ChecksumValidation

Copyright (C) 2012 Manu Carus (manu.carus@ethical-hacking.de)

About

ChecksumValidation is a framework for validation of international bank accounts (IBAN), credit card numbers, german identity cards, german passports and for german bank accounts. Validation is performed by computing checksums. Checksums for german bank accounts are maintained and documented by www.bundesbank.de. In sum, ChecksumValidation implements round about 150 algorithms. It may be freely used under GNU GPLv3 licence (see below).

ChecksumValidation is implemented in Microsoft .NET 2.0. It consists of:

- ChecksumValidation: a class library (DLL) for in-process use that implements all validation algorithms
- ChecksumSoapServer: a SOAP interface to service validation requests by the means of Web Services
- ChecksumTcpServer: a TCP interface to service validation requests by TCP/IP and a domain-specific language (DSL)
- ChecksumComServer: a COM interface for Windows inter-process communication
- ChecksumClient: a test utility which can be used either as a GUI and/or as a console application
- ChecksumUnitTest: a fully-fledged NUnit test for all implemented validation algorithms
- ChecksumUtility: a command-line tool for helper functions

Security

As bank accounts, credit cards, identity cards and passports represent sensitive data, security is a built-in feature: TCP traffic between a ChecksumValidation client and a Checksum TCP Server is always encrypted (AES-256, RFC 2898). SOAP traffic is encrypted on the basis of TLS/SSL link encryption.

Quickstart

Extract the ChecksumValidation.zip archive, e.g. to C:\Dev\ChecksumValidation. To invoke a GUI version of the ChecksumClient, enter the following commands into the command line:

```
>C:
>cd C:\Dev\ChecksumValidation\bin
>ChecksumClient -gui ---verbose
```

To use the command line version of the ChecksumClient, enter

```
>ChecksumClient -inproc iban:DE60700517550000007229 ---verbose
```

For more help, enter

```
>ChecksumClient -help
>ChecksumTcpServer -help
```

How to use: ChecksumClient.exe

ChecksumValidation Copyright (C) 2012 Manu Carus (manu.carus@ethical-hacking.de)

This program comes with ABSOLUTELY NO WARRANTY; for details type

'ChecksumClient -licence'.

This is free software, and you are welcome to redistribute it under certain conditions; refer to GNU GPLv3 <http://www.gnu.org/licenses/> for details.

Validates german bank accounts by computing a checksum according to the algorithms maintained and documented by www.bundesbank.de.

Also validates german identity cards, german passports, international bank accounts (IBAN) and credit cards.

The following interfaces to the server are available:

- Tcp Listener
- SOAP Web Service
- In-Proc
- COM

```
ChecksumClient command
               [-inproc|-tcp|-soap|-com]
               [-console|-gui]
               [-secure]
               [-password:pwd]
               [-verbose]
               [-silent]
               [-help]
               [-licence]
```

```
command         command to be sent to the checksum server

-inproc         client integrates a local checksum server (default)
-tcp            client connects via tcp to the checksum server
-soap           client connects via web service to the checksum server
-com            client connects via COM to the checksum server

-console        command line tool (default)
-gui            graphical user interface

[-secure]       indicates a secured tcp connection (e.g. ssl or ssh)
                default: false.

[-password:pwd] password used to secure the tcp connection
                (useful only in combination with -tcp)
                (required if -secure has not been set)
                default: console input if -console is set.
                         user input if -gui is set.

[-verbose]      verbose mode (can be extended to --verbose or ---verbose)
                (useful only in combination with -inproc)
[-licence]      displays the terms of licence for use of this software
[-silent]       silent mode (default: false)
[-help]         displays this text
```

Examples

```
ChecksumClient -inproc account:1234567897/37050299
ChecksumClient -soap iban:DE60700517550000007229
ChecksumClient -com to-iban:1234567890/37050299

ChecksumClient -tcp cache
ChecksumClient -tcp account:1234567897/37050299
ChecksumClient -tcp format-blz:37050299
ChecksumClient -tcp iban:DE60700517550000007229
ChecksumClient -tcp to-iban:1234567890/37050299
ChecksumClient -tcp format-iban:DE60700517550000007229
ChecksumClient -tcp "identity:2406055684D<<6810203<0705109<<<<<<6"
ChecksumClient -tcp "passport:2406055684D<<6810203M0705109<<<<<<<<<<<<<<6"
ChecksumClient -tcp credit-card:4509472140549006
ChecksumClient -tcp get-credit-card-type:4509472140549006
ChecksumClient -tcp stop
```

Description

The ChecksumClient tool offers a full featured GUI to test all validation function for all communication channels. You can start the GUI with the command:

```
>ChecksumClient -gui
```

With the -verbose option, you can adjust the output level of detail information, which can be helpful in some cases: use the option -verbose, --verbose, or even ---verbose to log additional calculation data, e.g.

```
>ChecksumClient -gui -verbose
>ChecksumClient -gui --verbose
```

```
>ChecksumClient -gui ---verbose
```

There are four different communication channels for checksum validation:

- In-Proc (.dll)
- COM
- TCP
- SOAP

You can choose the right option with the "Communication" drop-down list.

In-Proc can directly be used without any preliminary work.

Before invoking the COM server, you have to register the COM component as follows:

```
>regasm ChecksumComServer.dll
```

(You need administrative privileges for successful COM registration.)

Before invoking the TCP server, you have to start the TCP server in a separate command-line window:

```
>ChecksumTcpServer ---verbose
```

Make sure to enter a secure password when prompted (something like "#1TiavsPtu9!"). You have to enter the same password into the "Security Password" text box of the ChecksumClient GUI. The password is used to establish end-of-end-encryption of data between the TCP server and the client.

The TCP server can either be stopped by <Ctrl-C>, or by entering

```
>ChecksumClient -tcp stop
```

into another command-line window. Again, make sure to provide the same password when being prompted to.

Before invoking the SOAP server, you will have to host the physical directory

C:\dev\ChecksumValidation\soap by an ASP.NET Web Server (e.g. IIS).

In IIS, create a virtual directory, e.g. /ChecksumValidation, and activate SSL.

Copy the URL to the SOAP web service processor file Service.asmx in the virtual directory to the "SOAP Endpoint" edit field of the ChecksumClient GUI, e.g. https://localhost/Service.asmx

Make sure you have a secure connection. For test purposes, you may set the "Secure Connection" option in the "Security Settings for TCP and SOAP Server Communication" group in the ChecksumClient GUI).

The ChecksumClient also offers a console interface. In-Proc commands are as follows (samples):

```
>ChecksumClient -inproc ---verbose iban:DE60700517550000007229

>ChecksumClient -inproc ---verbose account:1234567897/37050299

>ChecksumClient -inproc ---verbose credit-card:4509472140549006

>ChecksumClient -inproc ---verbose
"identity:2406055684D<<6810203<0705109<<<<<<6"

>ChecksumClient -inproc ---verbose
"passport:2406055684D<<6810203M0705109<<<<<<<<<<<<<<6"

>ChecksumClient -inproc ---verbose format-blz:37050299

>ChecksumClient -inproc ---verbose to-iban:1234567890/37050299

>ChecksumClient -inproc ---verbose format-iban:DE60700517550000007229

>ChecksumClient -inproc ---verbose get-credit-card-type:4509472140549006
```

COM commands are as follows (samples):

```
>ChecksumClient -com ---verbose iban:DE60700517550000007229

>ChecksumClient -com ---verbose account:1234567897/37050299

>ChecksumClient -com ---verbose credit-card:4509472140549006

>ChecksumClient -com ---verbose "identity:2406055684D<<6810203<0705109<<<<<<6"

>ChecksumClient -com ---verbose
"passport:2406055684D<<6810203M0705109<<<<<<<<<<<<<<6"

>ChecksumClient -com ---verbose format-blz:37050299

>ChecksumClient -com ---verbose to-iban:1234567890/37050299

>ChecksumClient -com ---verbose format-iban:DE60700517550000007229

>ChecksumClient -com ---verbose get-credit-card-type:4509472140549006
```

TCP commands are as follows (samples):

```
>ChecksumClient -tcp ---verbose iban:DE60700517550000007229
>ChecksumClient -tcp ---verbose account:1234567897/37050299
>ChecksumClient -tcp ---verbose credit-card:4509472140549006
>ChecksumClient -tcp ---verbose "identity:2406055684D<<6810203<0705109<<<<<<6"
>ChecksumClient -tcp ---verbose
"passport:2406055684D<<6810203M0705109<<<<<<<<<<<<<<6"
>ChecksumClient -tcp ---verbose format-blz:37050299
>ChecksumClient -tcp ---verbose to-iban:1234567890/37050299
>ChecksumClient -tcp ---verbose format-iban:DE60700517550000007229
>ChecksumClient -tcp ---verbose get-credit-card-type:4509472140549006
```

Remember to start the TCP server before setting off commands, as described above.

Make sure to use a secure password, e.g. "#1TiavsPtu9!"

If you do not want to enter a password for every single command, you may provide for the -secure option, for test purposes only. Sample:

```
>ChecksumTcpServer ---verbose -secure
>ChecksumClient -tcp ---verbose -secure iban:DE60700517550000007229
```

In order to stop the TCP server, set off the command

```
>ChecksumClient -tcp stop
```

The stop command is only accepted from the local machine; you cannot stop the TCP server remotely.

SOAP commands are as follows (samples):

```
>ChecksumClient -soap iban:DE60700517550000007229
>ChecksumClient -soap account:1234567897/37050299
>ChecksumClient -soap credit-card:4509472140549006
>ChecksumClient -soap "identity:2406055684D<<6810203<0705109<<<<<<6"
>ChecksumClient -soap "passport:2406055684D<<6810203M0705109<<<<<<<<<<<<<<6"
>ChecksumClient -soap format-blz:37050299
>ChecksumClient -soap to-iban:1234567890/37050299
>ChecksumClient -soap format-iban:DE60700517550000007229
>ChecksumClient -soap get-credit-card-type:4509472140549006
```

Remember to host the Service.asmx file in an ASP.NET server, as described above.

The SOAP Endpoint URL has to be specified in the ChecksumClient.exe.config file, e.g.

```
<configuration>
  <appSettings>
    ...
    <add key="soap-endpoint"
         value="https://localhost/ChecksumValidation/Service.asmx" />
    ...
  </appSettings>
</configuration>
```

For test purposes, you may again use a plain text connection (HTTP instead of HTTPS),

but you have to provide for the -secure option to be conscious about this security

weakness. Sample:

```
>ChecksumClient -soap iban:DE60700517550000007229 -secure
```

How to use: ChecksumTcpServer.exe

ChecksumValidation Copyright (C) 2012 Manu Carus (manu.carus@ethical-hacking.de)

This program comes with ABSOLUTELY NO WARRANTY; for details type 'ChecksumTcpServer -licence'. This is free software, and you are welcome to redistribute it under certain conditions; refer to GNU GPLv3 <http://www.gnu.org/licenses/> for details.

Validates german bank accounts by computing a checksum according to the algorithms maintained and documented by www.bundesbank.de.

Also validates german identity cards, german passports, international bank accounts (IBAN) and credit cards.

```
ChecksumTcpServer [-port:<port>]
                  [-trace:file]
                  [-error:file]
                  [-secure]
                  [-password:pwd]
                  [-silent]
                  [-verbose]
                  [-help]
```

[-port:<port>] specifies the tcp port to listen to.

default: 65535.

[-trace:file] contains verbose output of the server process.

default: console.

[-error:file] error file containing detailed error messages in case.

default: console.

[-secure] indicates a secured tcp connection (e.g. ssl or ssh)
default: false.

[-password:pwd] password used to secure the tcp connection
(required if -secure has not been set)
default: environment variable $CHECKSUM_PASSWORD.

[-verbose] verbose mode.
can be extended to --verbose or ---verbose.
default: none.

[-licence] displays the terms of licence for use of this software

[-silent] silent mode.
default: false.

[-help] displays this text.

Examples

```
ChecksumTcpServer

ChecksumTcpServer -port:49152

ChecksumTcpServer -trace:ChecksumTcpServer.trace.txt
                  -error:ChecksumTcpServer.error.txt
                  -verbose
```

Description

Type

```
>ChecksumTcpServer -help
```

to get help.

The default port is 65535, but you can use any port by providing configuration data into the ChecksumTcpServer.exe.config file.

Start the TCP server by typing

```
>ChecksumTcpServer
```

If you have to automate this task and don't want to enter a passphrase interactively (to secure the TCP connection), you may set the environment variable CHECKSUM_PASSWORD.

Open a command-line window and type:

```
>set CHECKSUM_PASSWORD=#1TiavsPtu9!
>ChecksumTcpServer
```

Open a second command-line window and type:

```
>set CHECKSUM_PASSWORD=#1TiavsPtu9!
>ChecksumClient -tcp iban:DE60700517550000007229
```

For test purposes, you may provide for the -secure option:

Open a command-line window and type:

```
>ChecksumTcpServer -secure
```

Open a second command-line window and type:

```
>ChecksumClient -tcp iban:DE60700517550000007229 -secure
```

How to use: ChecksumSoapServer

In order to use SOAP Web Services in the context of ChecksumValidation, you have to set up an ASP.NET Web Server (as described above, see How to use: ChecksumClient.exe). Alternatively, you can just start the Visual Studio Solution \src\ChecksumValidation.sln and use the built-in ASP.NET Development Server. The SOAP Endpoint URL in this case is http://localhost:49152/Service.asmx

The WSDL comes by default with the Service.asmx file itself, e.g.

http://localhost:49152/Service.asmx?WSDL

Because the data to be validated is considered as sensitive information, security is enforced by SSL. You have to set up a secure connection (e.g. HTTPS) to successfully invoke a web service.

For test purposes, you may use a plain text connection (e.g. HTTP instead of HTTPS). But in order to be conscious about this security weakness, you have to provide for a specific SOAP header: With the web service request, the SOAP header "securitySettings" has to be set to "true" (or "yes" or "ok" or "1"). That's why you can't just use the default web form provided by Service.asmx. It will just output an error message to you, e.g.

```
System.Web.Services.Protocols.SoapException: security error! --->
System.Security.SecurityException: missing soap header

   at Service.VerifySecuritySettings() in
c:\dev\ChecksumValidation\src\ChecksumSoapServer\App_Code\Service.cs:line 535

   at Service.FormatBlz(String blz) in
c:\dev\ChecksumValidation\src\ChecksumSoapServer\App_Code\Service.cs:line 176

   --- End of inner exception stack trace ---

   at Service.FormatBlz(String blz) in
c:\dev\ChecksumValidation\src\ChecksumSoapServer\App_Code\Service.cs:line 194
```

How to use: ChecksumComServer.dll

Before you can use COM in the context of ChecksumValidation, you have to register the COM component ChecksumComServer.dll (as described above, see How to use: ChecksumClient.exe).

Refer to \src\ChecksumClient\Proxy\ComProxy.cs for a reference implementation of how to use COM from within a .NET class.

You can pretty easily use the COM server in any Windows COM application. Just set a COM reference to \bin\ChecksumComServer.dll and code like this:

```
Dim server As New ChecksumValidation.ChecksumServer.Server
Dim iban As String = server.ToIban("37050299", "1234567890")
```

COM identification data is:

```
ProgId: "ChecksumValidation.ChecksumServer.Server"
CLSID:  "{A81534F9-D11A-3917-BC4F-5E516E75FA0B}"
```

How to use: ChecksumValidation.dll

In order to use ChecksumValidation in-proc (.dll) in Visual Studio, just set a reference to \bin\ChecksumValidation.dll and code like this:

```
using ChecksumValidation.BankAccountValidation;
using ChecksumValidation.CreditCardValidation;
using ChecksumValidation.IbanValidation;
using ChecksumValidation.IdentityValidation;

...
TraceManager traceManager = new
TraceManager(TraceManager.VerboseMode.VeryVeryVerbose, Console.Out);
BankAccountValidator bankAccountValidator = new
BankAccountValidator(traceManager, ".");
CreditCardValidator ibanValidator = new IbanValidator(traceManager);
IbanValidator creditCardValidator = new CreditCardValidator(traceManager);
IdentityValidator identityValidator = new IdentityValidator(traceManager);

blz = "37050299";
account = "1234567890";

string iban = ibanValidator.ToIban(blz, account);
...
```

Refer to \src\ChecksumClient\Proxy\InProcProxy.cs for a reference implementation.

How to use: ChecksumUnitTest.dll

The ChecksumValidation framework comprises a full-featured unit test, which covers all 138 german bank code algorithms as well as unit tests for IBAN validation, credit card validation, german identity card validation and german passport validation.

Unit tests are performed using NUnit. Refer to http://nunit.org/?p=download and unzip the latest stable NUnit bin release to your local drive, e.g. to C:\dev\NUnit. For this ChecksumValidation release, I used NUnit-2.5.10.11092.zip from
http://launchpad.net/nunitv2/2.5/2.5.10/+download/NUnit-2.5.10.11092.zip

To perform a unit test, enter

```
>c:
>cd \dev\ChecksumValidation\bin
>copy c:\dev\NUnit\bin\net-2.0\nunit.framework.dll
>C:\dev\NUnit\bin\net-2.0\nunit-console.exe /nologo /out:NUnit.result.txt
/xml:NUnit.result.xml /err:NUnit.error.txt ChecksumUnitTest.dll
```

The test can take up to some minutes. The output will look something like

```
ProcessModel: Default    DomainUsage: Single
Execution Runtime: Default
................
Tests run: 16, Errors: 0, Failures: 0, Inconclusive: 0, Time: 25,1784 seconds
  Not run: 0, Invalid: 0, Ignored: 0, Skipped: 0
```

Refer to the output files

- NUnit.error.txt
- NUnit.result.txt
- NUnit.result.xml

for errors; for details, refer to the log file

- ChecksumValidation.UnitTest.log

Config Files

Configuration data is maintained by assembly config files:

- `ChecksumClient.exe.config`
- `ChecksumTcpServer.exe.config`
- `ChecksumUnitTest.dll.config`
- `ChecksumUtility.exe.config`

Refer to the <appSettings> section in these configuration files to adjust the application to your requirements.

Updates for German Bank Codes and Checksum Algorithms

Geman bank codes and checksum algorithms are maintained by www.bundesbank.de. Refer to

http://www.bundesbank.de/zahlungsverkehr/zahlungsverkehr_bankleitzahlen_download.php

http://www.bundesbank.de/zahlungsverkehr/zahlungsverkehr_pruefziffernberechnung.php

For this ChecksumValidation release, I used

http://www.bundesbank.de/download/zahlungsverkehr/bankleitzahlen/20120304/blz_20111205.txt

http://www.bundesbank.de/download/zahlungsverkehr/zv_pz201109.pdf

Licence

=======

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

See the GNU General Public License <http://www.gnu.org/licenses/> for more details.