INTOACCESS
THE INTEGRATORS

Net2JsonRestServer

Manual 1.1

INTOACCESS B.V. THE INTEGRATORS
A.Hofmanweg 5A · 2031 BH · Haarlem · The Netherlands
info@intoaccess.com · www.intoaccess.com · +31(0)23 7505 138

Table of contents

Principle

The Net2JsonRestServer is a wrapper service around the Net2 SDK, which can be called using HTTP POST calls with a JSON content.

This unlocks the Net2 SDK for programming languages (or operating systems) which have no dotnet bindings available. The image below shows in what context the application operates:

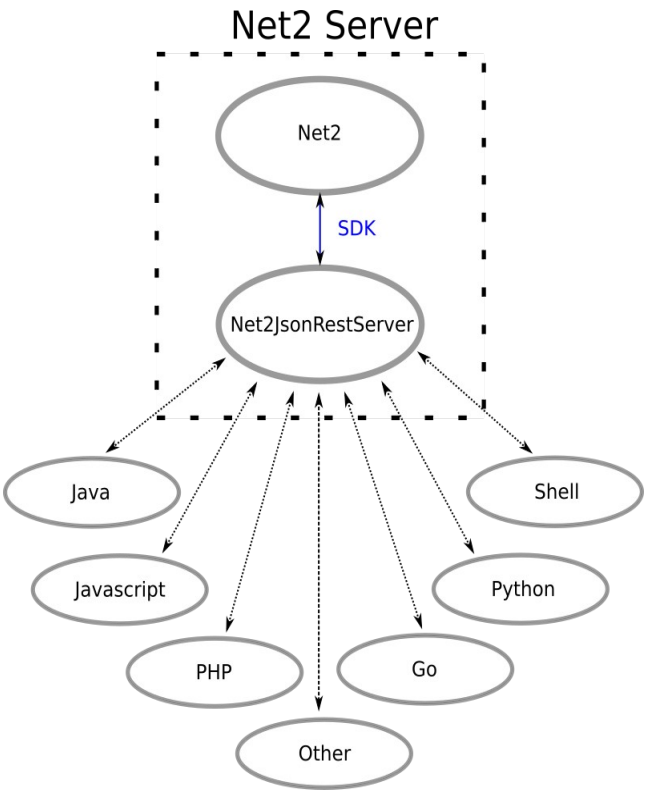

Image 1

Installation

The application can be installed, using a single setup file: Net2JsonRestServer.msi

It is not mandatory to install the software on the Net2 server, but we advise you to do so in order to have the most robust configuration.

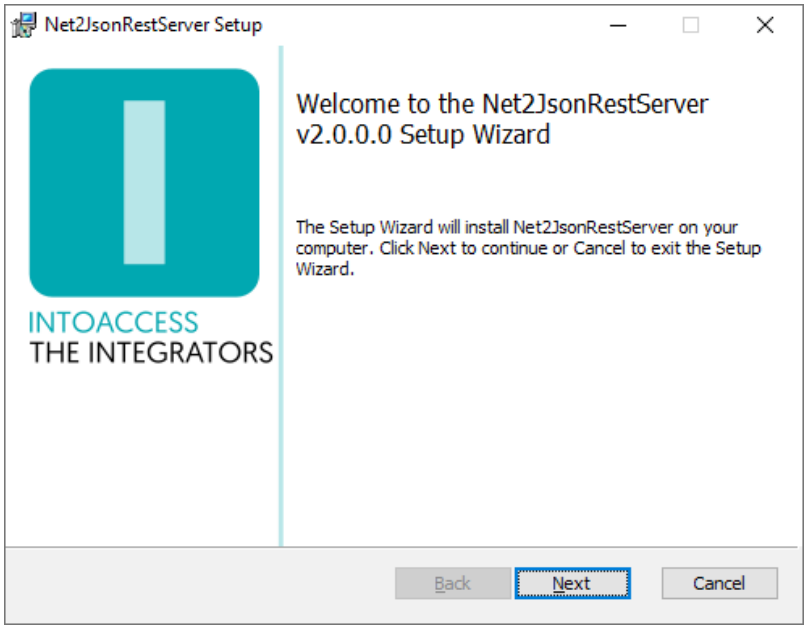

Image 2

The fist dialog window will display what application version you will install.

Note: the version number you see will most likely be different from the one displayed in Image 2.

Updates

Although a newer version should automatically replace any older version that may be present, you can choose to uninstall the existing version first. The configuration settings are not removed during this process, so after installing the new version, you do not have to configure everything all over again.

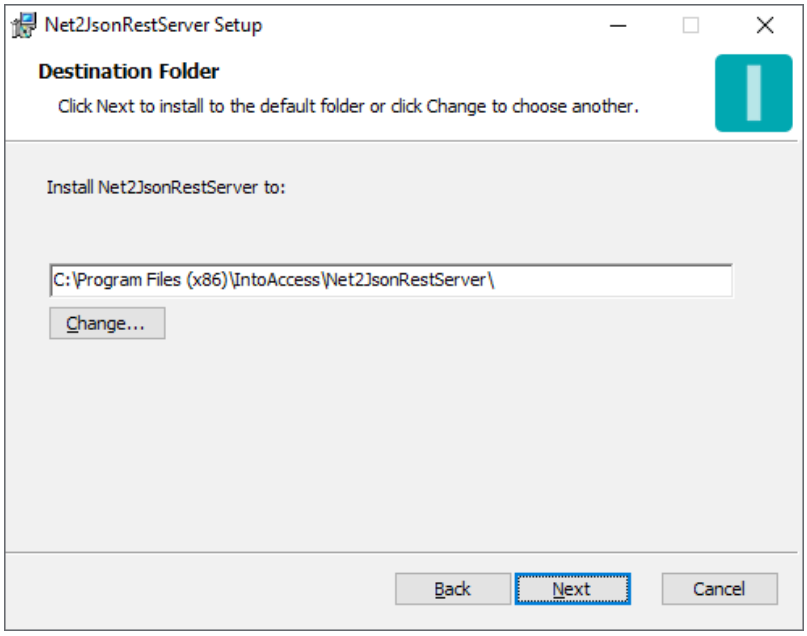

Image 3

The second dialog window shows where the application will be installed. The default value is typically fine.

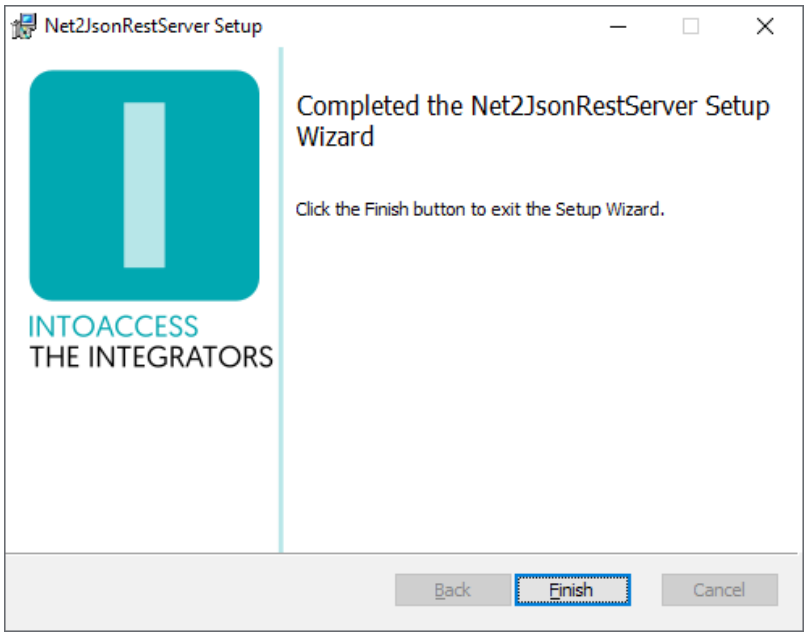

Image 4

The final dialog window will indicate whether the installation was successful.

Configuration

To help with the configuration, the *Net2JsonRestServer manager* application is available and allows you to configure:

- how the application should connect to Net2;
- how the REST service should be configured;
- if you require emails of application starts/stops/errors;

When the manager application is started, a splash screen is displayed for a short period. After that, the application will minimize to a 'tray icon' in the bottom right corner of the screen.

Image 6

Image 7

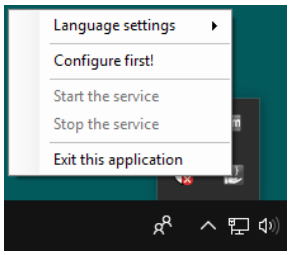

Image 8

By right clicking on the tray icon, the following pop-up menu will appear (providing it is not opened already):

- Language settings: Pick your language;
- Configure (first): Start application configuration;
- Start the service: Start the service (allowed after configuration);
- Stop the service: Stop the service (allowed after configuration);
- Exit this application: Exit the manager application.

The color of the tray icon is indicative of the service state. When the service is not running, its color is gray. The icon gets colored when the service is running.

Net2 Connection

The first configuration page is for configuring the Net2 connection. When you start the configuration for the first time, this will require a few steps, but after the settings are saved, the next time it will connect automatically.

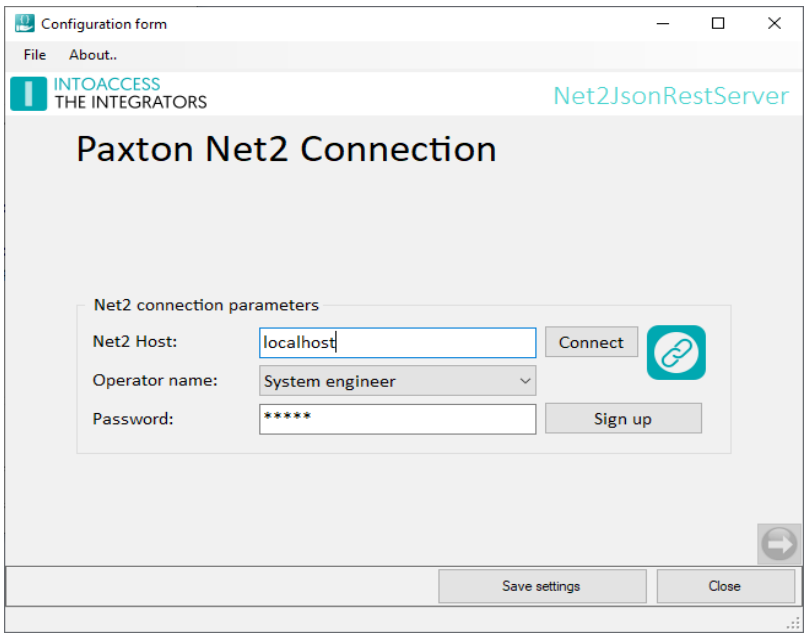

Image 9

- Enter the (ip)address of the Net2 server. If you have installed the attendance tool on the Net2 server, you can use the default 'localhost' value. Do not use an external adapter ip address in this case!!
- Click the Connect button; the application now tries to fetch the Net2 operators. (these users you can find under “Net2 operators” of the standard Net2 application)
- Select an operator with which the application should log on. It is required that this user has the “System Engineer” role.
- Enter the proper password.
- Click the Sign up button.

If all goes well, a message will appear that the connection was successful and the right hand arrow will change from gray to colored.

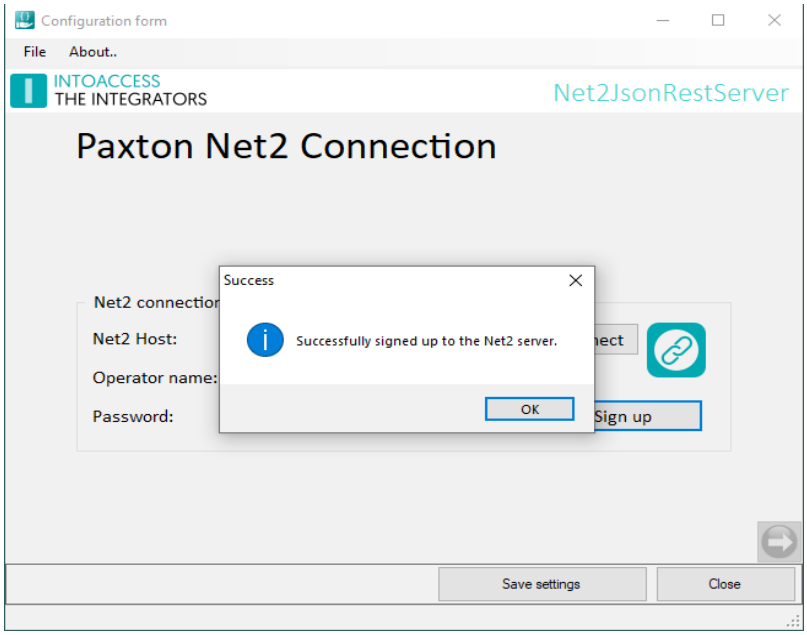

Image 10

After closing the success message, you can proceed to the next configuration window, by clicking on the right hand arrow.

REST Server Settings

In this configuration page, you can define how the REST server will behave:

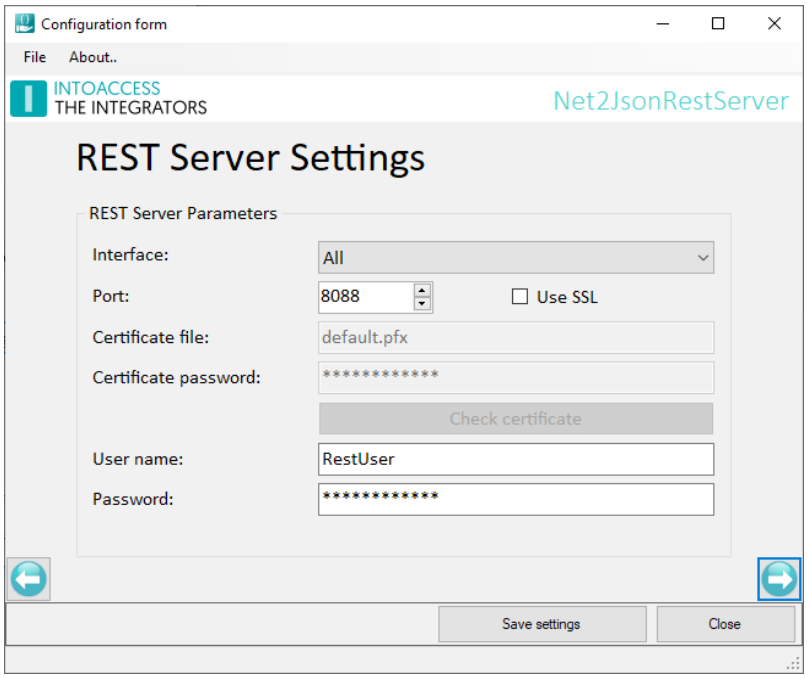

Image 11

- Interface:

  Here you can select if the service will listen on on all external interfaces or only locally.

  - All: The service can be accessed over all external(IPv4) interfaces that are connected to a network.
  - Only local: The service can only be accessed on the computer itself.
- Port:

  The TCP/IP port that the service listens on.
- Use SSL:

  If you enable this option, the communication will be encrypted. When communicating over a non trusted network, switching on this option is highly recomended.
- Certificate file:

  If SSL is used, a certificate file is required. The application offers a 'self signed' certificate by default, but not all client applications will accept this. An official certificate is issued by a 'certificate authority' and is specific for a domain. A possible supplier of certificates is Xolphin: https://www.xolphin.com/
- Certificate password:

  To load a PFX type certificate file, typically a password is required, which you can enter here.

By pressing the “Check certificate” button, you can verify if the application can properly interpret the certificate file.

- User name:

  In order to use the service, a client needs to authenticate itself. This is done by ‘Basic Authentication’, which is an HTTP communication standard.

  You can enter a user name in this field. By default this is “RestUser”.

- Password:

  A password is also part of the ‘Basic Authentication’ and can be entered here. By default it is “RestPassword”.

Mail Settings

The email configuration is optional and offers the possibility to have application messages sent to a system administrator.

This is specifically useful as an early warning system that the application is not functioning correctly.

Webmail

To use web mail providers (like Gmail), it may be required to lower the security settings of the mail account.

SMTP

The SMTP settings allow you to configure which SMTP server and port to use. If you select port 587, authenticated SMTP is also possible by supplying credentials.

Mail destination

To address multiple persons, additional email addresses can be added separated by a semi-colon (;).

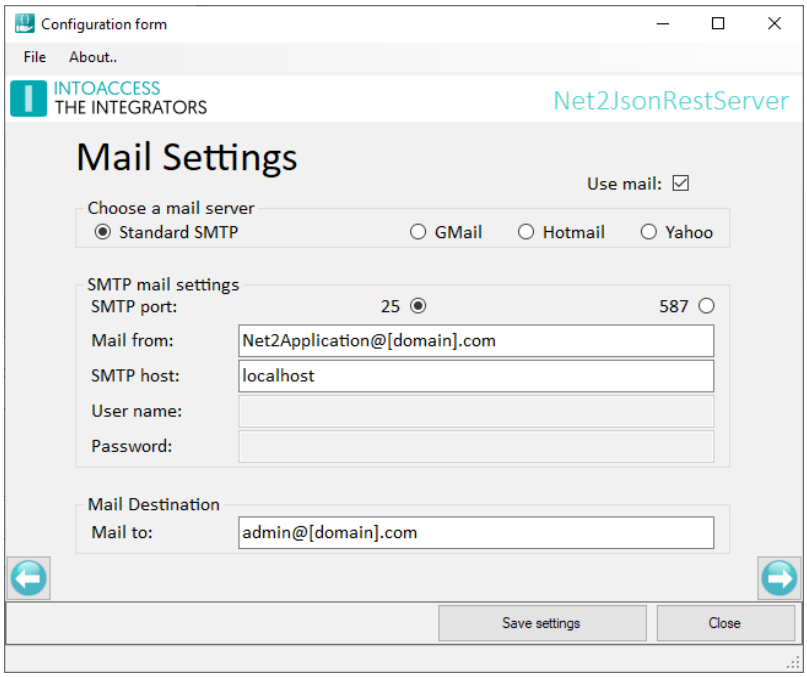

Image 12

Application Licence

The trial/test version as it can be downloaded from the IntoAccess website, is fully functional but will stop working after a certain date when it is not licensed.

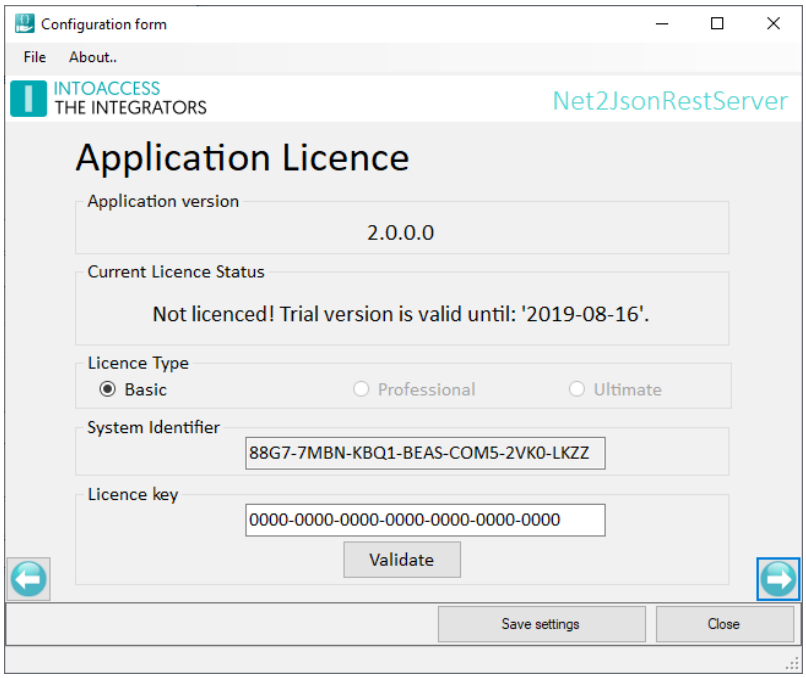

Image 13

After buying a license, you can copy the “System identifier” into an email or send a screenshot to IntoAccess and receive your license code.

Note: the System identifier differs per PC and therefore also the required license code.

Service Control

The service control window offers a way to stop and start the background service.

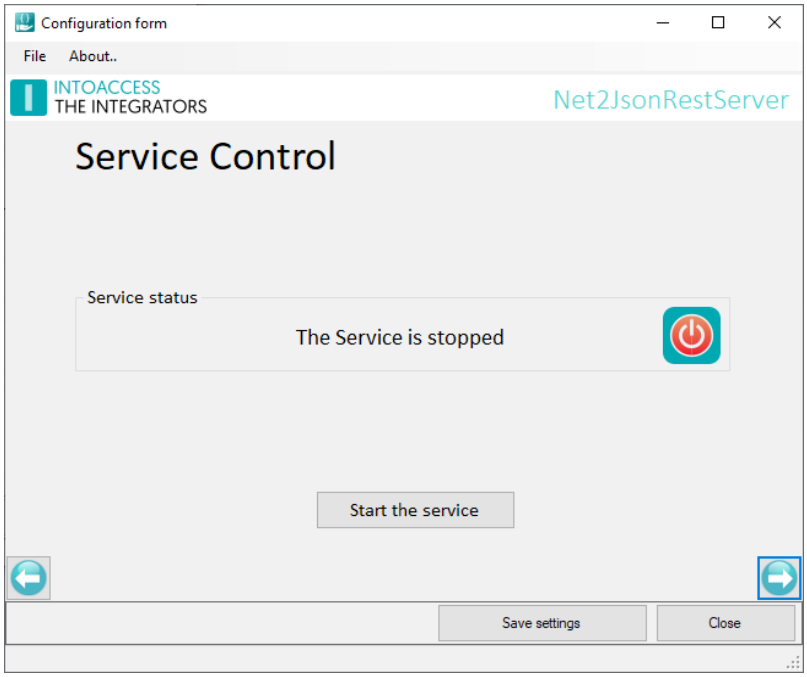

Image 14

Other ways to start and stop the service are:

- Using the tray icon pop-up menu;
- Using the Windows service manager (look for “Net2JsonRestServer”);

Log Settings

This page, see image 15, offers the possibility to review the last (max. 500) lines of the log file. The application will log its activity with a high level of detail. Especially when the application encounters an unexpected problem this log file might contain invaluable information, even for you as an end user.

Please have a look at the last lines of this file if the application refuses to start or otherwise behaves unexpectedly.

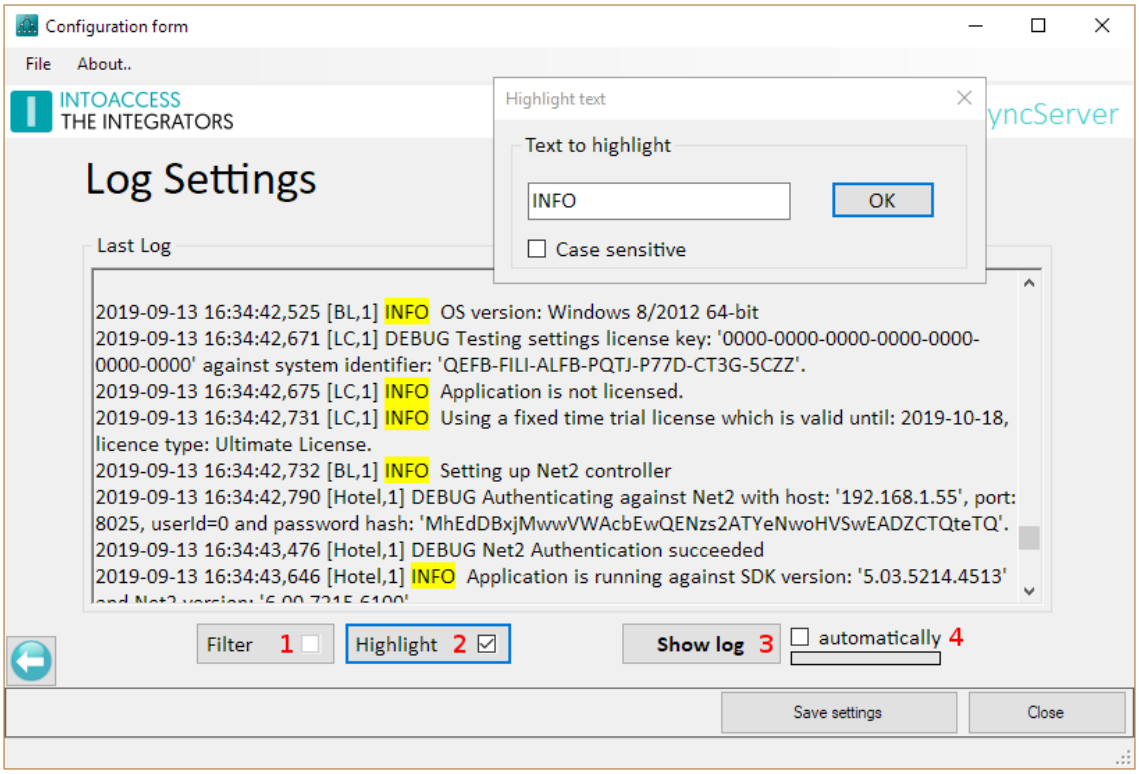

Image 15

You can resize the window in order to get a better overview of the content.
This page also offers the possibility to filter the log file on certain terms (1) and/or to mark certain terms (2). An obvious 'filter term' could be the word 'ERROR' or 'WARN'. If the application works properly, both terms should not appear in the log file.
Option (4) offers the possibility to automatically reload the log file at a fixed interval.

The log files are located in the folder:
c:\IntoAccess\Logging\Net2JsonRestServer

SDK calls

In this chapter you will find the implemented Net2 SDK calls. This is a subset of the total number of available calls, but can be extended on request.

The starting point for the wrapper is that all info that can be obtained using the QueryDB call, will not (also) be offered in another way.

Http response codes

The following codes are used:

- 200: success
- 400: bad request; sent when request is invalid (json or otherwise)
- 401: unauthorized; sent when no credentials ar found
- 403: forbidden; sent when credentials are invalid
- 404: not found; sent when the request path is not known
- 500: internal server error; catch all failure

Card types

The list of 'card types' and their id:

```
0     -     Unspecified
1     -     Proximity card
2     -     Proximity ISO card
3     -     Keyfob
4     -     Hands free token
5     -     WatchProx
6     -     Proximity ISO card no mag stripe
7     -     Vehicle number plate
8     -     Hands free key card
9     -     Fingerprint verification card
```

AddAccessLevel

Add new access level.

Request

url: "http(s)://<ip>:<port>/AddAccessLevel"

JSON

{

```
    "accessLevelName": "SomeAlName",
    "accessLevelDetails": [
         {
             "timezoneId": 1,
             "areaId": 1000001
         },
         {
             "timezoneId": 1,
             "areaId": 1000000
        }
    ]
}
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"accessLevelName": "SomeAccessLevel",
"accessLevelDetails": [{
"timezoneId": 1,
"areaId": 1000001
},
{
"timezoneId": 1,
"areaId": 1000000
}
]
}' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/AddAccessLevel"
```

AddCard

Add a new card/plate to a user.

Notes:

- The cardNo can be a card number “0-99999999” or a license plate.
- The cardTypeId should have the proper value (7) in case of a plate.
  (see chapter Card types)

Request

url: "http://<ip>:<port>/AddCard"

JSON

```
{
    "cardNo": "34567890",
    "cardTypeId": 1,
    "userId": 18,
}
```

Reply

True on succes, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"cardNo": "34567890",
"cardTypeId": 1,
"userId": 18,
}' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/AddCard"
```

AddNewUser

Add a new user.

Notes:

- String values can be left out or set to zero to leave them empty.
- If activationDate is not supplied or null, ‘today’ is used.

- If expiryDate is not supplied or null, there will be no expiration.
- Minimal date value: 1999-01-01.
- If pinCode is not set or null, no pin code is issued.
- If cardNumber is not set or null, no card number is issued.
- The first customFields value [0] is ignored and can be set to null.
- If no customFields are set, their values will be blank.

Request

url: "http://<ip>:<port>/AddNewUser"

JSON

```
{
    "accessLevelId": 1,
    "departmentId": 2,
    "antiPassbackInd": false,
    "alarmUserInd": false,
    "firstName": "John_50",
    "middleName": "Patrick_50",
    "surname": "Doe_50",
    "telephoneNo": "123456_30",
    "telephoneExtension": "78_10",
    "pinCode": "1234_8",
    "activationDate": "2019-05-01",
    "cardNumber": 12345678,
    "cardTypeId": 1,
    "active": true,
    "faxNo": "654321_30",
    "expiryDate": "2020-05-01",
    "customFields": [
        null,
        "Field1_100",
        "Field2_100",
        "Field3_50",
        "Field4_50",
        "Field5_50",
        "Field6_50",
        "Field7_50",
        "Field8_50",
        "Field9_50",
        "Field10_50",
```

```
        "Field11_50",
        "Field12_50",
        "Field13_memo",
        "Field14_50"
    ]
}
```

Reply

UserID of new user, or-101 (AddNewUserFailed) on failure.

JSON

```
123
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"accessLevelId": 1,
"departmentId": 1,
"antiPassbackInd":false,
"alarmuserInd": false,
"firstName": "John",
"middleName": "Patrick",
"surname": "Doe",
"telephoneNo": "123456",
"telephoneExtension": "12",
"pinCode": "",
"activationDate": "2020-01-01",
"cardNumber": 98989898,
"cardTypeId": 3,
"active": true,
"faxNo": "654321",
"expiryDate": "2023-01-01",
"customFields": [],
}' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/AddNewUser"
```

AddTimezone

Add a new time zone:

Notes:

- Multiple time zones with the same name are allowed.
- Minimal start time: 00:00:00
- Maximum start time: 23:59:59
- 0=holiday, 1=sunday (by default)
- Maximum number of time zones: 64

Request

url: "http://<ip>:<port>/AddTimezone"

JSON

```
{
    "timezoneName": "SomeTimezone",
    "timeSlots": [
        {
            "day": 1,
            "start": "00:00:00",
            "end": "23:59:59"
        },
        {
            "day": 2,
            "start": "17:00:00",
            "end": "19:00:00"
        }
    ]
}
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"timezoneName": "SomeTimezone",
"timeSlots": [
{
"day": 1,
"start": "00:00:00",
"end": "23:59:59"
```

```
},
{
"day": 2,
"start": "17:00:00",
"end": "19:00:00"
}
]
}
' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/AddTimezone"
```

ControlDoorEx

Control a door.

Parameters:

- address: ACU serial nummer
- relais: 0 = relais 1, 1 = relais
- function: 0 = Close, 1 = Timed open, 2 = Hold open
- doorOpenTime: Open time in msec
- ledFlash: 0 = no blinking, 1 = blink at reader 1, 2 =blink at reader 2, 3 = blink at both readers (possibly does not work according to spec)

Request

url: "http://<ip>:<port>/ControlDoorEx"

JSON

```
{
    "address": 123,
    "relais": 0,
    "function": 1,
    "doorOpenTime": 3000,
    "ledFlash": 3
}
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"address": 1156774,
"relais": 0,
"function": 1,
"doorOpenTime": 3000,
"ledFlash": 3
}' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/ControlDoorEx"
```

DeleteAccessLevel

Remove an access level.

Notes:

- The parameter is the access level id.
- Returns true, even if the access level does not exist.

Request

url: "http://<ip>:<port>/DeleteAccessLevel"

JSON

```
123
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='123' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/DeleteAccessLevel"
```

DeleteCard

Remove a card/plate.

Notes:

- The parameter is the card or plate number.
- Returns true, even if the card/plate does not exist.

Request

url: "http://<ip>:<port>/DeleteCard"

JSON

```
"34567890"
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='"34567890"' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/DeleteCard"
```

DeleteDepartment

Remove a department.

Notes:

- The parameter is the department id.
- Returns true, event if the department does not exist.

Request

url: "http://<ip>:<port>/DeleteDepartment"

JSON

```
123
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='123' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/DeleteDepartment"
```

DeleteTimezone

Remove a time zone.

Notes:

- The parameter is the time zone id.
- Returns true, even if the time zone does not exist.

Request

url: "http://<ip>:<port>/DeleteTimezone"

JSON

```
123
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='123' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/DeleteTimezone"
```

GetServerVersion

Returns the Net2JsonRestServer version.

Request

url: "http://<ip>:<port>/GetServerVersion"

JSON

```
<empty>
```

Reply

Version number string.

JSON

```
"1.0.0.0"
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/GetServerVersion"
```

LastErrorMessage

Returns the last Net2 error message.

Notes:

- It is the last error message of a (valid) Net2 call. An invalid call (like invalid JSON), will not set this value..
- If the previous (Net2) call was successful, the call will return an empty string.
- The GetServerVersion call (non Net2), will not rest the last Net2 error message.

Request

url: "http://<ip>:<port>/LastErrorMessage"

JSON

```
<empty>
```

Reply

Last error message string.

JSON

```
"Some error message"
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/LastErrorMessage"
```

PurgeUser

Remove a user completely.

Notes:

- The parameter is the user id.
- Removing a user will also remove all (specific) log references to it..

Request

url: "http://<ip>:<port>/PurgeUser"

JSON

```
123
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='1098' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/PurgeUser"
```

QueryDb

Exexute a query on the Net2 database.

Request

url: "http://<ip>:<port>/QueryDB"

JSON

```
"select * from AccessLevels"
```

Reply

JSON

```
{
    [
        {
```

```
            "AccessLevel": 0,
            "AccessLevelName": "No access"
        },
        {
            "AccessLevel": 1,
            "AccessLevelName": "All hours, all doors"
        },
        {
            "AccessLevel": 3,
            "AccessLevelName": "Working hours"
        }
    ]
}
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='"select * from AccessLevels"' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/QueryDB"
```

UpdateAccessLevel

Update an existing access level.

Notes:

- Multiple access levels with the same name are allowed.
- Maximum allowed number of access levels: 256.
- If the name is left empty, it will remain unchanged.

Request

url: "http://<ip>:<port>/UpdateAccessLevel"

JSON

```
{
    "accessLevelId": 11,
    "accessLevelName": "SomeOtherAccessLevel",
    "accessLevelDetails": [
        {
            "timezoneId": 2,
            "areaId": 1000001
        },
```

```
        {
            "timezoneId": 2,
            "areaId": 1000000
        }
    ]
}
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"accessLevelId": 11,
"accessLevelName": "SomeOtherAccessLevel",
"accessLevelDetails": [{
"timezoneId": 2,
"areaId": 1000001
},
{
"timezoneId": 2,
"areaId": 1000000
}
]
}' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/UpdateAccessLevel"
```

UpdateCard

Update an existing card/plate.

Notes:

- The cardNo can be a card number “0-99999999” or a license plate.
- The cardTypeId should have the proper value (7) in case of a plate.
  (see chapter Card types)

Request

url: "http://<ip>:<port>/UpdateCard"

JSON

```
{
    "cardNo": "34567890",
    "cardTypeId": 1,
    "userId": 18,
    "lostCard": false
}
```

Reply

True on succes, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"cardNo": "34567890",
"cardTypeId": 3,
"userId": 18,
"lostCard": false
}' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/UpdateCard"
```

UpdateDepartment

Update an existing department.

Notes:

* An update to a department with the same name, will return false.

Request

url: "http://<ip>:<port>/UpdateDepartment"

JSON

```
{
    "deptId": 123,
    "deptName": "OtherName"
}
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"deptId": 123,
"deptName": "OtherName"
}' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/UdpateDepartment"
```

UpdateTimezone

Update an existing time zone.

Ter kennisgeving:

- Multiple time zones with the same name are allowed.
- Minimal start time: 00:00:00
- Maximum end time: 23:59:59
- 0=holiday, 1=sunday (by default)
- Maximum number of time zones: 64

- If the name is left empty, it will remain unchanged.

Request

url: "http://<ip>:<port>/UpdateTimezone"

JSON

```
{
    "timezoneId": 4,
    "timezoneName": "SomeOtherTimezone",
    "timeSlots": [
        {
            "day": 4,
            "start": "00:00:00",
            "end": "23:59:59"
        },
        {
            "day": 5,
            "start": "17:00:00",
            "end": "19:00:00"
        }
    ]
}
```

Reply

True on success, false on failure.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"timezoneId": 4,
"timezoneName": "SomeOtherTimezone",
"timeSlots": [
{
"day": 4,
"start": "00:00:00",
"end": "23:59:59"
},
{
"day": 5,
```

```
"start": "17:00:00",
"end": "19:00:00"
}
]
}
' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/UpdateTimezone"
```

UpdateUserRecord

Update an existing user.

Notes:

- String values can be left out out or set to null, in order to leave them unchanged.
- if activationDate is not set or null, 'today' is used.
- If expiryDate is not set or null, there is no expiration.
- Minimal date value: 1999-01-01.
- If pinCode is not set or null, no pin code is issued.
- If cardNumber is not set or null, no card is issued.
- The first customFields value [0] is ignored and canbe set to null.
- If customFields is not set, the values remain unchanged.
- If an individual customField is not set or null, its value remains unchanged.

Request

url: "http://<ip>:<port>/UpdateUserRecord"

JSON

```
{
    "userId": 123,
    "accessLevelId": 1,
    "departmentId": 2,
    "antiPassbackInd": false,
    "alarmUserInd": false,
    "firstName": "John_50",
    "middleName": "Patrick_50",
    "surname": "Doe_50",
    "telephoneNo": "123456_30",
    "telephoneExtension": "78_10",
    "pinCode": "1234_8",
    "activationDate": "2019-05-01",
    "activeInd": true,
    "faxNo": "654321_30",
    "expiryDate": "2020-05-01",
    "customFields": [
        null,
        "Field1_100",
        "Field2_100",
        "Field3_50",
        "Field4_50",
        "Field5_50",
```

```
        "Field6_50",
        "Field7_50",
        "Field8_50",
        "Field9_50",
        "Field10_50",
        "Field11_50",
        "Field12_50",
        "Field13_memo",
        "Field14_50"
    ]
}
```

Reply

True on success, false on error.

JSON

```
true
```

Sample with wget

```
wget -O- --user=$USER --password=$PASSWORD \
--post-data='{
"userId": 50,
"accessLevelId": 0,
"departmentId": 1,
"antiPassbackInd":false,
"alarmuserInd": false,
"firstName": "John",
"middleName": "Patrick",
"surname": "Doe",
"telephoneNo": "123456",
"telephoneExtension": "12",
"pinCode": "",
"activationDate": null,
"activeInd": true,
"faxNo": "654321",
"expiryDate": null,
"customFields": [],
}' \
--header='Content-Type:application/json' \
"http://$SERVER:$PORT/UpdateUserRecord"
```

Manual Net2JsonRestServer
Version 1.1