MedITEX WebAPI Documentation

Document type:	Documentation
Version:	3.1
Date:	12.04.2017
Created by:	Andreas Kissener

Content

Interface description

The **MedITEX WebAPI** is a **REST server** which runs as a web service on a windows server machine. Requests and responses are formatted with **JSON**. At this point the API is designed at a **unidirectional** interface so it is not possible to manipulate data in the MedITEX database.
There are currently three functionalities offered by the MedITEX WebAPI:

GetDataStructure

Calling this function will return the **exposed data structure**, defined inside the **MedITEX database**.

Request:
[GET]
https://the-server:port/api/rest/TMedITEXWebAPI/GetDataStrcuture


```
{
  "Result": {
    "Cycle": {
      "Id":"INTEGER",
      "PersonId":"INTEGER",
      "TypeId":"SMALLINT",
      "Type":"VARCHAR(100)",
      "EggDonation":"SMALLINT",
      "StartDate":"DATE",
      "EndDate":"TIMESTAMP",
      "DateModified":"TIMESTAMP"
    },
    "Egg": {
      "Id":"INTEGER",
      "CycleId":"INTEGER",
      "DateModified":"TIMESTAMP"
    }
  }
}
```


Response:
The root node (**"GetDataStructureResult"**) contains all available tables, including their fields along with the field datatype.
Common data types are:

- NUMERIC
- INTEGER
- SMALLINT
- VARCHAR(X) [where 'X' is the field length]
- CHAR
- DATE
- TIMESTAMP

GetData(String table, String changedSince, String filter)

Calling this function with the appropriate **JSON parameters** will return all records which have been assigned to the given table and were changed since the given date.
The MedITEX WebAPI uses the ISO8601 format to process dates ([yyyy]-[mm]-[dd]T[hh]:[mm]:[ss].[zzz]Z]).

Parameters:

table: A **case-insensitive string** that matches a **table name**, returned by '**GetDataStructure'**.

[Optional]

changedSince: A **date/datetime string** in the **ISO 8601** format.

filter: Filter criteria for the requested data. Only **one filter criteria** can be applied per request. String values have to be put in [**'**] tokens.

Format: *FieldName [=, <, >, <=, >=, is, is not] Values*

E.g. 1) PatientId = 'TestPatient' (Patient request)
2) QualityScore is not null (EqqQuality request)
3) DateModified <= '2017-01-01'

The filter follows the **Firebird SQL** syntax. This means **date values** within the filter **cannot** be set in the **ISO 8601** format.

Request:

[PUT]

https://the-server:port/api/rest/TMedITEXWebAPI/GetData

[JSON]

{"table":"XXX","changedSince":"ZZZ"} [where 'XXX' is the table name and 'ZZZ' is a date string]


```
{
  "Result": [
    {
      "Drugs": {
        "Operation":"U",
        "Id":"3",
        "Name":"Puregon",
        "Type":"Injection",
        "UnitId":"1",
        "Unit":"IU",
        "DateModified":"2016-11-25 08:02:49"
      }
    },
    {
      "Drugs": {
        "Operation":"U",
        "Id":"4",
        "Name":"Puregon Pen",
        "Type":"Injection",
        "UnitId":"1",
        "Unit":"IU",
        "DateModified":"2017-03-20 08:46:10"
      }
    }
  ]
}
```


Response:

The root node (**"Result"**) contains an array of all records that has been changed since the requested date. Each record (named as the requested table name) contains the tables data as well as an **"Operation" entry** (**U : Insert/Update**; **D : Delete**) defining the changes on a certain table and all the fields that belong to this table.
If some fields are **not listed** within a JSON object then these values have a '**NULL'** value. **Boolean fields** are only '**TRUE**' if the value of these fields is '**1**'. **Non-listed fields** or any other value **besides** '**1**' is always interpreted as '**FALSE**'

***Note**: A delete operation only sends the **ID** of the record which has been deleted.*

GetUser(String username, String password)

Calling this function with the appropriate JSON parameters returns a status ID of the sent credentials.

Parameters:

username: The (case-sensitive) username of a MedITEX user.

password: The password belonging to the username.

Request:

[PUT]

https://the-server:port/api/rest/TMedITEXWebAPI/GetUser

[JSON]

{"username":"XXX","password":"ZZZ"} [where 'XXX' is the username and 'ZZZ' is the password]

Response:

{"GetUserResult":["XXX"]} [where 'XXX' is the returned status value]

Status values:

1 : Valid credentials

0 : Invalid credentials

-1 : System error while validating, no validation performed

Note:

There is no status differentiation between wrong usernames or wrong passwords for security reasons. As usernames and passwords might be generated within MedITEX we will not allow any gathering of information about existing usernames by brute-forcing random credentials over the WebAPI. In the future we might also implement an IP locking of too many failed login calls.

Configuration & Security

The MedITEX WebAPI is using an .ini file to load specific configurations, like the port or certificates which are required for SSL encryption.
If the .ini file is missing, a default one will be created.
A typical **WebAPI.ini** may look like this:

```
[WebAPIConfig]
MedITEX_Path=C:\Critex\MedITEX IVF\
Log_File=C:\Critex\MedITEX IVF\WebAPI\WebAPI.log
Port=63403
SessionTimeout=300000

[SSL]
Cert_File=C:\Critex\MedITEX IVF\WebAPI\webapi.crt
Key_File=C:\Critex\MedITEX IVF\WebAPI\webapi.key
Root_File=C:\Critex\MedITEX IVF\WebAPI\webapi.pem
```

MedITEX_Path: This path needs to define the MedITEX root folder. It is required to connect to the MedITEX database.

Log_File: Any errors will be logged in the defined file.

Port: The port which will be used by the web service.

Cert_/Key_/Root_File: All certificates needed to create a valid certificate chain.

***Note**: All communication with the MedITEX WebAPI has to be **encrypted**!*
Therefore it is necessary to define certificates in the WebAPI.ini from the MedITEX WebAPI package.

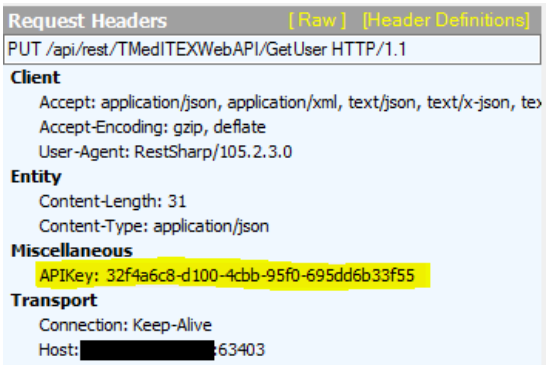
API Keys

An application connecting to the MedITEX WebAPI must authenticate to the API using an API key within the request header.

Each application will get their own API key which is stored in the clinics MedITEX database. An API key can be assigned to all exposed functions or only specific ones. If an invalid API key has been sent or the API key header is missing, each function will return an "**invalid API key"** response.

E.g.: {"Result": {"error": "Invalid API key"}}

Request header



[Request analysis using Fiddler]

Header name: "APIKey"
Header value: "[GUID]"

For testing purposes you may use a public APIKey for the MedITEX test server:
32f4a6c8-d100-4cbb-95f0-695dd6b33f55

Testserver

Server:

A **test server** has been set up at the following address:

http://7d54kyc8qyqo4u4d.myfritz.net:63403/api/rest/TMedITEXWebAPI/

For testing purposes it does **not** require an **HTTPs connection**.