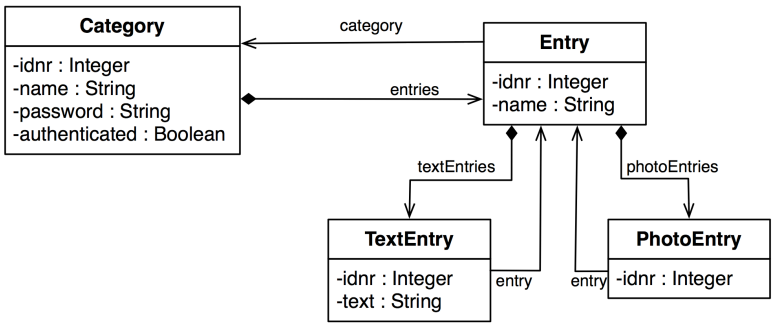
KeptSecret Server Protocol

This document describes the protocol used by *KeptSecret* v1.0

Datatypes used in the protocol

KeptSecret uses four datatypes as shown in the UML diagram in Picture 1:



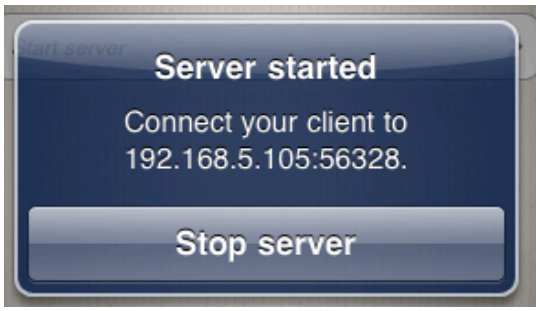
Picture 1: Class diagram of KeptSecret

A category is the uppermost class in the hierarchy. A category has zero or more entries. Each entry consists of a name, zero or more TextEntries and zero or more PhotoEntries and a backlink to the category it belongs to. TextEntries and PhotoEntries have backlink to their respective Entry object.

The protocol uses JSON[1] to format the objects.

Finding a KeptSecret instance

If the user starts the server mode of *KeptSecret*, he is presented with the info screen as shown in Picture 2.

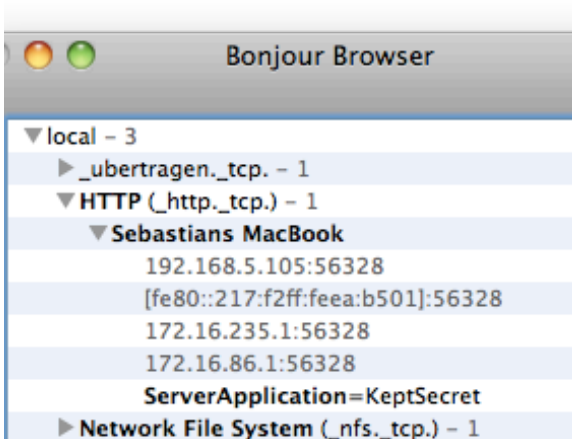


Picture 2: User info screen

This shows the IP address where you can reach *KeptSecret* and the port on which the server runs. This port is randomly assigned by the iPhone OS, so make sure you are not caching it.

[1] http://json.org

KeptSecret advertises itself using the Bonjour protocol. The type is ***_http._tcp.***
Additionally there is a TXTData called ***ServerApplication*** with the value of ***KeptSecret***:



Picture 3: Bonjour info for KeptSecret

Connecting to a *KeptSecret* instance

To connect a given *KeptSecret* instance, send a HTTP GET request to

`%ip:%port/auth=%password`

`%ip` is the IP address of the *KeptSecret* instance, **`%port`** the corresponding port. **`%password`** is the password for this instance.

The return of this call is an array which one or two elements. It contains at least the element **`Authentificated`** with the value of **`YES`**, if the password is correct and **`NO`** if it isn't. If the password is correct, the retuned array also has the element **`token`**, which contains an authentication token, which you use in the next call to the server.

Example:

```
{
  "Authentificated":"YES",
  "token":" CDE81CC2-E5FD-4E30-BD08-FDF1FAA6AF46"
}
```

The token is a one-way token. After using it in a successful call to the server, you get a new token as part of the answer.

Manipulating the *KeptSecret* database

To manipulate the entries in a KeptSecret database, you just send specialized HTTP requests to the server. Table 1 shows the allowed HTTP methods and the generic URI for this action

Method	Action	Generic URI
GET	Gets a list of the specific resource	%ip:%host/%resource?token=%token
GET	Gets the specific resource	%ip:%host/%resource/%id?token=%token
PUT	Modifies the specific resource	%ip:%host/%resource/%id?token=%token
POST	Adds a new resource	%ip:%host/%resource?token=%token
DELETE	Removes the specified method	%ip:%host/%resource/%id?token=%token

Table 1: KeptSecret HTTP methods

Allowed values for **`%resource`** are:

- categories
- entries
- textentries
- photoentries

The GET method allows you to get a list of all available resources or just a specific version with the given id.

If the call was completed successful, the return value is a list with the two entries **`token`** and **`payload`**. The entry for **`token`** contains the token for the next call. The token used for this call gets consumed is not longer valid. The entry for **`payload`** contains the actual return of this call.

Example:

```
{
  "token":"CDE81CC2-E5FD-4E30-BD08-FDF1FAA6AF46",
  "payload":[
    {
      "category" : {
        "password" : "",
        "authenticated" : true,
        "idnr" : 1,
        "name" : "Office",
        "entries" : []
      }
    },{
      "category" : {
        "password" : "9F9D51BC70EF21CA5C14F307980A29D8",
        "authenticated" : false,
        "idnr" : 2,
        "name" : "Jordan's Files",
        "entries" : []
      }
    }
  ]
}
```

If the token was not valid, the return is a list with just the entry **`Authentificated`** with the value **`NO`**.

Example:

```
{
  "Authentificated":"NO"
}
```

Except explicitly said, we will just look at **`payload`** attribute in following text.

Working with Categories

Categories are the basic building blocks for the *KeptSecret* datamodel. Contrary to other objects, categories can be secured with an additional password for extra security.

Getting categories

To get a list of all available categories, you send a GET request to

%ip:%host/categories?token=%token

The return value is a list of categories in the **payload** field. The value for the attribute **entries** is always an empty list.

As a category can have an own password, it may cannot be accessed at some time. If this is the case, the value of the attribute **authenticated** is **NO**. For locked categories, the **password** attribute has a random content.

To get a detailed return for a single category send a GET request to

%ip:%host/categories/%id?token=%token

where **%id** is the id number for this category. The returned category contains a complete entries array with all Entry-objects that are contained in this category.

If the category doesn't exist, the value of the **payload** attribute is an object called **Error** with the value **Category not found**.

Example:

```
{
  "token":"CDE81CC2-E5FD-4E30-BD08-FDF1FAA6AF46",
  "payload":{
    "Error":"Category not found"
  }
}
```

If the category exists, but is locked, the value of the **payload** attribute is an object called **CategoryAuthentificated** with the value **NO**.

Example:

```
{
  "token":"CDE81CC2-E5FD-4E30-BD08-FDF1FAA6AF46",
  "payload":{
    "CategoryAuthentificated":"NO"
  }
}
```

To unlock a category, send a GET request to

%ip:%host/categories/%id?token=%token&auth=%categorypassword

where **%categorypassword** is the corresponding password.

Adding categories

To add new category to *KeptSecret*, send a POST request to

%ip:%host/categories?token=%token

The attribute **idnr** must be set to **-1**. The value of the **payload** field is the new category object or an object called **Error** with the value **Malformed POST request**.

If you add a category with a password, it is considered unlocked for the rest of the communication.

Changing categories

To change a category, send a PUT request to

`%ip:%host/categories/%id?token=%token`

Make sure, that the **`idnr`** attribute of the sent request is equal to the **`%id`** attribute of the URI. This changes only the category itself. If you send something for the category's **`entries`** attribute, it will be ignored.

If **`idnr`** and **`%id`** are different, the **`payload`** contains an object **`Error`** with the value **`Malformed PUT request`**.

If the operation succeeds, the changed category object will be returned.

Deleting categories

To delete a category, send an DELETE request to

`%ip:%host/categories/%id?token=%token`

If the category was deleted, the value of the **`payload`** field is an object named **`CategoryDeleted`** with the value of **`YES`**.

Example:

```
{
  "token":"CDE81CC2-E5FD-4E30-BD08-FDF1FAA6AF46",
  "payload":{
    "CategoryDeleted":"YES"
  }
}
```

Working with Entries

Entries are the second level of ordering data in the *KeptSecret* database. As each Entry-object is assigned to a category, you must first unlock the associated category in order to work with an Entry-object.

If you try to work with an Entry-object in a locked Category, you will get an error message as return.

Example:

```
{
  "token":"CDE81CC2-E5FD-4E30-BD08-FDF1FAA6AF46",
  "payload":{
    "CategoryAuthentificated":"NO"
  }
}
```

Getting Entries

To get a detailed return for a single entry send a GET request to

`%ip:%host/entries/%id?token=%token`

where **`%id`** is the id number for this entry. The returned entry contains all the TextEntry- and PhotoEntry-objects assigned to this Entry-object.

If the entry doesn't exists, the value of the **`payload`** attribute is an object called **`Error`** with the value **`Entry not found`**.

Example:

```
{
  "token":"CDE81CC2-E5FD-4E30-BD08-FDF1FAA6AF46",
  "payload":{
    "Error":"Entry not found"
  }
}
```

Adding Entries

To add new entry to *KeptSecret*, send a POST request to

`%ip:%host/entries?token=%token`

The attribute **`idnr`** must be set to **`-1`**. The value of the **`payload`** field is the new Entry-object or an object called **`Error`** with the value **`Malformed POST request`**.

Changing Entries

To change an entry, send a PUT request to

`%ip:%host/entries/%id?token=%token`

Make sure, that the **`idnr`** attribute of the sent request is equal to the **`%id`** attribute of the URI.

If **`idnr`** and **`%id`** are different, the **`payload`** contains an object **`Error`** with the value **`Malformed PUT request`**.

If the operation succeeds, the changed entry-object will be returned.

Deleting Entries

To delete an entry, send an DELETE request to

`%ip:%host/entries/%id?token=%token`

If the entry was deleted, the value of the **`payload`** field is an object named **`EntryDeleted`** with the value of **`YES`**.

Example:

```
{
  "token":"CDE81CC2-E5FD-4E30-BD08-FDF1FAA6AF46",
  "payload":{
    "EntryDeleted":"YES"
  }
}
```

Working with TextEntries

Working with TextEntries follows the same rules as working with Entries.

Make sure to set the correct value for the **`entry`** attribute to which you want to add the TextEntry-object.

To confirm that a TextEntry has been deleted, the payload contains the object **`TextEntryDeleted`** with the value **`YES`**.

Working with PhotoEntries

PhotoEntries are different to any other object, because they consist of a metadata part and a binary one. If you create a PhotoEntry, you just create the metadata part. After that you can upload a binary file for this. Working with binary parts of a PhotoEntry object is the only part of the server protocol that does not consume a token.

Working with the metadata of a PhotoEntry is like working with any other Entry type.

To confirm that a PhotoEntry has been deleted, the payload contains the object **`PhotoEntryDeleted`** with the value **`YES`**.

To work with the binary part of a PhotoEntry object, add the parameter **`type`** with the value **`image`** to the URI of the request. The return for this request is the binary stream of the image data. The image is a JPEG image.

The concept of a PhotoEntry allows only metadata as request value for the POST method. This creates a PhotoEntry with an empty image. After that you can set (or change) the image by sending a PUT request. For this send the binary image data as the request body. You can either send JPEG or PNG data. *KeptSecret* stores this image with JPEG compression and a compression ratio of 80%.

Closing words

This document is the reference for the *KeptSecret* server protocol. If you find a bug, feel free to contact us on developer@keptsecretapp.com. If you find something that is sent from the server but not described in this guide make sure your application does not depend on this undocumented data, since it may change in further versions.