LocDirect HTTP API Version 1.0

Localize Direct HTTP API

Introduction

The LocDirect API use a standard HTTP request/response workflow. The caller sends a request to the API and then gets a response back from the API. The requests and responses can be either in XML or JSON and should always use UTF8 character encoding. The caller must ensure that the requests are correctly formatted, use escaped characters when required by the format and use UTF8 character encoding.

The API can then be accessed via:

`http://<server>:<port>/api/v1`

Important: The LocDirect Server is by default communicating on port 5070. This port is NOT the port that is used for HTTP-requests, instead it's the LocDirect port with an additional 0 at the end of this port number, so by default the HTTP API port is 50700. For example:

`http://my.server.com:50700/api/v1`

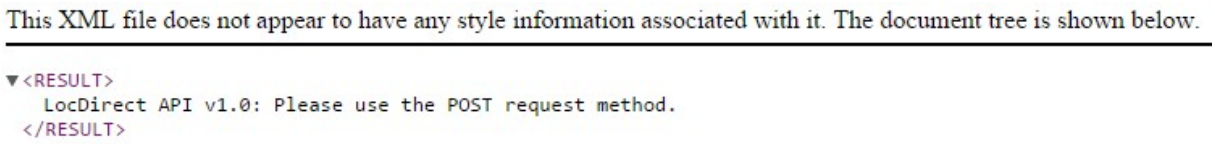
If you don't host your server yourself, you will get the API url and port when signing up or by contacting support@localizedirect.com.

Check API connection

It's easy to check if the API can be accessed. Simply open a web-browser and try to access the LocDirect API:

http://<server>:<port>/api/v1

If everything is working, then the web-browser will show something like this:



This indicates that the API responded successfully! If you don't get this response back, then a firewall could be blocking the API port or you might have the wrong <server> or <port> information.

Authentication

To authenticate, the caller send a Login-command to the API, with a valid user name and password. This will return an API key that then can be used on all API requests. The API key is called a *secId* and it will not change until the password of the API account changes. This means that the login command only needs to be called one time in order to receive the secId.

API Commands

The API commands follow a simple structure that is sent as POST-data in the HTTP request (using the POST http-method). The basic message structure is described below.

Request

When using XML, the root XML-element of a command is named EXECUTION and it consist of one or many TASK-elements. A TASK-element defines the specific task, like for example to update something on the server or to request data from the server. Below is an example of what a command may look like (this is only an example meant to explain the message structure):

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<EXECUTION secId="1212" client="API" version="1.0">
  <TASK name="UpdateUser">
    <OBJECT name="User">
         <userName>John Smith</userName>
         <userEmailAddress/>
    </OBJECT>
    <WHERE>
         <userId>123</userId>
    </WHERE>
  </TASK>
</EXECUTION>
```

The first part "`<?xml version="1.0" encoding="UTF-8"?>`" is important and should always be included.

JSON

When using JSON the structure is similar:

```
{
  "secId": "1212",
  "command": "User.UpdateUser",

  "data": {
   "userName": "John Smith",
   "userEmailAddress": ""
  },

  "where": {
    "userId": "123"
  }
}
```

When using XML, the EXECUTION-element has an attribute called "secId" and when using JSON there's a key-value pair with the same name. This is where the API key - the secId - returned from the login-message - needs to be set. When using XML there are two additional

attributes that needs to be set, “client” and “version”. The client-attribute describes what type of client is requesting the execution. This should always be set to “API”. The version attribute, should be set to the current API-version (currently 1.0).

When using XML, the TASK-element describes the task to be performed. This element have one attribute – the name-attribute. This attribute simply specifies the name of the task. In the example above, this happens to be a task called “GetUser”. The TASK-element must have two child elements called OBJECT and WHERE that can be used to specify the context of the task. Note that not all tasks have arguments, but the elements are still required as they are part of the message structure.

When using JSON, instead the name of the task/command is specified in the key-value pair called “command”, as in the example above.

When using XML, the OBJECT-element specifies the data that the task should use and when using JSON the key-value pair called “data” is used for the same purpose.

When using XML, the WHERE-element specifies arguments for the task and when using JSON the key-value pair called “where” is used for the same purpose.

Response

When a command has been sent to the server and executed, the server will always return a response (unless the connection is broken). If the command returns data, then a resultset-structure will be returned and if there are messages (error-, warning- or information-messages) from the server, then they will be returned in a result-message-structure.

Example XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<EXECUTION committed="true" client="API" version="1.0">
    <RESULTSET>
      <DATASETS>
        <DATASET>
         <userName>John Smith</userName>
         <userEmailAddress>john@smith.com</userEmailAddress>
        </DATASET>
      </DATASETS>
      <RESULT>
        <MESSAGE id="0" type="Information">
 This is an example message from the server!
        </MESSAGE>
      </RESULT>
    </RESULTSET>
  </TASK>
</EXECUTION>
```

Example JSON response:

```
{
  "datasets": [
    {
      "userName": "John Smith",
      "userEmailAddress": "john@smith.com"
```

```
    }
  ],
  "messages": [
     {
        "id": "0",
        "text": "This is an example message from the server!",
        "type": "Information"
     }
   ]
}
```

Apart from the resultset, these an attribute named “committed”. This indicates whether the request was successful or not. The LocDirect server is a fully transactional server, which means that a task can either only be fully performed (committed) or not at all (roll-backed).

A response from the API will always have a resultset. The response from each command is described in the API-messages documentation.

Logging in and retrieving an API key

In order to be able to use the API functionality, a user account first needs to be created. This needs to be done by an administrator using the LocDirect client application.

API key / SecId

On a successful login, the server will return the API key called “secId”. This id, must after the login be used on every API request. For example:

XML

```
<EXECUTION secId="123" client="API" version="1.0">
</EXECUTION>
```

JSON

```
{
   "secId": "123"
   ...
}
```

This way the server can verify that caller is valid and associate the API request with a specific user (for example when adding information in logs etc.).

The Login-Command

First you need to login, in order to receive a secId:

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<EXECUTION client="API" version="1.0">
  <TASK name="Login">
```

```
    <OBJECT name="Security" />
    <WHERE>
      <userName>John</userName>
      <password>htimS</password>
    </WHERE>
  </TASK>
</EXECUTION>
```

JSON

```
{
  "secId"="",
  "command": "Security.Login",

  "data": {},

  "where": {
    "userName": "John",
    "password": "htimS"
    }
}
```

This message will then be sent using an HTTP POST request where the XML/JSON request is the POST-data. If the username and password is correct, then the API will respond with:

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<EXECUTION committed="true" client="API" version="1.0">
    <RESULTSET>
      <DATASETS>
        <DATASET datatype="result">
          <secId>418EEBF8-D068-2137-8AB7-6443471BE836</secId>
          <moduleName>localizedirect-server</moduleName>
          <timeZone>Europe/Berlin</timeZone>
          <moduleVersion>2.9.144</moduleVersion>
        </DATASET>
      </DATASETS>
      <RESULT>
        <MESSAGE id="0" type="Information">The user 'John' has successfully
logged in!</MESSAGE>
      </RESULT>
    </RESULTSET>
</EXECUTION>
```

JSON

```
{
  "messages": [
     {
        "id": "0",
        "text": "The user 'John' has successfully logged in!",
        "type": "Information"
     }
```

```
   ],
  "datasets": [
    {
      "moduleVersion": "2.9.144",
      "moduleName": "localizedirect-server",
      "timeZone":"Europe/Berlin",
      "secId":"418EEBF8-D068-2137-8AB7-6443471BE836"
    }
  ]
}
```

The response confirms that the login was successful, by returning a resultset with the secId and some additional fields. The committed-attribute indicates that the task was successful.

If the user name or password are wrong, then a response like this will be returned:

XML

```
<RESULTSET>
  <DATASETS />
  <RESULT>
   <MESSAGE id="1" type="Error">Failed to login! User name or password is
invalid.</MESSAGE>
   </RESULT>
</RESULTSET>
```

JSON

```
{
  "messages": [
     {
       "id": "1",
       "text": "Failed to login! User name or password is invalid.",
       "type": "Error"
     }
   ],
  "datasets": []
}
```

After the user has logged in, the client can start to send other API.

IMPORTANT: As previously described all API commands, except the login command, will need use the secId that was returned by the login-task.

If the secId is not set, the API will respond with the following resultset (or the JSON equivalent):

```
<RESULTSET>
  <DATASETS />
  <RESULT>
   <MESSAGE id="10615" type="Error">You need to login to
continue!</MESSAGE>
  </RESULT>
</RESULTSET>
```

You should now be ready to go! Please refer to the document named "*LocDirect API – Messages*" to get information about the different API commands that can be used.

Using HTTPS

LocDirect can be set up so that it uses HTTPS instead of HTTP. If you are hosting your LocDirect server yourself, then you will need to have a valid HTTPS certificate for the domain where the server is located and then you can either use this certificate together with a web-proxy server (like for example Nginx) and install the certificate there and then route the traffic to the LocDirect server on the regular HTTP port or you can create a so called "keystore" that then LocDirect can be configured to use (this way LocDirect will handle the HTTPS requests and the certificate directly, but it will require some extra configuration).

For more detailed information on how to set this up, please contact our technical support at support@localizedirect.com.