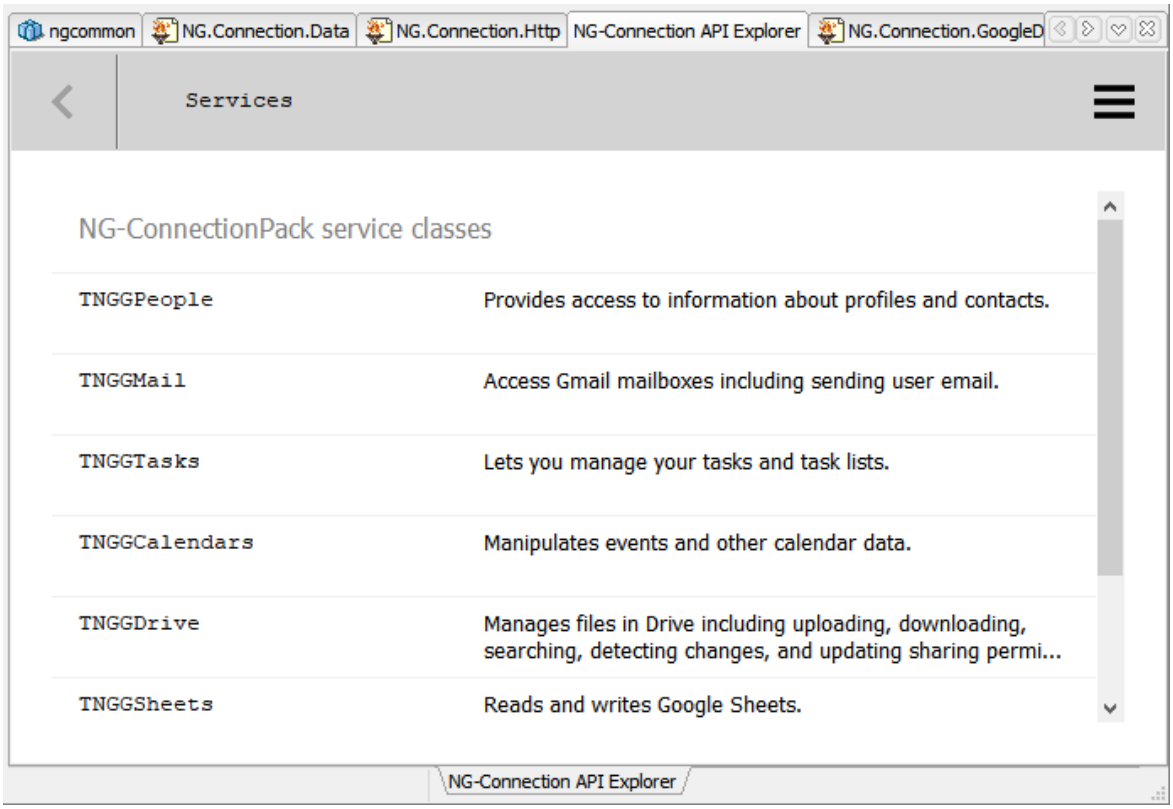
# 4 API Explorer

NG ConnectionPack provides visual tool, called API Explorer, which allows to test all supported services, by executing corresponding REST operations. API Explorer can be used directly inside Delphi IDE. API Explorer window can be shown by clicking on the corresponding menu item in Tools main Delphi menu:

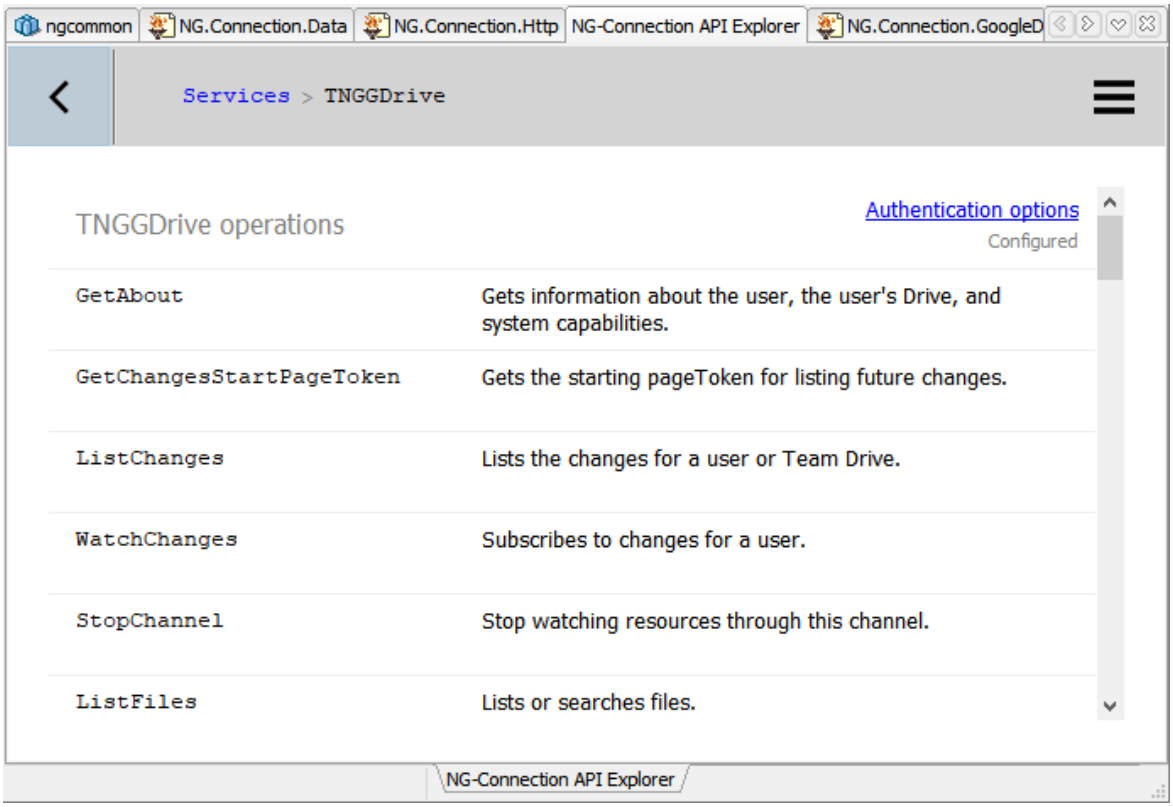



## Interface overview

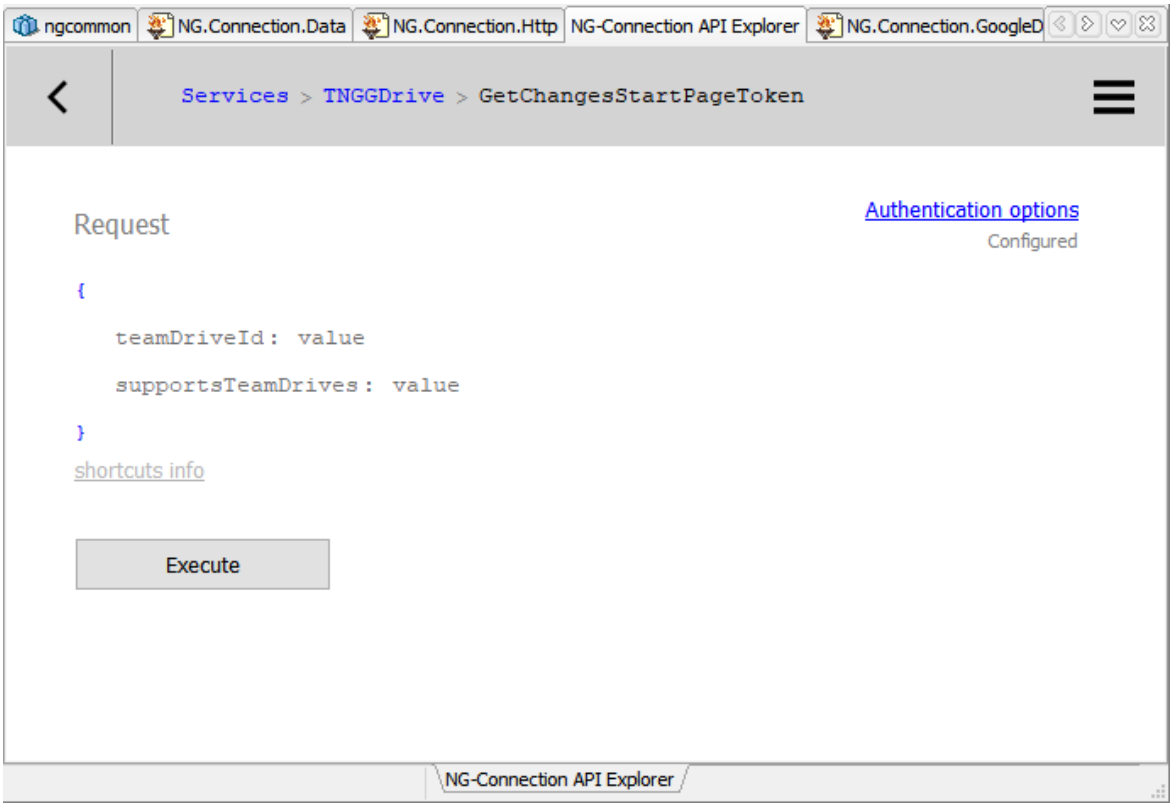
The main page of API explorer window shows supported services list:



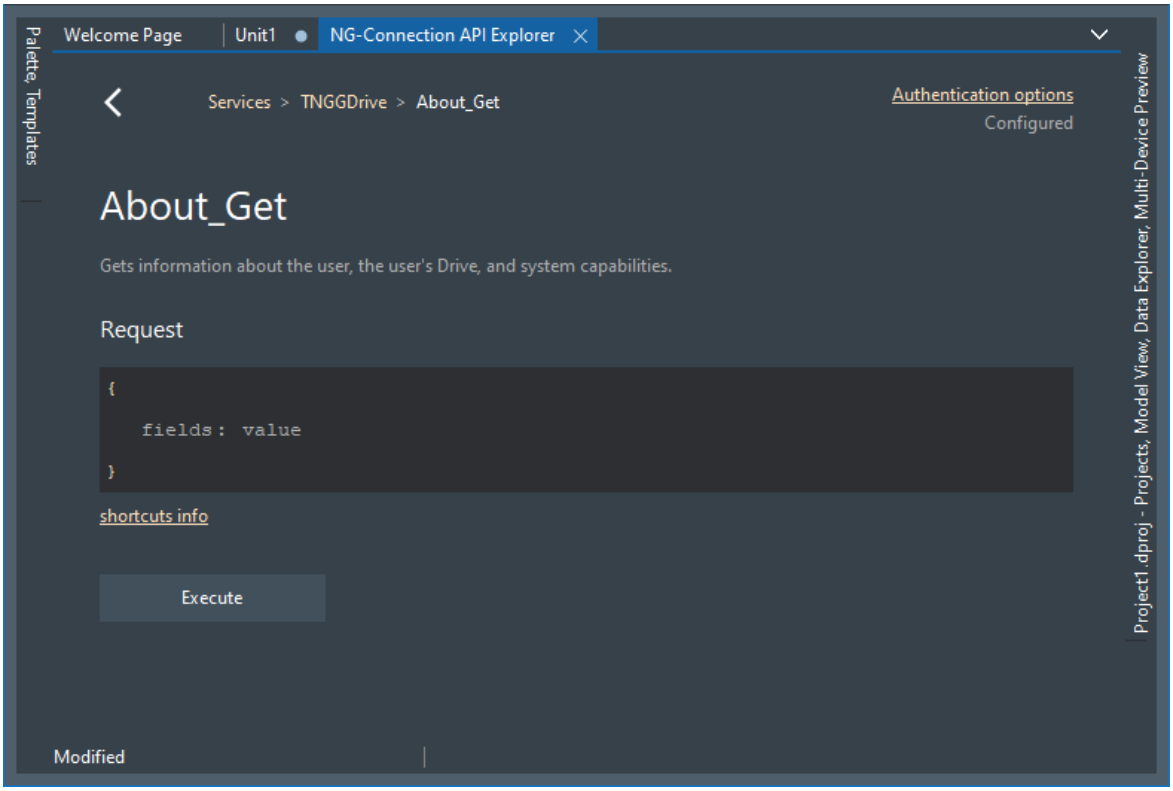

Double clicking on the service item will navigate to Service's page, where its operation list is shows:




Double clicking on operation item will navigate to operation execution page:



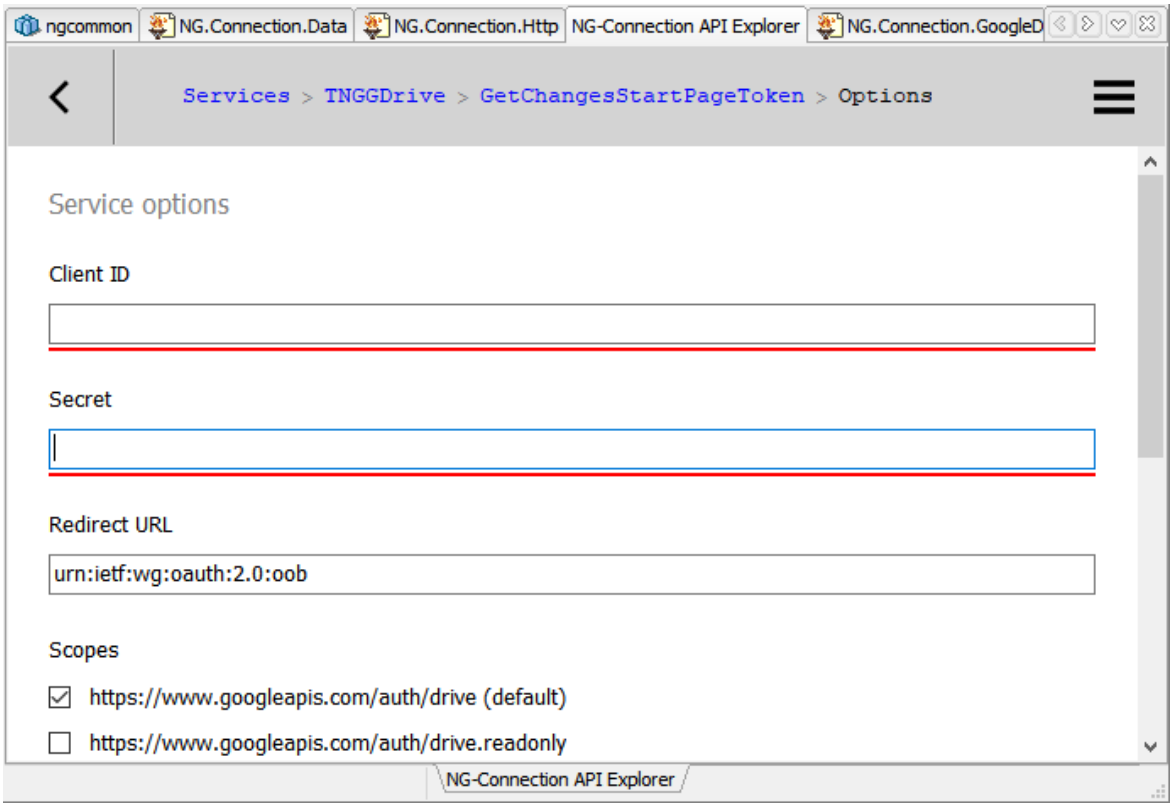

Dark theme:




On this page, operation request parameters can be set up and the operation can be executed. After operation execution, the resulting data will shows at the bottom of the page. If operation execution fails, an exception information will shows instead. Please read below about request editor usage.

# Authentication options

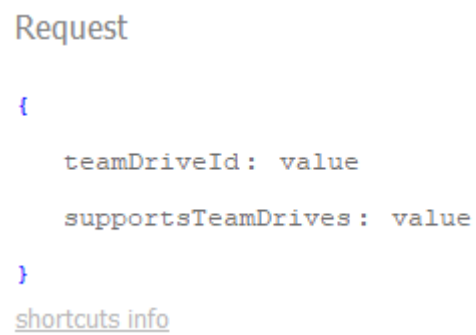
Before any of REST operation can be executed, authentication options should be specified for the corresponding service. As can be seen from above images, the link to authentication options page is shown at the service page and, as well, at operation execution page. Clicking on the link will navigate to service options page:




ClientID and Secret codes of the service should be set to appropriate values, as well, as Redirect URL. Please read more here. Scopes can be also adjusted. Option parameters are stored in Windows registry along with current access token, so there no requirement to logging in every time.

# Request Editor

Request parameters on operation execution page are shows with JSON like syntax in a special editor control:



The editor allows to overview provided request parameters and edit their values. Unspecified (null) properties are shows in gray color. To provide a value for primitive property, just click on the `value` text and type the value in shown in-place editor. To provide a value for array, map or object properties, select property line and press Insert button, or double click property line; array, map or object instance will be created as a result. To clear property value, select property line and press Ctrl+Delete key.

Since data objects usually declare a lot of properties, unspecified (null) properties is not shown by default. However, initially, unspecified (gray) properties are shown for root request object itself. To show unspecified properties, Insert Mode should be activated for this object:

- Select the line, after which unspecified properties need to be shown. This can be one of already object property line or the beginning line of the object itself. Press Insert key or double click object beginning line to show all unspecified properties after selected line.
- Select the line, before which unspecified properties need to be shown. This can be one of already object property line or the ending line of the object itself. Press Shift+Insert key or double click object ending line to show all unspecified properties before selected line.
- After editing values of requires properties press Escape key to cancel Insert Mode and hide all unspecified properties; now configured request parameters can be seen in a more compact way.

Insert Mode should be turned on for each object separately. However, Escape key turn off Insert Mode for all objects at once. Insert Mode example:

```
{
    id: 0
    permission: {
        id: 0
        allowFileDiscovery: value
        role: value
        domain: value
        type: value
        photoLink: value
        teamDrivePermissionDetails: array
        displayName: value
        emailAddress: value
        kind: value
        expirationTime: value
    }
```

Compacted mode, with hidden unspecified properties look like:

```
{
   id: 0
   permission: {
      id: 0
   }
}
```

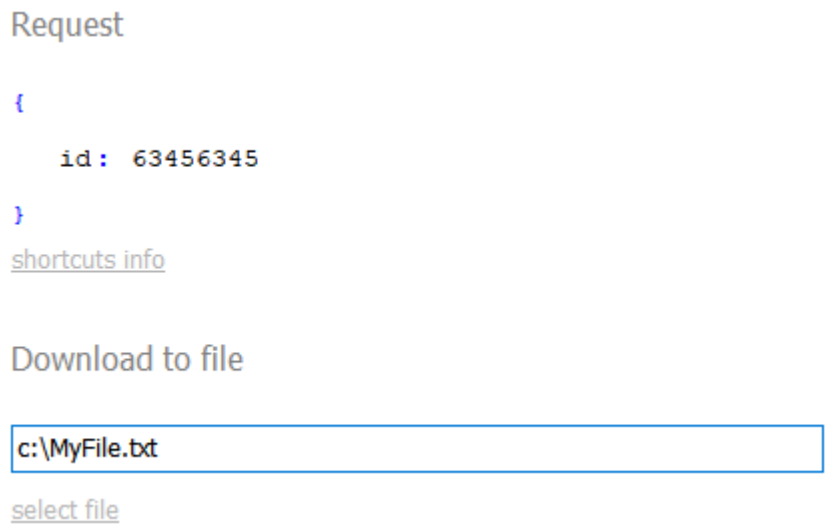
The editor supports NG ConnectionPack all data types. Numbers and string values can be entered as usual, `Boolean` values support `True` and `False` string literals, and as well can be specified as `0` or `1` numbers. Date and Date-Time values should be specified according to ISO 8601 format, like `2018-01-01T12:00:00Z`. Upload and Download values are handled in a special way, please read below.

# Upload and Download

If an operation require to upload stream data, its request object declares property of `NGUpload` type. For example, `data` property in `TNGGDrive.CreateFileUpload` operation request. In API explorer this property is shows as a single line, where uploading file path should be specified. As well, double clicking on the line, will show standard Open file dialog:

```
{
   data: c:\MyFile.txt
   uploadType: value
   keepRevisionForever: value
   supportsTeamDrives: value
   useContentAsIndexableText: value
   ocrLanguage: value
   ignoreDefaultVisibility: value
   metadata: object
}
```

If an operation requires to download stream data, its request's `Execute` method takes `NGDownload` parameter. In API Explorer, special text editor will be shown after request parameter editor to specify file path for downloading into:

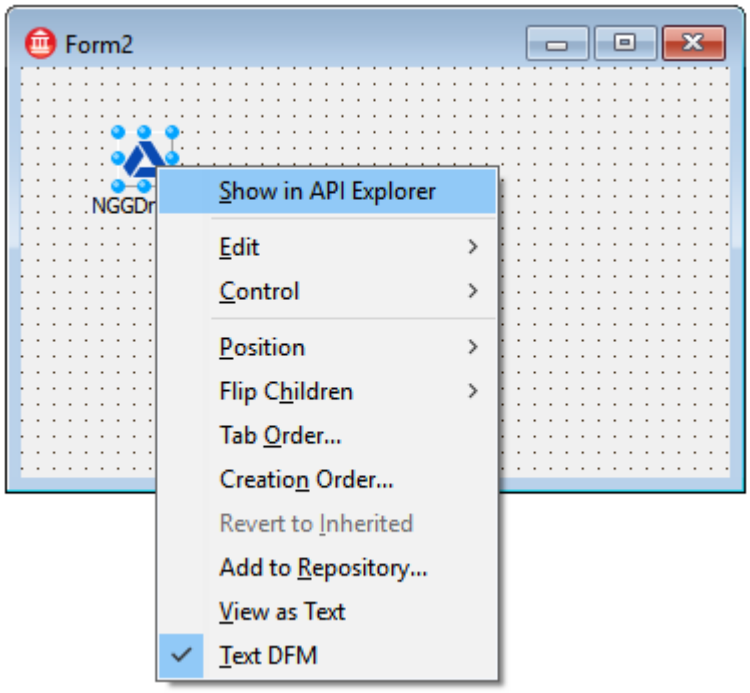



Clicking on select file label will open standard Save file dialog.

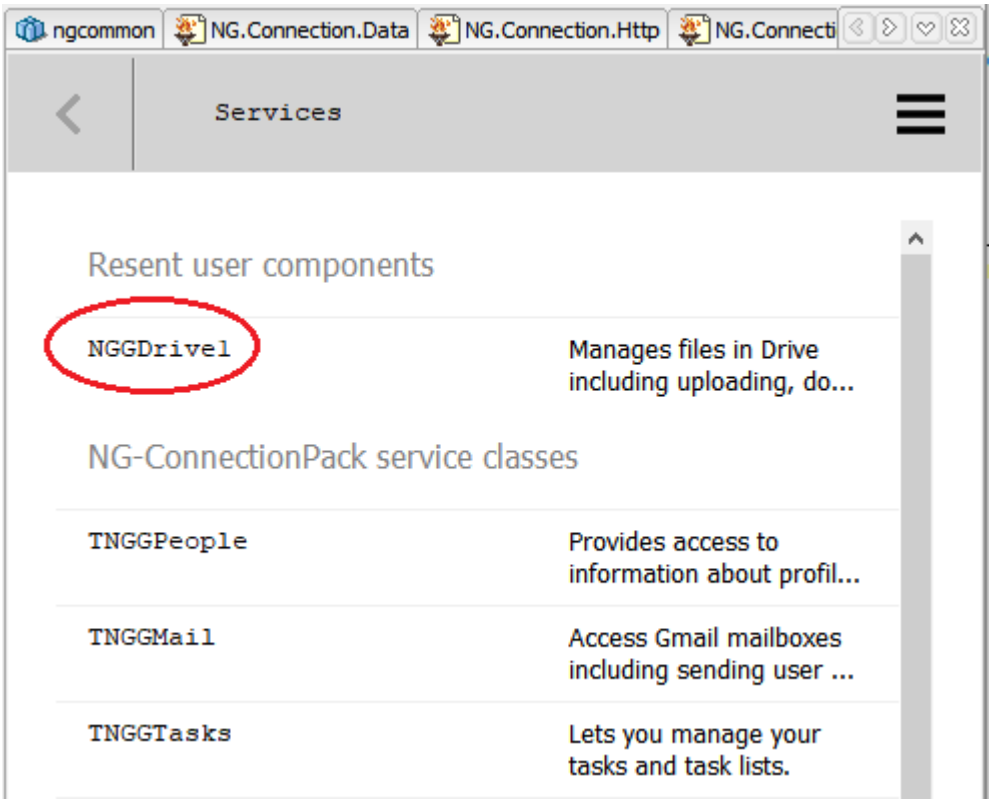
Currently, uploads and downloads can be performed only from/to files; there no other built-in content editor. However, downloaded file can be easily opened in Windows File Explorer, using the corresponding button, which will appear after successful operation execution.

# Exploring User Components

API Explorer also allows to explore user's components, which are components, placed on the form or data-module at design-time. This feature allows developers to test REST API with already defined in Object Inspector application settings: `ClientId`, `Secret` and `Scopes`. This can be done by right-clicking the component icon and selecting Show In API Explorer menu item:




After that, the component will be added to API Explorer services list, so its operations can be executed:




API Explorer does not remember users components list, so, the component reside in the list only until its parent form is not closed in IDE or the component is removed from the form. Also, in current version, API Explorer does not save access token for user's components.

# Your First Application (Google Drive)

# Part V

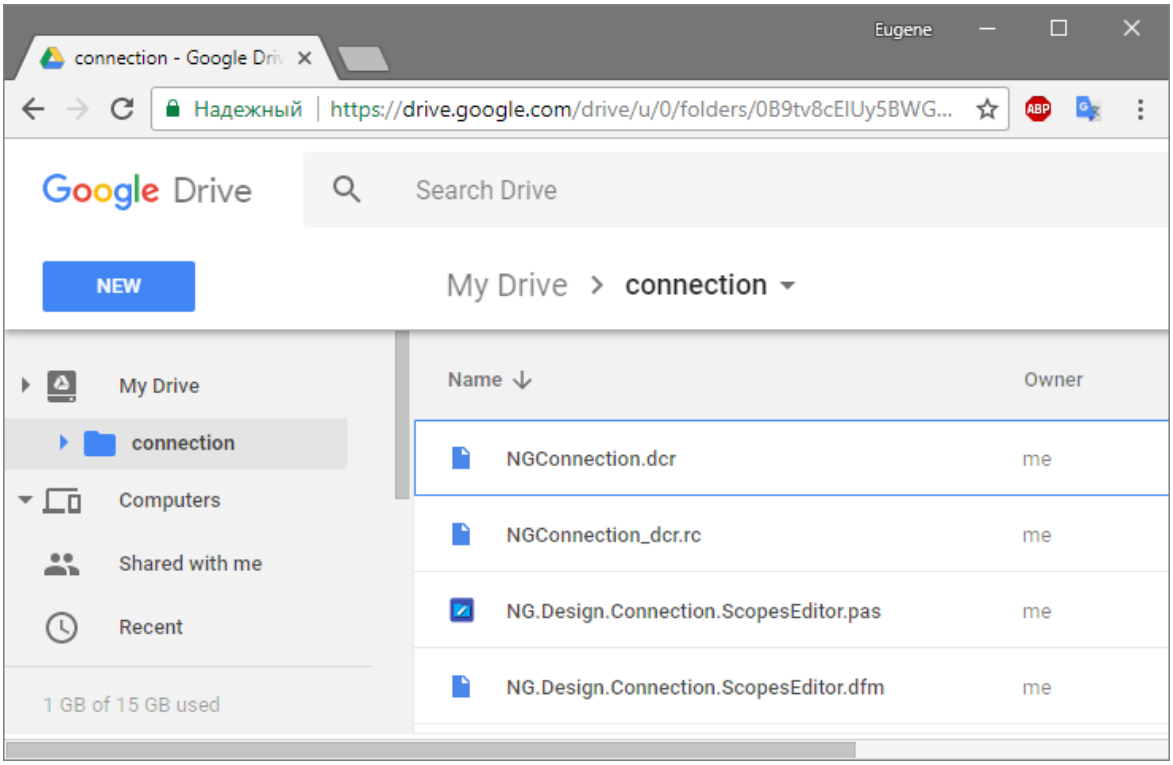
# 5 Your First Application (Google Drive)

This topic describes creation of a very simple application, which will show Google Drive user's file tree in TMemo control. The topic is written for new NG ConnectionPack users, and the application is **simplified as much as possible**. For more advanced, real-world apps, please refer to NG ConnectionPack demos.

## 1. General Google Drive Access

First of all, please make sure that you have Google account, and can access Google Drive via its primary web interface:

https://drive.google.com



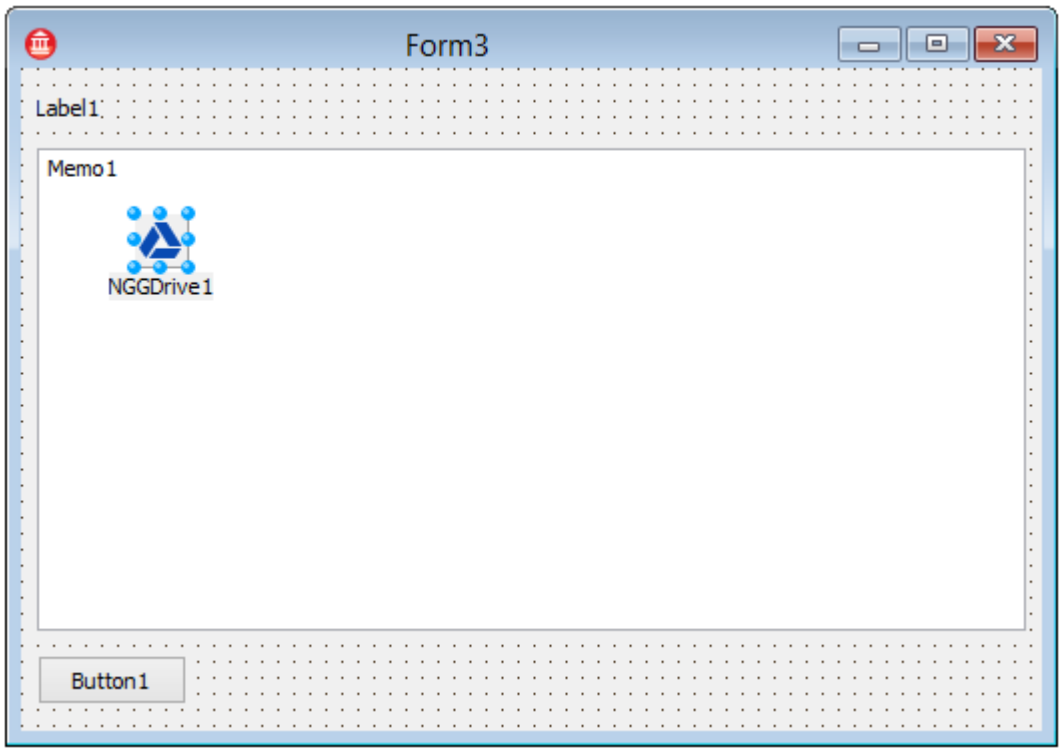

For testing purposes please make sure, that some files are available in your Google Drive. We uploaded NG ConnectionPack sources, for example.

## 2. Registering your Application

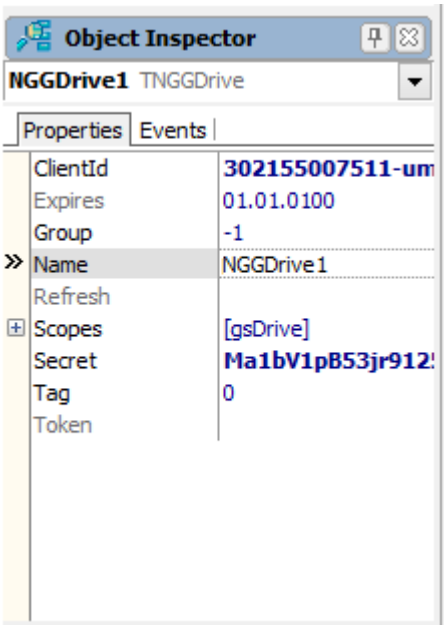
Please follow the procedure, described in Google Registration topic. Don't forget to enable Google Drive API:


## 3. Creating VCL Forms Application

The next step will be to create new VCL application in Delphi. Run Delphi IDE, create new VCL application, as usual, and place `TLabel`, `TMemo`, `TButton` and `TNGGDrive` components, like shown below:




We will use the memo to show downloaded files tree, and the label to show downloading progress. As a result of application registration, performed in the previous step, you received ClientID and Secret string values. Please copy-paste them to the corresponding properties of NGGDrive1 component using Delphi's Object Inspector:



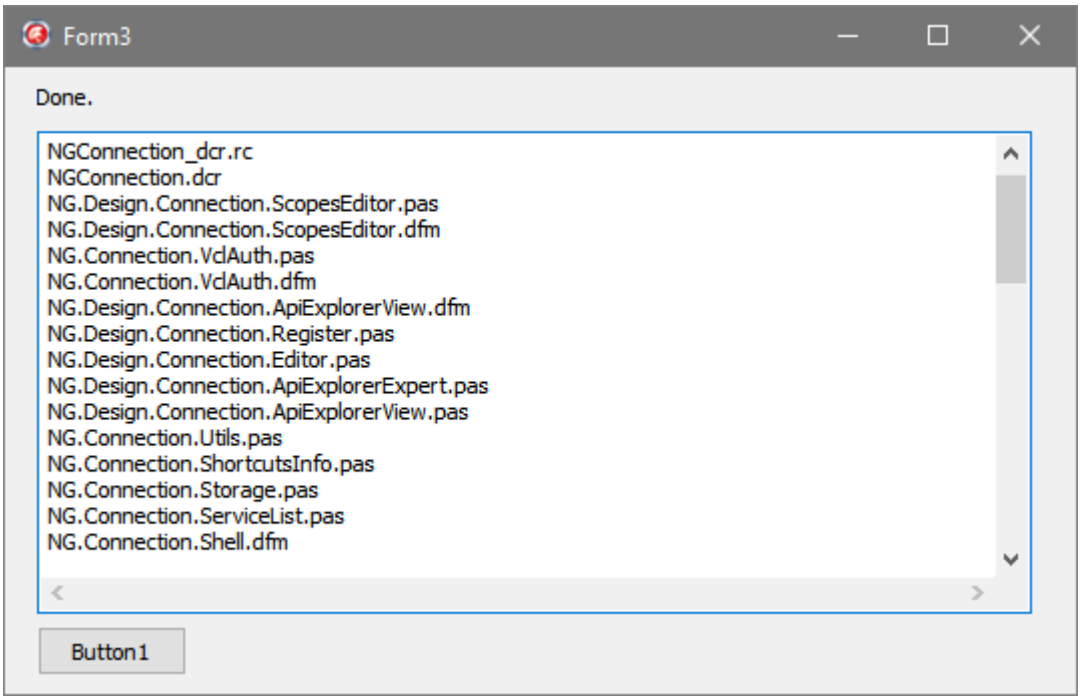

Double click the button on the form to create its `OnClick` event handler and write the following code in it:

```
procedure TForm3.Button1Click(Sender: TObject);
```

```
var
  lst: TNGFileList;
  fl:  TNGFile;
  i:   Integer;
begin
  FFileCount := 0;

  Label1.Caption := 'Working...';
  Label1.Refresh;

  lst := NGGDrive1.Files_List
                  .Q('trashed=false')
                  .Execute;

  for i := 0 to lst.Files.High do
  begin
    fl := lst.Files[i];
    Memo1.Lines.Add(fl.Name);
  end;

  Label1.Caption := 'Done.';
end;
```

# 4. Running the Application

Thats all! You can run the application. Since the authentication state is not saved/loaded for simplicity, the application will ask to authenticate the user at every run. Use your own Google Account credentials to grant the application rights to your Google Drive data. The application, then will retrieve full file list of your Google Drive files:




Congratulations! You've created your first application, which interacts with web based REST service via Internet, using LMD NG ConnectionPack components.

# Application Registration

# 6 Application Registration

The first thing required to connect REST services is to register your application. This is usually done by creating an account in the developer's zone of the corresponding service provider web site, and then filling the application description data, such as the Name, Logo, and some other parameters. Some services force to specify more important parameters, such as application type, so called redirect page, and a set of APIs used.

As a result you'll get application's `ClientID` and `Secret` codes. Provided `ClientID` and `Secret` values should be copied into corresponding properties of the NG ConnectionPack components.

Its important to note here that the account used for registering the application, and end-user accounts, which will be used later for authentication - are different. So, if you personally use the corresponding services, this does not mean that your personal data will be accessible to your application users.

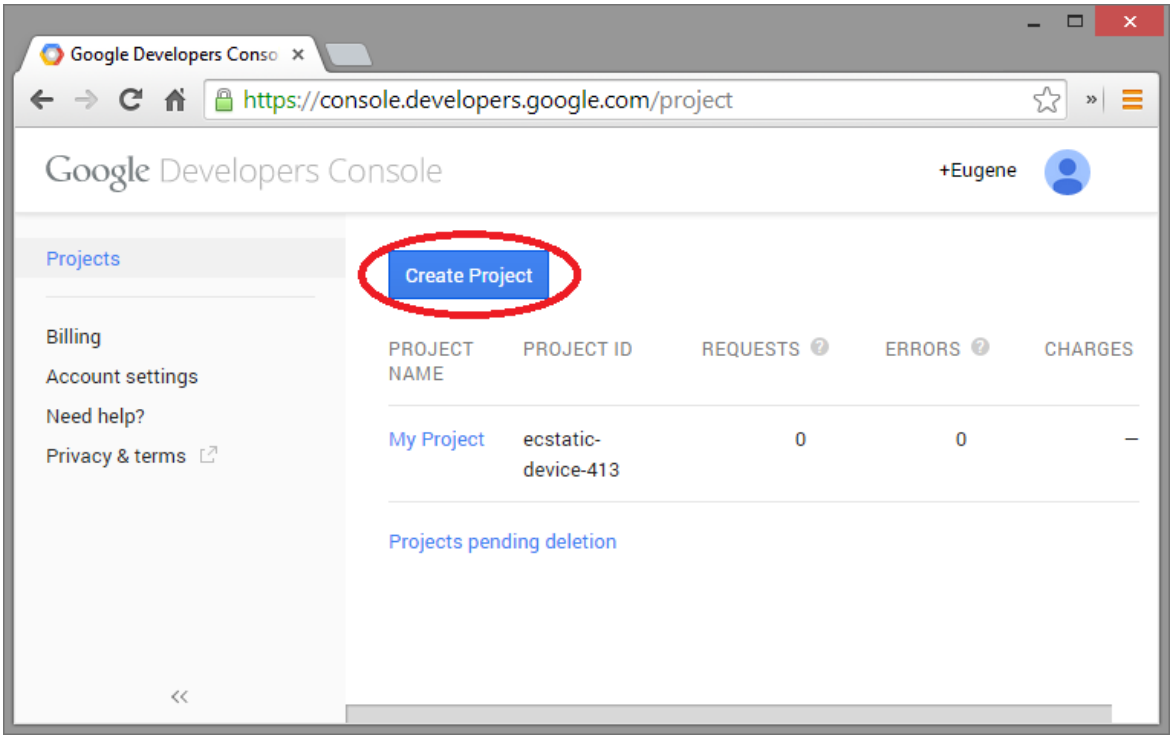
Provided ClientID and secret values should be considered as a private data of application. So, its not a good idea to show it to users. Also, note, that usually, service providers allow to register more than one applications from a single account.

Following is a step-by-step instructions of how to register application in all supported services:

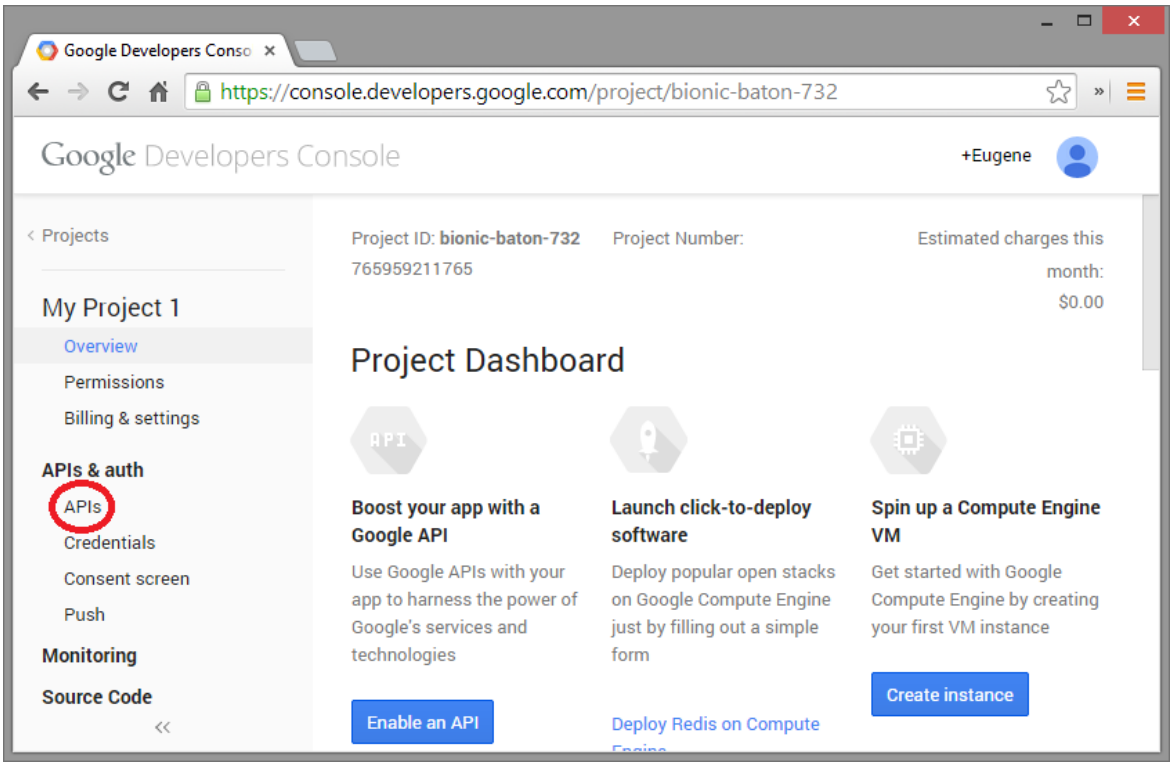
- Google Registration (for Google Drive, Google Calendars and Google Tasks)
- Dropbox Registration
- OneDrive Registration
- Box.NET Registration

## 6.1 Google Registration

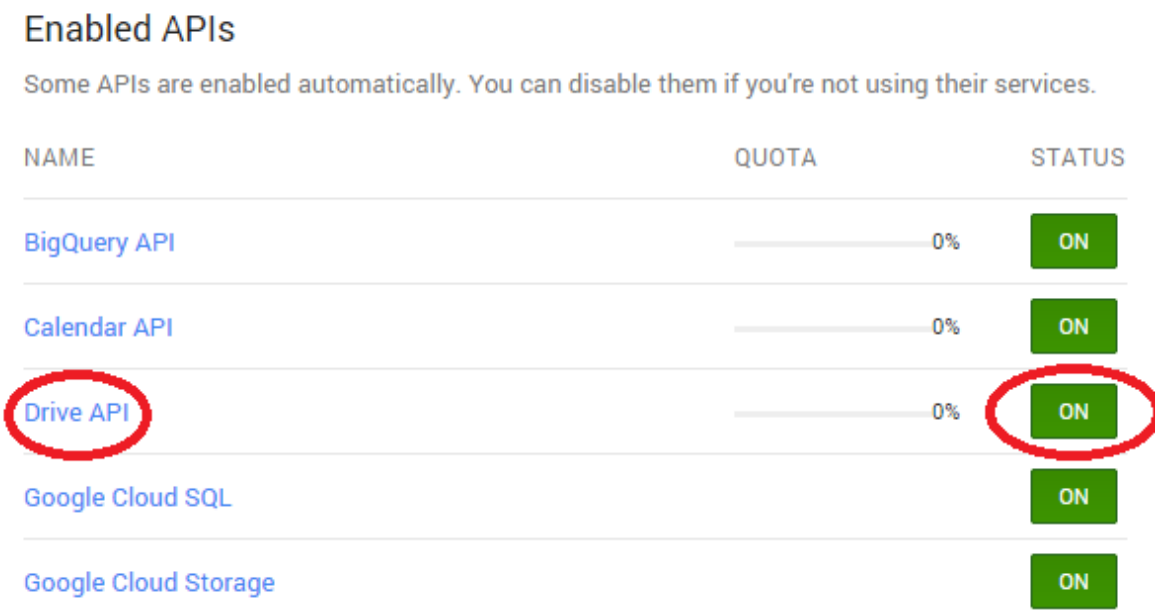
To register your application for using Google services, such as Google Drive, Google Calendars and Google Tasks, you need to have a general Google account. Please go to console.developers.google.com, which is currently looks like this:



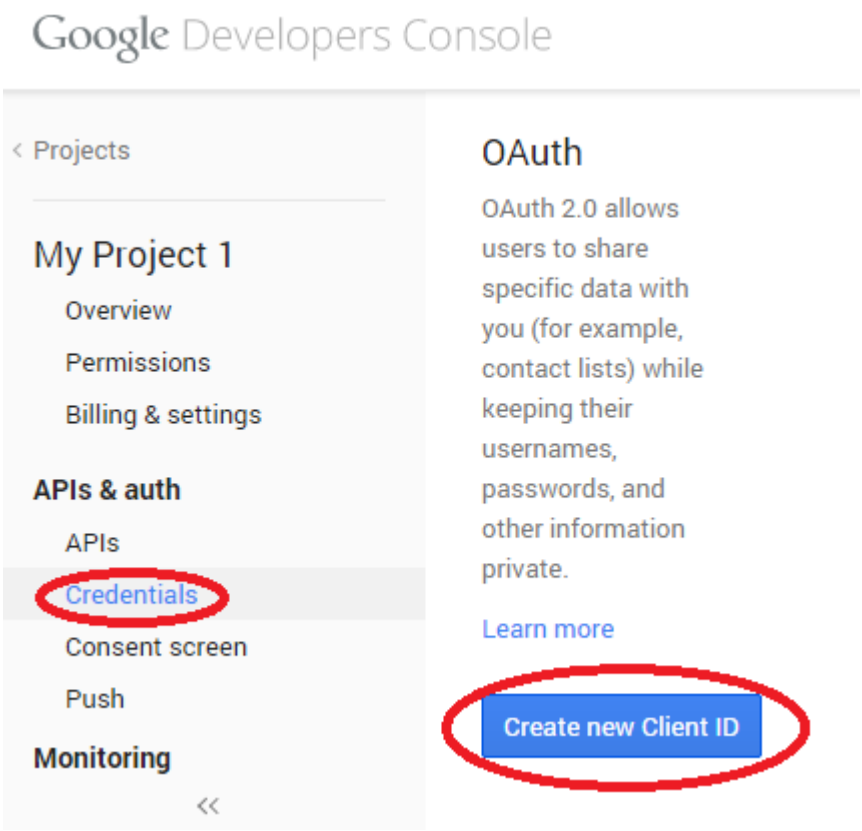
Then, click on Create Project button. The dialog will ask you to enter project's name, and after the project will be created, you will see the following project's page:



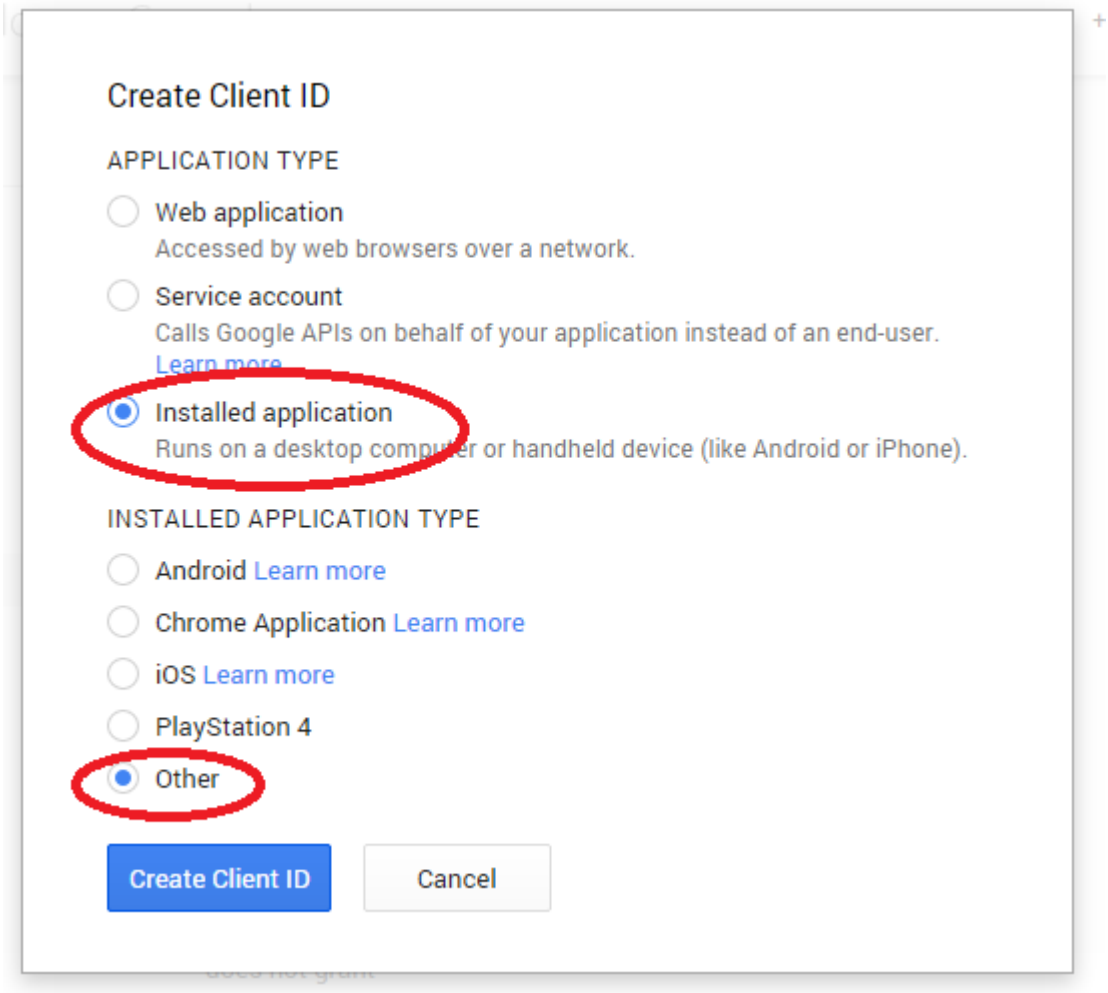

There two very important things to do here. First, go to APIs section and make sure that required APIs are enabled. You need to enable "Drive API" to use Google Drive, "Calendar API" to use Google Calendars, and "Tasks API" to use Google Tasks, for example:



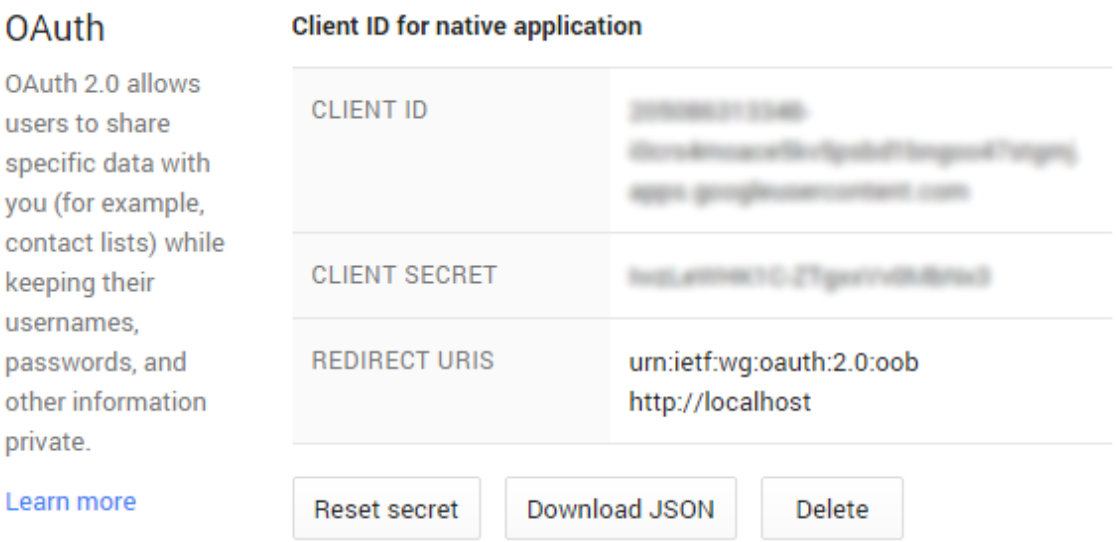
Second, go to Credentials section and click "Create new Client ID" button to setup authentication parameters:




In the appearing dialog select "**Installed application**" and "**Other**", as shown below, and click "Create Client ID":




As a result you will get a Client ID and Secret values:

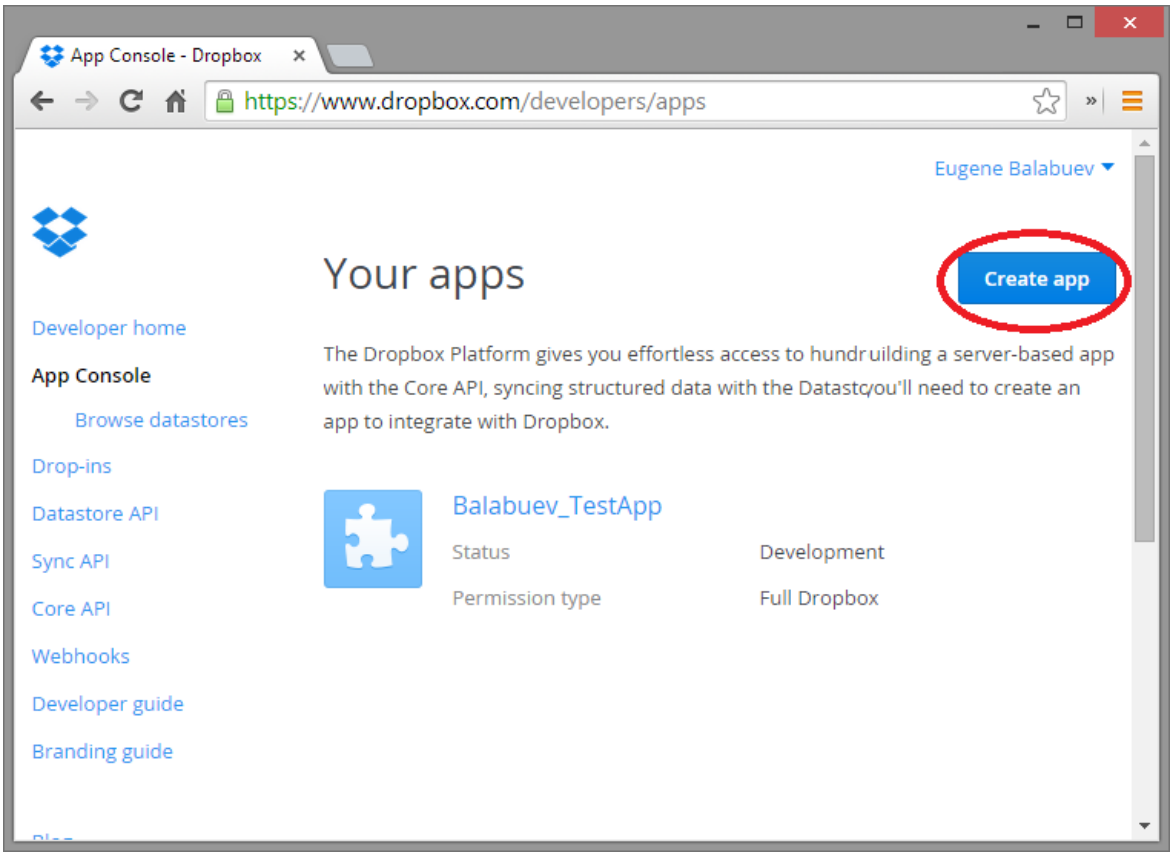


Please copy these values to the corresponding NG ConnectionPack components properties. As you can see from the description above there no need to create distinct projects for each of supported Google services, the same project can be used. You just need to enable corresponding APIs.

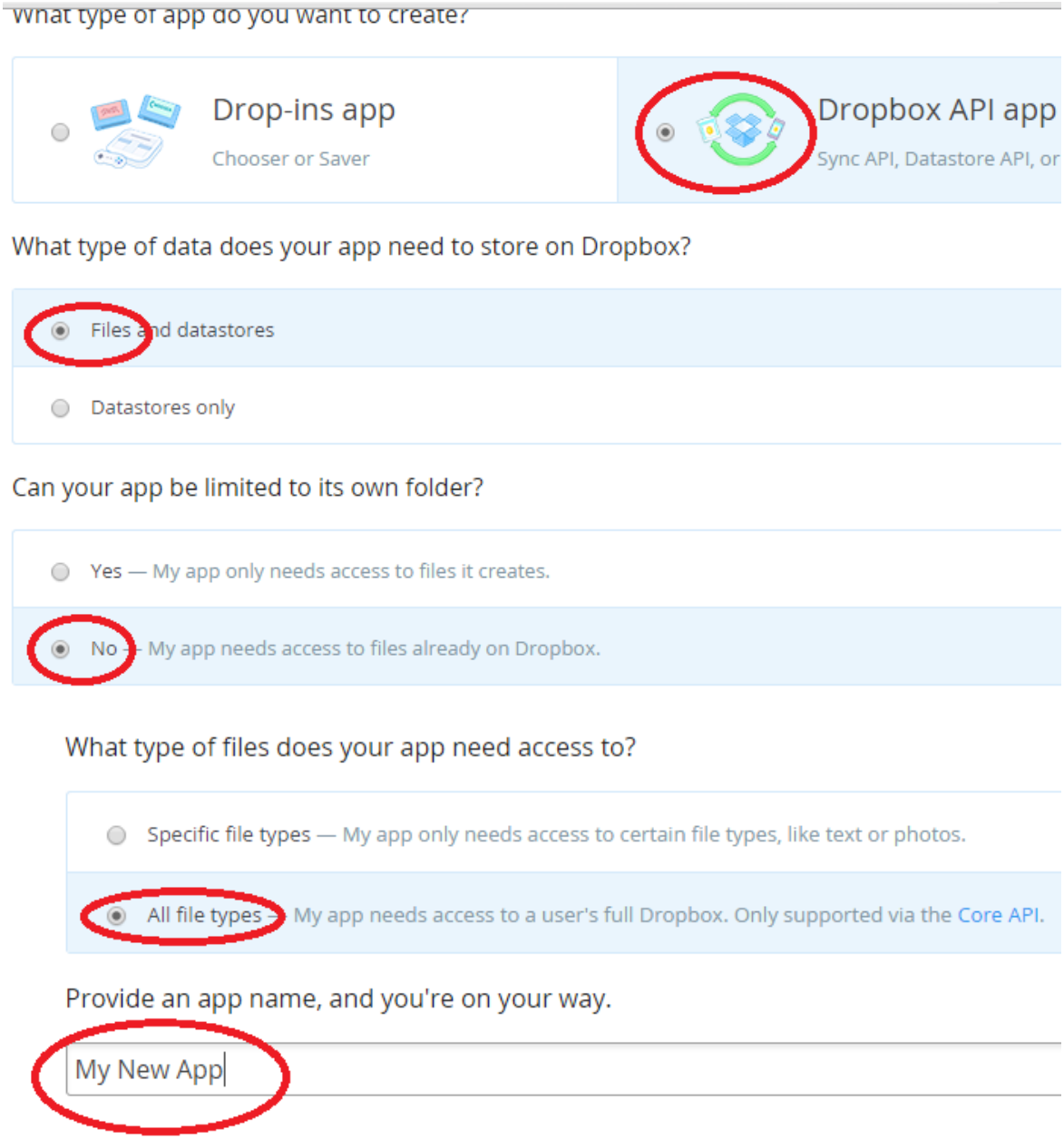
Note: Some time ago Google prohibits authentication using contained web view (`TWebBrowser`), because of security considerations, so, only system browser based authentication will work. As a result, **only localhost based redirect URL** can be specified in application's settings, like described in Authentication section.

## 6.2 Dropbox Registration

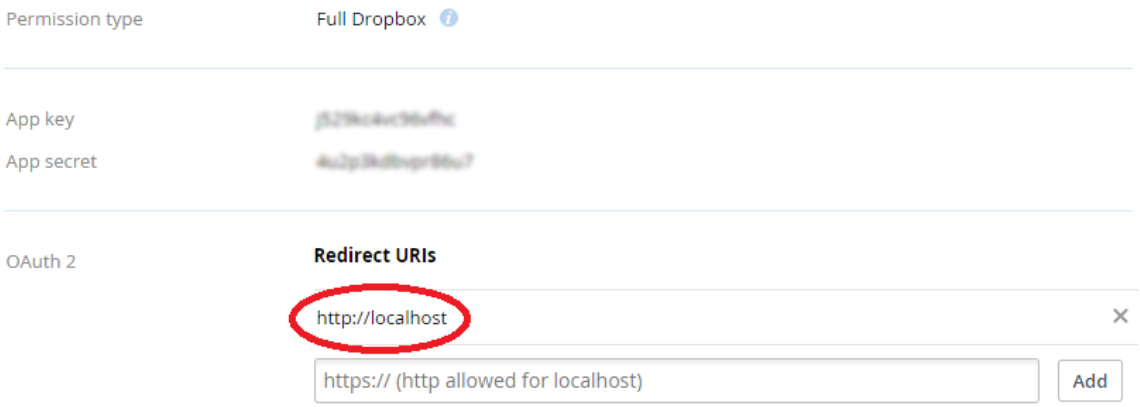
To register your application for using Dropbox service you need to go to dropbox apps console following this link: www.dropbox.com/developers/apps. Currently console looks like this:




Click "Create app" button and set the options as shown in the following pictures.



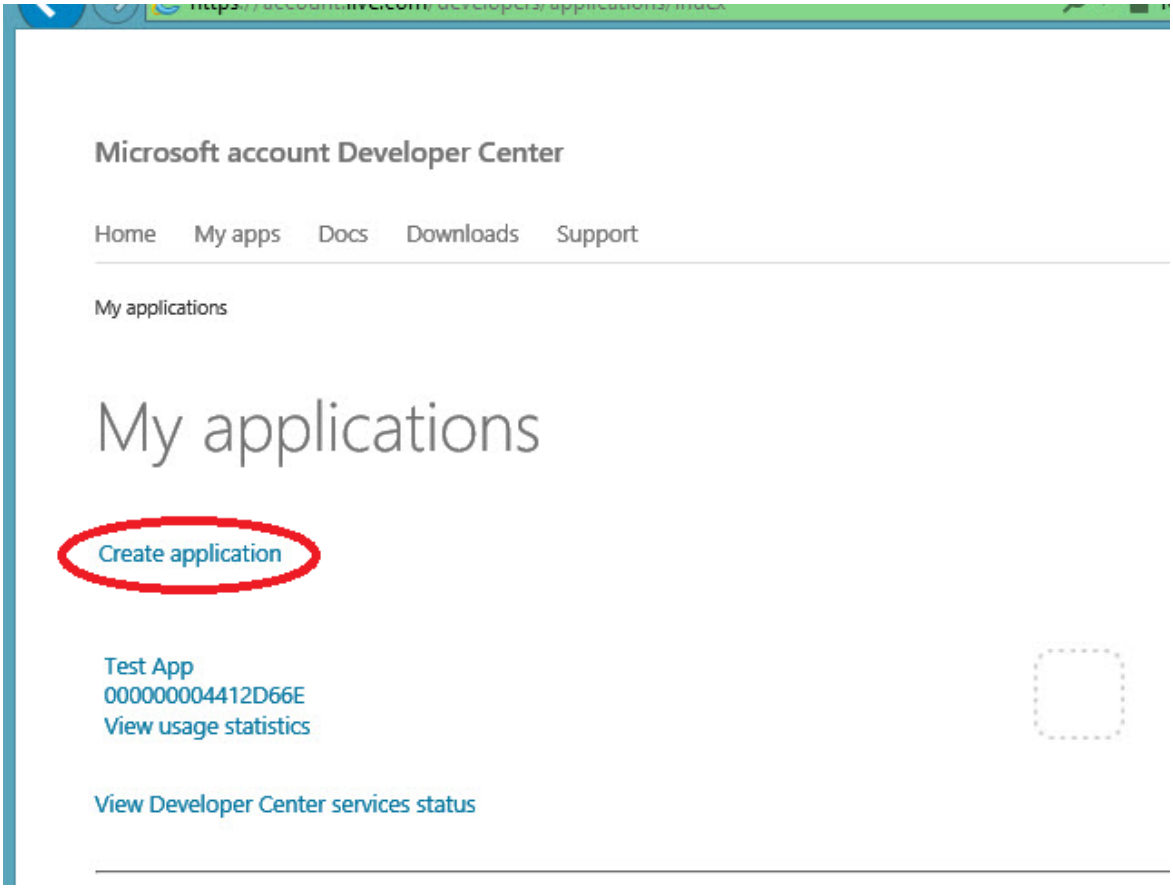

Click "Create app" button to submit. After the application has been created you will be redirected to the application details page. Its recommended to use "**http://localhost**" as a Redirect URI with system browser based authentication UI (see here).



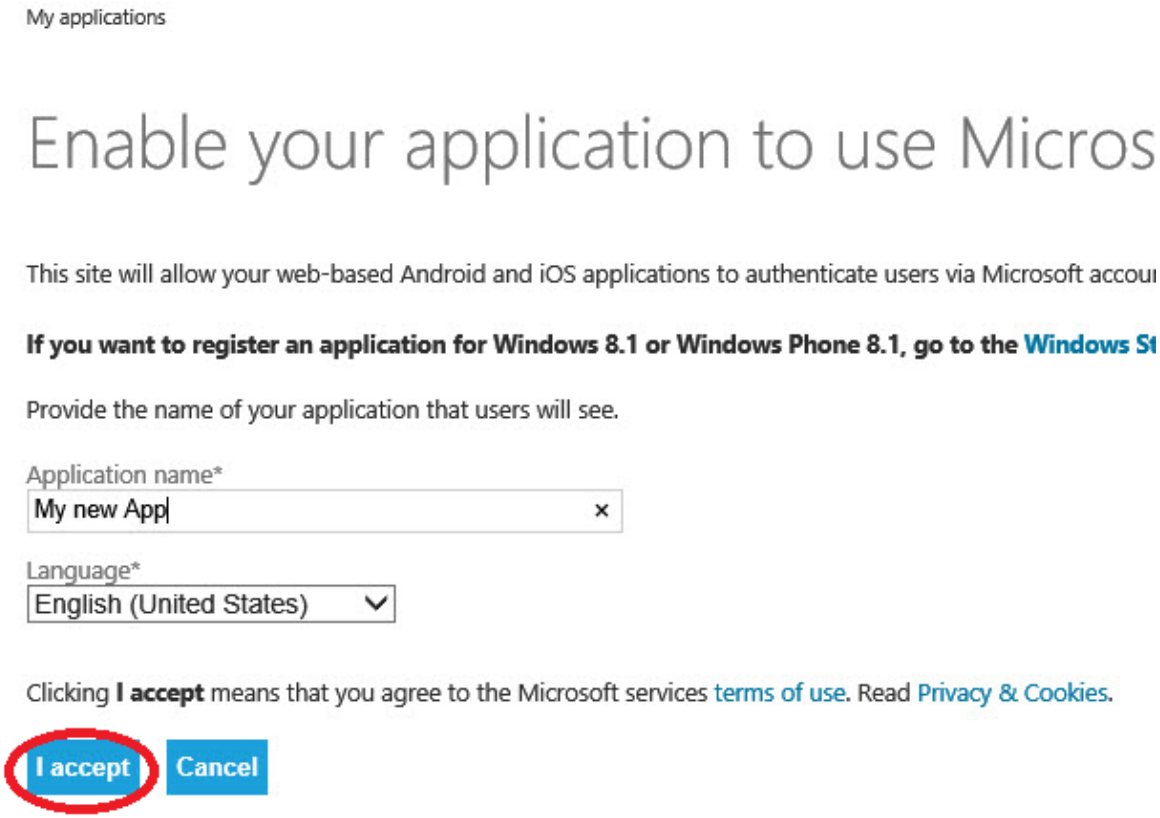

Also, copy "App key" and "App secret" values to `ClientId` and `Secret` properties of the `TNGDropBox` component. Please note, that NG ConnectionPack currently does not support Dropbox datastores. Also, the component does not support so called "sandbox", which is a private folder (in the user account space) where application can store its own data; the component only intended to be used for managing user's files.

## 6.3 Microsoft Registration

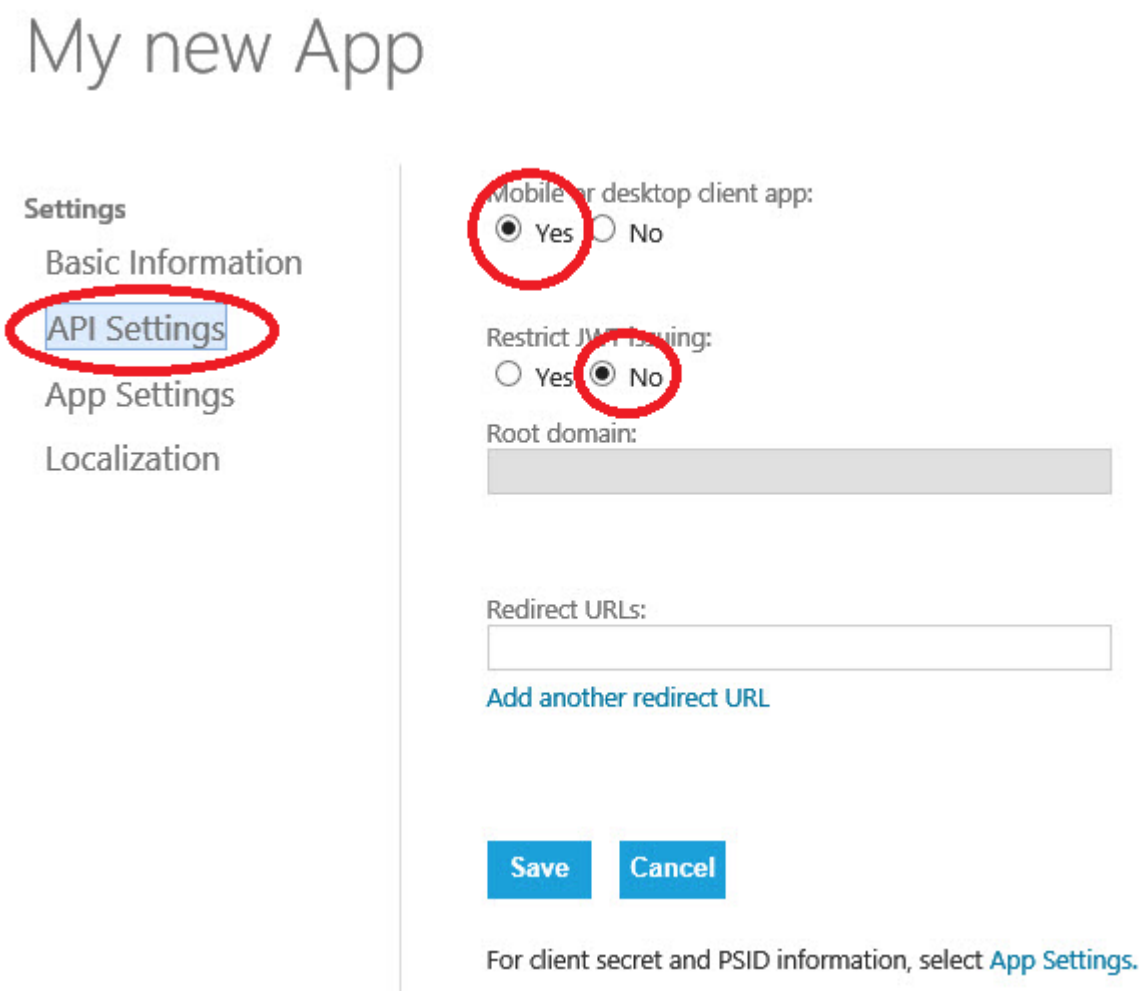
To register your application for using One Drive service you need to go to Microsoft Developer Center. To do this, please follow the link: https://account.live.com/developers/applications/index. Currently the page looks like this:



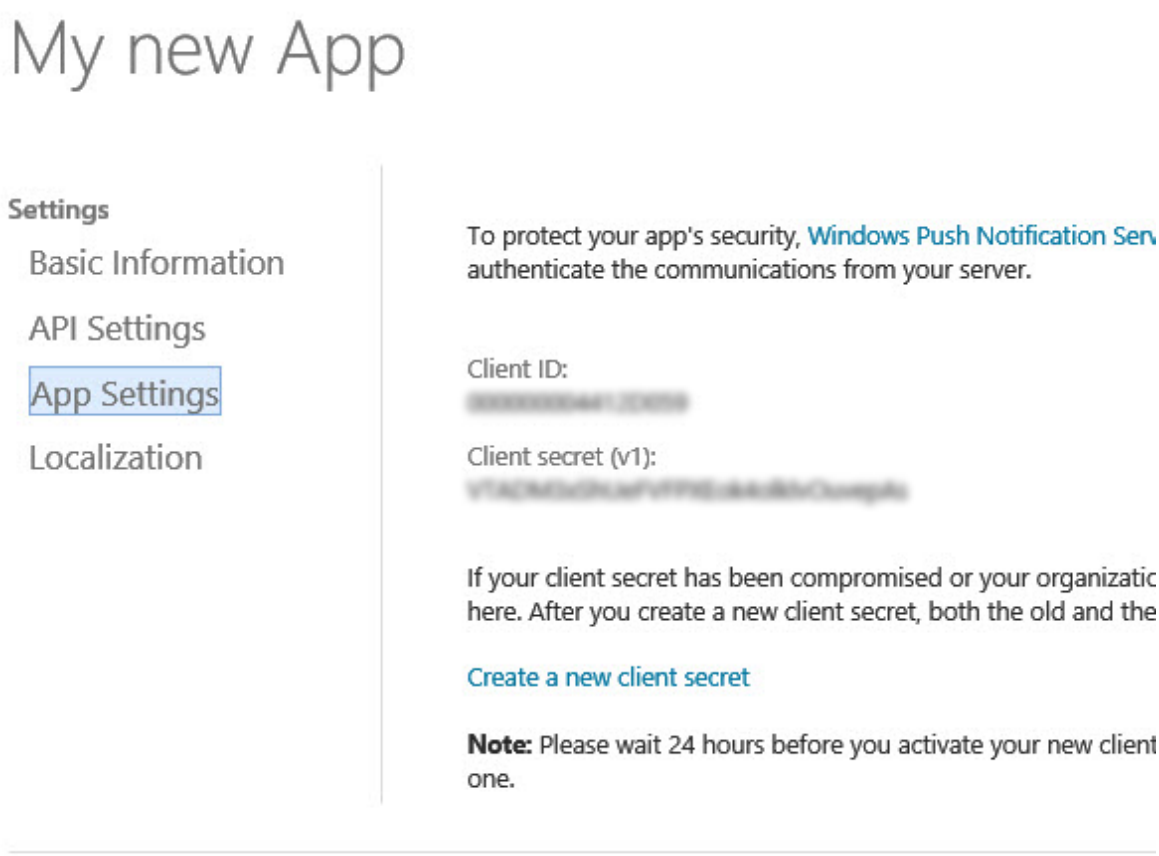
Click "Create application" link. In the following dialog, provide a Name for your application and submit byt clicking "I accept" button.




Then go to application's "API Settings" and set highlighted parameters as shown below, and click "Save" button:



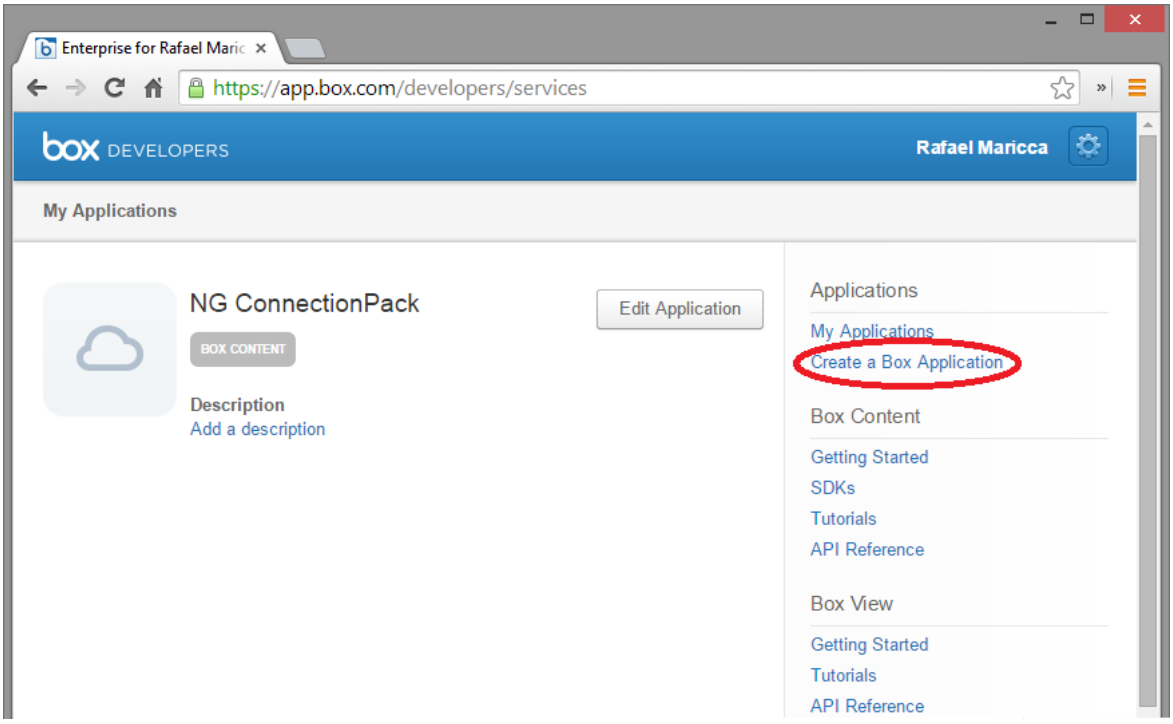

Please note, that you have to set "Mobile or desktop client app" parameter to "Yes". Then go to "App settings" section, where you'll see application's `ClientID` and `Secret` values:



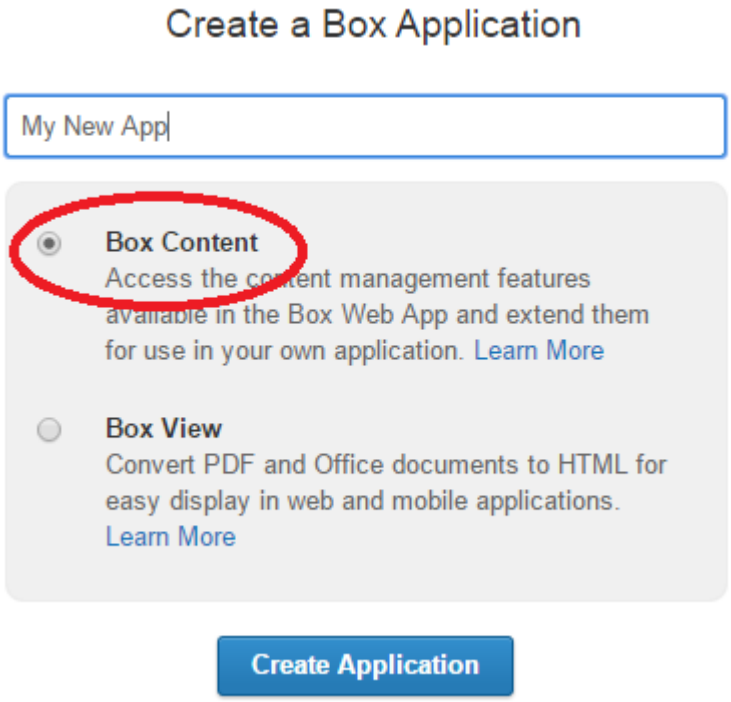

Copy these values into `ClientId` and `Secret` properties of the `TNGOneDrive` component.

## 6.4 Box.NET Registration

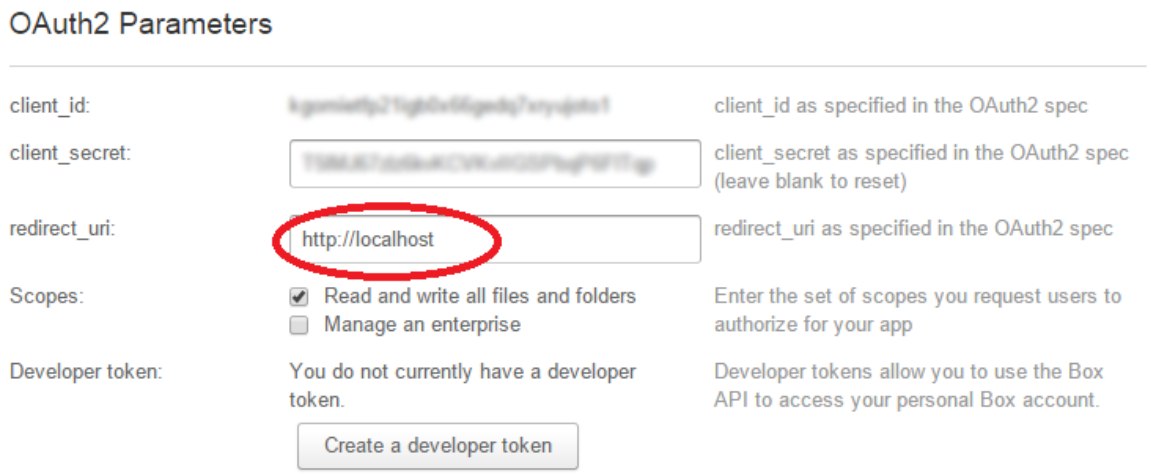
To register your application for using Box.NET service you need to go to box's apps console. To do this, please follow the link: https://developers.box.com and click "My Apps" button at the top-right corner. Currently console looks like this:




Click "Create a Box Application" link, as shown. In the appearing dialog provide application name and select "Box Content" option.




Click "Create Application" button to submit. Its recommended to use "**http://localhost**" as a Redirect URI with system browser based authentication UI (see here).




Also, copy "client_id" and "client_secret" values to `ClientId` and `Secret` properties of the `TNGBoxNet` component.

# Authentication

# Part VII

# 7 Authentication

Before the user will be able to authenticate himself to the service, you, as a developer should register your application following the procedure, described in Application Registration section. After that you should receive two string values: `ClientID` and `Secret`, which should be copied in the corresponding properties of used NG ConnectionPack components.

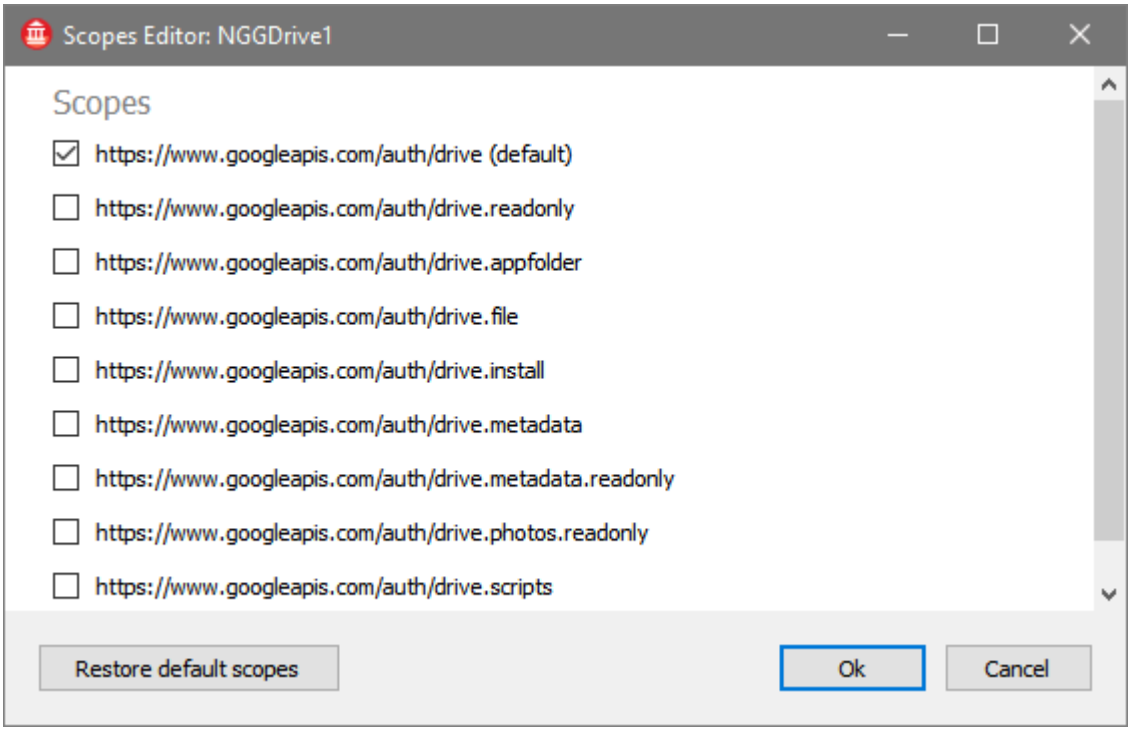
Its important to understand that the application registration and end-user authentication are two completely different things. While registering the application is done via your own developer's account during application developing, end-user authentication happens at run-time using user's account. And so, user's authentication, makes user's account resources, such as user's cloud files, accessible from your application.

## Scopes

In addition to `ClientID` and `Secret` settings, access scopes can be specified using `Scopes` property. Access scopes are strings that denote special permissions, which allows your application to perform some operations with user's data. For example, Google Drive service has full access scope, which allow to read and write files, and, as well, read-only scope, which allow only read files. Usually scopes are quite long strings in a form of web links. For example, Google Drive supports the following scopes:

- https://www.googleapis.com/auth/drive
- https://www.googleapis.com/auth/drive.readonly
- https://www.googleapis.com/auth/drive.appfolder
- https://www.googleapis.com/auth/drive.file
- https://www.googleapis.com/auth/drive.install
- https://www.googleapis.com/auth/drive.metadata
- https://www.googleapis.com/auth/drive.metadata.readonly
- https://www.googleapis.com/auth/drive.photos.readonly
- https://www.googleapis.com/auth/drive.scripts

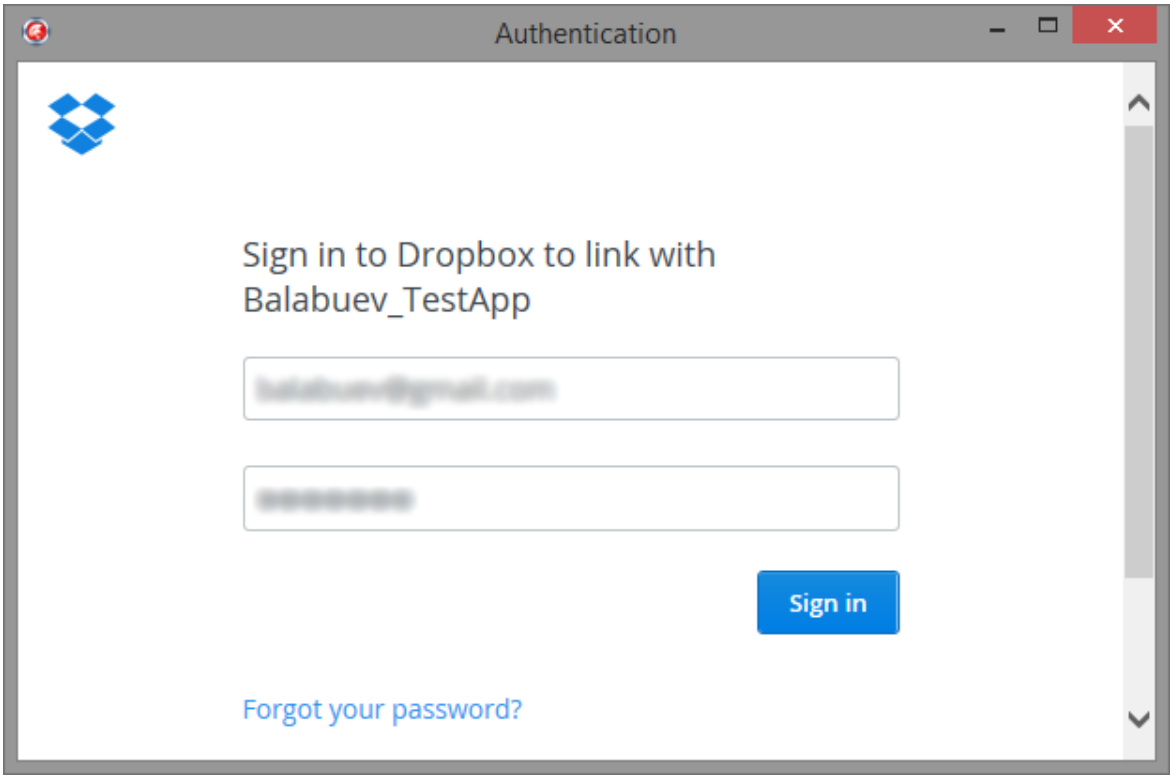
Several access scopes can be combined to declare functionality, required by application. In NG ConnectionPack all supported scopes are declared additionally as string constants inside corresponding service classes. For example, `TNGGDrive.DRIVE` or `TNGGDrive.READONLY` scopes. As well, NG ConnectionPack provides special design-time editor dialog for `Scopes` property, which simplifies scopes setup:

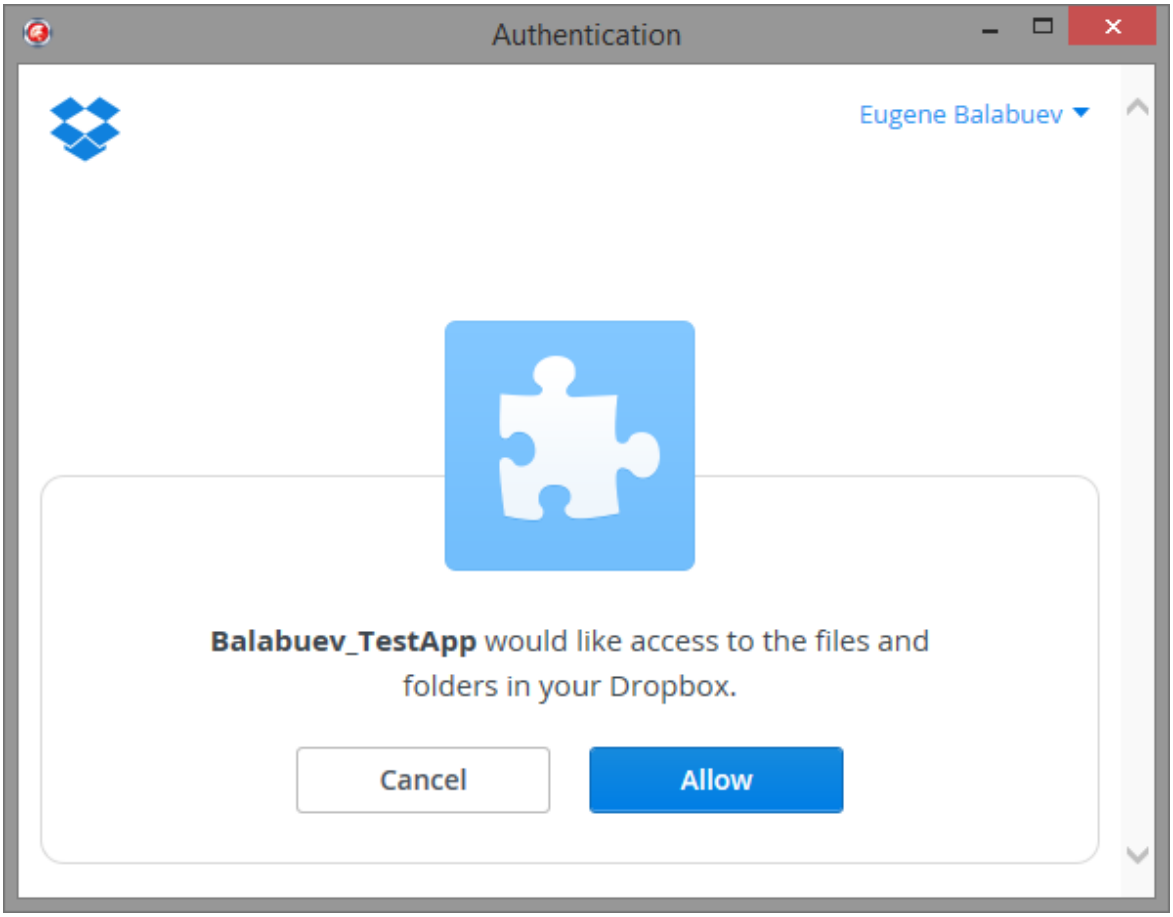



Please read corresponding service's documentation to learn about supported scopes.

# Authentication User Interface

Authentication is performed using special built-in modal dialog, which is shown by the NG ConnectionPack code on demand. The corresponding REST service login page is shown to the user along with your registered application name, logo and required access rights. After successful user login, the service provides so called `access_token` (and some other info), which is used subsequently to sign Http requests.




`Access_token` is used as a session identifier, and usually has a limited working time. This time can be different in different REST services, usually from 5 minutes to one hour (some services provides long running `access_token` values, e.g. Dropbox). Working time period is also returned as a result of

authentication process, and after this period has been elapsed, the token should be reacquired. NG ConnectionPack do this automatically, without displaying the dialog to the user. To acquire new `access_token` NG ConnectionPack performs special Htpp request providing so called `refresh_token`, which is also received as part of authentication result.

The process of refreshing token can be repeated almost infinite amount of times. So, the user will be able to use the application for a long time without re-logging in via authentication dialog. Because of this, the authentication state **should** be stored/loaded between application runs. NG ConnectionPack provides `SaveState` and `LoadState` methods, and its strongly recommended to use there methods.

In addition to preventing the user from logging in each time your application there is another important reason to persist authentication state betwen application runs: for some services the amount of tokens, which can be returned, is limited, and thus the requirement of storing authentication info between application runs are stated explicitly in the service's documentation.

Since it hard to predict, when the token will need to be refreshed at run-time, NG ConnectionPack makes the check before any HTTP request to the service. This implies, that initial authentication dialog can be shown as a result of such check. So, basically, NG ConnectionPack does not provide a way to explicitly show the dialog, but instead it provides `RefreshAccess` method, which can be called explicitly to ensure that authentication state is ok.

If the user closes authentication dialog via standard close button on the window header before the authentication process finishes (thus canceling authentication), `EAbort` standard exception will be raised. This means that the execution of code will be interrupted as with any usual exception, but there will be no UI error message. Please read about `EAbort` exception class and the corresponding `Abort` procedure in the Delphi documentation.

Its good to provide the possibility for the user to log out explicitly. `ResetToken` service's method can be used to achieve this. After `ResetToken` method call all current authentication information will be reset, and the user will be forced to authenticate again via authentication dialog.

# Built-in Authentication UI

NG ConnectionPack provides several different built-in realizations of abstract `TAuthUI` class. There two ways, which differs by features. One of the described below units should be explicitly used in user's project to enable corresponding AuthUI implementation.

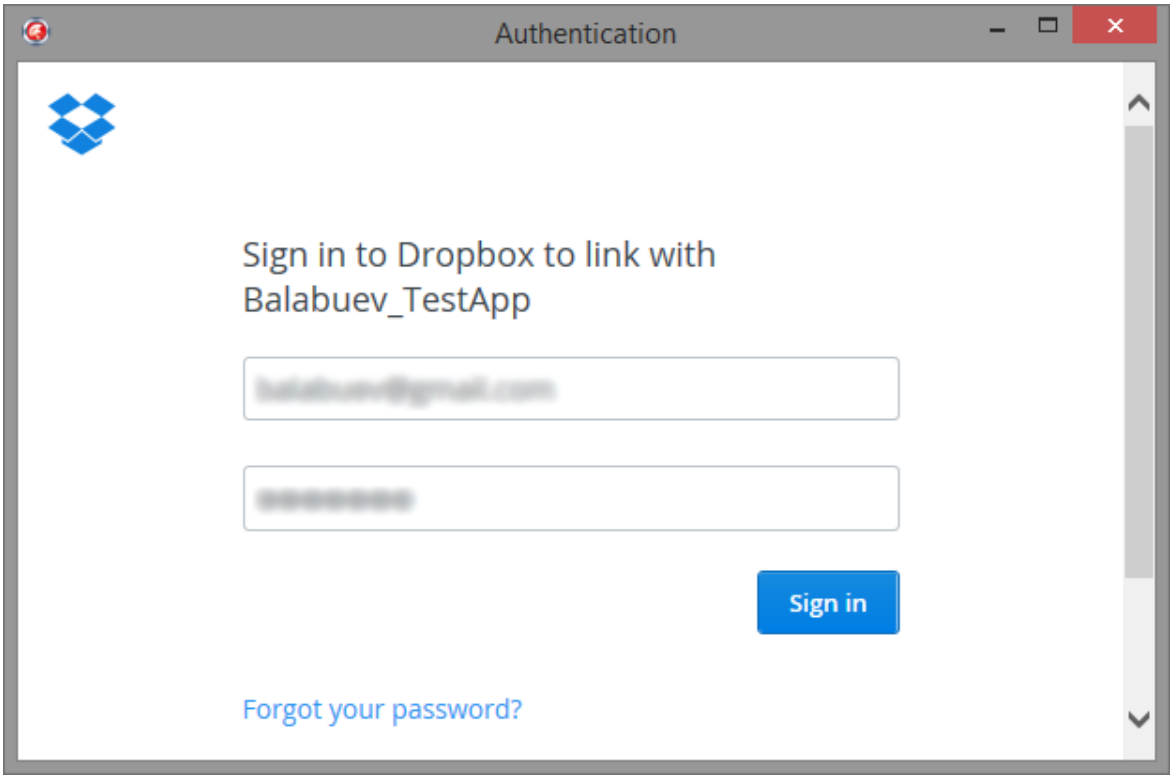
## Contained Web View

Units:

- `NG.Connection.VclAuth.pas` (for VCL projects)
- `NG.Connection.FmxAuth.pas` (for FMX projects)

This authentication UI shows modal dialog with the browser view inside. The code is able to intercept page navigation events, and so, non-localhost redirect URLs are supported. From 2020 version localhost URLs are also supported in this implementation.

Some time ago Google prohibit authentication using contained web view (`TWebBrowser`), because of security considerations, so, this way will not work with Google services.




Supported platforms:

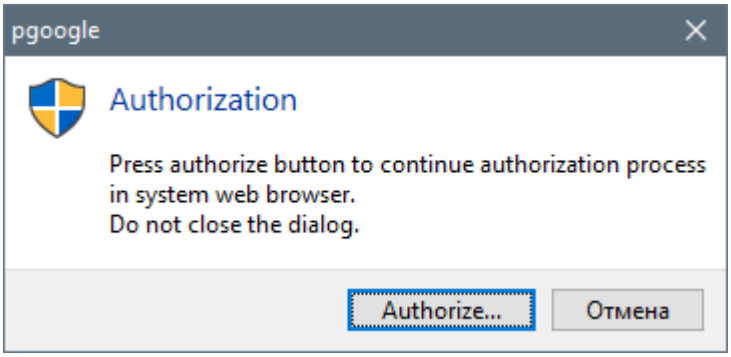
- Windows
- iOS

## System Browser

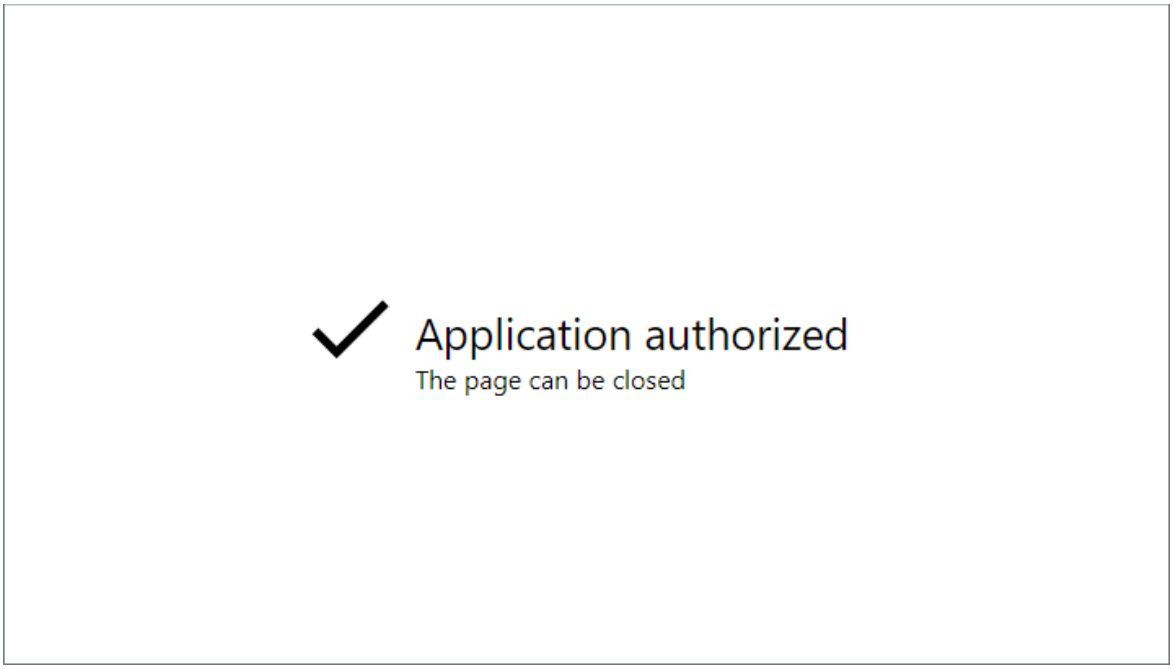
Units:

- `NG.Connection.SysVclAuth.pas` (for VCL projects)
- `NG.Connection.SysFmxAuth.pas` (for FMX projects)

This new (2020) authentication UI shows modal task dialog, which is able to open authentication web page in system browser. Using system browser, as opposed to contained web view, is the suggested way of performing authentication due to its higher security level. As well, users often have all authentication parameters (user names and passwords) pre-saved in the system browser, and so there no need to re-enter it many times. The code is **not** able to intercept page navigation events, and so, only localhost redirect URLs are supported.



After successful authorization the browser will be redirected to specified localhost address, and built-in local HTTP server will respond with the following page (text can be localized via resource-strings):




Supported platforms:

- Windows

### Redirect URLs

Redirect URLs to localhost is supported in the following two ways: http://localhost or http://127.0.0.1. Built-in tiny HTTP server is activated during authentication process to catch redirects to local URLs. Its port number can be specified in different ways:

- http://localhost **-** (no port number) default 80 port number will be used by local HTTP server.
- http://localhost:8000 - explicitly specified port number will be used by local HTTP server.
- http://localhost:[port] - (auto port number) local HTTP server will choose available port automatically.

So far dynamic port number scheme is supported only by Google services, and in this case redirect URLs should be specified as follows:

- "http://localhost:[port]" - in component's RedirectUrl property.
- "http://localhost" - (without port number) in Application Registration parameters.

## Custom Authentication UI

NG ConnectionPack provides `TAuthUI` abstract base class for those who want to implement his own customized authentication user interface. This is an advanced task, and NG ConnectionPack source code should be used as a reference implementation. After descending and implementing your own class from `TAuthUI` base class, `TRestService.OnGetAuthUIClass` event should be used to specify, that your custom class should be used instead of the built-in dialog.

In the case, when several services are connected into a group, which implies a shared authentication process, the event will be raised for all service components in the group in an unspecified order.

# Google API Key Authentication

Google services provides an additional ability to authenticate using API Key method instead of described above OAuth 2.0 method. Moreover, some advanced services requires to use this way. Please read the corresponding services documentation for more details.

# NGBeforeOAuth Global Callback

NG ConnectionPack provides NGBeforeOAuth(**var** Cancel, **var** Error) global callback, which can be used to intercept "on demand" authentication. Set `Cancel` parameter to `True` to silently cancel authentication process, and, additionally, `Error` parameter to some non-empty string to raise the error.

# State-less and Memory Management

# 8 State-less and Memory Management

NG ConnectionPack components implements *state-less* wrappers for corresponding REST services. Components do not store internally any returned data, such as file objects or calendar events.

This implies, that on each request new set of data will be returned. Most of such data types are really implemented as Delphi records (thanks to Delphi new language features, which allow the records to have properties and methods). However, `NGObject` data values are real Delphi objects and some of them need to be destroyed manually.

For example, to to retrieve file list from the Google Drive service the following code can be used:

```
var
  lst: TNGFileList;
  fl:  TNGFile;
  i:   Integer;
begin
  lst := NGGDrive1.ListFiles
                  .Q('trashed=false')
                  .Execute;
  try
    for i := 0 to lst.Files.High do
    begin
      fl := lst.Files[i];
      Memo1.Lines.Add(fl.Name);
    end;
  finally
    lst.Free;
  end;
end;
```

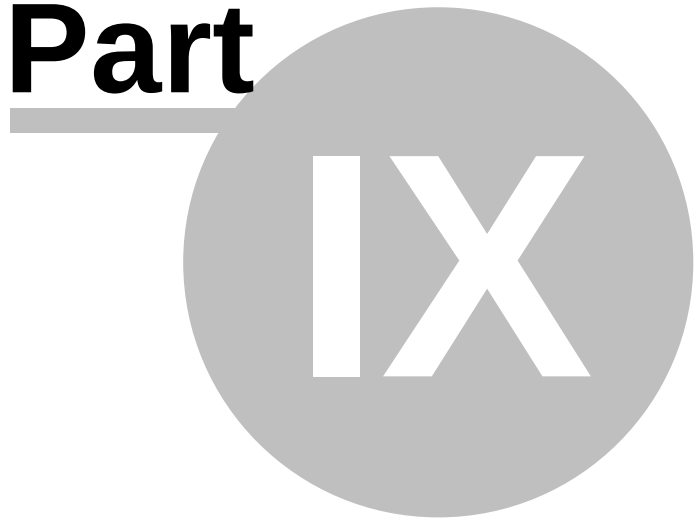
## Objects Lifetime

Instances of `NGObject` base class descendants are real Delphi objects, and thus, need to be destroyed eventually. The data of rest operation request and returned result is organizes as a tree (JSON is really a tree structure), where high-level object contain child objects directly as values of object properties or indirectly as a children of children or items of lists and maps. High-level object always owns all its child objects, and this, child objects should not be destroyed manually. Also, the user should not use manually created object in more than one place in that data tree, because then, it will be destroyed more than once; for each object property, which needs to have not null value, new unique object instance should be created.

Moreover, request data itself is implemented as a Delphi smart-record, and so, the user should never try to destroy request or any of its child objects manually.

However, rest operation result data is usually an object, which is real Delphi class instance, and so, it should be destroyed manually, after all required data has been read by the application.

# Service List



# 9 Service List

## Dropbox

| | |
|---|---|
| `TNGDropBox` | The Dropbox API allows developers to work with files in Dropbox, including advanced functionality like full-text search, thumbnails, and sharing.<br><br>https://www.dropbox.com/developers/documentation/http/documentation |

## Box.NET

| | |
|---|---|
| `TNGBoxNet` | The Box API gives you access to a set of secure content management features for use in your own app, such as file storage, preview, search, commenting, and metadata. It strives to be RESTful and is organized around the main resources from the Box web interface.<br>https://developer.box.com/v2.0/reference |

## Google Services

| | |
|---|---|
| `TNGGAbusiveExperienceReport` | View Abusive Experience Report data, and get a list of sites that have a significant number of abusive experiences.<br>https://developers.google.com/abusive-experience-report/ |
| `TNGGAcceleratedMobilePageUrl` | Retrieves the list of AMP URLs (and equivalent AMP Cache URLs) for a given list of public URL(s).<br>https://developers.google.com/amp/cache/ |
| `TNGGAdExchangeBuyer` | Accesses your bidding-account information, submits creatives for validation, finds available direct deals, and retrieves performance reports.<br>https://developers.google.com/ad-exchange/buyer-rest |
| `TNGGAdExchangeSeller` | Accesses the inventory of Ad Exchange seller users and generates reports.<br>https://developers.google.com/ad-exchange/seller-rest/ |
| `TNGGAdExperienceReport` | View Ad Experience Report data, and get a list of sites that have a significant number of annoying ads.<br>https://developers.google.com/ad-experience-report/ |
| `TNGGAdmin` | Fetches reports for the administrators of G Suite customers about the usage, collaboration, security, and risk for their users.<br>https://developers.google.com/admin-sdk/reports/ |
| `TNGGAdsense` | Accesses AdSense publishers' inventory and generates performance reports.<br>https://developers.google.com/adsense/management/ |
| `TNGGAnalytics` | Views and manages your Google Analytics data.<br>https://developers.google.com/analytics/ |

| | |
|---|---|
| TNGGAnalyticsReporting | Accesses Analytics report data.<br>https://developers.google.com/analytics/devguides/reporting/core/v4/ |
| TNGGAndroidDeviceProvisioning | Automates Android zero-touch enrollment for device resellers, customers, and EMMs.<br>https://developers.google.com/zero-touch/ |
| TNGGAndroidEnterprise | Manages the deployment of apps to Android for Work users.<br>https://developers.google.com/android/work/play/emm-api |
| TNGGAndroidManagement | The Android Management API provides remote enterprise management of Android devices and apps.<br>https://developers.google.com/android/management |
| TNGGAppEngine | The App Engine Admin API enables developers to provision and manage their App Engine applications.<br>https://cloud.google.com/appengine/docs/admin-api/ |
| TNGGAppsActivity | Provides a historical view of activity.<br>https://developers.google.com/google-apps/activity/ |
| TNGGAppState | The Google App State API.<br>https://developers.google.com/games/services/web/api/states |
| TNGGBigQuery | A data platform for customers to create, manage, share and query data.<br>https://cloud.google.com/bigquery/ |
| TNGGBigQueryDataTransfer | Transfers data from partner SaaS applications to Google BigQuery on a scheduled, managed basis.<br>https://cloud.google.com/bigquery/ |
| TNGGBlogger | API for access to the data within Blogger.<br>https://developers.google.com/blogger/docs/3.0/getting_started |
| TNGGBooks | Searches for books and manages your Google Books library.<br>https://developers.google.com/books/docs/v1/getting_started |
| TNGGCalendars | Manipulates events and other calendar data.<br>https://developers.google.com/google-apps/calendar/firstapp |
| TNGGCivicInfo | Provides polling places, early vote locations, contest data, election officials, and government representatives for U.S. residential addresses.<br>https://developers.google.com/civic-information |
| TNGGClassroom | Manages classes, rosters, and invitations in Google Classroom.<br>https://developers.google.com/classroom/ |
| TNGGCloudBilling | Allows developers to manage billing for their Google Cloud Platform projects programmatically.<br>https://cloud.google.com/billing/ |

| | |
|---|---|
| `TNGGCloudDebugger` | Examines the call stack and variables of a running application without stopping or slowing it down.<br>http://cloud.google.com/debugger |
| `TNGGCloudErrorReporting` | Groups and counts similar errors from cloud services and applications, reports new errors, and provides access to error groups and their associated errors.<br>https://cloud.google.com/error-reporting/ |
| `TNGGCloudFunctions` | API for managing lightweight user-provided functions executed in response to events.<br>https://cloud.google.com/functions |
| `TNGGCloudIot` | Registers and manages IoT (Internet of Things) devices that connect to the Google Cloud Platform.<br>https://cloud.google.com/iot |
| `TNGGCloudResourceManager` | The Google Cloud Resource Manager API provides methods for creating, reading, and updating project metadata.<br>https://cloud.google.com/resource-manager |
| `TNGGCloudShell` | Allows users to start, configure, and connect to interactive shell sessions running in the cloud.<br>https://cloud.google.com/shell/docs/ |
| `TNGGCloudTasks` | Manages the execution of large numbers of distributed requests. Cloud Tasks is in Alpha.<br>https://cloud.google.com/cloud-tasks/ |
| `TNGGCloudTrace` | Sends application trace data to Stackdriver Trace for viewing. Trace data is collected for all App Engine applications by default. Trace data from other applications can be provided using this API.<br>https://cloud.google.com/trace |
| `TNGGCloudUserAccounts` | Creates and manages users and groups for accessing Google Compute Engine virtual machines.<br>https://cloud.google.com/compute/docs/access/user-accounts/api/latest/ |
| `TNGGCompute` | Creates and runs virtual machines on Google Cloud Platform.<br>https://developers.google.com/compute/docs/reference/latest/ |
| `TNGGContainer` | The Google Kubernetes Engine API is used for building and managing container based applications, powered by the open source Kubernetes technology.<br>https://cloud.google.com/container-engine/ |
| `TNGGCustomSearch` | Searches over a website or collection of websites<br>https://developers.google.com/custom-search/v1/using_rest |
| `TNGGDataFlow` | Manages Google Cloud Dataflow projects on Google Cloud Platform.<br>https://cloud.google.com/dataflow |

| | |
|---|---|
| TNGGDataProc | Manages Hadoop-based clusters and jobs on Google Cloud Platform.<br>https://cloud.google.com/dataproc/ |
| TNGGDeployMentmanager | Declares, configures, and deploys complex solutions on Google Cloud Platform.<br>https://cloud.google.com/deployment-manager/ |
| TNGGDfaReporting | Manages your DoubleClick Campaign Manager ad campaigns and reports.<br>https://developers.google.com/doubleclick-advertisers/ |
| TNGGDialogFlow | An end-to-end development suite for conversational interfaces (e.g., chatbots, voice-powered apps and devices).<br>https://cloud.google.com/dialogflow-enterprise/ |
| TNGGDigitalAssetLinks | API for discovering relationships between online assets such as web sites or mobile apps.<br>https://developers.google.com/digital-asset-links/ |
| TNGGDlp | The Google Data Loss Prevention API provides methods for detection of privacy-sensitive fragments in text, images, and Google Cloud Platform storage repositories.<br>https://cloud.google.com/dlp/docs/ |
| TNGGDns | Configures and serves authoritative DNS records.<br>https://developers.google.com/cloud-dns |
| TNGGDoubleClickBidManager | API for viewing and managing your reports in DoubleClick Bid Manager.<br>https://developers.google.com/bid-manager/ |
| TNGGDoubleClickSearch | Reports and modifies your advertising data in DoubleClick Search (for example, campaigns, ad groups, keywords, and conversions).<br>https://developers.google.com/doubleclick-search/ |
| TNGGDrive | Manages files in Drive including uploading, downloading, searching, detecting changes, and updating sharing permissions.<br>https://developers.google.com/drive/ |
| TNGGFireBaseDynamicLinks | Programmatically creates and manages Firebase Dynamic Links.<br>https://firebase.google.com/docs/dynamic-links/ |
| TNGGFireBaseRemoteConfig | Firebase Remote Config API allows the 3P clients to manage Remote Config conditions and parameters for Firebase applications.<br>https://firebase.google.com/docs/remote-config/ |
| TNGGFitness | Stores and accesses user data in the fitness store from apps on any platform.<br>https://developers.google.com/fit/rest/ |
| TNGGFusionTables | API for working with Fusion Tables data.<br>https://developers.google.com/fusiontables |

| | |
|---|---|
| TNGGGames | The API for Google Play Game Services.<br>https://developers.google.com/games/services/ |
| TNGGGamesConfiguration | The Publishing API for Google Play Game Services.<br>https://developers.google.com/games/services |
| TNGGGamesManagement | The Management API for Google Play Game Services.<br>https://developers.google.com/games/services |
| TNGGGenomics | Upload, process, query, and search Genomics data in the cloud.<br>https://cloud.google.com/genomics |
| TNGGGroupsMigration | Groups Migration Api.<br>https://developers.google.com/google-apps/groups-migration/ |
| TNGGGroupsSettings | Lets you manage permission levels and related settings of a group.<br>https://developers.google.com/google-apps/groups-settings/get_started |
| TNGGIam | Manages identity and access control for Google Cloud Platform resources, including the creation of service accounts, which you can use to authenticate to Google and make API calls.<br>https://cloud.google.com/iam/ |
| TNGGIdentityToolKit | Help the third party sites to implement federated login.<br>https://developers.google.com/identity-toolkit/v3/ |
| TNGGKgSearch | Searches the Google Knowledge Graph for entities.<br>https://developers.google.com/knowledge-graph/ |
| TNGGLicensing | Views and manages licenses for your domain.<br>https://developers.google.com/google-apps/licensing/ |
| TNGGLogging | Writes log entries and manages your Stackdriver Logging configuration.<br>https://cloud.google.com/logging/docs/ |
| TNGGMail | Access Gmail mailboxes including sending user email.<br>https://developers.google.com/gmail/api/ |
| TNGGManufacturers | Public API for managing Manufacturer Center related data.<br>https://developers.google.com/manufacturers/ |
| TNGGMirror | Interacts with Glass users via the timeline.<br>https://developers.google.com/glass |
| TNGGML | An API to enable creating and using machine learning models.<br>https://cloud.google.com/ml/ |
| TNGGMonitoring | Manages your Stackdriver Monitoring data and configurations. Most projects must be associated with a Stackdriver account, with a few exceptions as noted on the individual method pages.<br>https://cloud.google.com/monitoring/api/ |

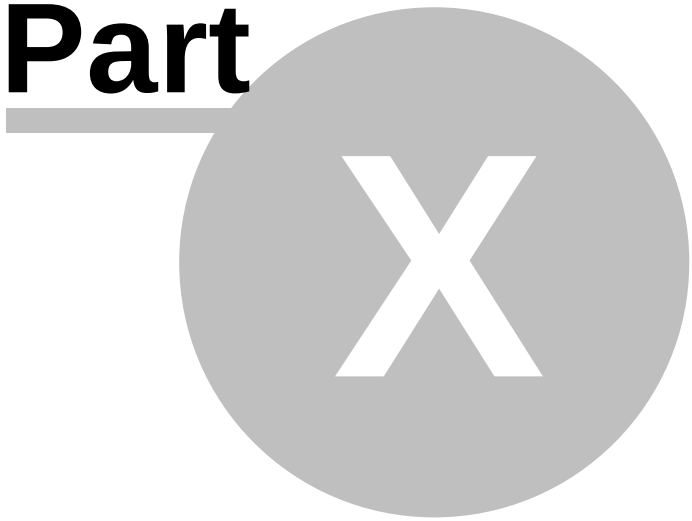
| | |
|---|---|
| TNGGPageSpeedOnline | Analyzes the performance of a web page and provides tailored suggestions to make that page faster. https://developers.google.com/speed/docs/insights/v2/getting-started |
| TNGGPartners | Searches certified companies and creates contact leads with them, and also audits the usage of clients. https://developers.google.com/partners/ |
| TNGGPeople | Provides access to information about profiles and contacts. https://developers.google.com/people/ |
| TNGGPlayCustomApp | An API to publish custom Android apps. https://developers.google.com/android/work/play/custom-app-api |
| TNGGPlusDomains | Builds on top of the Google+ platform for Google Apps Domains. https://developers.google.com/+/domains/ |
| TNGGPoly | The Poly API provides read-only access to assets hosted on https://poly.google.com. https://developers.google.com/poly/ |
| TNGGPrediction | Lets you access a cloud hosted machine learning service that makes it easy to build smart apps https://developers.google.com/prediction/docs/developer-guide |
| TNGGProximityBeacon | Registers, manages, indexes, and searches beacons. https://developers.google.com/beacons/proximity/ |
| TNGGPubSub | Provides reliable, many-to-many, asynchronous messaging between applications. https://cloud.google.com/pubsub/docs |
| TNGGReplicaPool | [Deprecated. Please use Instance Group Manager in Compute API] Provides groups of homogenous Compute Engine instances. https://developers.google.com/compute/docs/instance-groups/manager/v1beta2 |
| TNGGReplicaPoolUpdater | [Deprecated. Please use compute.instanceGroupManagers.update method. replicapoolupdater API will be disabled after December 30th, 2016] Updates groups of Compute Engine instances. https://cloud.google.com/compute/docs/instance-groups/manager/#applying_rolling_updates_using_the_updater_service |
| TNGGReseller | Creates and manages your customers and their subscriptions. https://developers.google.com/google-apps/reseller/ |
| TNGGResourceViews | The Resource View API allows users to create and manage logical sets of Google Compute Engine instances. https://developers.google.com/compute/ |

| | |
|---|---|
| TNGGRuntimeConfig | The Runtime Configurator allows you to dynamically configure and expose variables through Google Cloud Platform. In addition, you can also set Watchers and Waiters that will watch for changes to your data and return based on certain conditions.<br>https://cloud.google.com/deployment-manager/runtime-configurator/ |
| TNGGSafeBrowsing | Enables client applications to check web resources (most commonly URLs) against Google-generated lists of unsafe web resources.<br>https://developers.google.com/safe-browsing/ |
| TNGGScript | An API for managing and executing Google Apps Script projects.<br>https://developers.google.com/apps-script/api/ |
| TNGGSearchConsole | Provides tools for running validation tests against single URLs<br>https://developers.google.com/webmaster-tools/search-console-api/ |
| TNGGServiceControl | Google Service Control provides control plane functionality to managed services, such as logging, monitoring, and status checks.<br>https://cloud.google.com/service-control/ |
| TNGGServiceUser | Enables services that service consumers want to use on Google Cloud Platform, lists the available or enabled services, or disables services that service consumers no longer use.<br>https://cloud.google.com/service-management/ |
| TNGGSheets | Reads and writes Google Sheets.<br>https://developers.google.com/sheets/ |
| TNGGSiteVerification | Verifies ownership of websites or domains with Google.<br>https://developers.google.com/site-verification/ |
| TNGGSlides | An API for creating and editing Google Slides presentations.<br>https://developers.google.com/slides/ |
| TNGGSourceRepo | Access source code repositories hosted by Google.<br>https://cloud.google.com/source-repositories/docs/apis |
| TNGGSpanner | Cloud Spanner is a managed, mission-critical, globally consistent and scalable relational database service.<br>https://cloud.google.com/spanner/ |
| TNGGSpectrum | API for spectrum-management functions.<br>http://developers.google.com/spectrum |
| TNGGSpeech | Converts audio to text by applying powerful neural network models.<br>https://cloud.google.com/speech/ |

| | |
|---|---|
| TNGGSqlAdmin | Creates and configures Cloud SQL instances, which provide fully-managed MySQL databases. https://cloud.google.com/sql/docs/reference/latest |
| TNGGStorage | Stores and retrieves potentially large, immutable data objects. https://developers.google.com/storage/docs/json_api/ |
| TNGGStorageTransfer | Transfers data from external data sources to a Google Cloud Storage bucket or between Google Cloud Storage buckets. https://cloud.google.com/storage/transfer |
| TNGGStreetViewPublish | Publishes 360 photos to Google Maps, along with position, orientation, and connectivity metadata. Apps can offer an interface for positioning, connecting, and uploading user-generated Street View images. https://developers.google.com/streetview/publish/ |
| TNGGSurveys | Creates and conducts surveys, lists the surveys that an authenticated user owns, and retrieves survey results and information about specified surveys. |
| TNGGTagManager | Accesses Tag Manager accounts and containers. https://developers.google.com/tag-manager/api/v2/ |
| TNGGTaskQueue | Accesses a Google App Engine Pull Task Queue over REST. https://developers.google.com/appengine/docs/python/taskqueue/rest |
| TNGGTasks | Lets you manage your tasks and task lists. https://developers.google.com/google-apps/tasks/firstapp |
| TNGGTesting | Allows developers to run automated tests for their mobile applications on Google infrastructure. https://developers.google.com/cloud-test-lab/ |
| TNGGToolResults | Reads and publishes results from Firebase Test Lab. https://firebase.google.com/docs/test-lab/ |
| TNGGTpu | TPU API provides customers with access to Google TPU technology. https://cloud.google.com/tpu/ |
| TNGGTranslate | The Google Cloud Translation API lets websites and programs integrate with Google Translate programmatically. https://code.google.com/apis/language/translate/v2/getting_started.html |
| TNGGUrlshortener | Lets you create, inspect, and manage goo.gl short URLs https://developers.google.com/url-shortener/v1/getting_started |
| TNGGVault | Archiving and eDiscovery for G Suite. https://developers.google.com/vault |
| TNGGVision | Integrates Google Vision features, including image labeling, face, logo, and landmark detection, optical character recognition (OCR), and detection of explicit content, into |

| | |
|---|---|
| | applications.<br>https://cloud.google.com/vision/ |
| `TNGGWebFonts` | Accesses the metadata for all families served by Google Fonts, providing a list of families currently available (including available styles and a list of supported script subsets).<br>https://developers.google.com/fonts/docs/developer_api |
| `TNGGWebmasters` | View Google Search Console data for your verified sites.<br>https://developers.google.com/webmaster-tools/ |
| `TNGGYoutube` | Supports core YouTube features, such as uploading videos, creating and managing playlists, searching for content, and much more.<br>https://developers.google.com/youtube/v3 |
| `TNGGYoutubeAnalytics` | Retrieves your YouTube Analytics data.<br>http://developers.google.com/youtube/analytics/ |
| `TNGGYoutubeReporting` | Schedules reporting jobs containing your YouTube Analytics data and downloads the resulting bulk data reports in the form of CSV files.<br>https://developers.google.com/youtube/reporting/v1/reports/ |

# Terms of Use



# 10 Terms of Use

NG ConnectionPack allows to integrate cloud or remote 3rd party services (like Google, DropBox etc.) via public APIs made available by those service providers. NG ConnectionPack does not include any license to use these 3rd party services. To make use of the integration controls you need to register to these supported services and you're bound to the terms of those service providers. 3rd party service providers solely decide themselves who is allowed to use the services or which features are available to you.
This package neither includes any possible fees required to use 3rd party services (please make sure that you observe usage rules and pricing of the supported services) nor do we guarantee now or in the future usage or functionality of any of these 3rd party services.
Besides that LMD Innovative is not responsible for the use of NG ConnectionPack controls in your applications or projects.

Please check terms of 3rd party service providers which can be found at:

**Google**

https://developers.google.com/terms/
https://www.google.com/policies/terms/

**Dropbox**

https://www.dropbox.com/developers/reference/tos
https://www.dropbox.com/terms

**Box.Net**

https://www.box.com/legal/termsofservice
https://www.box.com/pricing/enterprise-license-agreement