CCSS
Centre commun de la sécurité sociale

Adresse postale : L-2975 Luxembourg
Guichets : 4, rue Mercier L-2144 Luxembourg
Heures d'ouverture : de 08h00 à 16h00
www.ccss.lu | ccss@secu.lu | Tél.: 40141-1 | Fax : 404481
DÉPARTEMENT ADMINISTRATIF

Structure du numéro d'identité (« matricule »)

Informations générales

La loi du 19 juin 2013 relative à l'identification des personnes physiques prévoit entre autre une modification de la structure du matricule national. L'entrée en vigueur de ce nouvel identifiant unique est fixée au 1er juillet 2014. Il se compose de la date de naissance de la personne à laquelle il est attribué, de trois chiffres séquentiels et de deux numéros de contrôle. Cette loi ne s'applique pas aux matricules employeurs à moins qu'il ne s'agisse de sociétés en nom personnel.

Tous les employeurs doivent utiliser le nouvel identifiant depuis le 1er juillet 2014.

Personnes morales - structure [1]

Le numéro d'identification des personnes morales est sur 13 positions avec :

a) l'année de la constitution ou, pour les personnes morales étrangères, celle de leur apparition sur le rôle d'une administration publique ou d'un établissement de sécurité sociale luxembourgeois autorisés à employer le numéro, année exprimée par 4 chiffres ou 4 zéros, selon que l'année de constitution est connue ou non ;
b) la forme juridique codifiée exprimée par 2 chiffres (20 à 99) ;
c) un numéro d'ordre à 4 chiffres distinguant les personnes morales constituées la même année (0001 à 9999) ;
d) un indicatif autovérificateur à une position numérique.

L'indicatif autovérificateur correspond à la différence entre 11 et le reste de la division par 11 de la somme des produits obtenus en multipliant chacun des 10 premiers chiffres du numéro d'identité par les facteurs respectifs 5, 4, 3, 2, 7, 6, 5, 4, 3 étant entendu que les numéros engendrant, lors de la division précitée, un reste de 1 ne sont pas attribués. Un reste de division zéro constitue le chiffre de contrôle.
Au cas où l'attribution du numéro suivant les modalités prévues dans le cadre du présent article s'avère impossible, il appartient au Centre Informatique de l'Etat d'attribuer un numéro d'après des critères alternatifs qu'il détermine et qui sont à approuver par le Ministre ayant dans ses attributions le Centre Informatique de l'Etat.

Personnes physiques - structure [2]

Le numéro d'identification des personnes physiques est sur 13 positions et se compose de 11 numériques identifiants et de 2 numériques de contrôle : AAAAMMJJXXXC_1C_2 avec :

- AAAA = année de naissance
- MM = mois de naissance
- JJ = jour de naissance
- XXX = numéro aléatoire unique par date de naissance
- C_1 = numéro de contrôle calculé sur AAAAMMJJXXX suivant l'algorithme LUHN 10
- C_2 = numéro de contrôle calculé sur AAAAMMJJXXX suivant l'algorithme VERHOEFF

Personnes physiques - algorithme

```
/**
 * Apply Luhn algorithm to compute check digit
 * This algorithm is used to compute the first check digit
 * during extended unique id generation
 *
 * @author Ciedmdr
 */
public class CheckDigitLuhn {
    /**
     * Computes the checksum C according Luhn algorithm
```

Bases légales :
[1] Règlement grand-ducal du 16 mars 1994
[2] Loi du 19 juin 2013 relative à l'identification des personnes physiques

```
     * @param String charset to compute Luhn check digit
     * @return the check digit
     */
    public static int computeCheckDigit(String iNumber ) {
        int checkSum = 0;
        int weight = 0;
        int weightedDigit = 0;
        for(int pos=0;pos<iNumber.length();pos++) {
            weight  = (pos%2==0)?2:1;
            weightedDigit = Character.digit(iNumber.charAt(iNumber.length()-pos-1),10) * weight;
            checkSum += (weightedDigit>9?weightedDigit-9:weightedDigit);
        }
        return (10 - checkSum%10) % 10;
    }
    /**
     * Verify the number in parameter (11 DIGITS + Luhn check digit = 12 DIGITS)
     * @param iNumber
     * @return true if checked
     */
    public static boolean checkDigit( String iNumber ) {
        int checkSum        = 0;
        int weight          = 0;
        int weightedDigit   = 0;
        for(int pos=0;pos<iNumber.length();pos++) {
            weight          =   (pos%2==0)?1:2;
            weightedDigit   =   Character.digit(iNumber.charAt(iNumber.length()-pos-1),10) * weight;
            checkSum        +=  (weightedDigit>9?weightedDigit-9:weightedDigit);
        }
        if (checkSum % 10 == 0)
            return true;
        else
            return false;
    }
}

/**
 * Apply Verhoeff algorithm to compute check digit
 * This algorithm is used to compute the second check digit during extended unique id generation
 * @author Ciedmdr
 */
public class CheckDigitVerhoeff {
    private static int inv( int iPos ) {
        int invTable[] = {0,4,3,2,1,5,6,7,8,9};
        return invTable[iPos];
    }
    private static int d(int j, int k) {
        int dTable[][] =  { {0,1,2,3,4,5,6,7,8,9},
                    {1,2,3,4,0,6,7,8,9,5},
                    {2,3,4,0,1,7,8,9,5,6},
                    {3,4,0,1,2,8,9,5,6,7},
                    {4,0,1,2,3,9,5,6,7,8},
                    {5,9,8,7,6,0,4,3,2,1},
                    {6,5,9,8,7,1,0,4,3,2},
                    {7,6,5,9,8,2,1,0,4,3},
                    {8,7,6,5,9,3,2,1,0,4},
                    {9,8,7,6,5,4,3,2,1,0}};
        return dTable[j][k];
    }
    private static int p(int i, int Ni) {
        int pTable[][] = {  {0,1,2,3,4,5,6,7,8,9},
                    {1,5,7,6,2,8,3,0,9,4},
                    {5,8,0,3,7,9,6,1,4,2},
                    {8,9,1,6,0,4,3,5,2,7},
                    {9,4,5,3,1,2,6,8,7,0},
                    {4,2,8,6,5,7,3,9,0,1},
                    {2,7,9,3,8,0,6,4,1,5},
                    {7,0,4,6,9,1,3,2,5,8}};
        return pTable[i % 8][Ni];
    }
    /**
     * Computes the checksum C as
     * C = inv(F_n (a_n)×F_(n-1) (a_(n-1) )×… ×F_1 (a_1 ) )
     * (with × being the multiplication in D_5)
     * @param String charset to compute Verhoeff check digit
     * @return the check digit
     */
    public static int computeCheckDigit(String iNumber ) {
        int checkSum = 0;
        for(int pos = 0; pos < iNumber.length(); pos++) {
            checkSum = d(checkSum,p(pos+1, Character.digit(iNumber.charAt(iNumber.length()-pos-1),10)));
        }
        return inv(checkSum);
    }
    /**
     * Verify the number in parameter (11 DIGITS + Verhoeff check digit = 12 DIGITS)
     * The verification computes and verified the following equation
     *  (F_n (a_n )×F_(n-1) (a_(n-1) )×…×F_1 (a_1 )×C) = 0
     * @param iNumber
     * @return true if checked
     */
    public static boolean checkDigit(String iNumber) {
        int checkSum = 0;
        for(int pos = 0; pos < iNumber.length(); pos++) {
            checkSum = d(checkSum,p(pos, Character.digit(iNumber.charAt(iNumber.length()-pos-1),10)));
        }
        if( checkSum == 0) {
            return true;
        }
        return false;
    }
}
```