ALGORITMO IUD

1 ABBREVIAZIONI E TERMINI

- CDA: Clinical Document Architecture
- IUD: Identificativo Unico di Documento
- SSN: Servizio Sanitario Nazionale
- GMT: Greenwich Mean Time
- ASL: Azienda Sanitaria Locale

2 RIFERIMENTI

[1] docB-SpecificheIntegrazioneServiceGatewayWSDL

3 INTRODUZIONE

Scopo del documento è la definizione di un codice univoco per l'identificazione dei documenti clinici CDA. Tale codice detto IUD deve avere la caratteristica di essere univoco nell'intero dominio sanitario nazionale nonché rispettare alcuni requisiti tecnici di stampa per mezzo di codice a barre, all'interno di documenti cartacei come il modulo prescrittivo SSN utilizzato per emettere una ricetta.

L'identificativo univoco del documento CDA è contenuto del tag <id> della classe ClinicalDocument, come per esempio:

```
<!--
    tag id (OBBLIGATORIO):
      - root: OID HL7 Regione Sardegna, ramo identificativi documenti
            elettronici
      - extension: identificativo univoco del documento
      - assigningAuthorityName: Regione Sardegna (opzionale)
-->
<id
  root = "2.16.840.1.113883.2.9.2.200.4.4"
  extension = "[IUD]"
  assigningAuthorityName = "Regione Sardegna"/>
```

Il tag contiene l'attributo root che identifica univocamente la Regione Sardegna e l'attributo extension che contiene l'identificativo univoco del documento. In particolare il valore della root può cambiare in dipendenza della tipologia di documento che si sta costruendo e cioè:

- Documento di Prescrizione (Farmaceutica/Specialistica/Ricovero):
  root := 2.16.840.1.113883.2.9.4.3.6 (MEF, ramo identificativi delle prescrizioni elettroniche)
- Documento di Erogazione (Farmaceutica/Specialistica):
  root := 2.16.840.1.113883.2.9.4.3.4 (MEF, ramo identificativi PLG - Cittadini iscritti SSN)
  root := 2.16.840.1.113883.2.9.4.3.5 (MEF, ramo identificativi PLG - Cittadini iscritti SASN)
- Documento di Accettazione/SDO:
  root := 2.16.840.1.113883.2.9.2.200.4.6 (Regione Sardegna, ramo identificativi nosologico)
- Documento di Prenotazione:
  root := 2.16.840.1.113883.2.9.2.200.4.9 (Regione Sardegna, ramo identificativi prenotazioni)

- Altri documenti [Referti, Lettera di Dimissione, PS-SSI, PS-EDS, Annullamento; Consenso Generale, Revoca Generale, Restrizione di Accesso (Revoca Puntuale), Certificati INPS/INAIL]
  root := 2.16.840.1.113883.2.9.2.200.4.4 (Regione Sardegna, ramo identificativi documenti)

Il documento **Errore. L'origine riferimento non è stata trovata.** già definisce una ipotesi di normalizzazione dello IUD che comunque non soddisfa tutti i vincoli di applicabilità espressi in questo documento.

4 DEFINIZIONE CODICE IUD

Il codice IUD costituisce un identificativo univoco del documento CDA composto da 16 caratteri appartenenti ad un alfabeto di 61 simboli come definito in seguito:

ALFABETO :=

{1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,y,x,z,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,Y,X,Z}

La cifra numerica 0 (zero) è stata esclusa dall'alfabeto per non generare confusione con i caratteri alfabetici o e O.

Per soddisfare il requisito di poter essere stampato mediante codice a barre sulla ricetta, il codice IUD non può essere più lungo di 16 cifre. Queste sono ottenute dalla concatenazione di tre elementi:

- un primo composto da 7 caratteri che rappresenta il codice operatore;
- un secondo composto da 8 caratteri che rappresenta il timestamp, ossia il numero di millisecondi trascorsi dal 01 Gennaio 1970, 00:00:00 GMT;
- un ultimo elemento composto da un solo carattere che rappresenta un codice di controllo ottenuto dai precedenti caratteri.

L'utilizzo di uno IUD così composto permette la stampa del codice a barre corrispondente all'interno di un'area libera del modulo prescrittivo SSN. L'utilizzo di un codice con più cifre non consentirebbe la sua stampa a meno di non utilizzare particolari tipologie di codici a barre che però risulterebbero illeggibili dai più comuni lettori ottici.

La struttura dello IUD, allora, risulta essere la seguente:

IUD ::= [XXXXXXX][YYYYYYYY][C], in cui

--> [XXXXXXX] ::= CODICE_OPERATORE$_{61}$

--> [YYYYYYYY] ::= TIMESTAMP$_{61}$

--> [C] ::= CODICE_CONTROLLO$_{61}$

dove:

- CODICE_OPERATORE$_{61}$ è la rappresentazione base 61 del relativo codice operatore in base 10
- TIMESTAMP$_{61}$ è la rappresentazione base 61 del relativo timestamp in base 10
- CODICE_CONTROLLO$_{61}$ è il codice di controllo espresso in base 61.

Con un alfabeto di 61 simboli e quindi una codifica base$_{61}$ si ottengono i seguenti massimi valori rappresentabili in base$_{10}$:

- codice operatore: 3.142.742.836.021 valori distinti
- timestamp: 191.707.312.997.281 valori distinti

Avendo riservato 7 cifre per la rappresentazione di CODICE_OPERATORE$_{61}$ vuol dire che il corrispondente CODICE_OPERATORE$_{10}$ deve essere composto da 12 cifre: esso quindi sarà ottenenuto della concatenazione del codice ASL e di un progressivo:

CODICE_OPERATORE$_{10}$ ::= <CODICE_ASL><PROGRESSIVO>, in cui

--> <CODICE_ASL> ::= 6 cifre decimali relative al codice ISTAT che identifica la ASL

(ad es. 200108 – ASL di Cagliari)

--> <PROGRESSIVO> ::= 6 cifre decimali a comporre un progressivo a partire da 000001

Il codice ASL è un codice definito a livello nazionale dal Ministero della Salute e quindi univoco.

Il CODICE_OPERATORE$_{10}$ è definito all'autorità responsabile dell'assegnazione di tale identificatore ad ogni suo operatore (quindi per Medir, così come definito dall'anagrafe operatori –ASO -, al momento della creazione dell'anagrafica relativa) e ha la caratteristica di essere univoco all'interno del dominio sanitario nazionale. Sarebbe comunque possibile utilizzare una diversa modalità di assegnazione di tale codice purché continui a valere la sua univocità a livello nazionale e il suo numero massimo di valori rappresentabili come espresso in precedenza.

Nel caso di Medir, quindi, il CODICE_OPERATORE$_{10}$ è quello assegnato dall'anagrafe operatori di Medir. Per gli scopi dell'integrazione degli applicativi client, ai quali è data responsabilità di generare il codice IUD, qualora il client non disponga di tale valore per il calcolo del codice IUD secondo l'algoritmo definito, è possibile ottenere il CUR dell'operatore sanitario mediante invocazione dell'operazione IdentifyPersonnel del servizio PersonnelManagement (cf. [1] **Errore. L'origine riferimento non è stata trovata.**).

Infine, le 8 cifre riservate per la rappresentazione del TIMESTAMP$_{61}$ consentono di poter utilizzare 14 cifre per rappresentare l'equivalente TIMESTAMP$_{10}$ e quindi poter esprimere valori fino al 17/12/8044 04:23:17:0280:

TIMESTAMP$_{10}$ ::= 14 cifre decimali

Un esempio di codice IUD ottenuto secondo quanto definito è il seguente. Dati:

- CODICE_ASL = 200108
- PROGRESSIVO = 123456
- CODICE_OPERATORE$_{10}$ = 200108123456
- TIMESTAMP$_{10}$ = 1193405602612
- CODICE_OPERATORE$_{61}$ = 4SVAnQh
- TIMESTAMP$_{61}$ = 1oaZmubl
- CODICE_CONTROLLO$_{61}$ = P

si ottiene: IUD = 4SVAnQh1oaZmublP

5 ALGORITMO DI GENERAZIONE

Indichiamo con N_b la rappresentazione del numero N in base b.

Consideriamo la base 61 come base per la rappresentazione del codice IUD e consideriamo il seguente alfabeto come l'insieme di simboli che consentono di esprimere un numero N in base 61:

ALFABETO =

{1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,y,x,z,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,Y,X,Z}

Consideriamo noto il CODICE_OPERATORE$_{10}$. La generazione IUD dovrà implementare i seguenti step:

1. Generazione del TIMESTAMP$_{10}$ come millisecondi trascorsi dal 01 Gennaio 1970 00:00:00 GMT
2. Conversione del CODICE_OPERATORE$_{10}$ in base 61 rappresentando anche gli "zeri" non significativi in modo da ottenere sempre la sequenza di 7 cifre del CODICE_OPERATORE$_{61}$. Lo "zero" dell'alfabeto utilizzato corrisponde al simbolo 1.
3. Conversione del TIMESTAMP$_{10}$ in base 61 rappresentando anche in questo caso gli "zeri" non significati in modo da ottenere sempre la sequenza di 8 cifre del TIMESTAMP$_{61}$
4. Concatenazione di CODICE_OPERATORE$_{61}$ e TIMESTAMP$_{61}$ in modo da ottenere la sequenza di 15 caratteri SEQUENZA$_{61}$

5. Generazione del codice di controllo $CODICE\_CONTROLLO_{10}$ ottenuto come somma dei valori decimali di ogni singolo carattere della $SEQUENZA_{61}$ e applicando al risultato finale l'operatore di modulo 61 (il valore decimale di ogni singolo carattere della sequenza è dato dalla sua posizione all'interno dell'alfabeto definito).
6. Conversione del codice di controllo $CODICE\_CONTROLLO_{10}$ in base 61 ottenendo $CODICE\_CONTROLLO_{61}$
7. Concatenazione di $SEQUENZA_{61}$ con $CODICE\_CONTROLLO_{61}$ per ottenere il codice finale IUD a 16 cifre

5.1 Implementazione in linguaggio Java

```
static public String generaIUD(String codiceOperatore10)
{
  String ALFABETO =
        "123456789abcdefghijklmnopqrstuvwyxzABCDEFGHIJKLMNOPQRSTUVWYXZ";
  char[] alfabetoSimbols = ALFABETO.toCharArray();
  long base = alfabetoSimbols.length;
  int MAX_LENGTH_COD_OP = 7;
  int MAX_LENGTH_TIMESTAMP = 8;

   //Codice dell'operatore in base 10
   long CODICE_OPERATORE10 = Long.valueOf(codiceOperatore10);

   /* ================================================================
    * 1. Generazione del TIMESTAMP10 come millisecondi trascorsi dal
    *    1 Gennaio 1970 00:00:00 GMT
    **/
   Calendar c = Calendar.getInstance(TimeZone.getTimeZone("GMT"));
   long TIMESTAMP10 = c.getTimeInMillis();

   /* ================================================================
    *  2. Conversione del CODICE_OPERATORE10 in base 61 rappresentando
    *     anche gli "zeri" non significativi in modo da ottenere sempre la
    *     sequenza di 7 cifre del CODICE_OPERATORE61. Lo "zero"
    *     dell'alfabeto utilizzato corrisponde al simbolo 1.
    **/
   StringBuffer textBuffer = new StringBuffer("1111111");
   int i = MAX_LENGTH_COD_OP;
   while (CODICE_OPERATORE10 >= base) {
     textBuffer.setCharAt(--i,
                 alfabetoSimbols[(int)(CODICE_OPERATORE10 % base)]);
     CODICE_OPERATORE10 = CODICE_OPERATORE10 / base;
   }
   textBuffer.setCharAt(--i,alfabetoSimbols[(int)CODICE_OPERATORE10]);

   String CODICE_OPERATORE61 = textBuffer.toString();

   /* ================================================================
    *  3. Conversione del TIMESTAMP10 in base 61 rappresentando anche in
    *     questo caso gli "zeri" non significati in modo da ottenere sempre
    *     la sequenza di 8 cifre del TIMESTAMP61
    **/
   textBuffer = new StringBuffer("11111111");
   i = MAX_LENGTH_TIMESTAMP;

   while (TIMESTAMP10 >= base) {
      textBuffer.setCharAt(--i,
```

```
                alfabetoSimbols[(int)(TIMESTAMP10 % base)]);
      TIMESTAMP10 = TIMESTAMP10 / base;
   }
   textBuffer.setCharAt(--i,alfabetoSimbols[(int)TIMESTAMP10]);

   String TIMESTAMP61 = textBuffer.toString();

   /* ================================================================
    *  4. Concatenazione di CODICE_OPERATORE61 e TIMESTAMP61 in modo da
    *     ottenere la sequenza di 15 caratteri SEQUENZA61
    **/
   String  SEQUENZA61 = CODICE_OPERATORE61 + TIMESTAMP61;

   /* ================================================================
    *  5. Generazione del codice di controllo CODICE_CONTROLLO10  ottenuto
    *     come somma dei valori decimali di ogni singolo carattere della
    *     SEQUENZA61 e applicando al risultato finale l'operatore di modulo
    *     61 (il valore decimale di ogni singolo carattere della sequenza è
    *     dalla sua posizione all'interno dell'alfabeto definito).
    **/
   int length = SEQUENZA61.length();
   int j = 0;
   long CODICE_CONTROLLO10 = 0;
   while (j < length) {
      CODICE_CONTROLLO10 += ALFABETO.indexOf("" + SEQUENZA61.charAt(j++));
   }

   /* ================================================================
    *  6. Conversione del codice di controllo CODICE_CONTROLLO10  in base
    *     61 ottenendo CODICE_CONTROLLO61
    **/
   String CODICE_CONTROLLO61 = "" +
          ALFABETO.charAt((int)CODICE_CONTROLLO10 % ALFABETO.length());

   /* ================================================================
    *  7.Concatenazione di SEQUENZA61 con CODICE_CONTROLLO61 per ottenere
    *    il codice finale IUD a 16 cifre
    **/
    return SEQUENZA61 + CODICE_CONTROLLO61;
}
```

5.2 Implementazione in linguaggio C#

```
public static String GeneraIUD(String codiceOperatore)
{
   String ALFABETO =
          "123456789abcdefghijklmnopqrstuvwyxzABCDEFGHIJKLMNOPQRSTUVWYXZ";
   char[] alfabetoSimbols = ALFABETO.ToCharArray();
   long base61 = alfabetoSimbols.Length;
   int MAX_LENGTH_COD_OP = 7;
   int MAX_LENGTH_TIMESTAMP = 8;

   //Codice dell'operatore in base 10
   long CODICE_OPERATORE10 = long.Parse(codiceOperatore);

   /* ================================================================
    * 1. Generazione del TIMESTAMP10 come millisecondi trascorsi dal
    *    01 Gennaio 1970 00:00:00 GMT
```

```
 **/
DateTime d = DateTime.UtcNow;
DateTime epoc = new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);
TimeSpan n = d.Subtract(epoc);
long TIMESTAMP10 = (long)n.TotalMilliseconds;

/* ========================================================================
 *  2. Conversione del CODICE_OPERATORE10 in base 61 rappresentando anche
 *     gli “zeri“ non significativi in modo da ottenere sempre la sequenza
 *     di 7 cifre del CODICE_OPERATORE61. Lo  “zero” dell’alfabeto
 *     utilizzato corrisponde al simbolo 1.
 **/
String text = "";

while (CODICE_OPERATORE10 >= base61)
{
   text = alfabetoSimbols[(int)(CODICE_OPERATORE10 % base61)] + text;
   CODICE_OPERATORE10 = CODICE_OPERATORE10 / base61;
}
text = alfabetoSimbols[(int)CODICE_OPERATORE10] + text;
String CODICE_OPERATORE61 = text.PadLeft(MAX_LENGTH_COD_OP, '1');

/* ========================================================================
 *  3. Conversione del TIMESTAMP10 in base 61 rappresentando anche in questo
 *     caso gli “zeri” non significati in modo da ottenere sempre la
 *     sequenza di 8 cifre del TIMESTAMP61
 **/
text = "";
while (TIMESTAMP10 >= base61)
{
   text = alfabetoSimbols[(int)(TIMESTAMP10 % base61)] + text;
   TIMESTAMP10 = TIMESTAMP10 / base61;
}
text = alfabetoSimbols[(int)TIMESTAMP10] + text;
String TIMESTAMP61 = text.PadLeft(MAX_LENGTH_TIMESTAMP, '1'); ;

/* ========================================================================
 *  4. Concatenazione di CODICE_OPERATORE61 e TIMESTAMP61 in modo da
 *     ottenere la sequenza di 15 caratteri SEQUENZA61
 **/
String SEQUENZA61 = CODICE_OPERATORE61 + TIMESTAMP61;

/* ========================================================================
 *  5. Generazione del codice di controllo CODICE_CONTROLLO10  ottenuto
 *     come somma dei valori decimali di ogni singolo carattere della
 *     SEQUENZA61 e applicando al risultato finale l’operatore di modulo 61
 *     (il valore decimale di ogni singolo carattere della sequenza è dato
 *     dalla sua posizione all’interno dell’alfabeto definito).
 **/
int length = SEQUENZA61.Length;
int j = 0;
long CODICE_CONTROLLO10 = 0;
while (j < length)
{
   CODICE_CONTROLLO10 += ALFABETO.IndexOf(SEQUENZA61.Substring(j++, 1));
}
```

```
   /* ===================================================================
    *  6. Conversione del codice di controllo CODICE_CONTROLLO10  in base 61
    *     ottenendo CODICE_CONTROLLO61
    **/
    String CODICE_CONTROLLO61 =
        "" + ALFABETO.Substring((int)CODICE_CONTROLLO10 % ALFABETO.Length, 1);

    /* ===================================================================
     *  7. Concatenazione di SEQUENZA61 con CODICE_CONTROLLO61 per ottenere
     *     il codice finale IUD a 16 cifre
     **/
    return SEQUENZA61 + CODICE_CONTROLLO61;
}
```

6 ALGORITMO DI VERIFICA

Per la verifica della correttezza del codice IUD occorre utilizzare il $CODICE\_CONTROLLO_{61}$ nel seguente modo: dato il codice IUD a 16 cifre

1. Generazione del codice di controllo $CODICE\_CONTROLLO\_VERIFICA_{10}$ ottenuto come somma dei valori decimali di ogni singolo carattere della sequenza dei primi 15 caratteri del codice IUD e applicando al risultato finale l'operatore di modulo 61 (il valore decimale di ogni singolo carattere della sequenza è dato dalla sua posizione all'interno dell'alfabeto definito).
2. Conversione del codice di controllo $CODICE\_CONTROLLO\_VERIFICA_{10}$ in base 61 ottenendo $CODICE\_CONTROLLO\_VERIFICA_{61}$
3. Confronto del $CODICE\_CONTROLLO\_VERIFICA_{61}$ con $CODICE\_CONTROLLO_{61}$ ottenuto dalla 16-esima cifra dello IUP.
4. Se i due codici differiscono il codice IUP non è un codice valido.

6.1 Implementazione in linguaggio Java

```
static public boolean checkValidity(String iud)
{
   String ALFABETO =
          "123456789abcdefghijklmnopqrstuvwyxzABCDEFGHIJKLMNOPQRSTUVWYXZ";

   /* ===================================================================
    *  1. Generazione del codice di controllo CODICE_CONTROLLO_VERIFICA10
    *     ottenuto come somma dei valori decimali di ogni singolo carattere
    *     della sequenza dei primi 15 caratteri del codice IUP e applicando
    *     al risultato finale l'operatore di modulo 61 (il valore decimale di
    *     ogni singolo carattere della sequenza è dato dalla sua posizione
    *     all'interno dell'alfabeto definito).
    **/
   int length = iud.length() - 1;
   int j = 0;
   long CODICE_CONTROLLO_VERIFICA10 = 0;
   while (j < length)
   {
     CODICE_CONTROLLO_VERIFICA10 += ALFABETO.indexOf("" + iud.charAt(j++));
   }

   /* ===================================================================
```

```
    *  2. Conversione del codice di controllo CODICE_CONTROLLO_VERIFICA10
    *     in base 61 ottenendo CODICE_CONTROLLO_VERIFICA61
    **/
   String CODICE_CONTROLLO_VERIFICA61 = "" +
     ALFABETO.charAt((int)CODICE_CONTROLLO_VERIFICA10 % ALFABETO.length());

   /* ================================================================
    * 3. Confrontare il CODICE_CONTROLLO_VERIFICA61 con CODICE_CONTROLLO61
    *    ottenuto dalla 16-esima cifra dello IUP.
    * 4. Se i due codici differiscono il codice IUD non è valido.
    **/
   return CODICE_CONTROLLO_VERIFICA61.equals(iud.substring(15,16));
}
```

6.2 Implementazione in linguaggio C#

```
static public bool checkValidity(String iud)
{
   String ALFABETO =
          "123456789abcdefghijklmnopqrstuvwyxzABCDEFGHIJKLMNOPQRSTUVWYXZ";

   /* ================================================================
    *  1. Generazione del codice di controllo CODICE_CONTROLLO_VERIFICA10
    *     ottenuto come somma dei valori decimali di ogni singolo carattere
    *     della sequenza dei primi 15 caratteri del codice IUP e applicando
    *     al risultato finale l'operatore di modulo 61 (il valore decimale di
    *     ogni singolo carattere della sequenza è dato dalla sua posizione
    *     all'interno dell'alfabeto definito).
    **/
   int length = iud.Length - 1;
   int j = 0;
   long CODICE_CONTROLLO_VERIFICA10 = 0;
   while (j < length)
   {
      CODICE_CONTROLLO_VERIFICA10 +=
          ALFABETO.IndexOf("" + iud.Substring(j++,1));
   }

   /* ================================================================
    *  2. Conversione del codice di controllo CODICE_CONTROLLO_VERIFICA10
    *     in base 61 ottenendo CODICE_CONTROLLO_VERIFICA61
    **/
   String CODICE_CONTROLLO_VERIFICA61 = "" +
    ALFABETO.Substring((int)CODICE_CONTROLLO_VERIFICA10 % ALFABETO.Length, 1);

   /* ================================================================
    * 3. Confrontare il CODICE_CONTROLLO_VERIFICA61 con CODICE_CONTROLLO61
    *    ottenuto dalla 16-esima cifra dello IUP.
    * 4. Se i due codici differiscono il codice IUD non è valido.
    **/
   return CODICE_CONTROLLO_VERIFICA61 == iud.Substring(15, 1);
}
```

7 STAMPA SU MODULO RICETTA SSN

Il codice IUD di 16 cifre è stampato tramite codice a barre sul modulo prescrittivo SSN in un'area libera di ridotte dimensioni sul lato sinistro dello spazio riservato al codice fiscale. Tale codice a barre sarà successivamente utilizzato nei processi di ritrovamento e associazione attraverso la sua lettura da parte di operatore sia ottica per mezzo di dispositivi per la lettura di codici a barre sia manuale attraverso tastiera.

Vi sono tre vincoli fondamentali che il codice IUD deve soddisfare:

- Stampa attraverso codice a barre in un'area di ridotte dimensioni.
- Leggibilità per mezzo dei più diffusi lettori ottici di codici a barre, quindi compatibilità.
- Leggibilità da parte di un operatore umano per la sua immissione tramite tastiera.

Per soddisfare i tre vincoli si utilizza:

- Un codice a barre nel formato "Code 128", il quale può codificare fino a 128 simboli per ogni cifra utilizzata.
- Il formato "Code 128" è una delle più diffuse codifiche per codici a barre e quindi supportata dai più diffusi lettori di codice a barre.
- La lunghezza del codice a barre in "Code 128" corrispondente alle 16 cifre dello IUD ha una dimensione tale da essere contenuto nell'area libera a sinistra del codice fiscale sul modulo di prescrizione SSN.
- Sul lato superiore del codice a barre e allineato a destra rispetto allo stesso codice si riportano le corrispondenti cifre dello IUD in modo da consentire una lettura ed immissione da tastiera dell'operatore.

Un esempio di codice a barre ottenuto dall'esempio riportato nel paragrafo 4 è il seguente:

L'immagine in Figura riportata in seguito mostra un esempio di stampa del codice IUD sul modulo di prescrizione SSN.

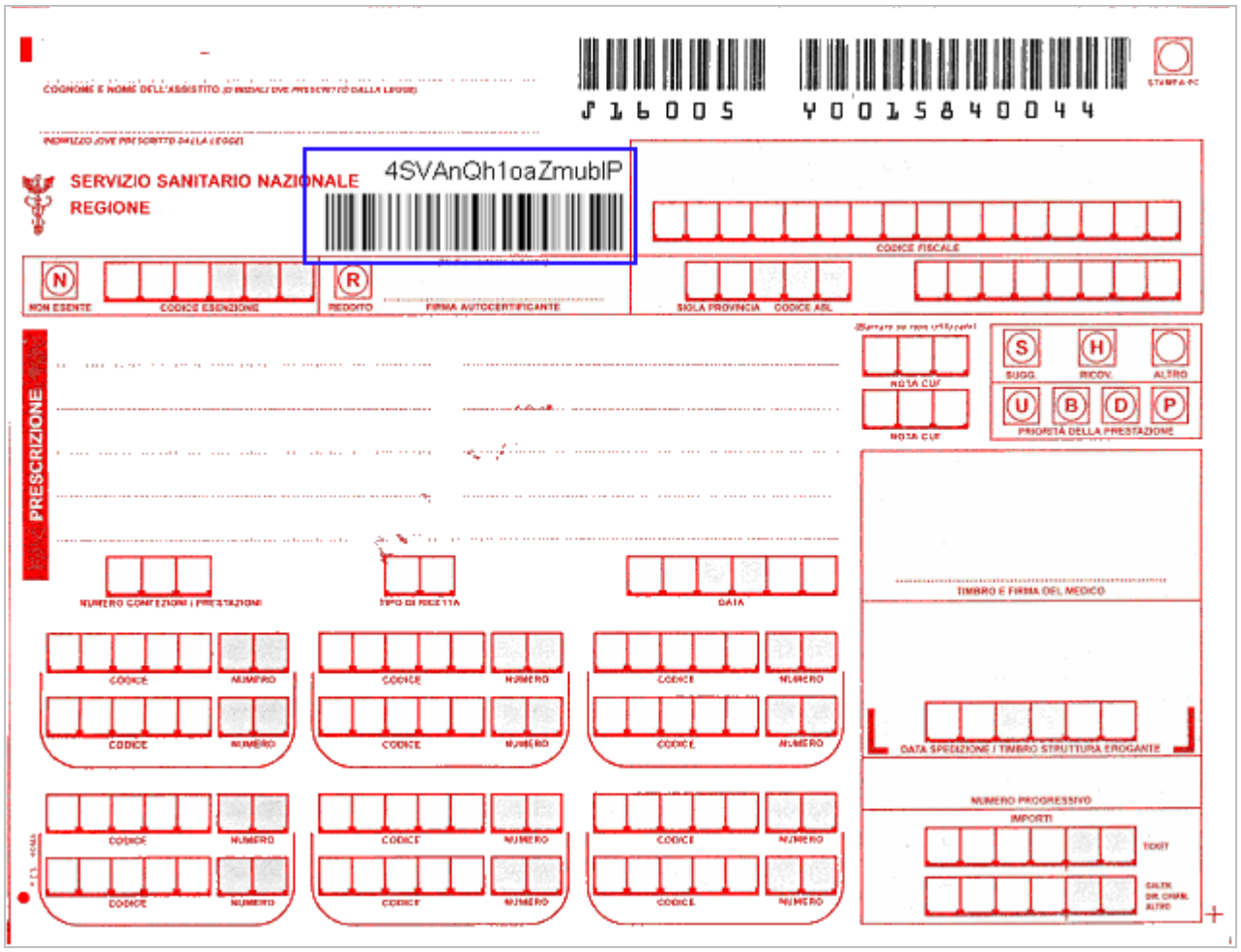

Figura 1 - Modulo prescrizione SSN