# LLM WEB SERVICE

# SOAP API Developer

# Reference

## CONTENTS

# PREFACE

## THIS DOCUMENT

This document describes the LLM SOAP Application Programming Interface (API) and service.

## INTENDED AUDIENCE

This document is written for programmers familiar with application programming standards such as the Simple Object Access Protocol (SOAP), the Web Services Description Language (WSDL), and XML Schema Definition (XSD) language.

## REVISION HISTORY

Revision history for *SOAP API Developer Reference*.

***TABLE P.1 Revision History***

| Date | Description |
|---|---|
| *04 Sept 2009* | *First Version of LLMWS is Published* |
| *10 Sept 2009* | *Error messages added to every LLMWS method* |
| *18 Sept 2009* | *Name of some structures have been changed* |
| *26 Mar 2010* | *Properties methods for seniors and users added* |
| | |
| | |
| | |

## LLM SOAP API OVERVIEW

The LLM SOAP API provides programmatic access to LLM features and services. Developers can build custom applications, tools, and services that correspond to the same services. Typical applications include add/edit and searching for registered seniors, add senior's progress to cognitive or physical training and add information about senior falling or alarms. The API is based on open standards known collectively as "Web Services," which include the Simple Object Access Protocol (SOAP), Web Services Definition Language (WSDL), and the XML Schema Definition language (XSD). These standards are supported by a wide range of development tools on a variety of platforms.

### SERVICES ARCHITECTURE

Like many web services, LLM SOAP is a combination of client-side and server-side schemas, hardware and software servers, and core services.

In an object-oriented processing model, the interface to SOAP requests/responses is an object in your application's native programming language. Your third-party SOAP client generates business-object interfaces and network stubs from LLM-provided WSDL and XSD files that specify the LLM message structure, its contents, and the LLM API service bindings.

A business application works with data in the form of object properties to send and receive data by calling object methods. The SOAP client handles the details of building the SOAP request, sending it to the LLM service, and converting the response back to an object.

### LLM WSDL/XSD SCHEMA DEFINITIONS

The LLM Web Services schema is required for developing applications with the LLM Web Services API. The following are the locations of the WSDL and LLMWS.

| | |
|---|---|
| LLM Web Service | http://nosmoke.med.auth.gr/LLMWebService/LLMWebService.asmx |
| LLM Schema | http://nosmoke.med.auth.gr/LLMWebService/LLMWebService.asmx?wsdl |

### API CONCEPTS AND TERMINOLOGY

Here are some basic concepts and terminology relating to LLM's API service and security authentication.

| **Term** | **Definition** |
|---|---|
| API Calls | LLM Application Programming Interface services, by which users and seniors can add information about senior progress and users interact with seniors data. |
| API Username and Password | A LLM-generated identifying user's account name and password that the user (or senior with less privileges) uses specifically for making API calls. You include your API username and password with every API call. The API username and password is the same as the LLM login username and password. |

## DOCUMENT ABBREVIATIONS

Here are some abbreviations used in this document.

| Abbreviation | Definition |
|---|---|
| LLMDB | LLM DataBase |
| LLMWS | LLM Web Service |
| LLMDR | LLM Developer Reference (This document) |
| ILC | Independent Living Component |
| CTC | Cognitive Training Component |
| PTC | Physical Training Component |

## SECURITY

The LLM SOAP API service is protected to ensure that only authorized LLM members (users and seniors) use it. There are two levels of security:

1. A required API username (Username field) and API password (Password field)

2. Secure Sockets Layer (SSL) data transport (not implemented yet)

## CHARACTER ENCODING, DATA TYPES AND FORMATS, AND CURRENCIES

This section details allowed character encoding and character sets, date data types, and formats.

### UTF-8 CHARACTER ENCODING

The LLM SOAP API service assumes that all data in SOAP requests is in Unicode, specifically, the Unicode (or UCS) Transformation Format, 8-bit encoding form (UTF-8). In SOAP responses, the service always returns data in UTF-8.

### DATE/TIME FORMATS

It'S TO BE UPDATED

The LLM SOAP API schema defines date/time values as Coordinated Universal Time (UTC/GMT), using ISO 8601 format, and of type ns:dateTime. An example date/time stamp is 2006-08-24T05:38:48Z

### ENUMERATIONS

| Enumeration | Values |
|---|---|
| UserRole | Administrator, Therapist, Relative, Unknown |
| UserCategory | User, Senior |
| PropertyTypeEnum | Unknown, String, Integer, Double, DateTime, Bool |

The enumeration UserCategory is used to describe if a member of the LLM Service is a User or a Senior. It is useful in some methods which return member instead of user or senior. This is because the web service deals different with Users and Seniors. But, in some situations, it is required from the system to return the member of the system, his/her category and Role. Namely, the LogIn function requires username and password and returns if there exists a member with the provided credentials, in which category belongs and if is he/she a User  provides his/her Role.

```
<s:simpleType name="UserCategory">
  <s:restriction base="s:string">
    <s:enumeration value="User"/>
    <s:enumeration value="Senior"/>
  </s:restriction>
</s:simpleType>
```

The enumeration UserRole is meaningful only if the member is a User.

```
<s:simpleType name="UserRole">
  <s:restriction base="s:string">
    <s:enumeration value="Administrator"/>
    <s:enumeration value="Therapist"/>
    <s:enumeration value="Relative"/>
    <s:enumeration value="UnKnown"/>
  </s:restriction>
</s:simpleType>
```

The enumeration PropertyTypeEnum is used to describe the type of a property.

```
<s:simpleType name="PropertyTypeEnum">
  <s:restriction base="s:string">
    <s:enumeration value="Unknown"/>
    <s:enumeration value="String"/>
    <s:enumeration value="Integer"/>
    <s:enumeration value="Double"/>
    <s:enumeration value="DateTime"/>
    <s:enumeration value="Bool"/>
  </s:restriction>
</s:simpleType>
```

## STRUCTURES

SOAP allows client programs to pass in almost any kind of data structure to the Web Service; that's why Web Services use SOAP instead of HTTP. The LLM API provides structures as inputs and outputs to the Methods. All structure are explained to the appropriate section of this document. Some of them are shown to the table below.

| Structure | Description |
|---|---|
| senior | |

| | |
|---|---|
| user | |
| CTCSeniorActivity | |
| PTCSeniorActivity | |
| ILCSeniorActivity | |

The structure Senior include in the WSDL file

```
<s:complexType name="senior">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="senior_id" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="lname" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="fname" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="address" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="city" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="country" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="birthdate" type="s:dateTime"/>
    <s:element minOccurs="0" maxOccurs="1" name="sex" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="passwd" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="active" type="s:boolean"/>
    <s:element minOccurs="1" maxOccurs="1" name="lastlogin" type="s:dateTime"/>
    <s:element minOccurs="0" maxOccurs="1" name="tel" type="s:string"/>
  </s:sequence>
</s:complexType>
```

## ROLES

The system is able to deal with 4 Member Roles. The Role with the less privileges is the Relative (User) who is able to get results about the relative senior. More privileges are attached to the Senior (Senior) who has not only the right to see his/hers progress but also to add information about it (through his/her LogIn to the ILC, CTC and PTC). The Therapist has the authority to add/edit, delete and get results about all seniors that were created from him/her (Seniors that are attached to the Therapist). Finally, the administrator has all the available rights.

## PROGRAMMING LANGUAGE

The LLM web service was developed within the .NET framework (IDE VS2008) using C#. Following sections include several examples written in .NET (C#) which implement the client –side call of web service's methods. Within the next versions of the LLMDR further examples will be documented in several other programming languages.

## ERROR HANDLING

Errors are used in order to provide feedback about successful or not call of the methods. The error structure is provided as the response of the LLMWS call of methods. It provides information about the privileges, etc. The next chapters of this document deal with the structures of seniors, users, activities etc. The response of each method is a structure that includes the requested object structure (senior, user, etc) and the error object structure. For this reason, the next chapters will deal only with the requested object structure. A full example is provided in order to clarify general concept.

### ERROR'S STRUCTURE

```
<s:complexType name="Error">–
   <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="ErrorCode" type="s:int"/>
      <s:element minOccurs="0" maxOccurs="1" name="ErrorMessage" type="s:string"/>
   </s:sequence>
</s:complexType>
```

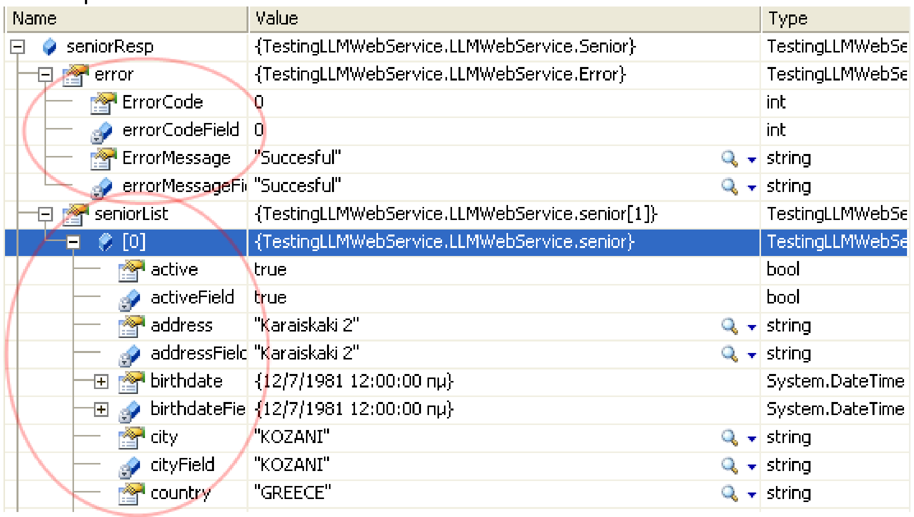
| **Attribute** | **Description** | **Type** |
|---|---|---|
| ErrorCode | The unique code of the error | (integer)O in case of successful response |
| ErrorMessage | The error's message | String |

| ErrorCode | ErrorMessage | Description |
|---|---|---|
| 0 | Succesful | |
| 1 | UnknownError | |
| 2 | AccessDenied | |
| 3 | InvalidCredentials | |

### EXAMPLE 1

The "save_senior" method (it is explained in the next chapters) returns a combined structure instead of providing a senior's tructure.

The response of the save senior is



| Name | Value | Type |
|---|---|---|
| seniorResp | {TestingLLMWebService.LLMWebService.Senior} | TestingLLMWebSe |
| error | {TestingLLMWebService.LLMWebService.Error} | TestingLLMWebSe |
| ErrorCode | 0 | int |
| errorCodeField | 0 | int |
| ErrorMessage | "Succesful" | string |
| errorMessageFi | "Succesful" | string |
| seniorList | {TestingLLMWebService.LLMWebService.senior[1]} | TestingLLMWebSe |
| [0] | {TestingLLMWebService.LLMWebService.senior} | TestingLLMWebSe |
| active | true | bool |
| activeField | true | bool |
| address | "Karaiskaki 2" | string |
| addressFielc | "Karaiskaki 2" | string |
| birthdate | {12/7/1981 12:00:00 πμ} | System.DateTime |
| birthdateFie | {12/7/1981 12:00:00 πμ} | System.DateTime |
| city | "KOZANI" | string |
| cityField | "KOZANI" | string |
| country | "GREECE" | string |

The returned structure is the “seniorResp”. This structure includes the error object and an array of seniors “seniorList” instead of a single senior object.

## SENIORS

### SENIOR'S STRUCTURE

```
<s:complexType name="senior">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="senior_id" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="lname" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="fname" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="address" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="city" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="country" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="birthdate" type="s:dateTime"/>
    <s:element minOccurs="0" maxOccurs="1" name="sex" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="passwd" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="active" type="s:boolean"/>
    <s:element minOccurs="1" maxOccurs="1" name="lastlogin" type="s:dateTime"/>
    <s:element minOccurs="0" maxOccurs="1" name="tel" type="s:string"/>
  </s:sequence>
</s:complexType>

<s:complexType name="SeniorR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="seniorList" type="tns:ArrayOfSenior"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| senior_id | The Unique Identifier of the Senior. It is auto-generated by the LLMWS (autoincrement) | Is an integer starting from 1. If the senior_id is 0 it means that the Senior is not saved in the LLMDB |
| lname | The Last Name of The Senior | String |
| fname | The First Name of The Senior | String |
| address | The home Address of The Senior | String |
| city | The City of The Senior | String (it will be changed to integer) |
| country | The Country of The Senior | String (it will be changed to integer) |
| birthdate | The Birthdate of The Senior | Date |
| sex | The Sex of The Senior | Although it is a string it should be 'M' for male or 'F' for female (it will be changed to enumeration) |
| username | The username of The Senior (required for LogIn) | String (max 30 characters) |
| passwd | The password of The Senior (required for LogIn) | String (max 30 characters) |
| active | If the Senior is Active or Not | True: active, False: Not Active |
| lastlogin | The date time of the last Log In | Date |
| tel | The telephone number of the Senior | String |

## ADD NEW SENIOR

The method that is used in order to add a new senior to the System is called
save_senior

```
SeniorR save_senior(senior seniorx, string username, string password);
```

The Senior (seniorx) parameter must have as senior_id = 0. This requirement is used in order to facilitate the method to separate if it is a new senior that should be added to the database or an already senior that he/she will be updated (explained in Section Edit Senior). The method save_senior returns a SeniorR structure with the error message and a list with one object (senior). The rest attributes of the senior Structure are store to the LLMDB. The username and password that are passed to the parameters are the user's credentials who is calling this method. If these credentials belong to a Senior or Relative the procedure is terminated and returns an error. In case of these credentials belonging to a Therapist or Administrator the procedure is running and the new added senior "belongs/attached" to the Therapist or Administrator that Called the method (see Attach Senior To User).

### EXAMPLE 1

```
LLMWebService.senior seniorx = new TestingLLMWebService.LLMWebService.senior();
 seniorx.active = true;
seniorx.address = "Karaiskaki 2";
seniorx.birthdate = new DateTime(1981, 07, 12);
seniorx.city = "KOZANI";
seniorx.country = "GREECE";
seniorx.fname = "Pan";
seniorx.senior_id = 0;
seniorx.lname = "Bam";
seniorx.sex = "M";
seniorx.tel = "112345678";
seniorx.username = "pan1";
seniorx.passwd = "pan1";

LLMWebService.SeniorR seniorResp = LLMWebServ.save_senior(seniorx, clsStaticVariables.UserName,
clsStaticVariables.PassWord);

if (seniorResp.error.ErrorCode == 0)
{
   seniorx = seniorResp.seniorList[0];
}
```

First a new instance of senior is created. Then each attribute of the structure is assigned a value. Finally, via the web service method *save_senior*, the new instance is being stored into the LLMDB. It must be stressed out that the method returns a structure (seniorResp) whose senior array contains (seniorList) the just saved senior with his ID, in case of successful storage.

## EDIT EXISTING SENIOR

The method that is used in order to edit an existing senior of the System is called
save_senior

```
SeniorR save_senior(senior seniorx, string username, string password);
```

The Senior (seniorx) parameter must have as senior_id the senior_id that will be edited. This requirement is used in order to facilitate the method to separate if it is a new Senior that should be added to the database (explained in Section Add New Senior) or an already Senior that he/she will be updated. The method save_senior returns a SeniorR structure. The rest attributes of the senior Structure are updated to the LLMDB. The username and password that are passed to the parameters are the user's credentials who is calling this method. If these credentials belong to a Senior or Relative the procedure is terminated and returns the appropriate error message. In case of these credentials belonging to a Therapist, the Therapist must have attached the specified Senior in order to update him/her. If the senior doesn't belong to the Therapist who calls the method the procedure is terminated and returns the appropriate error message. In case of the administrator the procedure runs independently of to whom the senior belongs to.

### EXAMPLE 1

```
int senior_id = 5;
LLMWebService.senior seniorx;
LLMWebService.SeniorR senior_arrayx = LLMWebServ.get_senior(senior_id, “username”, “password”);
if (senior_arrayx.seniorList.Length > 0)
{
   seniorx = senior_arrayx.seniorList[0];
   seniorx.fname = "George";
   LLMWebService.SeniorR seniorResp = LLMWebServ.save_senior(seniorx, “username”, “password”);
   if (seniorResp.error.ErrorCode == 0)
   {
      seniorx = seniorResp.seniorList[0];
   }
}
```

In this example, given the senior’s id we first get its corresponding instance (senior_arrayx.seniorList[0]) with the method *get_senior*. After that any changes are possible concerning senior’s structure attributes, e.g. its fname. Finally the altered instance of the senior structure is stored via *save_senior* method.

## DELETE SENIOR

The method that is used in order to delete an existing senior from the System is called
Delete_senior
`SeniorR Delete_senior(int senior_id, string username, string password);`
The senior_id must be equal to the senior that you want to delete. If the senior_id is 0 the procedure is terminated and returns in the SeniorR structure the appropriate error message. A Senior can be deleted only by a Therapist who has privileges to the selected Senior or by an Administrator. The user’s credentials facilitate the recognition of the user’s privileges.

### EXAMPLE 1

```
int senior_id = 5;
LLMWebService.SeniorR seniorx =  LLMWebServ.Delete_senior(senior_id, “username”, “password”);
```

In this example, given of the senior’s id, the LLMWS *Delete_senior* method deletes the corresponding senior user.

## GET SENIOR/SENIORS

The method that is used in order to get an existing senior/seniors from the System is called
get_senior
`SeniorR get_senior(int senior_id, string username, string password);`

The senior_id must be equal to the senior that you want to get. If the senior_id is 0 all the available Seniors returned from the System. The user credentials responsible for the Seniors that will be returned.
If the Member is a senior (based on the provided credentials) the system will return only his/her information. If a senior tries to get data from another senior the system returns an empty seniorList and the appropriate error message. The relative is able to get only information about his/hers relative Senior. The therapist can see information about all seniors attached to the specified by the credentials therapist. Finally, the administrator is able to get all the seniors.

## EXAMPLE 1

```
int senior_id = 3;
LLMWebService.SeniorR senior_arrayx = LLMWebServ.get_senior(senior_id, “username”, “password”);
```

As mentioned above, *get_senior* method can provide a certain senior user or a list with the seniors available to the user requested them. In this example, a senior with a certain id is requested and assigned to the array of senior structures as the one and only element of it.

## EXAMPLE 2

```
int senior_id = 0;
LLMWebService.SeniorR senior_arrayx = LLMWebServ.get_senior(senior_id,”username”, “password”);

senior_arrayx.seniorList[0]  //First element access of seniorList, which contains a senior structure.
senior_arrayx.seniorList[0].city = "Thessaloniki"; // Set value of city attribute of the first element of seniorList.
senior_arrayx.error.ErrorMessage //Access to the error structure and specifically to the ErrorMessage attribute.
```

The example above represents the case of using *get_senior* method to acquire an array  with all the available senior users. For this reason, senior_id is assigned a zero value.

## USERS

### USER'S STRUCTURE

```
<s:complexType name="user">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="user_id" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="lname" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="fname" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="password" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="role_id" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="role_name" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="Role" type="tns:UserRole"/>
  </s:sequence>
</s:complexType>

<s:complexType name="UserR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="userList" type="tns:ArrayOfUser"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| user_id | The Unique Identifier of the User. It is auto-generated by the LLMWS (autoincrement) | Is an integer starting from 1. If the user_id is 0 it means that the User is not saved in the LLMDB |
| lname | The Last Name of The User | String |
| fname | The First Name of The User | String |
| username | The username of The User (required for LogIn) | String (max 30 characters) |
| password | The password of The User (required for LogIn) | String (max 30 characters) |
| role_id | The role Id of the User | The Id of the role as it is stored in the LLMDB |
| role_name | The role name in string format of the User | String |
| Role | The Role of the User (enumeration) | Enumeration (UserRole) |

### ADD NEW USER

The method that is used in order to add a new user to the System is called
save_user
UserR save_user(user userx, string username, string password);
The user (userx) parameter must have as user_id = 0. This requirement is used in order to facilitate the method to separate if it is a new user that should be added to the database or an already user that he/she will be updated (explained in Section Edit User). The method save_user returns a UserR structure with the error message and a list with one object (user). The rest attributes of the User Structure are store to the LLMDB. The username and password that are passed to the parameters are the user's credentials who is calling this method. If these credentials belong to a

Senior,Relative or Therapist the procedure is terminated and returns the appropriate error message. In case of these credentials belonging to an Administrator the procedure is running.

In case of adding or editing a user it is obligated to provide only the Role and not the role_name and role_id.

### EXAMPLE 1

```
LLMWebService.user userx = new TestingLLMWebService.LLMWebService.user();
userx.fname = "RALTEC";
userx.user_id = 0;
userx.lname = "RALTEC";
userx.username = "raltec";
userx.password = "raltec";
userx.Role = TestingLLMWebService.LLMWebService.UserRole.Administrator;
LLMWebService.UserR userResp = LLMWebServ.save_user(userx, “username”, “password”);
if (userResp.error.ErrorCode == 0)
{
  userx = userResp.userList[0];
}
```

In this example, a new instance of user is created. Then each attribute of the structure is assigned a value. Finally, via the web service method *save_user*, the new instance is being stored into the LLMDB. It must be stressed out that the method returns a structure (userResp) whose user array contains (userList) the just saved senior with his ID, in case of successful storage.

## EDIT EXISTING USER

The method that is used in order to edit an existing user of the System is called
save_ user

```
UserR save_user(user userx, string username, string password);
```

The user (userx) parameter must have as user_id the user_id that will be edited. This requirement is used in order to facilitate the method to separate if it is a new User that should be added to the database (explained in Section Add New User) or an already User that he/she will be updated. The method save_user returns a UserR structure. The rest attributes of the Senior Structure are updated to the LLMDB. The username and password that are passed to the parameters are the user’s credentials who is calling this method. If these credentials belong to a Senior, Relative or Therapist the procedure is terminated and returns the appropriate error message. In case of the administrator the procedure runs.

In case of adding or editing a user it is obligated to provide only the Role and not the role_name and role_id.

### EXAMPLE 1

```
int user_id = 5;
LLMWebService.user userx;
LLMWebService.UserR user_arrayx = LLMWebServ.get_user(user_id, “username”, “password”);
if (user_arrayx.userList.Length > 0)
{
  userx = user_arrayx.userList[0];
   userx.fname = "George";
```

```
    LLMWebService.UserR userResp = LLMWebServ.save_user(userx, “username”, “password”);
    if (userResp.error.ErrorCode == 0)
    {
        userx = userResp.userList[0];
    }
}
```

In this example, given the user’s id we first get its corresponding instance (user_arrayx.userList[0]) with the method *get_user*. After that any changes are possible concerning user’s structure attributes, e.g. its fname. Finally the altered instance of the user structure is stored via *save_user* method.

## DELETE USER

The method that is used in order to delete an existing user from the System is called
Delete_user

```
UserR Delete_user(int user_id, string username, string password);
```

The user_id must be equal to the user that you want to delete. If the user_id is 0 the procedure is terminated and returns the appropriate error message. A User can be deleted only by an Administrator. The user’s credentials facilitate the recognition of the user’s privileges.

### EXAMPLE 1

```
int user_id = 5;
LLMWebService.UserR userx =  LLMWebServ.Delete_user(user_id,”username”,“password”);
```

In this example, given of the user’s id, the LLMWS *Delete_user* method deletes the corresponding user.

## GET USER/USERS

The method that is used in order to get an existing user / users from the System is called
get_user

```
UserR get_user(int user_id, string username, string password);
```

The user_id must be equal to the user that you want to get. If the user_id is 0 the all available Users are returned from the System. Every user can get only his/her data apart from the Administrator who is able to get all Users.

### EXAMPLE 1

```
int user_id = 3;
LLMWebService.UserR user_arrayx = LLMWebServ.get_user(user_id, “username”, “password”);

user_arrayx.userList[1].lname = "Smith"; //Set last name of second element of userList, which is a list of user structures.
Int err_code = user_arrayx.error.ErrorCode;//Get ErrorCode of structure error, which is contained in user_arrayx User structure.
```

As mentioned above, *get_user* method can provide a certain user or a list with the users available to the user requested them. In this example, a user with a certain id is requested and assigned to the array of user structures as the one and only element of it.

## EXAMPLE 2

```
int user_id = 0;
LLMWebService.UserR user_arrayx = LLMWebServ.get_user(user_id,”username”, “password”);
```

The example above represents the case of using *get_user* method to acquire an array with all the available users (only the administrator is allowed to perform this action). For this reason, user_id is assigned a zero value.

## MEMBER

### MEMBER'S STRUCTURE

```
<s:complexType name="userorsenior">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="usersenior_id" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="usercategory" type="tns:UserCategory"/>
    <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="password" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="role_id" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="role_name" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="Role" type="tns:UserRole"/>
  </s:sequence>
</s:complexType>
```

```
<s:complexType name="UserOrSeniorR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="userorseniorList" type="tns:ArrayOfUserorsenior"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| usersenior_id | The Unique Identifier of the User or The Senior | Is an integer that is equal to the user_id (if it is a user), equal to the senior_id (if it is a Senior) or 0 if there is not exist a Member with the provided credentials |
| usercategory | If it is a User or a Senior | Enumeration (UserCategory) |
| username | The username of The User or the Senior | String (max 30 characters) |
| password | The password of The User or the Senior | String (max 30 characters) |
| role_id | The role Id of the User (if it is a User) | The Id of the user's role as it is stored in the LLMDB |
| role_name | The role name in string format of the User (if it is a User) | String |
| Role | The Role of the User (if it is a User) | Enumeration (UserRole) |

### GET MEMBER

The method that is used in order to get an existing member from the System is called find_username_and_password

```
UserOrSeniorR find_username_and_password(string username, string password);
```

The method returns a structure of UserOrSeniorR. If there is not a user or senior with the provided credentials the appropriate error message is provided by the error object of the UserOrSeniorR structure. If the credentials belong to a senior then the usercategory attribute is Senior, else the usercategory attribute is User. In the case in which the usercategory attribute is User the Role attribute gives the User's Role (Administrator, Therapist, Relative, UnKnown).
Based on the returned structure, the programmer is able to get the User's or Senior's information by using the Get User or Get Senior according to the usercategory and the usersenior_id.

## EXAMPLE 1

LLMWebService. UserOrSenior userorseniorx = LLMWebServ.find_username_and_password(“username”, “password”);

In this example, method *find_username_and_password* is called and returns an instance of the userorsenior structure.

## USER SENIOR RELATIONSHIP

### USER – SENIOR RELATIONSHIP'S STRUCTURE

```
<s:complexType name="UserSeniorRelation">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="userrelationid" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="user_id" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="senior_id" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="IsCreator" type="s:boolean"/>
  </s:sequence>
</s:complexType>


<s:complexType name="UserSeniorRelationR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="userseniorrelationList" type="tns:ArrayOfUserseniorrelation"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| userrelationid | The Unique Identifier of the User or The Senior | The unique identifier of the user-senior relation (auto – generated by the LLMWS) |
| user_id | The User's user_id | User_id Integer |
| senior_id | The Senior's senior_id | senior_id Integer |
| IsCreator | If the User is the one who registered the Senior to the LLM System | True or False |

### ATTACH SENIOR TO USER

The method that is used in order to attach a Senior to a User is called
add_senior_to_user

```
UserSeniorRelationR add_senior_to_user(int user_id, int senior_id, string username, string password);
```

This method attaches a Senior to a User. If the User that provides thecredentials is an Administrator then he/she can attach any Senior to any User. This method is called internally when the method Add New Senior or Edit Existing Senior is called. In this case, it is automatically attaches the added or edited senior to the User (based on the Credentials)

#### EXAMPLE 1

```
int user_id = 22;
int senior_id = 18;
LLMWebServ. UserSeniorRelationR  userseniorrelationx = LLMWebServ. add_senior_to_user( user_id, senior_id,
“username”, “password”);
```

## DELETE SENIOR – USER RELATIONSHIP

The method that is used in order to delete an existing relationship between a user and a senior is called Delete_senior_relations

```
UserSeniorRelationR Delete_senior_relations(int user_id, int senior_id, string username, string password);
```

If the user recognised by the credentials is a Therapist or a Relative , then they can delete only their relationship to the Senior. If the User is an Administrator he/she can Delete any relationship or delete All relationships to a Senior by providing the senior_id and as a user_id 0.

### EXAMPLE 1

```
int user_id = 22;
int senior_id = 18;
LLMWebServ.UserSeniorRelationR userseniorrelationx  = LLMWebServ. Delete_senior_relations ( user_id, senior_id,
“username”, “password”);
```

## LLM COMPONENTS AND ACTIVITIES

Each component, like ILC, CTC and PTC, supports a finite predefined number of activities. These well-described (by the component's providers) activities behave as the elementary entities of the Seniors Interaction. For example, the movement activity in the kitchen may be named "KitchenActivity1", the movement activity in the bedroom may be named "BedroomActivity1". These two activities are different entities. Another example may include the procedure of software that shows to the senior 3 screens (with different multimedia material each time). This may be a CTC Activity. Another activity may be a procedure of software that shows to the senior 4 screens (with different multimedia material each time). In conclusion, activity is a set of exercises that is treated by the LLM System as an Activity. The activity is performed by a Senior and a score for this activity is generated
The LLM System supports a finite number of activities. This number stems from the activities that are supported by the CTC, the ILC and the PTC. These activities are provided by the providers of the components and added to the LLM System once. If a new Activity is available, the administrator is responsible for adding it as a supported activity of the LLM System.

The term Component is used in order to describe the system that is used by the ILC, CTC and PTC. For example, The Components of the PTC may be a treadmill er100 model, a treadmill er200 model, a Balanced Board, a Wiimote, etc. Each of the above activities may be available to more than one Components. For example, the "walking" activity may take place at the treadmill er100 model or the treadmill er200 model. If a new Component is available, the administrator is responsible for adding it as a supported Component of the LLM System.

## INTEPENDENT LIVING COMPONENT ILC

### ILC ACTIVITY'S STRUCTURE

```
<s:complexType name="ILCActivity">
   <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int"/>
      <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string"/>
      <s:element minOccurs="0" maxOccurs="1" name="Units" type="s:string"/>
   </s:sequence>
</s:complexType>

<s:complexType name="ILCACTIVITYR">
   <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="ILCActivityList" type="tns:ArrayOfILCActivity"/>
      <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
   </s:sequence>
</s:complexType>
```

| Attribute | Description | Type |
|---|---|---|
| ID | ILC Activity ID | The unique identifier of the ILC Activity |
| Name | ILC Activity Name | String |
| Units | The ILC activity progress Units | String |

The ILC Activity is an integrated well-described activity that a Senior can achieve. For example, the movement activity in the kitchen may be named "KitchenActivity1", the movement activity in the bedroom may be named "BedroomActivity1". These two activities are different entities.

### ADD ILC ACTIVITY

The method that is used in order to add a supported by the system ILC Activity called
AddILCActivity

```
ILCACTIVITYR AddILCActivity(ILCActivity ilcactx, string username, string password);
```

This method adds an ILC Activity as supported by the LLM System. Only an administrator has the privilege to use this method. This method is described in this document for development purposes only and will be absent in the final version of the LLMDR.

#### EXAMPLE 1

```
LLMWebService.ILCActivity ilcactx = new TestingLLMWebService.LLMWebService.ILCActivity();
ilcactx.ID = 0;
ilcactx.Name = "BedroomActivity1";
ilcactx.Units = "seconds";
LLMWebService.ILCACTIVITYR ilcactivity = LLMWebServ.AddILCActivity(ilcactx, "username", "password");
if (ilcactivity.error.ErrorCode == 0)//access to error's structure ErrorCode attribute
        ilcactx = ilcactivity.ILCActivityList[0]; //get the first element of ILCActivityList, which is a ILCActivity structure.
```

In the above example, a new instance of ILCActivity is created. Then each attribute of the structure is assigned a value. Finally, via the web service method *AddILCActivity*, the new instance is being stored into the LLMDB. It must be stressed out that the method returns an ILCACTIVITYR structure with an array (one element) of the new ILCActivity, in case of successful addition.

## GET ILC ACTIVITY

The method that is used in order to get all supported by the system ILC Activities called GetILCActivity

```
ILCACTIVITYR GetILCActivity(int ilcactivity_id, string username, string password);
```

This method gets the specified by the “ilcactivity_id” ILCActivity or all the available ILC Activities that are supported by the LLM System.Every Member (User or Senior) has the privilege to use this Method.

### EXAMPLE 1

```
int ilcact_id = 0;
LLMWebService.ILCACTIVITYR ilcactivity = LLMWebServ.GetILCActivity(ilcact_id, “username”,”password”);
```

The above sample of code represents the case of getting all the supported by the system ILCActivities, using the *GetILCActivity* method. It must be stressed out that “ilcactivity_id” is set to zero, thus indicating the fact that the method will return as a result all the available ILCActivities.

### EXAMPLE 2

```
int ilcact_id = 3;
LLMWebService.ILCACTIVITYR ilcactivity = LLMWebServ.GetILCActivity(ilcact_id, “username”,”password”);
String ilc_activity_name = ilcactivity.ILCActivityList[0].Name; //Access Name attribute of first element of ILCActivityList
String error_message_name = ilcactivity.error.ErrorMessage;//Access ErrorMessage(Name) attribute of error structure
```

The above sample of code represents the case of getting a specific ILCActivity corresponding to the provided “ilcactivity_id” , using the *GetILCActivity* method.

## ILC COMPONENT’S STRUCTURE

```
<s:complexType name="ILCComponent">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="Comments" type="s:string"/>
  </s:sequence>
</s:complexType>

<s:complexType name="ILCCOMPONENTR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="ILCComponentList" type="tns:ArrayOfILCComponent"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| Attribute | Description | Type |
|---|---|---|
| ID | ILC Component ID | The unique identifier of the ILC Component |
| Name | ILC Component Name | String |
| Comments | The ILC Component Description | String |

## ADD ILC COMPONENT

The method that is used in order to add a supported by the system ILC Component called AddILCComponent

```
ILCCOMPONENTR AddILCComponent(ILCComponent ilccmptx, string username, string password);
```

This method adds an ILC Component as supported by the LLM System. Only an administrator has the privilege to use this method. This method is described in this document for development purposes only and will be absent in the final version of the LLMDR.

### EXAMPLE 1

```
LLMWebService.ILCComponent ilccmptx = new TestingLLMWebService.LLMWebService.ILCComponent();
ilccmptx.ID = 0;
ilccmptx.Name = "AAL";
ilccmptx.Comments = "ALTEC";
LLMWebService.ILCCOMPONENTR ilccomp = LLMWebServ.AddILCComponent(ilccmptx,"username", "password");
if (ilccomp.error.ErrorCode == 0) //Access to ErrorCode attribute of error structure which is contained in
                                              ILCCOMPONENT
   ilccmptx = ilccomp.ILCComponentList[0];
```

First, a new instance of ILCComponent structure is created. After that, each attribute of the structure is assigned a value and the whole structure is passed as a parameter to the *AddILCComponent* method of the LLMWS. Finally the method returns as a result an ILCCOMPONENTR structure with an array (one element) of the new ILCComponent, in case of successful addition.

## GET ILC COMPONENT

The method that is used in order to get all supported by the system ILC Components called GetILCComponent

```
ILCCOMPONENTR GetILCComponent(int ilccomponent_id, string username, string password);
```

This method gets the specified by the "ilccomponent _id" or all the available ILC Components that are supported by the LLM System.Every Member (User or Senior) has the privilege to use this Method.

### EXAMPLE 1

```
int ilccmp_id = 0;
LLMWebService.ILCCOMPONENTR ilccomp = LLMWebServ.GetILCComponent(ilccmp_id, "username", "password");
```

The above sample of code represents the case of getting all the supported by the system ILCComponents, using the *GetILCComponent* method. It must be stressed out that “ilccomponent_id” is set to zero, thus indicating the fact that the method will return as a result, the available ILCComponents.

### EXAMPLE 2

```
int ilccmp_id = 2;
LLMWebService.ILCCOMPONENTR ilccomp = LLMWebServ.GetILCComponent(ilccmp_id, “username”, “password”);
String provider_name = ilccomp.ILCComponentList[0].Comments;//Get Comments attribute of first element of
                                                                ILCComponentList
int error_code = ilccomp.error.ErrorCode;//Get ErrorCode attribute o ferror structured which is contained in ilccomp
                                           ILCCOMPONENT structure.
```

The above sample of code represents the case of getting a specific ILCComponent corresponding to the provided “ilccomponent_id” , using the *GetILCComponent* method.

## ILC SENIOR’S ACTIVITY STRUCTURE

```
<s:complexType name="ILCSeniorActivity">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="senior_id" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="seniorslname" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="seniorsfname" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="ilcactivityid" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="ilcactivityname" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="ilcid" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="ilcname" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="datetimestart" type="s:dateTime"/>
    <s:element minOccurs="1" maxOccurs="1" name="datetimeend" type="s:dateTime"/>
    <s:element minOccurs="0" maxOccurs="1" name="score" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="level" type="s:int"/>
  </s:sequence>
</s:complexType>
```

```
<s:complexType name="ILCSENIORACTIVITYR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="ILCSeniorActivityList" type="tns:ArrayOfILCSeniorActivity"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| ID | ILC Senior Activity ID | The unique identifier of the ILC Senior Activity |
| senior_id | Senior ID | The unique identifier of the Senior |
| seniorslname | Senior’s Last Name | String |
| seniorsfname | Senior’s First Name | String |
| ilcactivityid | ILC Activity ID | The unique identifier of the ILC Activity |
| ilcactivityname | ILC Activity Name | String |

| | | |
|---|---|---|
| ilcid | ILC Component ID | The unique identifier of the ILC Component |
| ilcname | ILC Component Name | String |
| datetimestart | Date Time Activity Started | Datetime |
| datetimeend | Date Time Activity Ended | Datetime |
| score | The score achieved by the Senior | String |
| level | The Difficulty Level of the Activity | Integer |

## ADD ILC SENIOR'S ACTIVITY

The method that is used in order to add an activity that is performed by a senior and the score achieved is called AddILCSeniorActivity

```
ILCSENIORACTIVITYR AddILCSeniorActivity(ILCSeniorActivity ilcsenacttx, string username, string password);
```

This method adds an activity that is performed by a senior. It behaves as a log file of the senior's progress and activities. The obligated fields that must be provided are:
senior_id: The senior that performs the activity (see Get Senior/Seniors)
ilcactivityid: The ILC Activity that is performed by the Senior (see Get ILC Activity)
ilcid: The ILC Component that is used by the senior in order to perform the Activity (see Get ILC Component)
datetimestart: The date and time that the activity started
datetimeend: The date and time that the activity ended
score: The score that achieved by the Senior (The Units of these are described in the Activity, see Get ILC Activity)
level: The level of difficulty of the performed ILC Activity (e.g. 1 - 10)

A senior has the privilege to use this method only for himself, the Therapist for all his/her Seniors and an Administrator for all the Seniors. The common scenario is the Senior to provide his activities (by LogIn to the LLM System).

### EXAMPLE 1

```
LLMWebService.ILCSeniorActivity ilcactx = new TestingLLMWebService.LLMWebService.ILCSeniorActivity();
ilcactx.ID = 0;
ilcactx.senior_id = 18;
ilcactx.ilcactivityid = 1;
ilcactx.ilcid = 1;
ilcactx.datetimestart = new DateTime(2009, 6, 21,10,00,00);
ilcactx.datetimeend = new DateTime(2009, 6, 21, 11, 00, 00);
ilcactx.score = "good";
ilcactx.level = 2;
LLMWebService.ILCSENIORACTIVITYR ilcsenact = LLMWebServ.AddILCSeniorActivity(ilcactx, “username”, “password”);
if (ilcsenact.error.ErrorCode == 0)
  ilcactx = ilcsenact.ILCSeniorActivityList[0];//Get first ILCSeniorActivity from the ILCSeniorActivityList.
```

A new instance of ILCSeniorActivity is created and each attribute of it is assigned with a valid value. Finally, the *AddILCSeniorActivity* method stores the new instance of ILCSeniorActivity.

## GET ILC SENIOR'S ACTIVITY

The method that is used in order to get all performed ILC Activities by a senior called GetILCSeniorActivity

```
ILCSENIORACTIVITYR GetILCSeniorActivity(int ilcsenioractivity_id, int senior_id, int ilcactivity_id, int ilc_id, int user_id, string username, string password);
```

This method gets all the performed ILC Senior's Activities (Log) based on the following criterias:
ilcsenioractivity_id: if it is 0 then the criteria is not used

senior_id: if it is 0 then the criteria is not used, else the method results include only for the specified senior.
ilcactivity_id: if it is 0 then the criteria is not used, else the method results include only for the specified activity.
ilc_id: if it is 0 then the criteria is not used, else the method results include only for the specified ilc component.
user_id: if it is 0 then the criteria is not used, else the method results include only for the specified ilc component.

If the provided credentials belong to a Senior, the results include only his/her performance. If the provided credentials belong to a Therapist, the results include only his/her Seniors performance. If the User is an Administrator he/she has the privilege to get all Seniors' performance.

### EXAMPLE 1

```
LLMWebService.ILCSENIORACTIVITYR ilcsenact = LLMWebServ.GetILCSeniorActivity(0,0,0,0,0, "username",
"password");
String senior_score = ilcsenact.ILCSeniorActivityList[0].score;//Get the score attribute of ILCSeniorActivity, which is the
                                                                   first element of the ILCSeniorActivityList contained in
                                                                   ilcsenact ILCSENIORACTIVITY structure returned by the
                                                                   method.
```

The above example represents the case of getting all the supported by the system ILCSeniorActivities, using the *GetILCSeniorActivity* method. It must be stressed out that "ilcsenioractivity_id", "senior_id", "ilcactivity_id", "ilc_id", "user_id", are set to zero, thus indicating the fact that the method will return as a result, the whole set of ILCSeniorActivities.

### EXAMPLE 2

```
int ilcsenioract_id = 4;
LLMWebService.ILCSENIORACTIVITYR ilcsenact = LLMWebServ.GetILCSeniorActivity(ilcsenioract_id,0,0,0,0, "username",
"password");
```

In this example, a specific ILCSeniorActivity is returned, passing its unique "ilcsenioractivity_id" as a parameter to the *GetILCSeniorActivity* method.

### EXAMPLE 3

```
int ilcsenior_id = 16;
LLMWebService.ILCSENIORACTIVITYR ilcsenact = LLMWebServ.GetILCSeniorActivity(0, ilcsenior_id, 0, 0, 0, "username",
"password");
```

The above example represents the case that the *GetILCSeniorActivity* method returns a list of ILCSeniorActivities that are performed by a specific senior, given his/her unique id. . Following the same convention, a user can retrieve e.g. all the results achieved by all seniors using all kind of ILC equipment concerning a specific activity such as "moving in living room" for example. This is accomplished by setting ilcactivity_id equal to the corresponding id of "moving in living room" activity and concurrently by setting all other arguments, except for UserName and Password, to zero.

## COGNITIVE TRAINING COMPONENT CTC

### CTC ACTIVITY'S STRUCTURE

```
<s:complexType name="CTCActivity">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="Units" type="s:string"/>
  </s:sequence>
</s:complexType>

<s:complexType name="CTCACTIVITYR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="CTCActivityList" type="tns:ArrayOfCTCActivity"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| ID | CTC Activity ID | The unique identifier of the CTC Activity |
| Name | CTC Activity Name | String |
| Units | The CTC activity progress Units | String |

The CTC Activity is an integrated well-described activity that a Senior can achieve. For example, the procedure of software that shows to the senior 3 screens (with different multimedia material each time). This may be a CTC Activity. Another activity may be a procedure of software that shows to the senior 4 screens (with different multimedia material each time). These two activities are different entities.

## ADD CTC ACTIVITY

The method that is used in order to add a supported by the system CTC Activity called AddCTCActivity

```
CTCACTIVITYR AddCTCActivity(CTCActivity ctcactx, string username, string password);
```

This method adds a CTC Activity as supported by the LLM System. Only an administrator has the privilege to use this method. This is method is described in this document for the purpose of development only and will be absent in the final version of the LLMDR.

### EXAMPLE 1

```
LLMWebService.CTCActivity ctcactx = new TestingLLMWebService.LLMWebService.CTCActivity();
ctcactx.ID = 0;
ctcactx.Name = "MemoryTest";
ctcactx.Units = "percent";
LLMWebService.CTCACTIVITYR ctcactivity = LLMWebServ.AddCTCActivity(ctcactx, “username”, “password”);
if (ctcactivity.error.ErrorCode == 0)//Get ErrorCode attribute of error strucure contained in ctcactivity CTCACTIVITY
                                   structure
  ctcactx = ctcactivity.CTCActivityList[0];
```

In the above example, a new instance of CTCActivity is created. Then each attribute of the structure is assigned a value. Finally, via the web service method *AddCTCActivity*, the new instance is being stored into the LLMDB. It must be stressed out that the method returns an CTCACTIVITYR structure with an array (one element) of the new CTCActivity, in case of successful addition.

## GET CTC ACTIVITY

The method that is used in order to get all supported by the system ILC Activities called GetCTCActivity

```
CTCACTIVITYR GetCTCActivity(int ctcactivity_id, string username, string password);
```

This method gets the specified by the “ctcactivity_id” or all the available CTC Activities that are supported by the LLM System.Every Member (User or Senior) has the privilege to use this Method.

### EXAMPLE 1

```
int ctcact_id = 0;
LLMWebService.CTCACTIVITYR ctcactivity = LLMWebServ.GetCTCActivity(ctcact_id, “username”, “password”);
String measure = ctcactivity.CTCActivityList[0].Units;//Get Units attribute of CTCActivity structure (CTCActivityList[0]),
                                                which is contained in CTCACTIVITY structure ctcactivity.
```

The above sample of code represents the case of getting all the supported by the system CTCActivities, using the *GetCTCActivity* method. It must be stressed out that “ctcactivity_id” is set to zero, thus indicating the fact that the method will return as a result all the available CTCActivities.

### EXAMPLE 2

```
int ctcact_id = 2;
LLMWebService.CTCACTIVITYR ctcactivity = LLMWebServ.GetCTCActivity(ctcact_id, “username”, “password”);
```

The above sample of code represents the case of getting a specific CTCActivity corresponding to the provided “ctcactivity_id” , using the *GetCTCActivity* method.

## CTC COMPONENT’S STRUCTURE

```
<s:complexType name="CTCComponent">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="Comments" type="s:string"/>
  </s:sequence>
</s:complexType>

<s:complexType name="CTCCOMPONENTR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="CTCComponentList" type="tns:ArrayOfCTCComponent"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| ID | CTC Component ID | The unique identifier of the CTC Component |
| Name | CTCComponent Name | String |
| Comments | The CTC Component Description | String |

## ADD CTC COMPONENT

The method that is used in order to add a supported by the system ILC Component called AddCTCComponent

```
CTCCOMPONENTR AddCTCComponent(CTCComponent ctccmptx, string username, string password);
```

This method adds an CTC Component as supported by the LLM System. Only an administrator has the privilege to use this method. This is method is described in this document for the purpose of development only and will be absent in the final version of the LLMDR.

### EXAMPLE 1

```
LLMWebService.CTCComponent ctccmptx = new LLMWebService.CTCComponent();
ctccmptx.ID = 0;
ctccmptx.Name = "BrainAge";
ctccmptx.Comments = "Nintendo";
LLMWebService.CTCCOMPONENTR ctccomp = LLMWebServ.AddCTCComponent(ctccmptx, “username”, “password”);
if (ctccomp.error.ErrorCode == 0)//Get ErrorCode attribute of error structure contained in CTCCOMPONENT structure
                                            ctccomp.
  ctccmptx = ctccomp.CTCComponentList[0];
```

First, a new instance of CTCComponent structure is created. After that, each attribute of the structure is assigned a value and the whole structure is passed as a parameter to the *AddCTCComponent* method of the LLMWS. Finally the method returns as a result an CTCCOMPONENTR structure with an array (one element) of the new CTCComponent, in case of successful addition.

## GET CTC COMPONENT

The method that is used in order to get all supported by the system ILC Components called GetCTCComponent

```
CTCCOMPONENTR GetCTCComponent(int ctccomponent_id, string username, string password);
```

This method gets the specified by the “ctccomponent _id” or all the available CTCComponents that are supported by the LLM System.Every Member (User or Senior) has the privilege to use this Method.

### EXAMPLE 1

```
int ctccmp_id = 0;
LLMWebService.CTCCOMPONENTR ctccomp = LLMWebServ.GetCTCComponent(ctccmp_id, “username”, “password”);
int ctc_component_id = ctccomp.CTCComponentList[0].ID;//Get ID attribute of CTCComponent structure, which is the
                                                        first element of CTCComponentList attribute of ctccomp
                                                        structure.
```

The above sample of code represents the case of getting all the supported by the system CTCComponents, using the *GetCTCComponent* method. It must be stressed out that “ctccomponent_id” is set to zero, thus indicating the fact that the method will return as a result, the available CTCComponents

### EXAMPLE 2

```
int ctccmp_id = 1;
LLMWebService.CTCCOMPONENTR ctccomp = LLMWebServ.GetCTCComponent(ctccmp_id, “username”, “password”);
```

The above sample of code represents the case of getting a specific CTCComponent corresponding to the provided “ctccomponent_id” , using the *GetCTCComponent* method.

## CTC SENIOR’S ACTIVITY STRUCTURE

```
<s:complexType name="CTCSeniorActivity">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="senior_id" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="seniorslname" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="seniorsfname" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="ctcactivityid" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="ctcactivityname" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="ctcid" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="ctcname" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="datetimestart" type="s:dateTime"/>
    <s:element minOccurs="1" maxOccurs="1" name="datetimeend" type="s:dateTime"/>
    <s:element minOccurs="0" maxOccurs="1" name="score" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="level" type="s:int"/>
  </s:sequence>
</s:complexType>
```

```
<s:complexType name="CTCSENIORACTIVITYR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="CTCSeniorActivityList" type="tns:ArrayOfCTCSeniorActivity"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| ID | ILC Senior Activity ID | The unique identifier of the ILC Senior Activity |
| senior_id | Senior ID | The unique identifier of the Senior |
| seniorslname | Senior’s Last Name | String |
| seniorsfname | Senior’s First Name | String |
| ctcactivityid | CTC Activity ID | The unique identifier of the CTC Activity |
| ctcactivityname | CTC Activity Name | String |
| ctcid | CTC Component ID | The unique identifier of the CTC Component |
| ctcname | CTC Component Name | String |
| datetimestart | Date Time Activity Started | Datetime |
| datetimeend | Date Time Activity Ended | Datetime |
| score | The score achieved by the Senior | String |
| level | The Difficulty Level of the Activity | Integer |

## ADD CTC SENIOR’S ACTIVITY

The method that is used in order to add an activity that performed of a senior and the score achieved is called AddCTCSeniorActivity

```
CTCSENIORACTIVITYR AddCTCSeniorActivity(CTCSeniorActivity ctcsenacttx, string username, string password);
```

This method add an activity that is performed by a senior. It behaves as a log file of the senior’s progress and activities. The obligated fields that must be provided are:
senior_id: The senior that performs the activity (see Get Senior/Seniors)
ctcactivityid: The CTC Activity that is performed by the Senior (see Get CTC Activity)
ctcid: The CTC Component that is used by the senior in order to perform the Activity (see Get CTC Component)
datetimestart: The date and time that the activity started
datetimeend: The date and time that the activity ended
score: The scored that achieved by the Senior (The Units of these are described in the Activity, see Get CTC Activity)
level: The level of difficulty of the performed CTC Activity

A senior has the privilege to use this method only for himself, the Therapist for all his/her Seniors and an Administrator for all the Seniors. The common scenario is the Senior to provide his activities (by LogIn to the LLM System).

### EXAMPLE 1

```
LLMWebService.CTCSeniorActivity ctcactx = new TestingLLMWebService.LLMWebService.CTCSeniorActivity();
ctcactx.ID = 0;
ctcactx.senior_id = 16;
ctcactx.ctcactivityid = 1;
ctcactx.ctcid = 1;
ctcactx.datetimestart = new DateTime(2009, 6, 21, 10, 00, 00);
ctcactx.datetimeend = new DateTime(2009, 6, 21, 11, 00, 00);
ctcactx.score = "16";
ctcactx.level = 8;
LLMWebService.CTCSENIORACTIVITYR ctcsenact = LLMWebServ.AddCTCSeniorActivity(ctcactx, “username”,
“password”);
if (ctcsenact.error.ErrorCode == 0)//Get ErrorCode attribute of error structure which is contained in ctcsenact
                                      CTCSENIORACTIVITY structure.
  ctcactx = ctcsenact.CTCSeniorActivityList[0];
```

A new instance of CTCSeniorActivity is created and each attribute of it is assigned with a valid value. Finally, the *AddCTCSeniorActivity* method stores the new instance of CTCSeniorActivity.

## GET CTC SENIOR’S ACTIVITY

The method that is used in order to get all performed ILC Activities by a senior called GetCTCSeniorActivity

```
CTCSENIORACTIVITYR GetCTCSeniorActivity(int ctcsenioractivity_id, int senior_id, int ctcactivity_id, int ctc_id, int
user_id, string username, string password);
```

This method gets all the performed CTC Senior’s Activities (Log) based on the following criterias:
ctcsenioractivity_id: if it is 0 then the criteria is not used
senior_id: if it is 0 then the criteria is not used, else the method results include only for the specified senior.
ctcactivity_id: if it is 0 then the criteria is not used, else the method results include only for the specified activity.
ctc_id: if it is 0 then the criteria is not used, else the method results include only for the specified ilc component.

user_id: if it is 0 then the criteria is not used, else the method results include only for the specified ilc component.

If the provided credentials belong to a Senior, the results include only his/her performance. If the provided credentials belong to a Therapist, the results include only his/her Seniors performance. If the User is an Administrator he/she has the privilege to get all Seniors' performance.

### EXAMPLE 1

```
LLMWebService.CTCSENIORACTIVITYR ctcsenact = LLMWebServ.GetCTCSeniorActivity(0, 0, 0, 0, 0, "UserName",
"PassWord");
DateTime start_of_activity = ctcsenact.CTCSeniorActivityList[0].datetimestart;//Get date and time of senior's start of
                                                                                first activity stored in
                                                                                CTCSeniorActivityList
```

The above example represents the case of getting all the supported by the system CTCSeniorActivities, using the *GetCTCSeniorActivity* method. It must be stressed out that "ctcsenioractivity_id", "senior_id", "ctcactivity_id", "ctc_id", "user_id", are set to zero, thus indicating the fact that the method will return as a result, the whole set of CTCSeniorActivities.

### EXAMPLE 2

```
int ctcsenioract_id = 2;
LLMWebService. CTCSENIORACTIVITYR ctcsenact = LLMWebServ.GetCTCSeniorActivity(ctcsenioract_id, 0, 0, 0, 0,
"username", "password");
```

In this example, a specific CTCSeniorActivity is returned, passing its unique "ctcsenioractivity_id" as a parameter to the *GetCTCSeniorActivity* method.

### EXAMPLE 3

```
int ctcsenior_id = 16;
LLMWebService. CTCSENIORACTIVITYR ctcsenact = LLMWebServ.GetCTCSeniorActivity(0, ctcsenior_id, 0, 0, 0,
"username", "password");
```

The above example represents the case that the *GetCTCSeniorActivity* method returns a list of CTCSeniorActivities that are performed by a specific senior, given his/her unique id. Following the same convention, a user can retrieve e.g. all the results achieved by all seniors using all kind of CTC software concerning a specific task such as memory. This is accomplished by setting ctcactivity_id equal to the corresponding id of memory task and concurrently by setting all other arguments, except for UserName and Password, to zero.

## PHYSICAL TRAINING COMPONENT PTC

### PTC ACTIVITY'S STRUCTURE

```
<s:complexType name="PTCActivity">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="Units" type="s:string"/>
  </s:sequence>
</s:complexType>


<s:complexType name="PTCACTIVITYR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="PTCActivityList" type="tns:ArrayOfPTCActivity"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| ID | PTC Activity ID | The unique identifier of the PTC Activity |
| Name | PTC Activity Name | String |
| Units | The PTC activity progress Units | String |

The PTC Activity is an integrated well-described activity that a Senior can achieve.

## ADD PTC ACTIVITY

The method that is used in order to add a supported by the system CTC Activity called
AddPTCActivity

```
PTCACTIVITYR AddPTCActivity(PTCActivity ptcactx, string username, string password);
```

This method adds a PTC Activity as supported by the LLM System. Only an administrator has the privilege to use this method. This is method is described in this document for the purpose of development only and will be absent in the final version of the LLMDR.

### EXAMPLE 1

```
LLMWebService.PTCActivity ptcactx = new LLMWebService.PTCActivity();
ptcactx.ID = 0;
ptcactx.Name = "flexibility";
ptcactx.Units = "percent";
LLMWebService.PTCACTIVITYR ptcactivity = LLMWebServ.AddPTCActivity(ptcactx, “username”, “password”);
if (ptcactivity.error.ErrorCode == 0)//Get ErrorCode attribute of error structure, which is contained in PTCACTIVITY
                                     structure
  ptcactx = ptcactivity.PTCActivityList[0];
```

In the current example, a new instance of PTCActivity is created. Then each attribute of the structure is assigned a value. Finally, via the web service method *AddPTCActivity*, the new instance is being stored into the LLMDB. It must be stressed out that the method returns an PTCACTIVITYR structure with an array (one element) of the new PTCActivity, in case of successful addition.

## GET PTC ACTIVITY

The method that is used in order to get all supported by the system PTC Activities called
GetPTCActivity

```
PTCACTIVITYR GetPTCActivity(int ptcactivity_id, string username, string password);
```

This method gets the specified by the “ptcactivity_id” or all the available PTC Activities that are supported by the LLM System.Every Member (User or Senior) has the privilege to use this Method.

### EXAMPLE 1

```
int ptcact_id = 0;
LLMWebService.PTCACTIVITYR ptcactivity = LLMWebServ.GetPTCActivity(ptcact_id, “username”, “password”);
String ptc_activity_name = ptcactivity.PTCActivityList[2].Name;//Get Name attribute of PTCActivity structure
                                                                 (PTCActivityList[2]), contained in PTCACTIVITY
                                                                 structure ptcactivity.
```

The above sample of code represents the case of getting all the supported by the system PTCActivities, using the *GetPTCActivity* method. It must be stressed out that “ptcactivity_id” is set to zero, thus indicating the fact that the method will return as a result all the available PTCActivities.

### EXAMPLE 2

```
int ptcact_id = 3;
LLMWebService.PTCACTIVITYR ptcactivity = LLMWebServ.GetPTCActivity(ptcact_id, “username”, “password”);
```

The above sample of code represents the case of getting a specific PTCActivity corresponding to the provided “ptcactivity_id” , using the *GetPTCActivity* method.

## PTC COMPONENT’S STRUCTURE

```
<s:complexType name="PTCComponent">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="Name" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="Comments" type="s:string"/>
  </s:sequence>
</s:complexType>


<s:complexType name="PTCCOMPONENTR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="PTCComponentList" type="tns:ArrayOfPTCComponent"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| ID | PTC Component ID | The unique identifier of the PTC Component |
| Name | PTCComponent Name | String |
| Comments | The PTC Component Description | String |

## ADD PTC COMPONENT

The method that is used in order to add a supported by the system ILC Component called AddPTCComponent

```
PTCCOMPONENTR AddPTCComponent(PTCComponent ptccmptx, string username, string password);
```

This method adds an PTC Component as supported by the LLM System. Only an administrator has the privilege to use this method. This is method is described in this document for the purpose of development only and will be absent in the final version of the LLMDR.

### EXAMPLE 1

```
LLMWebService.PTCComponent ptccmptx = new LLMWebService.PTCComponent();
ptccmptx.ID = 0;
ptccmptx.Name = "TmCardio";
ptccmptx.Comments = "Techmed";
LLMWebService.PTCCOMPONENTR ptccomp = LLMWebServ.AddPTCComponent(ptccmptx, “username”, “password”);
if (ptccomp.error.ErrorCode == 0)// Get ErrorCode attribute of error structure, which is contained in PTCCOMPONENT
                                  structure

 ptccmptx = ptccomp.PTCComponentList[0];
```

First, a new instance of PTCComponent structure is created. After that, each attribute of the structure is assigned a value and the whole structure is passed as a parameter to the *AddPTCComponent* method of the LLMWS. Finally the method returns as a result a PTCCOMPONENTR structure with an array (one element) of the new PTCComponent, in case of successful addition.

## GET PTC COMPONENT

The method that is used in order to get all supported by the system PTC Components called GetPTCComponent

```
PTCCOMPONENTR GetPTCComponent(int ptccomponent_id, string username, string password);
```

This method gets the specified by the “ptccomponent _id” or all the available PTCComponents that are supported by the LLM System.Every Member (User or Senior) has the privilege to use this Method.

### EXAMPLE 1

```
int ptccmp_id = 0;
LLMWebService.PTCCOMPONENTR ptccomp = LLMWebServ.GetPTCComponent(ptccmp_id, “username”, “password”);
```

The above sample of code represents the case of getting all the supported by the system PTCComponents, using the *GetPTCComponent* method. It must be stressed out that “ptccomponent_id” is set to zero, thus indicating the fact that the method will return as a result, the available PTCComponents.

### EXAMPLE 2

```
int ptccmp_id = 2;
LLMWebService. PTCCOMPONENTR ptccomp = LLMWebServ.GetPTCComponent(ptccmp_id, “username”, “password”);
String ptc_component_name = ptccomp.PTCComponentList[0].Name;//Get Name attribute of PTCComponent structure
                                                           (PTCComponentList[0]), the first element of
```

PTCComponentList, contained in ptccomp PTCCOMPONENT structure.

The above sample of code represents the case of getting a specific PTCComponent corresponding to the provided "ptccomponent_id" , using the *GetPTCComponent* method.

## PTC SENIOR'S ACTIVITY STRUCTURE

```
<s:complexType name="PTCSeniorActivity">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="ID" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="senior_id" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="seniorslname" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="seniorsfname" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="ptcactivityid" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="ptcactivityname" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="ptcid" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="ptcname" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="datetimestart" type="s:dateTime"/>
    <s:element minOccurs="1" maxOccurs="1" name="datetimeend" type="s:dateTime"/>
    <s:element minOccurs="0" maxOccurs="1" name="score" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="level" type="s:int"/>
  </s:sequence>
</s:complexType>
```

```
<s:complexType name="PTCSENIORACTIVITYR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="PTCSeniorActivityList" type="tns:ArrayOfPTCSeniorActivity"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| ID | ILC Senior Activity ID | The unique identifier of the ILC Senior Activity |
| senior_id | Senior ID | The unique identifier of the Senior |
| seniorslname | Senior's Last Name | String |
| seniorsfname | Senior's First Name | String |
| ptcactivityid | PTC Activity ID | The unique identifier of the PTC Activity |
| ptcactivityname | PTC Activity Name | String |
| ptcid | PTC Component ID | The unique identifier of the PTC Component |
| ptcname | PTC Component Name | String |
| datetimestart | Date Time Activity Started | Datetime |
| datetimeend | Date Time Activity Ended | Datetime |
| score | The score achieved by the Senior | String |
| level | The Difficulty Level of the Activity | Integer |

## ADD PTC SENIOR'S ACTIVITY

The method that is used in order to add an activity that performed of a senior and the score achieved is called AddPTCSeniorActivity

```
PTCSENIORACTIVITYR AddPTCSeniorActivity (PTCSeniorActivity ptcsenacttx, string username, string password);
```

This method add an activity that is performed by a senior. It behaves as a log file of the senior's progress and activities. The obligated fields that must be provided are:
senior_id: The senior that performs the activity (see Get Senior/Seniors)
ptcactivityid: The PTC Activity that is performed by the Senior (see Get PTC Activity)
ptcid: The PTC Component that is used by the senior in order to perform the Activity (see Get PTC Component)
datetimestart: The date and time that the activity started
datetimeend: The date and time that the activity ended
score: The scored that achieved by the Senior (The Units of these are described in the Activity, see Get PTC Activity)
level: The level of difficulty of the performed PTC Activity

A senior has the privilege to use this method only for himself, the Therapist for all his/her Seniors and an Administrator for all the Seniors. The common usage is the Senior to provide his activities (by LogIn to the LLM System).

### EXAMPLE 1

```
LLMWebService.PTCSeniorActivity ptcactx = new LLMWebService.PTCSeniorActivity();
ptcactx.ID = 0;
ptcactx.senior_id = 16;
ptcactx.ptcactivityid = 1;
ptcactx.ptcid = 1;
ptcactx.datetimestart = new DateTime(2009, 6, 21, 10, 00, 00);
ptcactx.datetimeend = new DateTime(2009, 6, 21, 11, 00, 00);
ptcactx.score = "5";
ptcactx.level = 2;
LLMWebService.PTCSENIORACTIVITYR ptcsenact = LLMWebServ.AddPTCSeniorActivity(ptcactx, "username",
"password");
if (ptcsenact.error.ErrorCode == 0)//Get ErrorCode attribute of error structure contained in ptcsenact
                                                PTCSENIORACTIVITY structure.
 ptcactx = ptcsenact.PTCSeniorActivityList[0];
```

A new instance of PTCSeniorActivity is created and each attribute of it is assigned with a valid value. Finally, the *AddPTCSeniorActivity* method stores the new instance of PTCSeniorActivity.

## GET PTC SENIOR'S ACTIVITY

The method that is used in order to get all performed ILC Activities by a senior called GetPTCSeniorActivity

```
PTCSENIORACTIVITYR GetPTCSeniorActivity(int ptcsenioractivity_id, int senior_id, int ptcactivity_id, int ptc_id, int
user_id, string username, string password);
```

This method gets all the performed PTC Senior's Activities (Log) based on the following criterias:
ptcsenioractivity_id: if it is 0 then the criteria is not used
senior_id: if it is 0 then the criteria is not used, else the method results include only for the specified senior.
ptcactivity_id: if it is 0 then the criteria is not used, else the method results include only for the specified activity.
ptc_id: if it is 0 then the criteria is not used, else the method results include only for the specified ilc component.
user_id: if it is 0 then the criteria is not used, else the method results include only for the specified ilc component.

If the provided credentials belong to a Senior, the results include only his/her performance. If the provided credentials belong to a Therapist, the results include only his/her Seniors performance. If the User is an Administrator he/she has the privilege to get all Seniors' performance.

### EXAMPLE 1

```
LLMWebService.PTCSENIORACTIVITYR ptcsenioractivity = LLMWebServ.GetPTCSeniorActivity(0, 0, 0, 0, 0, “username”, 
“password”);
int level_of_difficulty = ptcsenioractivity.PTCSeniorActivityList[0].level;//Get difficulty level of first PTCSeniorActivity
                                                                            (PTCSeniorActivityList[0]) from the
                                                                            PTCSeniorActivityList
```

The above example represents the case of getting all the supported by the system PTCSeniorActivities, using the *GetPTCSeniorActivity* method. It must be stressed out that “ptcsenioractivity_id”, “senior_id”, “ptcactivity_id”, “ptc_id”, “user_id”, are set to zero, thus indicating the fact that the method will return as a result, the whole set of PTCSeniorActivities.

## EXAMPLE 2

```
int ptcsenioract_id = 2;
LLMWebService.PTCSENIORACTIVITYR ptcsenioractivity = LLMWebServ.GetPTCSeniorActivity(ptcsenioract_id, 0, 0, 0, 0, 
“username”, “password”);
```

In this example, a specific PTCSeniorActivity is returned, passing its unique “ptcsenioractivity_id” as a parameter to the *GetPTCSeniorActivity* method.

## EXAMPLE 3

```
int ptcsenior_id = 16;
LLMWebService. PTCSENIORACTIVITYR ptcsenioractivity = LLMWebServ.GetPTCSeniorActivity(0, ptcsenior_id, 0, 0, 0, 
“userName”, “password”);
```

The above example represents the case that the *GetPTCSeniorActivity* method returns a list of PTCSeniorActivities that are performed by a specific senior, given his/her unique id. Following the same convention, a user can retrieve e.g. all the results achieved by all seniors using all kind of PTC equipment concerning a specific task such as flexibility. This is accomplished by setting ptcactivity_id equal to the corresponding id of flexibility task and concurrently by setting all other arguments, except for UserName and Password, to zero.

## PROPERTIES

### PROPERTY’S STRUCTURE

```
<s:complexType name="property">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="property_id" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="propertyName" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="propertyTypeString" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyType" type="tns:PropertyTypeEnum"/>
    <s:element minOccurs="0" maxOccurs="1" name="valueRange" type="s:string"/>
    <s:element minOccurs="0" maxOccurs="1" name="propertyDescription" type="s:string"/>
  </s:sequence>
</s:complexType>

<s:complexType name="PropertyR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="properyList" type="tns:ArrayOfProperty"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| **Attribute** | **Description** | **Type** |
|---|---|---|
| property_id | Property ID | The unique identifier of the Property |
| propertyName | Property Name | String |
| propertyTypeString | Not used | Not used |
| propertyType | Property Type | PropertyTypeEnum |
| valueRange | The range of the value specified by the developers needs | String |
| propertyDescription | Property’s Description | String |

The Property can describe a senior or a user (preferences, etc).

### ADD PROPERTY

The method that is used in order to add a new property to the LLM system called add_property

```
PropertyR add_property(property propertyx, string username, string password);
```

This method adds a Property as supported by the LLM System. Only an administrator has the privilege to use this method. In order for a new property to be store, the property’s property_id must be set to zero 0. Contrary, id the property_id is specified see “edit existing property” session.

#### EXAMPLE 1

```
LLMWebService.property propertyx = new TestingLLMWebService.LLMWebService.property();
propertyx.property_id = 0;
propertyx.propertyName = "PreferedNumOfColors";
propertyx.propertyType = TestingLLMWebService.LLMWebService.PropertyTypeEnum.Integer;
propertyx.propertyDescription = "The number of different colors displayed that dont cinfuse the senior";
propertyx.valueRange = "1-500";
```

```
PropertyR propertyResp = LLMWebServ.add_property(propertyx,UserName,PassWord);

if (propertyResp.error.ErrorCode == 0)
{
   propertyx = propertyResp.properyList[0];
}
```

In the current example, a new instance of Property is created. Then each attribute of the structure is assigned a value. Finally, via the web service method *add_property*, the new instance is being stored into the LLMDB. It must be stressed out that the method returns a `PropertyR` structure with an array (one element) of the new Property, in case of successful addition.

## EDIT EXISTING PROPERTY

The method that is used in order to edit an existing senior of the System is called
add_property

```
PropertyR add_property(property propertyx, string username, string password);
```

For editing an existing property the property's property_id (of the instance propertyx) must be equal to the one that must be edited. Only an administrator has the privilege to use this method.

### EXAMPLE 1

```
LLMWebService.property propertyx = new TestingLLMWebService.LLMWebService.property();
propertyx.property_id = 5;
propertyx.propertyName = "PreferedNumOfColors";
propertyx.propertyType = TestingLLMWebService.LLMWebService.PropertyTypeEnum.Integer;
propertyx.propertyDescription = "The number of different colors displayed that dont cinfuse the senior";
propertyx.valueRange = "1-500";

PropertyR propertyResp = LLMWebServ.add_property(propertyx,UserName,PassWord);

if (propertyResp.error.ErrorCode == 0)
{
   propertyx = propertyResp.properyList[0];
}
```

## DELETE PROPERTY

The method that is used in order to delete an existing property from the System is called
Delete_property

```
PropertyR Delete_property(int property_id, string username, string password);
```

The property_id must be equal to the property's id that you want to delete. If the property_id is 0 the procedure is terminated and returns in the PropertyR structure the appropriate error message.

### EXAMPLE 1

```
int property_id = 5;

PropertyR properyResp = LLMWebServ.Delete_property(property_id,UserName,PassWord);
```

In this example, given of the property’s id, the LLMWS *Delete_property* method deletes the corresponding property.

## GET PROPERTY

The method that is used in order to get all property instances manipulated by the LLM system called get_property

```
PropertyR get_property(int property_id, string propertyName, string username, string password);
```

This method gets the specified by the “property_id” or “propertyName” or all the available Properties that are supported by the LLM System. Every Member (User or Senior) has the privilege to use this Method.

### EXAMPLE 1

```
int property_id = 5;

PropertyR propertyResp = LLMWebServ.get_property(property_id,”” ,UserName,PassWord);
```

The above sample of code represents the case of getting all the supported by the system Properties, using the get_property method. It must be stressed out that “property_id” is set to zero, thus indicating the fact that the method will return as a result all the available PTCActivities (in case of empty propertyName).

## SENIOR PROPERTIES’ STRUCTURE

```
<s:complexType name="seniorproperty">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="seniorproperty_id" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="seniorid" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyid" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="propertyvaluestring" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyvaluedouble" type="s:double"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyvaluedatetime" type="s:dateTime"/>
    <s:element minOccurs="0" maxOccurs="1" name="propertyvaluestringalternative" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyvaluedoublealternative" type="s:double"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyvaluedatetimealternative" type="s:dateTime"/>
    <s:element minOccurs="1" maxOccurs="1" name="Property" type="tns:property"/>
  </s:sequence>
</s:complexType>
```

```
<s:complexType name="SeniorPropertyR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="seniorpropertyList" type="tns:ArrayOfSeniorproperty"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| Attribute | Description | Type |
|---|---|---|
| seniorproperty_id | Senior Property Relation ID | Unique identifier |
| seniorid | Senior ID | The unique identifier of the Senior |
| propertyid | Property ID | I The unique identifier of the Property |
| propertyvaluestring | Value for the specified property attached to a senior (String Type) | String |
| propertyvaluedouble | Value for the specified property attached to a senior (Double Type) | Double |
| propertyvaluedatetime | Value for the specified property attached to a senior (Datetime Type) | Datetime |
| propertyvaluestringalternative | Alternative, See propertyvaluestring | String |
| propertyvaluedoublealternative | Alternative, See propertyvaluedouble | Double |
| propertyvaluedatetimealternative | Alternative, See propertyvaluedatetime | Datetime |
| Property | The Property Instance the property id refers to. | property |

## ADD A PROPERTY TO A SENIOR

The method that is used in order to add a property to a senior is called
AddSeniorProperty

```
SeniorPropertyR AddSeniorProperty(seniorproperty seniorpropertyx, string username, string password);
```

This method adds a property to a senior and a value of this property.
The obligated fields of the seniorpropertyx that must be provided are:
Seniorid:
Propertyid:

A senior has the privilege to use this method only for himself, the Therapist for all his/her Seniors and an Administrator for all the Seniors. In order for a new property to be store, the property's property_id must be set to zero 0. Contrary, id the property_id is specified see "edit existing senior's property" session.

### EXAMPLE 1

```
LLMWebService.seniorproperty senprox = newLLMWebService.seniorproperty();
senprox.seniorproperty_id = 0;
senprox.seniorid = 23;
senprox.propertyid = 3;

senprox.propertyvaluestring = "Blue";
senprox.propertyvaluedouble = 0;

LLMWebService.SeniorPropertyR senproR = LLMWebServ.AddSeniorProperty(senprox,UserName,PassWord);
```

A new instance of SeniorProperty is created and each attribute of it is assigned with a valid value. Finally, the AddSeniorProperty method stores the new instance of SeniorProperty.

## EDIT EXISTING SENIOR'S PROPERTY

The method that is used in order to edit an existing senior's property is called
AddSeniorProperty

```
SeniorPropertyR AddSeniorProperty(seniorproperty seniorpropertyx, string username, string password);
```

This method alters a property to a senior and a value of this property.
The obligated fields of the seniorpropertyx that must be provided are:
seniorproperty_id:
Seniorid:
Propertyid:

A senior has the privilege to use this method only for himself, the Therapist for all his/her Seniors and an Administrator for all the Seniors. In order for updating an already existing property, the seniorpropertyx's seniorproperty_id must be specified.

### EXAMPLE 1

```
LLMWebService.seniorproperty senprox = newLLMWebService.seniorproperty();
senprox.seniorproperty_id = 157;
senprox.seniorid = 23;
senprox.propertyid = 3;

senprox.propertyvaluestring = "Red";
senprox.propertyvaluedouble = 0;

LLMWebService.SeniorPropertyR senproR = LLMWebServ.AddSeniorProperty(senprox,UserName,PassWord);
```

## DELETE SENIOR'S PROPERTY

The method that is used in order to delete an existing property of a senior is called
DeleteSeniorProperty

```
SeniorPropertyR DeleteSeniorProperty(int seniorproperty_id, int senior_id, string username, string password);
```

The seniorproperty_id must be equal to the property's seniorproperty_id that you want to delete. If the property_id is 0 and senior_id is specified then all the propertie's values will be deleted from the LLMDB.

### EXAMPLE 1

```
int seniorid = 0;
int seniorpropertyid = 3;
LLMWebService.SeniorPropertyR senproR = LLMWebServ.DeleteSeniorProperty(seniorpropertyid,
seniorid,UserName,PassWord);
```

In this example, given of the seniorpropertyid, the LLMWS DeleteSeniorProperty method deletes the corresponding property's instance from a senior (and not the property from the properties table).

## GET SENIOR'S PROPERTIES

The method that is used in order to get senior's properties is called
GetSeniorProperty

```
SeniorPropertyR GetSeniorProperty(int seniorproperty_id, int seniorid, int propertyid, string propertyName, string username, string password);
```

This method gets all the Senior's Properties (and values):
seniorproperty_id: if it is 0 then the criteria is not used, else the specified senior property relation property will be returned
seniorid: if it is 0 then the criteria is not used, else the method results include only for the specified senior.
propertyid: if it is 0 then the criteria is not used, else the method results include only for the specified property.
propertyName: if it is "" (empty) then the criteria is not used, else the method results include only for the specified property's name.

A senior has the privilege to use this method only for himself, the Therapist for all his/her Seniors and an Administrator for all the Seniors. In order for updating an already existing property, the seniorpropertyx's seniorproperty_id must be specified.

### EXAMPLE 1

```
int seniorid = 23;
int propertyid = 3;
int seniorpropertyid = 0;

LLMWebService.SeniorPropertyR senproR = LLMWebServ.GetSeniorProperty(seniorpropertyid, seniorid, propertyid,
"",UserName,PassWord);
```

## USER PROPERTIES’ STRUCTURE

```
<s:complexType name="userproperty">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="userproperty_id" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="userid" type="s:int"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyid" type="s:int"/>
    <s:element minOccurs="0" maxOccurs="1" name="propertyvaluestring" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyvaluedouble" type="s:double"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyvaluedatetime" type="s:dateTime"/>
    <s:element minOccurs="0" maxOccurs="1" name="propertyvaluestringalternative" type="s:string"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyvaluedoublealternative" type="s:double"/>
    <s:element minOccurs="1" maxOccurs="1" name="propertyvaluedatetimealternative" type="s:dateTime"/>
    <s:element minOccurs="1" maxOccurs="1" name="Property" type="tns:property"/>
  </s:sequence>
</s:complexType>

<s:complexType name="UserPropertyR">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="userpropertyList" type="tns:ArrayOfUserproperty"/>
    <s:element minOccurs="1" maxOccurs="1" name="error" type="tns:Error"/>
  </s:sequence>
</s:complexType>
```

| Attribute | Description | Type |
|---|---|---|
| userproperty_id | User Property Relation ID | Unique identifier |
| userid | User ID | The unique identifier of the User |
| propertyid | Property ID | The unique identifier of the Property |
| propertyvaluestring | Value for the specified property attached to a senior (String Type) | String |
| propertyvaluedouble | Value for the specified property attached to a senior (Double Type) | Double |
| propertyvaluedatetime | Value for the specified property attached to a senior (Datetime Type) | Datetime |
| propertyvaluestringalternative | Alternative, See propertyvaluestring | String |
| propertyvaluedoublealternative | Alternative, See propertyvaluedouble | Double |
| propertyvaluedatetimealternative | Alternative, See propertyvaluedatetime | Datetime |
| Property | The Property Instance the property id refers to. | property |

## ADD A PROPERTY TO A USER

The method that is used in order to add a property to a user is called
AddUserProperty

```
UserPropertyR AddUserProperty(userproperty userpropertyx, string username, string password);
```

This method adds a property to a user and a value of this property.
The obligated fields of the userpropertyx that must be provided are:
userid:
Propertyid:

A user has the privilege to use this method only for himself and an Administrator for all the users. In order for a new property to be store, the property's property_id must be set to zero 0. Contrary, if the property_id is specified see "edit existing user's property" session.

### EXAMPLE 1

```
LLMWebService.userproperty senprox = newLLMWebService.userproperty();
senprox.userproperty_id = 0;
senprox.userid = 23;
senprox.propertyid = 3;

senprox.propertyvaluestring = "Blue";
senprox.propertyvaluedouble = 0;

LLMWebService.UserPropertyR senproR = LLMWebServ.AddUserProperty(senprox,UserName,PassWord);
```

A new instance of UserProperty is created and each attribute of it is assigned with a valid value. Finally, the AddUserProperty method stores the new instance of UserProperty.

## EDIT EXISTING USER'S PROPERTY

The method that is used in order to edit an existing user's property is called
AddUserProperty

```
UserPropertyR AddUserProperty(userproperty userpropertyx, string username, string password);
```

This method alters a property to a user and a value of this property.
The obligated fields of the userpropertyx that must be provided are:
userproperty_id:
userid:
propertyid:

A userhas the privilege to use this method only for himself and an Administrator for all the users. In order for updating an already existing property, the userpropertyx's userproperty_id must be specified.

### EXAMPLE 1

```
LLMWebService.userproperty senprox = newLLMWebService.userproperty();
senprox. userproperty_id = 157;
senprox. userid = 23;
senprox.propertyid = 3;

senprox.propertyvaluestring = "Red";
senprox.propertyvaluedouble = 0;

LLMWebService.UserPropertyR senproR = LLMWebServ.AddUserProperty(senprox,UserName,PassWord);
```

## DELETE USERS'S PROPERTY

The method that is used in order to delete an existing property of a user is called
DeleteUserProperty

```
UserPropertyR DeleteUserProperty(int userproperty_id, int user_id, string username, string password);
```

The userproperty_id must be equal to the property's userproperty_id that you want to delete. If the property_id is 0 and user_id is specified then all the propertie's values will be deleted from the LLMDB.

### EXAMPLE 1

```
int userid = 0;
int userpropertyid = 3;
LLMWebService.UserPropertyR senproR = LLMWebServ.DeleteUserProperty(userpropertyid,
userid,UserName,PassWord);
```

In this example, given of the userpropertyid, the LLMWS DeleteUserProperty method deletes the corresponding property's instance from a user (and not the property from the properties table).

## GET USER'S PROPERTIES

The method that is used in order to get user's properties is called
GetUserProperty

```
UserPropertyR GetUserProperty(int userproperty_id, int userid, int propertyid, string propertyName, string username,
string password);
```

This method gets all the user's Properties (and values):
userproperty_id: if it is 0 then the criteria is not used, else the specified user property relation property will be returned
userid: if it is 0 then the criteria is not used, else the method results include only for the specified user.
propertyid: if it is 0 then the criteria is not used, else the method results include only for the specified property.
propertyName: if it is "" (empty) then the criteria is not used, else the method results include only for the specified property's name.

A user has the privilege to use this method only for himself and an Administrator for all the users. In order for updating an already existing property, the userpropertyx's userproperty_id must be specified.

### EXAMPLE 1

```
int userid = 23;
int propertyid = 3;
int userpropertyid = 0;

LLMWebService.UserPropertyR senproR = LLMWebServ.GetUserProperty(userpropertyid, userid, propertyid,
"",UserName,PassWord);
```

## REFERENCES