```
/*
* Copyright(c) 2000-2018 Deutsche Post DHL Group. All rights
reserved.
*
* Redistribution and use in source and binary forms, with or
without
* modification, are permitted provided that the following
conditions are met:
*
*  1. Redistributions of source code must retain the above
copyright notice,
*     this list of conditions and the following disclaimer.
*  2. Redistributions in binary form must reproduce the above
copyright
*     notice, this list of conditions and the following
disclaimer in the
*     documentation and/or other materials provided with the
distribution.
*  3. Neither the name of the copyright holder nor the names
of its
*     contributors may be used to endorse or promote products
derived from
*     this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
CONTRIBUTORS BE
* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*/
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
```

```
 * Deutsche Post DHL Group - Internetmarke.
 *
 * Example of an MD5 signing / hashing algorithm, to generate
the hashes to
 * be enclosed in the Internetmarke interface SOAP headers.
 *
 * You may freely take the code for your Internetmarke
handling stubs.
 */
public class MD5Sample {
      /** Logging output for the hashing algorithm. */
      private static final Logger logger =
Logger.getAnonymousLogger();

      /** The identification of the MD5 hashing algorithm. */
      public static final String MD5 = "MD5";
      /** Encoding style for the DPDHL Internetmarke. */
      public static final String ENCODING_STYLE = "UTF-8";
      /** Separator character between the fields to encode /
hash. */
      public static final String FIELDS_SEPARATOR = "::";
      /** Maximum string size for the fields to be hashed. */
      public static final int MAX_FIELD_STRING_SIZE = 256;
      /** Maximum length of the hash string representation. */
      public static final int MAX_HASH_STRING_SIZE = 8;

      /**
       * Main function to start the example algorithm from the
command line.
       * @param args Command line arguments.
       */
      public static void main(String args[]) {

            DateFormat formatter = new SimpleDateFormat(
"ddMMyyyy-HHmmss" );
            String theDate = formatter.format( new Date() );

            // String theDate = "28012014-142729"; // set a
fixed date as an example

            // Example: To be replaced by the partner id
provided by Deutsche Post
            String thePartnerID = "DPPAR";
            // Example: To be replaced by the secret provided
by Deutsche Post
            String thePartnerSecret = "Das_Ist_der_neue_Key";
            // Example: To be replaced by the key phase
provided by Deutsche Post
            String theKeyPhase = "1";

            MD5Sample md5Sample = new MD5Sample();

            // The main part, hashing of the given felds.
            byte[] hash = md5Sample.sign(new String[] {
                        thePartnerID, theDate,theKeyPhase },
                        thePartnerSecret);
```

```
           if (hash != null) {
                String theSignature = md5Sample.hashToString
(hash);

                // Debugging output...
                logger.info("Hash signature (as string): " +
theSignature);
           }
      }

    /**
     * Signs a given set of parameter fields.
     *
     * @param theFields Set of fields that are included in the
signature and
     * are to be signed.
     * @param theSignatureKey Signature key, added to the list
of fields.
     * @return byte[] Signature of the field set.
     */
      public byte[] sign(String[] theFields, String
theSignatureKey) {

           if (theFields != null && theFields.length > 0) {

                // Prepare a string buffer to assemble fields
string
                StringBuffer fieldsString = new StringBuffer
(MAX_FIELD_STRING_SIZE);

                // move all fields to be signed into string
buffer with separators
                for (int i = 0; i < theFields.length; i++) {
                     if (theFields[i] != null) {
                          fieldsString.append(theFields
[i].trim());
                     }
                     fieldsString.append(FIELDS_SEPARATOR);
                }
                fieldsString.append(theSignatureKey);

                // Debugging output....
                logger.info("String representation of fields:
" + fieldsString.toString());

                try {

                     // Get the hashing algorithm
                     MessageDigest theDigest =
MessageDigest.getInstance(MD5);
                     theDigest.reset();

                     // hash the fields string
                     theDigest.update(fieldsString.toString
().getBytes(ENCODING_STYLE));
```

```
                        return theDigest.digest();

                  } catch (NoSuchAlgorithmException nsae) {
                        logger.log(Level.SEVERE, nsae.toString(),
nsae);
                  } catch (UnsupportedEncodingException uee) {
                        logger.log(Level.SEVERE, uee.toString(),
uee);
                  }
            }

            // Some sort of error occurred, just return null
            return null;
      }

      /**
       * Converts the byte hash into a string representation
       * (human-readable), Base16 encoding truncated to 8
characters.
       * @param hash The hash (byte string) to be converted.
       * @return String String representation of the hash.
       */
      public String hashToString(byte[] hash) {

            StringBuffer hashStringRepresentation = new
StringBuffer();

            for (int i = 0; i < hash.length; i++) {
                  String hexByteString = Integer.toHexString
(0xFF & hash[i]);
                  if (hexByteString.length() == 1) {
                        hashStringRepresentation.append("0");
                  }
                  hashStringRepresentation.append
(hexByteString);
            }

            return hashStringRepresentation.substring(0,
MAX_HASH_STRING_SIZE);
      }
}
```