Consiglio Nazionale delle Ricerche

Il nuovo Sistema di Controllo Sintattico Ver. 1

A. Del Soldato, M. Martinelli, S. Ruberti, I.Serrecchia

IIT TR-16/2004

Technical Report

Dicembre 2004

Istituto di Informatica e Telematica

INDICE:

1 Premessa

In Italia la registrazione del "country code Top Level Domain" (ccTLD) 'it' e il servizio di assegnazione e mantenimento dei nomi a dominio è svolto dalla Registration Authority (di seguito indicata come RA).

Le attività del Registro vengono svolte dal Consiglio Nazionale delle Ricerche (CNR) tramite l'Istituto di Informatica e Telematica (IIT) al quale IANA (Internet Assigned Number Authority) ora diventato ICANN (Internet Corporation for Assigned Names and Numbers) ha delegato il servizio sulla base di riconosciute competenze acquisite dai tecnici dello IIT che sono stati i promotori, a partire dagli anni '80 della diffusione del protocollo IP nell'ambiente della ricerca italiana [MRTV99].

Le procedure per la registrazione e la modifica dei Provider/Maintainer (P/M) e dei nomi a dominio prevedono l'invio, al Registro dello IIT, di una specifica documentazione descritta nelle Regole e Procedure Tecniche di registrazione [PTR202]. Questa documentazione è soggetta a successive fasi di verifica effettuate, sia da sistemi automatici, che da operatori umani addetti al controllo di congruenza dei dati contenuti nei vari documenti.

Nelle procedure attuali, la fase di verifica più consistente è effettuata da un sistema automatico di controllo sintattico sui moduli elettronici inviati sia per la registrazione che per la modifica di un P/M o di un nome a dominio.

In questo documento è descritta questa fase di controllo sintattico ed in modo particolare è descritto il nuovo sistema software che la implementa e che integra il sistema preesistente.

I *Moduli Tecnici* inviati per la registrazione dei nomi a dominio nonché gli oggetti *mntner* per la registrazione dei P/M sono inviati al Registro via posta elettronica all'indirizzo *domain@nic.it*.

Ogni modulo (di seguito indicato genericamente con *Modulo Elettronico*) è sottoposto ad una procedura automatica (di seguito indicata semplicemente come controllo sintattico) che esegue i controlli sintattici e semantici sui dati in esso contenuti e ne invia l'esito al P/M in un messaggio di posta elettronica.

2 Scopo

Il sistema di controllo sintattico utilizzato fino ad oggi dal Registro è un programma scritto in linguaggio Perl sviluppato e distribuito da RIPE e successivamente adeguato alle esigenze del Registro attraverso la definizione di nuovi attributi all'interno delle varie classi di oggetti e l'aggiunta di controlli resi necessari dalle regole e dalle procedure tecniche di registrazione relative ai nomi a dominio in Italia.

Sebbene RIPE metta a disposizione il codice sorgente del sistema, ogni rilascio di una nuova versione del programma, implica una reintegrazione da parte del Registro delle proprie caratteristiche e funzionalità.

Inoltre, il fatto che il programma distribuito da RIPE implementi contemporaneamente, sia la parte dei controlli sui moduli che la gestione del database WHOIS, pone grossi problemi in una realtà, come quella del Registro italiano, in cui la registrazione degli oggetti nel database non è direttamente conseguente ai controlli automatici, a causa delle necessarie verifiche incrociate che sono effettuate con la documentazione cartacea.

Per questi motivi nasce l'esigenza di progettare e implementare un nuovo sistema di controllo sintattico con la stessa interfaccia, ma indipendente dalla gestione del database WHOIS ed il più possibile completo ed efficiente nei confronti dei controlli propri del Registro stesso.

La nuova progettazione del sistema permetterà l'integrazione di molti controlli precedentemente effettuati dagli operatori addetti alla registrazione e il rafforzamento del sistema stesso con l'inserimento di controlli più restrittivi sia a livello puramente sintattico, che di consistenza dei dati e di autorizzazione.

Inoltre, il fatto di avere una gestione svincolata dei controlli sui moduli di registrazione renderà indipendente la gestione del database WHOIS contenente gli oggetti registrati, che potrà, in un futuro, essere ridisegnato al fine di ottenere una maggiore efficienza e flessibilità anche in relazione alle piattaforme di supporto.

3 La registrazione e la modifica di un nome a dominio

I P/M, che intendono stipulare con la RA un servizio di registrazione di nomi a dominio, devono sottoscrivere un contratto ed inviare, via posta elettronica o via Web [MD99], un modulo specifico contenente un oggetto *mntner* [MM00] con i propri dati salienti.

Una volta che la RA ha controllato la documentazione ricevuta ed ha registrato l'oggetto *mntner* nel database WHOIS, il P/M può registrare e successivamente modificare i nomi a dominio da lui gestiti.

La registrazione di un nome a dominio, prevede due fasi principali:

a. l'invio, via fax o posta di superficie, di una Lettera di Assunzione di Responsabilità (LAR), che può essere inviata sia dal P/M che dal richiedente il nome a dominio;

b. l'invio, tramite posta elettronica o via Web [MD00], di un modulo tecnico [MM001].

A queste due fasi segue un'ulteriore fase di verifica di congruenza e correttezza dei dati inviati la quale darà seguito alla registrazione, nel database WHOIS, dell'oggetto *mntner* o dell'oggetto *domain* e degli oggetti *person* e *role* contenuti nel *Modulo Tecnico*.

Lo schema in *figura n.1* indica le varie fasi di verifica che sono effettuate nel caso di invio, da parte del P/M dell'oggetto *mntner* (per l'attivazione del contratto) o del modulo tecnico di registrazione di un nome a dominio.

I *Moduli Tecnici* inviati per la registrazione dei nomi a dominio nonchè gli oggetti *mntner* per la registrazione dei P/M sono inviati al Registro via posta elettronica all'indirizzo *domain@nic.it*.

Ogni modulo (di seguito indicato genericamente con *Modulo Elettronico*) è sottoposto ad una procedura automatica (di seguito indicata semplicemente come controllo sintattico) che esegue i controlli sintattici e semantici sui dati in esso contenuti e ne invia l'esito al P/M in un messaggio di posta elettronica.

Nel caso in cui il controllo sintattico di un *Modulo Tecnico* vada a buon fine, il modulo è sottoposto ad un'ulteriore verifica automatica per il controllo di validità dei nameserver dichiarati nell'oggetto *domain*; il risultato viene inviato via posta elettronica al P/M.

Se anche il controllo sui nameserver ha esito positivo, il modulo è sottoposto ad una ulteriore verifica di congruenza da parte di un operatore. L'operatore comunica gli eventuali errori al P/M oppure registra gli oggetti contenuti nel modulo all'interno del database WHOIS.

Le verifiche per l'oggetto *mntner* sono analoghe a quelle effettuate sul *Modulo Tecnico* ad esclusione del controllo sui nameserver.

La modifica di ogni oggetto *mntner*, *domain*, *person* e *role* registrati nel database è subordinata all'invio di un nuovo modulo elettronico contenente l'oggetto da modificare. Un modulo elettronico può contenere più oggetti appartenenti a classi diverse, solo se sono tra loro referenziati.

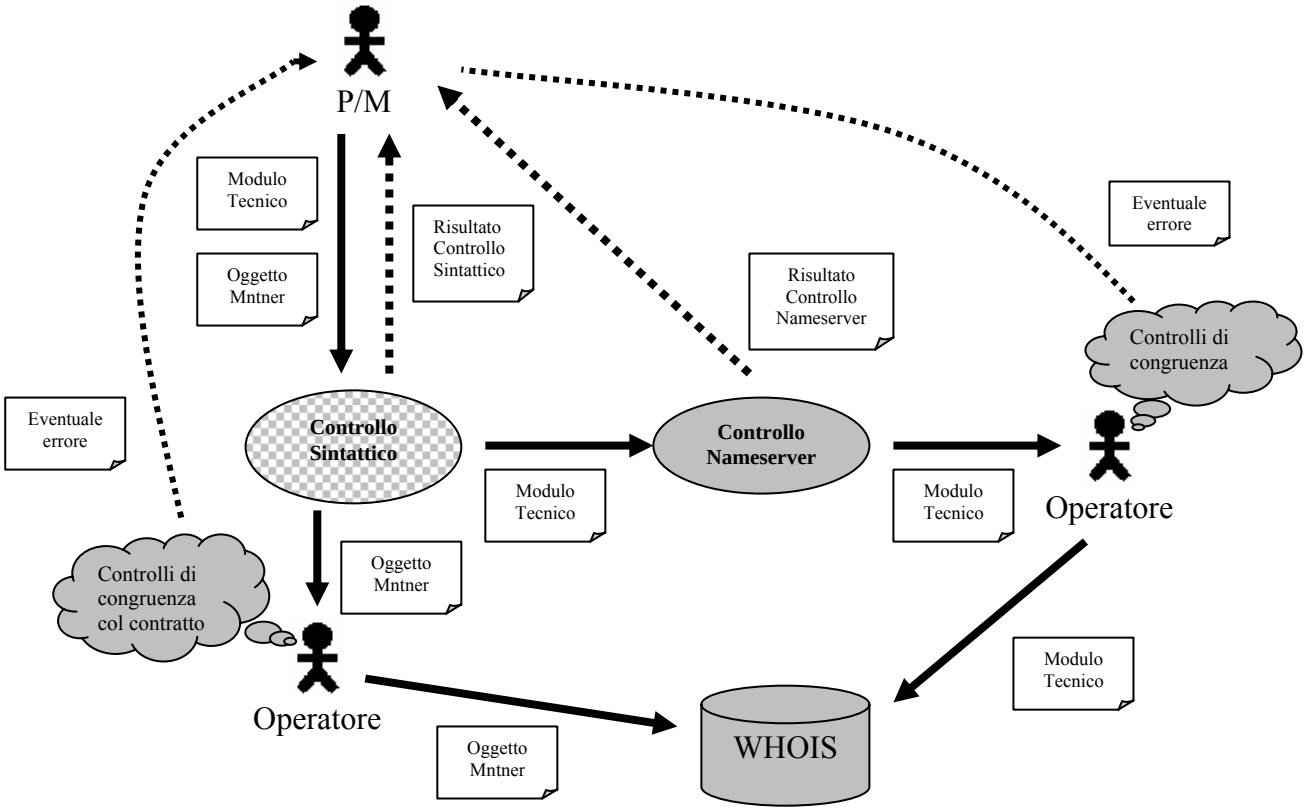


figura n.1

3.1 Nuove caratteristiche principali

Il nuovo sistema automatico per il controllo sintattico dovrà rendere il processo di registrazione più veloce e più sicuro sotto i seguenti aspetti:

a) semplificare i controlli incrociati con la documentazione cartacea alleviando il lavoro degli operatori;

b) automatizzare la maggior parte dei controlli attualmente effettuati manualmente dagli operatori, nonché l'eventuale invio delle notifiche di errore;

c) accorciare i tempi di attesa per l'eventuale rinvio di un modulo elettronico corretto, notificando ai P/M gli eventuali errori subito dopo la fase di controllo sintattico;

d) agevolare il mantenimento di dati congruenti all'interno del database WHOIS del Registro;

e) Contribuire ad eliminare alcune inconsistenze attualmente riscontrate nei dati contenuti nei database, conseguenza sia di errori umani, sia di artifici o della non adempienza stretta delle regole e procedure da parte dei P/M, come ad esempio persone registrate nel database WHOIS con *nic-handle* assegnati ad altre persone, oggetti *person* presenti nel database WHOIS ma non in quello dei *nic-handle*, numeri telefonici o codici fiscali improbabili, ecc.

In particolare, il nuovo sistema di controllo sintattico dovrà permettere di automatizzare tutte le verifiche descritte nei punti seguenti, ad eccezione di quelle che coinvolgono dati su supporto cartaceo (punti a. e b.). Questi ultimi controlli, che comunque richiedono l'intervento di un operatore, saranno agevolati facendo in modo che, se necessario, il sistema inserisca automaticamente nel modulo elettronico i contatti da analizzare, evitando così l'interrogazione manuale del database WHOIS.

a) Corrispondenza del valore inserito nell'attributo *org* di un oggetto *domain* con la denominazione della persona fisica o giuridica richiedente riportata nella Lettera di Assunzione di Responsabilità (LAR).

b) Corrispondenza tra la persona referenziata nell'attributo *admin-c* di un oggetto *domain* e colui che ha sottoscritto la LAR. Nel caso in cui l'oggetto *person* corrispondente all'*admin-c* non fosse contenuto nel modulo elettronico, si rende attualmente necessario il reperimento dei dati da parte dell'operatore attraverso un'interrogazione al database WHOIS.

c) Esistenza, del database WHOIS, di un nome a dominio diverso assegnato alla persona fisica richiedente. Questa verifica implica attualmente un'interrogazione al database WHOIS per assicurare il rispetto della regola di univocità di assegnazione di un nome a dominio ad una persona fisica.

d) Presenza all'interno del database WHOIS di tutti gli oggetti *person* che sono referenziati nel modulo elettronico e non inseriti all'interno modulo stesso. Questo controllo, necessario per assicurare che tutti i contatti tecnici e amministrativi dichiarati per i nomi a dominio ed i mntner siano anch'essi registrati all'interno del database WHOIS, richiede attualmente spesso più interrogazioni al database.

e) Controllo sulla validità dei *nic-handle* utilizzati all'interno del modulo elettronico. Al fine di garantire la validità dei *nic-handle* registrati all'interno del database WHOIS, l'operatore attualmente controlla che:

   - gli oggetti *person* inseriti in un modulo elettronico contenente un oggetto *domain* o *mntner*, siano referenziati nel modulo stesso (l'esistenza dei *nic-handle* viene attualmente verificata dal vecchio controllo sintattico);
   - i *nic-handle* associati ad ogni oggetto *person* non siano già stati assegnati ad un'altra persona.

Questi controlli che servono a garantire che nel database WHOIS non siano presenti né *nic-handle* non precedentemente richiesti né falsi *nic-handle* richiedono, attualmente, almeno un accesso ai database WHOIS e/o HANDLE per ogni oggetto *person*.

4 Il nuovo sistema di controllo sintattico

Il nuovo sistema di controllo sintattico costituisce la prima fase di controllo in tutto il sistema per la registrazione dei domini[1].

Il sistema si colloca (vedi *figura n.2*) tra il P/M, dal quale riceve il modulo elettronico e al quale comunica l'esito dei controlli e un operatore umano, oppure tra il P/M e il sistema automatico per la verifica della configurazione dei nameserver, se il modulo è un *modulo elettronico*.

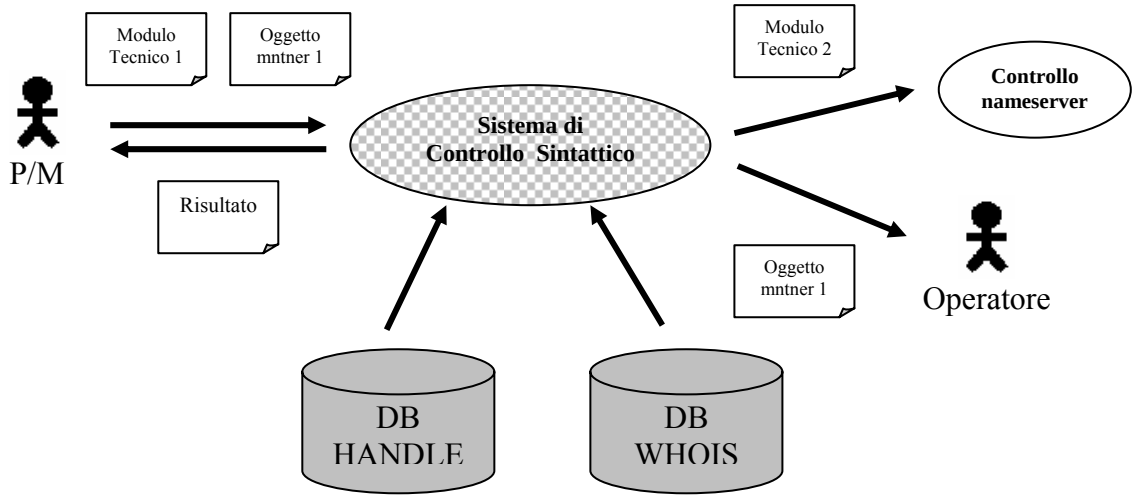


figura n.2

Come mostrato nella figura sopra riportata, il sistema si interfaccia anche con i database HANDLE e WHOIS del Registro, i quali vengono interrogati per poter effettuare alcune verifiche di consistenza dei dati presenti nel modulo e per il reperimento degli oggetti *person* eventualmente aggiunti al modulo tecnico prima che questo venga inviato per le fasi di verifica successive.

Il sistema comunica con i database del Registro tramite interrogazioni whois. I database sono il cuore del Registro e contengono tutte le informazioni relative ai nomi a dominio registrati presso il Registro italiano come i nic-handle assegnati da IT-NIC (database HANDLE), i nomi a dominio italiani, i P/M ed i loro contatti tecnici e amministrativi in oggetti di classe domain, mntner e person rispettivamente (database WHOIS).

L'interfaccia con i P/M e con il controllo dei nameserver avviene tramite messaggi di posta elettronica. Il formato del messaggio di posta che contiene la risposta del nuovo sistema sarà analogo a quello del sistema precedente, pur includendo nuovi tipi di messaggi di errore. Il modulo

[1] A parte i controlli effettuati sulla LAR da un operatore nel momento in cui essa è pervenuta al Registro, che sono indipendenti dai controllo sui moduli elettronici.

elettronico restituito in uscita verso il controllo dei nameserver sarà lo stesso inviato dal P/M opportunamente formattato e con l'eventuale aggiunta di oggetti *person* e/o *role*.

4.1 Controlli effettuati dal sistema

Il nuovo sistema di controllo sintattico riceve in ingresso un file contenente un messaggio di posta elettronica. Questo messaggio è prima formattato e ripulito e poi analizzato dal sistema in varie fasi, in ognuna delle quali sono eseguiti determinati tipi di controlli.

Le varie tipologie di controllo sono:

- ormato del messaggio ed esistenza di un modulo elettronico 'valido' all'interno del corpo del messaggio;
- Controllo sintattico dei singoli attributi contenuti negli oggetti che compongono il modulo elettronico;
- Controllo di consistenza dei dati inseriti nel modulo in relazione al modulo stesso e ai dati già registrati nei database WHOIS e HANDLE;
- Controllo di autenticazione del P/M che intende registrare o modificare gli oggetti contenuti nel modulo;

Nelle sezioni seguenti sono analizzate in dettaglio le verifiche previste dal sistema per ogni tipologia di controllo sopra elencata.

4.1.1 Controllo della tipologia del messaggio ricevuto e del suo formato

Il messaggio di posta elettronica, che il sistema riceve in ingresso, deve contenere nel corpo del messaggio un *Modulo Tecnico* o un oggetto *mntner* in formato ASCII.

Il sistema analizza il messaggio e lo formatta al fine di separare l'intestazione dai singoli oggetti contenuti nel modulo elettronico e eliminando tutte le righe che sintatticamente non possono far parte della struttura di un oggetto.

Inoltre riscrive, tutte le righe che descrivono un oggetto nella forma

<attributo>:<spazio><valore>

trasformando, se necessario il nome dell'attributo in lettere minuscole.

In questa fase il sistema controlla che il modulo elettronico:

a) sia contenuto nel corpo del messaggio ed in formato ASCII. Non sono ammessi, per esempio, i moduli elettronici inviati come allegato o in formato HTML;

b) contenga almeno un oggetto *domain*, *mntner* o *person*;

c) non contenga più di un oggetto *domain* o *mntner*;

d) non contenga oggetti *domain* e *mntner* contemporaneamente.

Se questa fase genera almeno un errore, il sistema produce ed invia al P/M il messaggio di posta elettronica contenente l'esito dei controlli effettuati. In caso contrario, il sistema procede con la fase di controllo sintattico dei dati.

4.1.2 Controllo sintattico dei dati

In questa fase viene verificata la correttezza sintattica vera e propria di ogni oggetto contenuto nel modulo elettronico.

Per ogni oggetto viene controllato che:

a) siano stati inseriti tutti gli attributi obbligatori;

b) non siano stati inseriti attributi multipli per gli attributi dell'oggetto che sono univoci;

c) la sintassi del valore di ogni attributo sia corretta.

Il fallimento del controllo sintattico su un attributo impedirà l'eventuale controllo di consistenza sull'attributo stesso, come descritto nella sezione seguente.

4.1.3 Controllo di consistenza dei dati

Il sistema verifica la consistenza dei dati inseriti sia in relazione al contenuto del database WHOIS sia in relazione ai dati contenuti nel modulo stesso. I controlli di consistenza sono eseguiti sia a livello di oggetto che a livello di singolo attributo contenuto in un oggetto; quest'ultimo tipo di controllo è eseguito solo se il valore contenuto nell'attributo da verificare non ha generato un errore sintattico di qualsiasi tipo.

Di seguito sono elencati tutti i controlli di consistenza previsti dal sistema:

a) ogni oggetto *person* e *role*, contenuto nel modulo elettronico, deve essere referenziato come contatto tecnico o amministrativo almeno in un oggetto domain, mntner o role, se presenti all'interno del modulo elettronico;

b) ogni *nic-handle* indicato in ogni oggetto contenuto nel modulo deve essere stato precedentemente assegnato dal Registro e quindi presente nel database HANDLE del Registro stesso;

c) Il *nic-handle* associato ad ogni oggetto *person* o *role* contenuto nel modulo non deve essere già stato assegnato ad un diverso contatto all'interno del database WHOIS, se l'oggetto era già stato registrato precedentemente, o all'interno del database HANDLE;

d) Il valore dell'attributo *pin*, eventualmente presente nel *Modulo Tecnico*, non deve essere associato ad un diverso nome a dominio all'interno del database WHOIS;

e) Il valore nell'attributo *x400-domain,* contenuto in un eventuale oggetto *domain*, deve essere correttamente compilato in relazione al valore inserito nell'attributo *domain*;

f) il contatto referenziato come *postmaster* in un eventuale oggetto *domain* deve contenere almeno un attributo *e-mail* con l'indirizzo di posta elettronica;

g) l'oggetto domain, se presente nel modulo, deve contenere quegli attributi che permettano di identificarlo come dominio interamente delegato o di sola posta elettronica (*nserver*, *zone-c*, oppure *mailgate* e *gate-c*);

h) il contatto referenziato come *admin-c* all'interno di un eventuale oggetto domain deve corrispondere ad un oggetto *person* e non ad un oggetto *role* il quale non potrebbe teoricamente rappresentare il firmatario della LAR.

Nel caso in cui almeno un controllo di consistenza fallisca, il sistema genera ed invia al maintainer il messaggio di posta elettronica contenente l'esito dei controlli effettuati, altrimenti procede nella fase di verifica delle autenticazioni.

4.1.4 Controlli di autenticazione

Il sistema implementa due tipi di controlli di autenticazione:

a. basata sul contenuto del campo *From*: dell'intestazione del messaggio contenente il modulo elettronico. Il valore del campo *From*: deve corrispondere al valore dell'espressione regolare specificata in uno degli attributi *auth* di tipo *MAIL-FROM* dell'oggetto *mntner* associato all'oggetto correntemente esaminato;

b. basata sul valore dell'attributo password. Il valore dell'attributo password inserito in un oggetto, una volta codificato, deve corrispondere al valore specificato in uno degli attributi

auth di tipo *CRYPT-PW* dell'oggetto *mntner* associato all'oggetto corrente. La codifica della password è eseguita utilizzando l'algoritmo di crittografia Data Encryption Standard (DES).

Tutti gli oggetti domain, mntner, role e person sono sottoposti al controllo di autenticazione, se è presente l'attributo mnt-by nell'oggetto stesso. Per gli oggetto domain e mntner tale attributo è obbligatorio.

Il controllo di autenticazione sugli oggetti è eseguito sulla base della password solo se l'attributo password è stato inserito nell'oggetto, in caso contrario l'autenticazione è basata sull'indirizzo di posta elettronica.

Per gli oggetti *person* e *role*, contenenti l'attributo opzionale *mnt-by*, il controllo di autenticazione è effettuato con lo stesso criterio degli oggetti domain. All'interno di un modulo elettronico si possono verificare due differenti scenari:

1. l'oggetto person contiene l'attributo password;
2. l'oggetto person non contiene l'attributo password.

Nel primo caso il criterio di autenticazione si basa sulla password specificata nell'oggetto stesso, nel secondo caso viene invece preso in considerazione il valore dell'attributo password indicato nell'oggetto domain o mntner eventualmente presente nel modulo.

Se l'oggetto analizzato è già stato registrato nel database WHOIS con un attributo mnt-by diverso da quello indicato, il sistema genera un errore di autenticazione ed il P/M che gestisce l'oggetto è avvisato del tentativo di modifica tramite messaggi di posta elettronica che vengono inviati agli indirizzi specificati negli attributi upd-to dell'oggetto mntner indicato nell'attributo mnt-by.

Al termine di questa fase, il sistema genera ed invia al mittente del messaggio contenente il modulo, l'esito del controllo sintattico che, in questo caso, può essere sia positivo che negativo.

Nel caso in cui tutti i controlli sul messaggio siano andati a buon fine ed il modulo elettronico contenga un oggetto *domain*, il sistema invia il modulo elettronico al software che controlla la configurazione dei nameserver o, altrimenti, lo invia ad un operatore. Il modulo elettronico, che dopo i vari controlli viene generato dal sistema, è il modulo elettronico originale inviato dal P/M opportunamente riformattato ed eventualmente completato con gli oggetti person necessari per la registrazione dei dati nel database WHOIS o per i successivi controlli effettuati dagli operatori.

Nelle sezioni seguenti sono descritti i formati dei messaggi di posta sopra indicati.

4.2 Interfaccia verso gli altri sistemi di controllo

Nel caso in cui tutti i controlli eseguiti dal sistema siano andati a buon fine, esso invia il modulo elettronico al programma per il controllo dei nameserver o ad un operatore a seconda della tipologia del modulo analizzato.

Il modulo elettronico contenuto nel messaggio in uscita dal sistema, è lo stesso modulo inviato dal P/M opportunamente formattato ed eventualmente integrato con l'aggiunta di oggetti *person* e *role* come descritto nella *sezione 5.14*.

Il sistema aggiunge, infatti:

1. l'oggetto *person* associato al *nic-handle* inserito come *admin-c* in un oggetto *domain* o *mntner*, se non presente nel modulo;
2. gli oggetti *person* e/o *role* referenziati come contatti in un oggetto *domain*, *mntner* o *role* che non sono presenti nel modulo e non sono registrati nel database WHOIS.

Ogni oggetto del modulo è inoltre formattato secondo i seguenti criteri:

1. gli attributi sono riordinati seguendo un criterio sequenziale predefinito per ogni singola classe;
2. ogni riga è riscritta nel formato

   <etichetta>:<spazi><valore>

   in modo che i valori degli attributi siano tra loro allineati;
3. il valore degli attributi *domain* e *x400-domain* è trasformato in lettere minuscole e per il secondo viene aggiunto il carattere ';' nel caso in cui sia stato omesso.

Inoltre, per ogni oggetto presente nel modulo, il sistema valuta l'ultimo attributo *changed* inserito e, nel caso la data non corrisponda a quella odierna, viene aggiunto un attributo *changed* con valore '*hostmaster@nic.it data_odierna*' in coda agli attributi *changed* già esistenti. Questa operazione non interessa gli oggetti aggiunti automaticamente al modulo dal sistema.

4.3 Interfaccia verso il P/M mittente

Il risultato del controllo sintattico viene inviato dal sistema al P/M mittente del messaggio contenente il modulo elettronico esaminato.

Il messaggio ha un formato standard che comprende un'intestazione fissa e una chiusura che varia a seconda del risultato finale dei controlli.

Per ogni oggetto esaminato dal sistema viene segnalato il risultato globale dei controlli effettuati e inoltre, nel caso si sia verificato almeno un errore o un warning, riporta l'oggetto seguito dai singoli messaggi di segnalazione.

Fanno eccezione i messaggi che segnalano il formato errato dell'intero messaggio o del modulo elettronico per i quali, essendo di carattere globale, non viene riportato nessuno oggetto all'interno della risposta.

I messaggi di errore che si riferiscono ad un attributo specifico sono nella forma:

**ERROR* syntax error in "<nome_attributo>" valore: <MESSAGGIO DI ERRORE>*

I messaggi inerenti il formato del modulo o del messaggio e gli errori di autenticazione o di consistenza, che riguardano un intero oggetto, sono invece espressi nella forma:

**ERROR*: <MESSAGGIO DI ERRORE>*

I messaggi di warning sono, invece, nel seguente formato:

**WARNING* in "<nome_attributo>" valore: <MESSAGGIO DI WARNING>*

Seguono alcuni esempi di risposta generati dal sistema.

4.3.1 Esempio 1 (Segnalazione di esito positivo dei controlli)

Questo esempio mostra il caso in cui il controllo non ha generato errori. Il modulo analizzato sarà inviato alle altre fasi di controllo per la registrazione degli oggetti in esso contenuti.

```
Your e-mail:

>  From:    paperino@nic.it
>  Subject: Modifica dominio nic.it
>  Date:    Mon, 21 Jan 2002 12:51:40 +0100
>  Msg-Id:  <200201221150.MAA99182@master.nic.it>

has been processed by the IT-NIC Syntax Phase database software and
produced the following output:

---------------------------------------------------------------------
Syntax Check Phase OK: [domain] nic.it

domain:         nic.it
x400-domain:    c=it; admd=garr; prmd=nic;
org:            ccTLD ".it" Registry - IIT/CNR
descr:          Italian 2nd-level domain
descr:          Italian Network Information Center
descr:          Via Giuseppe Moruzzi 1
descr:          I-56124 PISA, Italy
admin-c:        FD317-ITNIC
tech-c:         DV73-ITNIC
tech-c:         MM13-ITNIC
postmaster:     DV73-ITNIC
postmaster:     MM13-ITNIC
zone-c:         DV73-ITNIC
zone-c:         MM13-ITNIC
nserver:        193.205.245.5 dns.nic.it
nserver:        193.205.245.8 dns2.nic.it
nserver:        194.119.192.34 nameserver.cnr.it
remarks:        Fully-managed
mnt-by:         RA-MNT
changed:        hostmaster@nic.it 20000317
changed:        hostmaster@nic.it 20010622
source:         IT-NIC
*WARNING* in "domain" value: value lowercased
*WARNING* in "x400-domain" value: value lowercased

Syntax Check Phase OK: [person] Franco Denoth

Syntax Check Phase OK: [person] Daniele Vannozzi

Syntax Check Phase OK: [person] Maurizio Martinelli


Objects that just generated a WARNING have been reset to the correct value
and have gotten over the syntax control phase.
Objects that just generated an *ERROR* have not gotten over the syntax
```

control phase. Please, submit them again.

WARNING!
Forms containing FAILED objects are NOT submitted to the semantic control phase.

If you have any questions about an error or warning message, please contact <hostmaster@nic.it>.

Sincerely Yours,
IT-NIC Database Maintenance Department Robot

4.3.2 Esempio 2 (Segnalazione di errore sugli attributi degli oggetti)

Questo esempio mostra il caso in cui il sistema abbia rilevato degli errori sintatici sui singoli attributi. Il processo di registrazione, in questo caso, si interrompe dopo l'invio di questo messaggio.

```
Your e-mail:

>  From:    paperino@nic.it
>  Subject: Modifica dominio nic.it
>  Date:    Mon, 21 Jan 2002 12:51:40 +0100
>  Msg-Id:  <200201221150.MAA99182@master.nic.it>

has been processed by the IT-NIC Syntax Phase database software and
produced the following output:

------------------------------------------------------------------------
Syntax Check Phase FAILED: [domain] nic?.it

domain:         nic?.it
x400-domain: c=it; admd=garr; prmd=nic;
org:            ccTLD ".it" Registry - IIT/CNR
descr:          Italian 2nd-level domain
descr:          Italian Network Information Center
descr:          Via Giuseppe Moruzzi 1
descr:          I-56124 PISA, Italy
admin-c:        FD317
tech-c:         DV73-ITNIC
tech-c:         MM13-ITNIC
postmaster:     DV73-ITNIC
postmaster:     MM13-ITNIC
zone-c:         DV73-ITNIC
zone-c:         MM13-ITNIC
nserver:        193.205.245.5 dns.nic.it
nserver:        193.205.245.8 dns2.nic.it
nserver:        194.119.192.34 nameserver.cnr.it
remarks:        Fully-managed
mnt-by:         RA-MNT
changed:        hostmaster@nic.it 20000317
changed:        hostmaster@nic.it 20010622
source:         IT-NIC
*WARNING* in "domain" value: value lowercased
*ERROR* syntax error in "domain" value: illegal name
*ERROR* syntax error in "admin-c" value: is NOT a valid nic-handle

Syntax Check Phase OK: [person] Franco Denoth

Syntax Check Phase OK: [person] Daniele Vannozzi

Syntax Check Phase FAILED: [person] Maurizio Martinelli

person:         Maurizio Martinelli
```

address: ccTLD ".it" Registry
address: Istituto di Informatica e Telematica - CNR
address: Via G. Moruzzi 1
address: 56124 Pisa
address: Italy
phone: +39 050 3139811
fax-no: +39 050 542420
e-mail: martinelli@nic.it
nic-hdl: MM13-ITNIC
notify: martinelli@nic.it
changed: martinelli@nic 20000110
changed: martinelli@nic.it 20010227
source: IT-NIC
ERROR syntax error in "changed" value: the e-mail part is not a valid address

Objects that just generated a WARNING have been reset to the correct value
and have gotten over the syntax control phase.
Objects that just generated an *ERROR* have not gotten over the syntax
control phase. Please, submit them again.

WARNING!
Forms containing FAILED objects are NOT submitted to the semantic
control phase.

If you have any questions about an error or warning message, please
contact <hostmaster@nic.it>.

Sincerely Yours,
IT-NIC Database Maintenance Department Robot

4.3.3 Esempio 3 (Segnalazione di errore sul formato del modulo)

Questo esempio mostra il caso in cui il formato del modulo elettronico non è valido. Il sistema segnala l'errore e non mostra gli oggetti componenti il modulo. Anche in questo caso, come per il precedente, il sistema di controllo sintattico si interrompe dopo l'invio di questo messaggio al P/M.

Your e-mail:

> From: paperino@nic.it
> Subject: Modifica dominio nic.it
> Date: Mon, 21 Jan 2002 12:51:40 +0100
> Msg-Id: <200201221150.MAA99182@master.nic.it>

has been processed by the IT-NIC Syntax Phase database software and
produced the following output:

ERROR: ** Domain and maintainer objects are not allowed in the same message **

If you have any questions about an error or warning message, please
contact <hostmaster@nic.it>.

Sincerely Yours,
IT-NIC Database Maintenance Department Robot

4.3.4 Esempio 4 (Segnalazione di errore di autenticazione e di consistenza)

Questo esempio mostra il caso in cui un P/M non autoritativo (PIPPO-MNT) abbia cercato di modificare il dominio 'nic.it' inserendo nel modulo anche un oggetto person non referenziato dall'oggetto domain inviato. Il sistema segnala l'errore di autenticazione e di consistenza al P/M mittente ed invierà un messaggio di segnalazione della tentata modifica anche a tutti gli indirizzi di posta elettronica indicati negli attributi *upd-to* dell'oggetto *mntner* RA-MNT, che costituisce il P/M autorizzato per tale dominio.

Anche questo caso porta all'interruzione di controllo sintattico del nome a dominio.

```
Your e-mail:

>  From:    paperino@nic.it
>  Subject: Modifica dominio nic.it
>  Date:    Mon, 21 Jan 2002 12:51:40 +0100
>  Msg-Id:  <200201221150.MAA99182@master.nic.it>

has been processed by the IT-NIC Syntax Phase database software and
produced the following output:

--------------------------------------------------------------------------
Syntax Check Phase FAILED: [domain] nic.it

domain:         nic.it
x400-domain:    c=it; admd=garr; prmd=nic;
org:            ccTLD ".it" Registry - IIT/CNR
descr:          Italian 2nd-level domain
descr:          Italian Network Information Center
descr:          Via Giuseppe Moruzzi 1
descr:          I-56124 PISA, Italy
admin-c:        FD317-ITNIC
tech-c:         DV73-ITNIC
tech-c:         MM13-ITNIC
postmaster:     MM13-ITNIC
zone-c:         DV73-ITNIC
zone-c:         MM13-ITNIC
nserver:        193.205.245.5 dns.nic.it
nserver:        193.205.245.8 dns2.nic.it
nserver:        194.119.192.34 nameserver.cnr.it
remarks:        Fully-managed
mnt-by:         PIPPO-MNT
changed:        Stefano.Trumpy@iat.cnr.it 19991102
changed:        hostmaster@nic.it 20020220
source:         IT-NIC
*ERROR*:        authorisation failed, request forwarded to maintainer


Syntax Check Phase FAILED: [person] Arianna Del Soldato

person:         Arianna Del Soldato
```

address: ccTLD ".it" Registry
address: Istituto di Informatica e Telematica - CNR
address: Via G. Moruzzi 1
address: 56124 Pisa
address: Italy
phone: +39 050 3152088
e-mail: delsoldato@nic.it
nic-hdl: ADS2-ITNIC
changed: arianna@nic.it 20020222
changed: hostmaster@nic.it 20020222
source: IT-NIC
ERROR: ** Unreferenced "person" in "domain" or role(s) object **

Objects that just generated a WARNING have been reset to the correct value
and have gotten over the syntax control phase.
Objects that just generated an *ERROR* have not gotten over the syntax
control phase. Please, submit them again.

WARNING!
Forms containing FAILED objects are NOT submitted to the semantic
control phase.

If you have any questions about an error or warning message, please
contact <hostmaster@nic.it>.

Sincerely Yours,
IT-NIC Database Maintenance Department Robot

4.4 Interfaccia verso altri P/M

Se, nel corso dei controlli di autenticazione, il sistema analizza un oggetto già registrato con un attributo *mnt-by* diverso da quello inserito, viene generato un errore di autenticazione nel messaggio di risposta al mittente e viene avvisato, della tentata corruzione, il P/M gestore dell'oggetto.

La notifica del tentativo di modifica è effettuata tramite dei messaggi di posta elettronica che sono inviati agli indirizzi specificati negli attributi *upd-to* dell'oggetto *mntner* gestore dell'oggetto.

Di seguito è riportato un esempio di questo tipo di comunicazione.

4.4.1 Esempio

Dear Maintainer,

This is to notify you that some objects in which you are mentioned as
a maintainer were requested to be changed, but *failed* the proper
authorisation for any of the mentioned maintainers.
Please contact the sender of these changes about changes that
need to be made to the following objects.

The mail message causing these failures had the following mail headers:

- From: paperino@nic.it
- Subject: Modifica dominio nic.it
- Date: Mon, 21 Jan 2002 12:51:40 +0100
- Msg-Id: <200201221150.MAA99182@master.nic.it>

IT-NIC Database Maintainer Forwarding Department

UPDATE REQUESTED FOR:

domain: nic.it
x400-domain: c=it; admd=garr; prmd=nic;
org: ccTLD ".it" Registry - IIT/CNR
descr: Italian 2nd-level domain
descr: Italian Network Information Center
descr: Via Giuseppe Moruzzi 1
descr: I-56124 PISA, Italy
admin-c: FD317-ITNIC
tech-c: DV73-ITNIC
tech-c: MM13-ITNIC
postmaster: MM13-ITNIC
zone-c: DV73-ITNIC
zone-c: MM13-ITNIC
nserver: 193.205.245.5 dns.nic.it
nserver: 193.205.245.8 dns2.nic.it
nserver: 194.119.192.34 nameserver.cnr.it
remarks: Fully-managed

mnt-by: PIPPO-MNT
changed: Stefano.Trumpy@iat.cnr.it 19991102
changed: hostmaster@nic.it 20020220
source: IT-NIC

If you have any questions about an error or warning message, please
contact <hostmaster@nic.it>.

Sincerely Yours,
IT-NIC Database Maintenance Department Robot

Il messaggio segnala l'errore e riporta sia l'intestazione del messaggio analizzato dal sistema, sia l'oggetto che ha causato l'errore, così come presente nel modulo elettronico esaminato.

4.5 Segnalazione di errori irreversibili

Oltre ai messaggi contenenti la risposta del controllo sintattico, il sistema invia altre due comunicazioni nel caso in cui l'esecuzione del sistema sia stata interrotta per mancata risposta da uno dei database del Registro o per fallimento nella lettura di un file temporaneo.

Ogni segnalazione è inviata contemporaneamente, ma con contenuti diversi, sia al mittente del modulo elettronico, sia allo staff tecnico del Registro che dovrà provvedere ad analizzare il problema riscontato.

Seguono i due esempi di segnalazione.

4.5.1 Segnalazione di errore inviato al mittente

Il messaggio contiene l'intestazione del messaggio di posta analizzato dal sistema, seguito dalla segnalazione dell'errore stesso e da un invito a ritrasmettere il modulo al sistema di controllo sintattico.

Your e-mail:

> From: paperino@nic.it
> Subject: Modifica dominio nic.it
> Date: Mon, 21 Jan 2002 12:51:40 +0100
> Msg-Id: <200201221150.MAA99182@master.nic.it>

has been processed by the IT-NIC syntax phase database software
and has produced the following output:
WHOIS connection refused: too many connection

Please submit the form again.

If you have any questions about an error or warning message, please
contact <hostmaster@nic.it>.

Sincerely Yours,
IT-NIC Database Maintenance Department Robot

4.5.2 Segnalazione di errore inviato al Registro

Il messaggio contiene un'intestazione ed una chiusura standard ed è suddiviso in due sezioni principali contenenti le informazioni necessarie per identificare la causa dell'errore.

Una prima sezione, chiamata 'ERROR', indica il database nel quale si è verificato l'errore (WHOIS o HANDLE), il nome della procedura o del programma che ha rilevato l'errore e l'argomento della query effettuata al database.

Una seconda sezione, chiamata 'ON MODULE' riporta per intero il messaggio di posta elettronica che ha causato l'errore.

There has been an execution error on the SYNTAX CHECK Program!

ERROR:
Connection Error to the WHOIS database.
Main program con nic-handle: VF249-ITNIC

ON MODULE:

>
…
>
…
>

Messaggio analizzato compresa l'intestazione

Automatic Mail

4.6 Log di sistema

Il sistema registra tutti i messaggi di posta elettronica inviati ai P/M in appositi file di log che sono creati giornalmente e denominati secondo la sintassi *aaaammgg* che identifica l'anno, il mese e il giorno in cui il messaggio è stato inviato.

In particolare:

- i messaggi contenenti la risposta del controllo sintattico (*sezione 4.3*) e la segnalazione di tentata corruzione (*sezione 4.4*) sono memorizzati in una stessa directory chiamata *acklog;*
- i messaggi di segnalazione di errore fatale sono, invece, memorizzati in directory chiamata *errlog*.

Le directory *acklog* e *errlog* appartengono alla stessa directory generale di log la cui posizione nel file system è definita all'interno del sistema stesso.

Alla stessa directory generale di log appartiene, inoltre, la directory *tmp* in cui il sistema crea i file temporanei utilizzati per la creazione dei messaggi inviati.

5 Struttura del sistema

Il sistema di controllo sintattico è un insieme di programmi e di moduli scritti in linguaggio Perl. I file contenenti i programmi ed i moduli sono stati suddivisi in due directory: una principale che contiene tutti i programmi e le procedure che eseguono le varie tipologie di controllo ed una sotto-directory CONF contenente i file di definizione e configurazione. Segue l'elenco dei file che compongono il sistema:

Programmi:

main.pl: contiene il programma principale che riceve in ingresso il messaggio contenente il modulo elettronico da controllare, esegue i controlli e spedisce i messaggi di risposta ai P/M e al Registro.

GetMailHeader.pl: contiene la funzione che esegue il parsing dell'intestazione del messaggio di posta elettronica contenente il modulo e ne memorizza i campi principali.

Moduli:

Format.pm: contiene le procedure che eseguono la formattazione di oggetti come l'intero messaggio analizzato e il attributo *x400-domain;*

CheckObjects.pm: contiene la funzione che controlla la sintassi di ogni singolo oggetto (*domain*, *mntner*, *role*, *person*) contenuto nel modulo;

CheckFields.pm: contiene le funzioni che eseguono il controllo sintattico di ogni singolo attributo di un oggetto;

CheckAuth.pm: contiene le funzioni che eseguono il controllo di autenticazione sui singoli oggetti (*domain*, *mntner*, *role*, *person*);

CheckCons.pm: contiene le funzioni che eseguono i controlli di consistenza;

WHOISDBinterface.pm: contiene le funzioni di interfaccia con i database WHOIS e HANDLE;

WriteObject.pm: contiene la funzione che formatta e stampa un oggetto, all'interno dei messaggi di posta elettronica restituiti in uscita;

GlobalCheck.pm: contiene le definizioni delle variabili globali contenenti i parametri di configurazione del sistema;

utility.pm: contiene le funzioni di generico utilizzo come per il calcolo della data odierna il il lock dei file ecc.

Directory CONF

CONF/Messages: contiene le sezioni di testo standard che compongono i messaggi di risposta;

CONF/SetMsgs.pm: contiene la funzione che inizializza una variabile globale per ogni sezione dichiarata nel file *Messages*;

CONF/SyntaxError.pm: contiene la definizione di ogni singolo messaggio di errore o warning che il sistema può segnalare all'interno di un messaggio di risposta;

CONF/SyntaxConf.pm: contiene le strutture dati che definiscono le caratteristiche strutturali e sintattiche di ogni classe di oggetto prevista da un modulo elettronico.

Le sezioni seguenti di questo documento descrivono in dettaglio le funzionalità ed il contenuto di ogni file sopra elencato partendo dalla descrizione dei file di configurazione, per poi descrivere il programma principale e tutti i moduli da esso utilizzati.

5.1 CONF/Messages

Questo file contiene la definizione delle sezioni di testo standard che compongono i vari messaggi di posta elettronica generati e spediti dal sistema.

Il file ha una struttura propria che deve essere rispettata per consentire il corretto funzionamento del sistema. In particolare:

- una sezione di testo è un'insieme di righe tra loro contigue ed è separata da un'altra sezione da almeno una riga vuota;
- ogni sezione è identificata da un'etichetta alla quale deve corrispondere la dichiarazione di un'omonima variabile nel modulo *GlobalCheck.pm*;
- ogni riga di testo appartenente ad una sezione è delimitata a sinistra dall'etichetta che identifica la sezione stessa;
- il file termina con un'etichetta ENDCONF.

Questo file viene analizzato dalla procedura *&SetMsgs::GetMessages()* che provvede ad assegnare le varie sezioni alle variabili globali omonime dichiarate nel file *GlobalCheck.pm* in modo che possano essere utilizzate dai vari programmi del sistema.

Di seguito sono riportate le sezioni di testo definite nel file precedute da una breve descrizione.

- **MHEADER**

  È l'intestazione del messaggio di posta elettronica contenente la risposta del controllo sintattico. L'intestazione non contiene il campo To: che viene inserito dinamicamente al momento della creazione del messaggio di posta prendendolo dal campo 'From:' o 'Reply-To:' del messaggio analizzato dal sistema. Inoltre, il testo è parametrico rispetto alle variabili $SUBJECT e $CHECKRESULT che dovranno contenere rispettivamente il valore del campo 'Subject:' del messaggio analizzato e un'etichetta SUCCESS o FAILED che segnala il successo o il fallimento dei controlli effettuati:

  MHEADER From: IT-NIC Database Management <hostmaster\@nic.it>
  MHEADER Subject: Re: $SUBJECT - $CHECKRESULT
  MHEADER Reply-To: hostmaster\@nic.it
  MHEADER Precedence: bulk

- **MAILTXT**

  È il testo che viene inserito subito dopo l'intestazione del messaggio contenente la risposta del controllo sintattico. Il testo contiene quattro variabili che dovranno essere espanse con il valore degli attributi contenuti nell'intestazione del messaggio di posta elettronica esaminato:

  MAILTXT
  MAILTXT Your e-mail:
  MAILTXT
  MAILTXT > From: $FROM
  MAILTXT > Subject: $SUBJECT
  MAILTXT > Date: $MDATE
  MAILTXT > Msg-Id: $MSGID
  MAILTXT
  MAILTXT has been processed by the IT-NIC Syntax Phase database software and
  MAILTXT produced the following output:
  MAILTXT
  MAILTXT ----------

- **USRHEADER**

  Contiene l'intestazione del messaggio di posta elettronica contenente il modulo elettronico che viene inviato dal sistema verso le fasi successive del processo di registrazione. L'intestazione coincide con quella del messaggio di posta esaminato ad eccezione del campo 'To:' che viene impostato da programma in base alla tipologia dei controlli che dovranno essere successivamente effettuati sul modulo:

  USRHEADER From: $FROM
  USRHEADER Subject: $SUBJECT
  USRHEADER Date: $MDATE
  USRHEADER Msg-Id: $MSGID
  USRHEADER

- **USRERRORHEADER**

  Contiene l'intestazione del messaggio di posta elettronica che viene inviato all'utente nel caso in cui siano presenti troppe connessioni con uno dei database del Registro. L'intestazione è parametrica rispetto all'attributo 'Subject' del messaggio analizzato.

  USRERRORHEADER From: IT-NIC Database Management <hostmaster\@nic.it>
  USRERRORHEADER Subject: Re: $SUBJECT - TOO MANY CONNECTION
  USRERRORHEADER Reply-To: hostmaster\@nic.it
  USRERRORHEADER Precedence: bulk

- **USRERRORTEXT**

  È il testo contenuto nel corpo del messaggio di posta elettronica che viene inviato all'utente nel caso in cui siano presenti troppe uno dei database del Registro. La sezione è parametrica rispetto ai campi contenuti nell'intestazione del messaggio analizzato.

  USRERRORTEXT
  USRERRORTEXT Your e-mail:
  USRERRORTEXT
  USRERRORTEXT > From: $FROM
  USRERRORTEXT > Subject: $SUBJECT
  USRERRORTEXT > Date: $MDATE
  USRERRORTEXT > Msg-Id: $MSGID
  USRERRORTEXT
  USRERRORTEXT has been processed by the IT-NIC syntax phase database software
  USRERRORTEXT and has produced the following output:
  USRERRORTEXT WHOIS connection refused: too many connection
  USRERRORTEXT
  USRERRORTEXT Please submit the form again.
  USRERRORTEXT

- **FWHEADER**

  È l'intestazione del messaggio di posta elettronica che viene inviato ad un P/M nel caso in cui si sia verificato un tentativo di modifica di un oggetto da lui gestito da parte di un altro P/M non autorizzato:

FWHEADER From: IT-NIC Database Management <hostmaster\@nic.it>

FWHEADER Subject: Requested IT-NIC database object changes

FWHEADER Reply-To: hostmaster\@nic.it

- **FWTEXT**

  È la prima parte del corpo del messaggio di posta elettronica che viene inviato ad un P/M nel caso in cui si sia verificato un tentativo di modifica di un oggetto da lui gestito da parte di un altro P/M non autorizzato. La sezione è parametrica rispetto ai campi contenuti nell'intestazione del messaggio analizzato.

  FWTEXT

  FWTEXT Dear Maintainer,

  FWTEXT

  FWTEXT This is to notify you that some objects in which you are mentioned as

  FWTEXT a maintainer were requested to be changed, but *failed* the proper

  FWTEXT authorisation for any of the mentioned maintainers.

  FWTEXT Please contact the sender of these changes about changes that

  FWTEXT need to be made to the following objects.

  FWTEXT

  FWTEXT The mail message causing these failures had the following mail headers:

  FWTEXT

  FWTEXT - From: $FROM

  FWTEXT - Subject: $SUBJECT

  FWTEXT - Date: $MDATE

  FWTEXT - Msg-Id: $MSGID

  FWTEXT

  FWTEXT IT-NIC Database Maintainer Forwarding Department

  FWTEXT

  FWTEXT ---

  FWTEXT UPDATE REQUESTED FOR:

  FWTEXT

- **NICTEXT**

  Testo standard che viene stampato in testa al corpo del messaggio di posta elettronica inviato allo staff tecnico del Registro per una segnalazione di errore verificatosi durante l'esecuzione del programma. La sezione è parametrica rispetto alla variabile $TYPE che dovrà contenere le informazioni necessarie ad identificare il problema.

  NICTEXT There has been an execution error in the SYNTAX CHECK Program!
  NICTEXT
  NICTEXT ERROR:
  NICTEXT $TYPE
  NICTEXT
  NICTEXT ON MODULE:
  NICTEXT

- **CHECKOK**

  È la stringa che viene stampata nel messaggio di risposta del sistema di controllo sintattico in testa ad ogni oggetto i cui controlli siano andati a buon fine. La stringa è parametrica rispetto alle variabili $OBJECT e $NAME che dovranno contenere la classe (*domain*, *mntner*, *person, role*) e il nome dell'oggetto rispettivamente:

  CHECKOK Syntax Check Phase OK: [$OBJECT] $NAME
  CHECKOK

- **CHECKFAILED**

  È la stringa che viene stampata nel messaggio di risposta del sistema di controllo sintattico per ogni oggetto il cui controllo sintattico non sia andato a buon fine. La stringa è parametrica rispetto alle variabili $OBJECT e $NAME che dovranno contenere la classe dell'oggetto (*domain*, *mntner*, *person, role*) e il nome dell'oggetto rispettivamente:

  CHECKFAILED Syntax Check Phase FAILED: [$OBJECT] $NAME
  CHECKFAILED

- **ACKERR**

  È il messaggio che viene stampato in coda al messaggio di risposta del sistema di controllo sintattico nel caso in cui si sia verificato almeno un errore o un warning negli oggetti esaminati:

  ACKERR
  ACKERR Objects that just generated a WARNING have been reset to the correct value
  ACKERR and have got over the syntax control phase.
  ACKERR Objects that just generated an *ERROR* have not gotten over the syntax
  ACKERR control phase. Please, submit them again.
  ACKERR
  ACKERR
  ACKERR WARNING!
  ACKERR Forms containing FAILED objects are NOT submitted to the semantic
  ACKERR control phase.
  ACKERR

- **ACKOK**

  È il testo che viene stampato in coda al messaggio di risposta del sistema di controllo sintattico nel caso in cui non si sia verificato nè un errore nè un warning in tutti gli oggetti esaminati:

  ACKOK
  ACKOK No syntax error/warnings were found in your request.
  ACKOK Your request has gotten over the syntax control phase. Congratulations.
  ACKOK
  ACKOK Now your request is submitted to the DNS verification tool, and in case
  ACKOK of success, it is sent to the Registration Authority staff for the
  ACKOK semantic control phase (*domain* name verification, cross-checks with
  ACKOK the documents of the requesting organization, etc.). This phase may take
  ACKOK up to 10 working days as clearly stated in the Italian Registration rules
  ACKOK (http://www.nic.it/RA/domini/regole/regolamento.pdf).
  ACKOK

- **ACKSIG**

  È la firma apposta in calce ai messaggi di posta elettronica contenenti il risultato dei controlli:

  ACKSIG ----------

  ACKSIG

  ACKSIG If you have any questions about an error or warning message, please

  ACKSIG contact <hostmaster\@nic.it>.

  ACKSIG

  ACKSIG Sincerely Yours,

  ACKSIG IT-NIC Database Maintenance Department Robot

5.2 CONF/SetMsgs.pm

Questo modulo contiene la funzione *&GetMessages()* utilizzata dal programma principale per l'inizializzazione delle variabili globali contenenti le sezioni di testo che verranno utilizzate per comporre i messaggi di posta elettronica generati dal sistema.

5.2.1 GetMessages()

Questa funzione assegna le sezioni di testo elencate nel file passato come parametro, a specifiche variabili globali dichiarate nel modulo *GlobalConf.pm*.

IN

- **$confname:** nome del file contenente le sezioni di testo da assegnare alle variabili globali; le sezioni di testo sono elencate nel file mettendo in testa ad ogni riga un'etichetta che identifica la sezione seguita da uno spazio bianco; inoltre la fine del file è segnata dall'etichetta identificatrice ENDCONF.

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalConf::etichetta:** una per ogni etichetta letta, sono variabili inizializzate dinamicamente dalla funzione e denominate in base al nome delle etichette delle sezioni.

DESCRIZIONE

La funzione analizza il file scartando le righe bianche e quelle di commento e termina non appena trova un'etichetta denominata ENDCONF. Per tutte le altre righe, che presuppone siano sezioni di testo da memorizzare:

- suddivide l'etichetta dalla riga di testo;
- riferisce dinamicamente una variabile dichiarata nel modulo *GlobalConf.pm* e omonima all'etichetta letta e vi appende la riga di testo.

La dichiarazione delle variabili all'interno di un file, in questo caso il modulo Perl *GlobalCheck.pm* (Vedi *sezione 5.5*), si è resa necessaria per mantenere visibile il valore di quest'ultima durante tutta l'esecuzione del sistema.

5.3 CONF/SyntaxError.pm

Questo modulo contiene tutti i messaggi previsti dal sistema per la segnalazione degli errori nel messaggio di risposta del sistema di controllo sintattico.

Ogni messaggio ha un suo codice identificativo; i messaggi sono memorizzati in un vettore hash denominato *%ERROR_MESSAGES* il quale contiene tutte le associazioni *(codice_errore, messaggio)*.

Come mostrato di seguito, alcuni dei messaggi previsti sono parametrici rispetto alle variabili; questi dovranno essere valutati prima di essere stampati.

Le sezioni seguenti descrivono tutte le associazioni contenute nella struttura *%ERROR_MESSAGES* suddivise a seconda della tipologia di errore:

- errori sintattici sul valore dei singoli attributi di un oggetto;
- errori di consistenza;
- errori di autenticazione.

5.3.1 Errori sintattici sul valore dei singoli attributi di un oggetto

CODICE DI ERRORE	**MESSAGGIO**
ERRORI DI SEGNALAZIONE SUGLI ATTRIBUTI MULTIPLI:	
ERR_UNKNOWN_ATTRIBUTE	'unknown attribute in $OBJECT object'
ERR_MULTIPLE_FIELD	‘multiple lines are not allowed’
ERR_MISSING_FIELD	‘mandatory field missing’
Attributo person:	
ERR_SYNTAX_NAME	"must contain at least two components"
ERR_ILLEGAL_NAME	"personal title not allowed"
ERR_SYNTAX_PERS	"can contain only the ‘ and alphanumeric characters"
Attributo Domain:	

ERR_SYNTAX_DOMNAME	"illegal name"
WARN_SYNTAX_HYPHEN_DOMNAME	"The use of two consecutive hyphens is not recommended"
WARN_SYNTAX_LOWERCASE_DOMNAME	"value lowercased"
Attributi contenenti il nome di un maintainer:	
ERR_ILLEGAL_MNT	"'--' not allowed"
ERR_SYNTAX_MNT	"is not a valid maintainer name"
Attributi contenenti il codice di un *nic-handle* :	
ERR_SYNTAX_NICHDL	"is NOT a valid nic-handle"
Attributo Changed:	
ERR_SYNTAX_ IN_CHANGED	"illegal value"
ERR_SYNTAX_EMAIL_IN_CHANGED	"the e-mail part is not a valid address"
ERR_SYNTAX_DATE_IN_CHANGED	"the date part is not a valid YYYYMMDD value"
ERR_SYNTAX_NOEMAIL_IN_CHANGED	"'e-mail address' is missing"
ERR_SYNTAX_NODATE_IN_CHANGED	"'date' is missing"
ERR_SYNTAX_SWAP_EMAIL_DATE_IN_CHANGED	"the e-mail address and the date are in reverse order"
ERR_SYNTAX_OUT_OF_DATE_IN_CHANGED	"'date' is in the future"
Attributo source:	
ERR_SYNTAX_SOURCE	"must contain IT-NIC"
Attributo x400-domain:	
ERR_SYNTAX_X4001	"'=' is missing"
ERR_SYNTAX_C	"must contain 'c=it' "
ERR_SYNTAX_ADMD	"'admd' tag is not valid "
ERR_SYNTAX_PRMD	"'prmd' tag is not valid"

ERR_SYNTAX_PRMD_LONG	"prmd tag too long, max length is 16 characters"
ERR_SYNTAX_O	"'o' tag is not valid"
ERR_SYNTAX_Ou	"'ou' tag is not valid"
ERR_SYNTAX_CHECK_X400	"the number of the domain name components don't match with the number of the tags"
ERR_INCONSISTENT_X400DOMAIN	'subfield "$field" does not match with the 'domain' name'
WARN_LOOK_X400DOMAIN	"ATTENZIONEEEEE CONTROLLARE CAMPO X400"
WARN_LOWERCASE_X400	"value lowercased "
WARN_SEMICOLON_X400	"final ';' missing"
Attributo auth:	
ERR_SYNTAX_MAIL	"is not a regular expression"
ERR_SYNTAX_CRYPT	"the length of the password is incorrect"
ERR_SYNTAX_NOEMPTY	"MAIL-FROM or CRYPT-PW value missing"
ERR_SYNTAX_AUTH	"is incorrect"
ERR_SYNTAX_EMAIL	"is not a valid RFC822 address"
Attributo nserver:	
ERR_SYNTAX_IP_IN_NSERVER	"the first component is not an IP address"
ERR_SYNTAX_IN_NSERVER	"illegal value"
ERR_SYNTAX_FQDN_IN_NSERVER	"the last component is not a domain name"

ERR_SYNTAX_NOIP_IN_NSERVER	"the first component ‘IP address’ is missing"
ERR_SYNTAX_NOFQDN_IN_NSERVER	"the last component ‘Domain Name’ is missing"
ERR_SYNTAX_SWAP_IP_FQDN_IN_NSERVER	"the IP address and Domain name are in reverse order"
WARN_SYNTAX_FQDN_IN_NSERVER	"the last component contains a not recommended character"
Attributo DomNet:	
ERR_SYNTAX_DOMNET	"is not a network number"
Attributo Phone and Fax:	
ERR_SYNTAX_PHONE	"illegal value"
Attributo e-mail	
ERR_SYNTAX_MAIL	"illegal value"
Attributo pin:	
ERR_SYNTAX_PIN	'illegal characters'
Attributi a testo libero:	
ERR_STRING_TOOLONG	"length must be less than 255 characters"

5.3.2 Errori di consistenza

CODICE DI ERRORE	**MESSAGGIO**
Attributi contenenti un *nic-handle* :	
ERR_NICHDL_NOT_FOUND	'"$nichdl" DOES NOT EXIST'
Attributo pin:	
ERR_REGISTERED_PIN	“** Individuals can register only one domain name **"
ERR_INVALID_PIN	“invalid pin”

Attributo postmaster:	
ERR_NO_VALID_POSTMASTER	"object associated to ‘postmaster’ must contain the ‘e-mail’ field"
Attributo admin-c:	
ERR_NO_VALID_ADMINC	"'admin-c' field must be associated to a 'person' object"
Attributi contenenti nic-handle:	
ERR_NO_VALID_NICHDLE	"must be associated to a 'person' object"
Oggetto person:	
ERR_UNREFERENCED_OBJECT	‘** Unreferenced “$contact” in ”$object” or role(s) object **’
ERR_ASSIGNED_PERSON	“nic-handle already assigned to another person”
Formato del messaggio e del modulo analizzato:	
ERR_CONTENT_TYPE	"** Data should be included in the body message only in plain text **"
ERR_SUBJECT	"** The keyword 'new' is not allowed in the 'Subject' of the message **"
ERR_NO_OBJECT_FOUND	"** No objects were found in your message **"
ERR_TOO_MANY_DOMAIN	"** Multiple domain objects are not allowed **"
ERR_TOO_MANY_MNTNER	"** Multiple maintainer objects are not allowed **"
ERR_INCONSISTENT_MESSAGE	"** Domain and maintainer objects are not allowed in the same message **"
Configurazione della tipologia del dominio:	
ERR_BAD_CONF1	“nserver and mailgate, or related fields inserted: incompatible fields”
ERR_BAD_CONF2	“zone-c and mailgate fields inserted: incompatible fields”
ERR_NO_CONF	“domains need nserver or mailgate fields”
ERR_MISS_ZONEC	“nserver field needs zone-c field”

ERR_MISS_GATEC	“mailgate field needs gate-c field”
ERR_MISS_NSERVER	“there must be two authoritative nservers, at least”

5.3.3 Errori di autenticazione

CODICE DI ERRORE	**MESSAGGIO**
ERR_MNTNOTAUTH	"authorisation failed, request forwarded to maintainer"
ERR_AUTH	"authorisation failed"
ERR_MNTNOTFOUND	'unknown maintainer(s) “$mntner” referenced'

5.4 CONF/SyntaxConf.pm

Questo file contiene la definizione degli oggetti che compongono un modulo elettronico secondo le specifiche definite in [MM00], [MM001] e [ADS02].

Il sistema prevede quattro classi di oggetti: *domain*, *mntner*, *person* e *role* e, per ogni tipologia, questo file definisce:

- gli attributi della classe stessa con riferimento alla loro molteplicità e obbligatorietà;
- l'ordine con il quale gli attributi devono comparire all'interno di un oggetto appartenente a quella classe;
- le proprietà sintattiche da controllare per ogni attributo della classe.

Queste caratteristiche sono descritte attraverso la definizione e l'inizializzazione di tre strutture dati per ogni classe:

- **%DOMAIN_FIELD**, **%MNTNER_FIELD**, **%PERSON_FIELD** e **%ROLE_FIELD**, definiscono gli attributi delle classi *domain*, *mntner, person* e *role* rispettivamente, tenendo conto della loro obbligatorietà e molteplicità;
- **@DOMAIN_FIELD**, **@MNTNER_FIELD**, **@PERSON_FIELD** e **@ROLE_FIELD**, nei quali è definito l'ordine in cui appaiono gli attributi all'interno degli oggetti appartenenti ad ogni classe;
- **%DOMAIN_FUNCTION**, **%MNTNER_ FUNCTION**, **%PERSON_ FUNCTION** e **%ROLE_FUNCTION**, che definiscono le caratteristiche sintattiche degli attributi contenuti in ogni oggetto.

In questo file sono, inoltre, definite tre funzioni che consentono al sistema l'accesso alle strutture dati sopra elencate.

Al fine di puntualizzare i requisiti sintattici e semantici ritenuti validi sulle varie classi di oggetti, di seguito viene riportata la definizione delle strutture dati sopra citate nonché la descrizione delle funzioni.

5.4.1 Definizione degli attributi di ogni classe

Le strutture dati **%DOMAIN_FIELD**, **%MNTNER_FIELD, %PERSON_FIELD** e **%ROLE_FIELD** definite in questo modulo contengono associazioni del tipo

(*attributo, [obbligatorio, multivalore]*)

per ogni attributo previsto nella classe che rappresentano. Dove: '*attributo'* è il nome dell'attributo e (*obbligatorio, multivalore)* sono due *flag* che indicano se il valore è obbligatorio e/o multivalore.

Di seguito sono riportate tre tabelle che descrivono le associazioni per ogni attributo appartenente ad ogni classe.

Classe *domain*		**Classe *mntner***		**Classe *person***		**Classe *role***	
'password'	[0, 0]	'password'	[0, 0]	'password'	[0, 0]	'password'	[0, 0]
'domain'	[1, 0]	'mntner'	[1, 0]	'person'	[1, 0]	'role'	[1, 0]
'x400-domain'	[1, 0]	'descr'	[1, 1]	'address'	[1, 1]	'address'	[1, 1]
'org'	[1, 0]	'admin-c'	[1, 1]	'phone'	[1, 1]	'phone'	[1, 1]
'org-unit'	[0, 1]	'tech-c'	[0, 1]	'fax-no'	[0, 1]	'fax-no'	[0, 1]
'pin'	[0, 0]	'upd-to'	[1, 1]	'e-mail'	[0, 1]	'e-mail'	[1, 1]
'descr'	[0, 1]	'mnt-nfy'	[0, 1]	'nic-hdl'	[1, 0]	'trouble'	[0, 1]
'admin-c'	[1, 1]	'auth'	[1, 1]	'remarks'	[0, 1]	'admin-c'	[1, 1]
'tech-c'	[1, 1]	'remarks'	[0, 1]	'notify'	[0, 1]	'tech-c'	[1, 1]
'postmaster'	[1, 1]	'notify'	[0, 1]	'mnt-by'	[0, 0]	'nic-hdl'	[1, 0]
'zone-c'	[0, 1]	'mnt-by'	[1, 0]	'changed'	[0, 1]	'remarks'	[0, 1]
'nserver'	[0, 1]	'created'	[0, 0]	'source'	[1, 0]	'notify'	[0, 1]
'dom-net'	[0, 1]	'changed'	[0, 1]	);		'mnt-by'	[0, 0]
'gate-c'	[0, 1]	'source'	[1, 0]			'changed'	[0, 1]
'mailgate'	[0, 1]					'source'	[1, 0]
'remarks'	[0, 1]						
'notify'	[0, 1]						
'x400-mta'	[0, 0]						
'x400-routing'	[0, 0]						
'mnt-by'	[1, 0]						
'created'	[0, 0]						
'changed'	[0, 1]						
'source'	[1, 0]						

5.4.2 Ordinamento degli attributi all'interno degli oggetti

I vettori **@DOMAIN_FIELD**, **@MNTNER_FIELD**, **@PERSON_FIELD** e **@PERSON_FIELD** definiscono l'ordine in cui appaiono gli attributi all'interno di un oggetto appartenente ad una determinata classe. Ogni struttura dati contiene la lista ordinata degli attributi contenuti nella classe che rappresenta.

Segue la definizione dei vettori.

Classe *domain*

@DOMAIN_FIELD = ('password', 'domain', 'x400-domain', 'org', 'org-unit', 'pin', 'descr', 'admin-c', 'tech-c', 'postmaster', 'zone-c', 'nserver', 'dom-net', 'gate-c', 'mailgate', 'remarks', 'notify', 'x400-mta', 'x400-routing', 'mnt-by', 'changed', 'source')

Classe *mntner*

@MNTNER_FIELD = ('password', 'mntner', 'descr', 'admin-c', 'tech-c', 'upd-to', 'mnt-nfy', 'auth', 'remarks', 'notify', 'mnt-by', 'changed', 'source')

Classe *role*

@ROLE_FIELD = ('password', 'role', 'address', 'phone', 'fax-no', 'e-mail', 'trouble', 'admin-c', 'tech-c', 'nic-hdl', 'remarks', 'notify', 'mnt-by', 'changed', 'source')

Classe *person*

@PERSON_FIELD = ('password', 'person', 'address', 'phone', 'fax-no', 'e-mail', 'nic-hdl', 'remarks', 'notify', 'mnt-by', 'changed', 'source')

5.4.3 Caratteristiche sintattiche del valore degli attributi contenuti in ogni classe

Le strutture dati **%DOMAIN_FUNCTION**, **%MNTNER_FUNCTION**, **%PERSON_FUNCTION** e **%ROLE_FUNCTION** definiscono le caratteristiche sintattiche degli attributi contenuti in ogni oggetto. Ogni struttura dati contiene, per ogni attributo previsto nella classe,un'associazione del tipo *(attributo, nome_funzione)* dove:

- *attributo* è il nome dell'attributo e
- *nome_funzione* è il nome di una funzione (*&nome_funzione()*) nella quale viene definita la sintassi del valore dell'attributo attraverso dei controlli sintattici che verranno effettuati sul valore dell'attributo stesso.

Ad ogni *nome_funzione* corrisponde la definizione di un'omonima funzione nel modulo *CheckFields.pm,* la quale viene eseguita dal sistema - in particolare dalla funzione *&CheckObjects::CheckObject* - per eseguire i controlli sintattici sul valore nell'attributo associato.

Di seguito sono riportate tre tabelle che descrivono le associazioni per ogni attributo appartenente ad ogni classe.

Classe *person*		**Classe *role***	
'password'	'TextFreeOK'	'password'	'TextFreeOK'
'person'	'PersonOK'	'role'	'RoleOK'
'address'	'TextFreeOK'	'address'	'TextFreeOK'
'phone'	'TelNumberOK'	'phone'	'TelNumberOK'
'fax-no'	'TelNumberOK'	'fax-no'	'TelNumberOK'
'e-mail'	'EmailOK'	'e-mail'	'EmailOK'
'nic-hdl'	'NicHandleOK'	'trouble'	'TextFreeOK'
'remarks'	'TextFreeOK'	'admin-c'	'NicHandleOK'
'notify'	'EmailOK'	'tech-c'	'NicHandleOK'
'mnt-by'	'MntnerOK'	'nic-hdl'	'NicHandleOK'
'changed'	'ChangedOK'	'remarks'	'TextFreeOK'
'source'	'SourceOK'	'notify'	'EmailOK'
		'mnt-by'	'MntnerOK'
		'changed'	'ChangedOK'
		'source'	'SourceOK'

Classe *domain*		Classe *mntner*	
'password'	'TextFreeOK'	'password'	'TextFreeOK'
'domain'	'DomainOK'	'mntner'	'MntnerOK'
'x400-domain'	'X400DomainOK'	'descr'	'TextFreeOK'
'org'	'TextFreeOK'	'admin-c'	'NicHandleOK'
'org-unit'	'TextFreeOK'	'tech-c'	'NicHandleOK'
'pin'	'PinOK'	'upd-to'	'EmailOK'
'descr'	'TextFreeOK'	'mnt-nfy'	'EmailOK'
'admin-c'	'NicHandleOK'	'auth'	‘AuthOK’
'tech-c'	'NicHandleOK'	'remarks'	'TextFreeOK'
'postmaster'	'NicHandleOK'	'notify'	'EmailOK'
'zone-c'	'NicHandleOK'	'mnt-by'	'MntnerOK'
'nserver'	'NserverOK'	'created'	'TextFreeOK'
'dom-net'	'DomNetOK'	'changed'	'MntnerOK'
'gate-c'	'NicHandleOK'	'source'	'SourceOK'
'mailgate'	'NserverOK'		
'remarks'	'TextFreeOK'		
'notify'	'EmailOK'		
'x400-mta'	‘’		
'x400-routing'	‘’		
'mnt-by'	'MntnerOK'		
'created'	‘’		
'changed'	'ChangedOK'		
'source'	'SourceOK'		

5.4.4 GetOBJECT_FIELD()

Questa funzione restituisce un puntatore alla struttura dati contenente la descrizione degli attributi di una determinata classe.

IN

- **$type:** indica la classe della quale si vuole ottenere la descrizione degli attributi (*domain*, *mntner*, *person, role*);

OUT

- **$OBJECT:** puntatore al vettore hash contenente la descrizione degli attributi.

DESCRIZIONE

La funzione restituisce il puntatore alla struttura *%DOMAIN_FIELD*, *%MNTNER_FIELD*, *%PERSON_FIELD* o *%ROLE_FIELD* a seconda che il parametro in ingresso sia '*domain*', '*mntner*', '*person*' o '*role'* rispettivamente.

5.4.5 GetEmptyOBJECT_FIELD()

Questa funzione restituisce il puntatore ad una struttura dati analoga a *%DOMAIN_FIELD*, *%MNTNER_FIELD*, *%PERSON_FIELD* o *%ROLE_FIELD* definita in questo modulo, ma con i flag di esistenza e molteplicità inizializzati a zero.

Queste strutture dati consentono al sistema di tener traccia dell'esistenza e della molteplicità degli attributi contenuti in un oggetto esaminato.

IN

- **$type:** indica la classe della quale si vuole ottenere la descrizione degli attributi (*domain*, *mntner*, *person, role*);

OUT

- **$OBJECT:** puntatore al vettore hash contenente la descrizione degli attributi con i flag di molteplicità inizializzati a zero.

DESCRIZIONE

La funzione seleziona una delle strutture *%DOMAIN_FIELD*, *%MNTNER_FIELD*, *%PERSON_FIELD* o *%ROLE_FIELD* definite in questo modulo, a seconda che il parametro in ingresso sia '*domain*', '*mntner*' '*person*' o '*role'* rispettivamente, azzera i flag e restituisce il puntatore in uscita.

5.4.6 GetOBJECT_FUNCTION()

Questa funzione restituisce il puntatore alla struttura dati contenente l'associazione dei controlli sintattici con ogni attributo contenuto in una determinata classe.

IN

- **$type:** indica la classe della quale si vuole la descrizione delle caratteristiche sintattiche degli attributi (*domain*, *mntner*, *person, role*);

OUT

- **\%OBJECT:** puntatore al vettore hash contenente l'associazione tra gli attributi e le funzioni che ne determinano le caratteristiche sintattiche.

DESCRIZIONE

La funzione restituisce il puntatore ad una delle strutture *%DOMAIN_FUNCTION*, *%MNTNER_ FUNCTION*, *%PERSON_ FUNCTION* o *%ROLE_FUNCTION* definite in questo modulo, a seconda che il parametro in ingresso sia '*domain*', '*mntner*', '*person*' o '*role'* rispettivamente.

5.5 GlobalCheck.pm

Questo file contiene la definizione delle variabili globali contenenti i parametri di configurazione del sistema, in particolare contiene:

- un parametro che determina l'esecuzione del sistema in modalità TEST;
- i comandi Unix eseguiti;
- la definizione, in relazione alla modalità di esecuzione, delle directory utilizzate per la creazione dei file generati dal sistema;
- i nomi dei file temporanei, in relazione alla modalità di esecuzione;
- gli indirizzi di posta elettronica a cui vengono inviati i messaggi generati dal sistema nel caso venga eseguito in modalità TEST;
- le costanti utilizzate dalla funzione *flock()* per la definizione della tipologia di lock che il sistema deve eseguire.

In questo file sono, inoltre, dichiarate tutte le variabili globali che durante l'esecuzione del sistema conterranno le sezioni di testo utilizzate per la composizione dei vari messaggi di posta elettronica generati (vedi sezioni 4.1 e 4.2).

5.5.1 Modalità di esecuzione del sistema

Tutto il sistema di controllo sintattico può essere eseguito in modalità test assegnando il valore 1 (uno) anziché 0 (zero) alla variabile globale **$TESTMODE** definita in questo file.

Sulla base di questa variabile, sempre in questo file, è definito l'ambiente in cui viene eseguito il sistema come, ad esempio, le directory utilizzate e il nome dei file creati dal programma principale.

Inoltre, in modalità test il sistema stampa sullo standard output una serie di messaggi che permettono di verificare il percorso seguito e il risultato dei vari controlli via via effettuati a tempo di esecuzione. L'utilizzo della variabile $TESTMODE per questo scopo non verrà citato nella descrizione dei vari programmi che compongono il sistema poiché i messaggi possono variare nella quantità e nel dettaglio a seconda delle esigenze che possono sorgere durante le fasi di test.

5.5.2 Comandi Unix

Nel modulo *GlobalCheck.pm* sono inizializzate le seguenti variabili contenenti i comandi Unix utilizzati dal sistema:

$fastmail: comando *fastmail* utilizzato per l'invio dei messaggi di comunicazione allo staff tecnico del Registro;

$sendmail: comando *sendmail* utilizzato dalla funzione *main::&SendMail()* per inviare, ai P/M, i messaggi di posta elettronica generati dal sistema;

$cat: comando *cat* utilizzato dalla funzione *&main::CreateAckMail()* per la creazione del messaggio di posta elettronica contenente il risultato dei controlli effettuati dal sistema;

$whois: comando utilizzato dal sistema per effettuare le interrogazioni al database WHOIS del Registro;

$whoisHandle: comando utilizzato dal sistema per effettuare le interrogazioni al database HANDLE del Registro;

5.5.3 Definizione delle directory utilizzate

Il modulo *GlobalCheck.pm* definisce le directory che determinano l'ambiente di lavoro del sistema; quest'ultimo è dipendente dalla modalità di esecuzione e quindi dal valore della variabile $TESTMODE. Le directory definite sono le seguenti:

$LOGDIR: directory di base all'interno della quale viene creato l'ambiente dei log del sistema di controllo sintattico;

$ACKDIR: directory in cui sono creati i file di log dei messaggi di risposta contenenti il risultato dei controlli inviato all'utente; il nome dei file di log viene creato in *main.pl* a tempo di esecuzione nel formato aaaammgg dove 'aaaa, mm, gg' indicano la data odierna;

$ERRORDIR: directory in cui sono creati i file di log dei messaggi di errore inviati all'utente nel caso in cui il sistema si arresti per un malfunzionamento; il nome dei file di log viene creato nel *main.pl* a *run time* nel formato aaaammgg dove 'aaaa, mm, gg' indicano la data odierna;

$TMPDIR: directory nella quale sono creati i file temporanei contenenti i vari messaggi di posta elettronica creati dal sistema.

5.5.4 Definizione dei nomi dei file temporanei

Durante la sua esecuzione, il sistema genera dei file temporanei utilizzati per la creazione dei vari messaggi di posta elettronica che sono inviati per comunicare sia con il P/M mittente che con lo staff tecnico del Registro, nonché con gli altri sistemi che compongono il processo globale di registrazione di un nome a dominio.

Questi file sono creati nella directory *$TMPDIR,* decritta nella sezione precedente, ed hanno dei nomi prestabiliti dipendenti dalla modalità di esecuzione del sistema. Di seguito sono elencate le variabili contenenti i nomi dei file temporanei:

$ResultCheckMail: contiene il messaggio di posta elettronica che varrà inviato all'utente come risposta dei vari controlli effettuati;

$BodyCheckMail: contiene il corpo del messaggio che andrà a comporre l'intero messaggio di posta elettronica memorizzato in *$ResultCheckMail*;

$RACheckMail: contiene il messaggio di posta elettronica con il modulo elettronico che verrà inviato agli altri sistemi di controllo del processo di registrazione, nel caso in cui i controlli effettuati dal sistema siano andati a buon fine;

$RAErrorMail: contiene il messaggio di posta elettronica che verrà inviato allo staff tecnico del Registro nel caso si sia verificato un malfunzionamento durante l'esecuzione del sistema;

$UsrErrorMail: contiene il messaggio di posta elettronica che verrà inviato all'utente nel caso si sia verificato un malfunzionamento durante l'esecuzione del sistema;

$FwdMessage: è la prima parte del nome del file che contiene i messaggi di posta elettronica inviati ad un P/M, nel caso in cui questo abbia subito un tentativo di corruzione da parte di un altro P/M non autoritativo. La seconda parte del nome del file è creato a tempo di esecuzione per ogni messaggio inviato.

Nel modulo *GlobalCheck.pm* sono inoltre dichiarati i nomi di due file utilizzati dal sistema per l'esecuzione dei controlli ed anch'essi dipendenti dalla modalità di esecuzione:

$mailFile: file temporaneo in cui viene salvato il messaggio di posta elettronica esaminato dal sistema;

$MessageFile: riferimento assoluto del file *Message* in cui sono definite tutte le sezioni di testo standard che compongono i vari messaggi di posta elettronica generati.

5.5.5 Destinatari dei messaggi di posta elettronica

I vari messaggi di posta elettronica generati dal sistema sono inviati ai reali destinatari o ad indirizzi interni al Registro a seconda della modalità di esecuzione indicata nella variabile *$TESTMODE*. Questa diversificazione consente, in modalità test, di poter controllare le risposte in uscita dal sistema.

Di seguito sono indicate le variabili che contengono gli indirizzi di posta elettronica predefiniti:

$TEST_TOPM: indirizzo del destinatario del messaggio, che in modalità test, riceverà il messaggio di risposta dei controlli effettuati. A regime questo indirizzo è determinato a tempo di esecuzione e corrisponde al mittente del messaggio di posta correntemente esaminato;

$TEST_FWDMAIL: indirizzo del destinatario che, in modalità test, riceverà i messaggi contenenti un avviso di tentata corruzione di un oggetto. A regime i destinatari vengono individuati a tempo di esecuzione prendendo gli indirizzi indicati negli attributi *upd-to* dell'oggetto *mntner* associato al P/M che ha subito il tentativo di corruzione;

$ERRORMAIL: indirizzo corrispondente allo staff tecnico del Registro che riceverà i messaggi di malfunzionamento del sistema. Questo indirizzo è dipendente dalla modalità di esecuzione del sistema;

$DNSCHECK: indirizzo corrispondente al sistema di controllo dei nameserver. Questo indirizzo è dipendente dalla modalità di esecuzione del sistema;

$DBDOMAIN: indirizzo al quale viene rinviato il modulo elettronico di un oggetto *mntner* o di oggetti *person* nel caso in cui questo abbia superato con successo tutti i controlli effettuati dal sistema. Questo indirizzo è dipendente dalla modalità di esecuzione del sistema.

5.5.6 Modalità di lock di un file

I file di log, una volta creati, vengono acceduti dalle varie istanze del sistema per la registrazione dei propri messaggi. L'accesso ai file viene quindi effettuato previa esecuzione di un comando di lock, eseguito chiamando la funzione predefinita Perl *flock*. In questo modulo sono definite le variabili globali che definiscono le modalità di lock consentite al sistema:

$LOCK_SH: lock condiviso;

$LOCK_EX: lock esclusivo;

$LOCK_NB: ritorna immediatamente invece di bloccarsi in attesa di poter locare;

$LOCK_UN: rilascia un lock precedente.

5.5.7 Dichiarazione delle variabili globali che conterranno le sezioni standard di testo

In questo modulo sono dichiarate tutte le variabili globali che, durante l'esecuzione del sistema, conterranno le sezioni di testo definite nel file *Message*. Le variabili sono inizializzate dalla funzione *&SetMsgs::GetMessages()* (*sezione 5.2*) chiamata dal programma principale e utilizzate dal sistema per la creazione dei vari messaggi di posta elettronica da esso inviati.

Al fine di permettere una corretta esecuzione di tutto il sistema, queste variabili devono essere in numero uguale alle sezioni definite nel file *Message* (*sezione 5.1*) e omonime alle etichette che contraddistinguono le sezioni.

Di seguito è indicato il nome delle variabili e una breve descrizione della sezione di testo che andranno a contenere.

$MHEADER: intestazione del messaggio di risposta, inviato al P/M, contenente il risultato dei controlli;

$MAILTXT: sezione fissa del corpo del messaggio di risposta, inviato al P/M, contenente il risultato dei controlli;

$USRHEADER: intestazione del messaggio contenente il modulo elettronico da inviare alle fasi successive del processo di registrazione;

$USRERRORHEADER: intestazione del messaggio che viene inviato all'utente per una eventuale segnalazione di errore;

$USRERRORTEXT: testo del messaggio che viene inviato all'utente per una eventuale segnalazione di errore;

$FWHEADER: intestazione del messaggio che viene inviato ad un P/M nel caso si sia verificata una tentata corruzione di un oggetto da lui gestito;

$FWTEXT: sezione fissa del corpo del messaggio che viene inviato ad un P/M nel caso ti tentata corruzione di un oggetto da lui gestito;

$NICTEXT: sezione fissa del messaggio inviato ai tecnici del Registro per la segnalazione di un errore;

$CHECKOK: utilizzata per creare il messaggio di risposta da inviare all'utente, contiene la stringa stampata in testa ad ogni oggetto che ha passato con successo tutti i controlli sintattici;

$CHECKFAILED: utilizzata per creare il messaggio di risposta da inviare all'utente, contiene la stringa stampata in testa ad ogni oggetto in cui si è verificato almeno un warning o un errore;

$ACKERR: utilizzata per creare il messaggio di risposta da inviare all'utente, contiene la chiusura del messaggio nel caso in cui si sia verificato almeno un warning o un errore sugli oggetti;

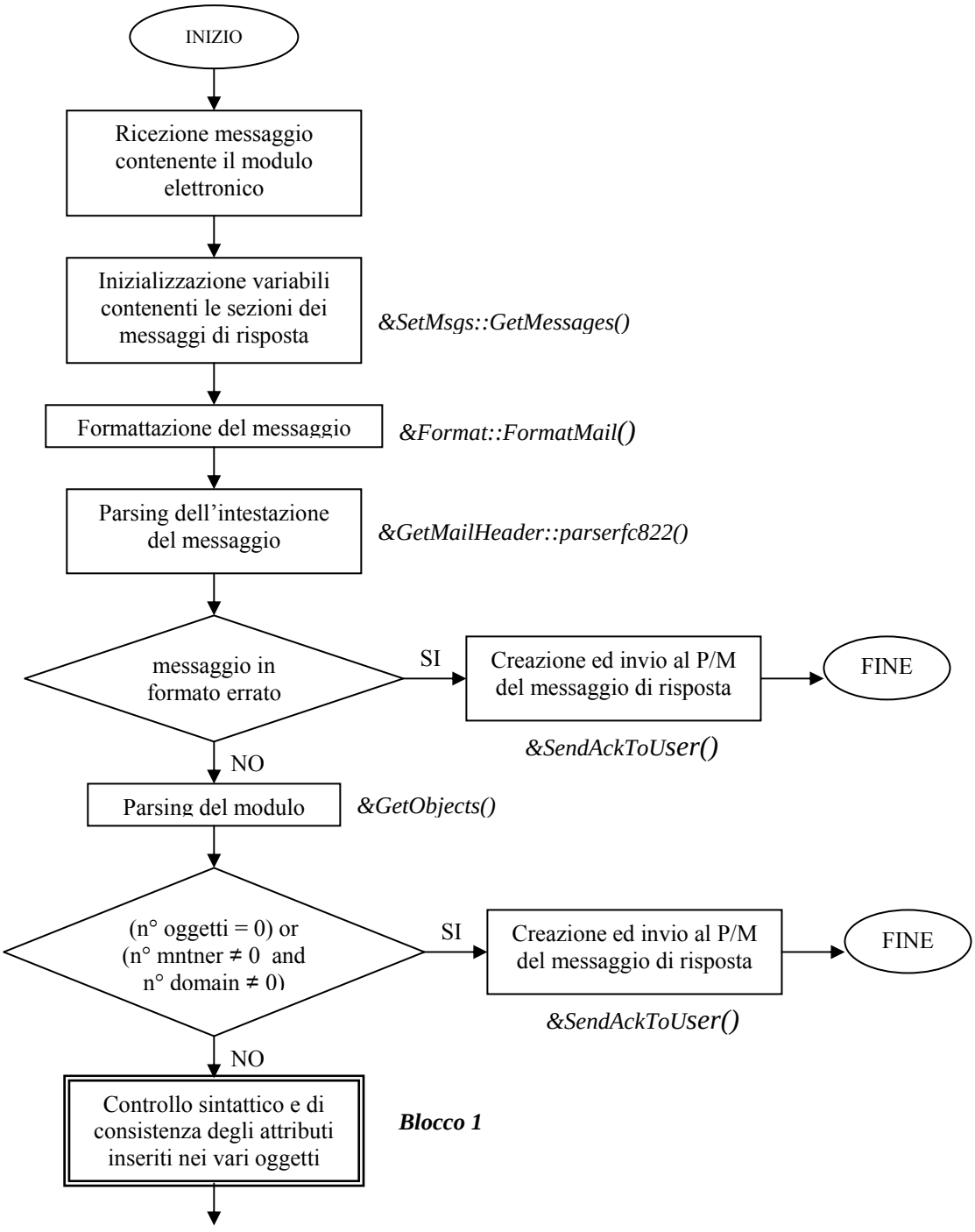
$ACKOK: utilizzata per creare il messaggio di risposta da inviare all'utente, contiene la chiusura del messaggio nel caso in cui non si sia verificato né un errore né un warning sugli oggetti;

$ACKSIG: firma utilizzata nei vari messaggi di posta elettronica creati.

5.6 syntax.pl

Questo file contiene il programma principale di tutto il sistema di controllo sintattico. Esso riceve in ingresso il messaggio con il modulo elettronico da esaminare, esegue tutti i vari controlli avvalendosi di tutti gli altri moduli e programmi che compongono il sistema ed infine invia i messaggi contenenti rispettivamente l'esito dei controlli ed il modulo da registrare.

Nel diagramma in *figura n.3* sono indicate le fasi principali che costituiscono l'intero sistema.



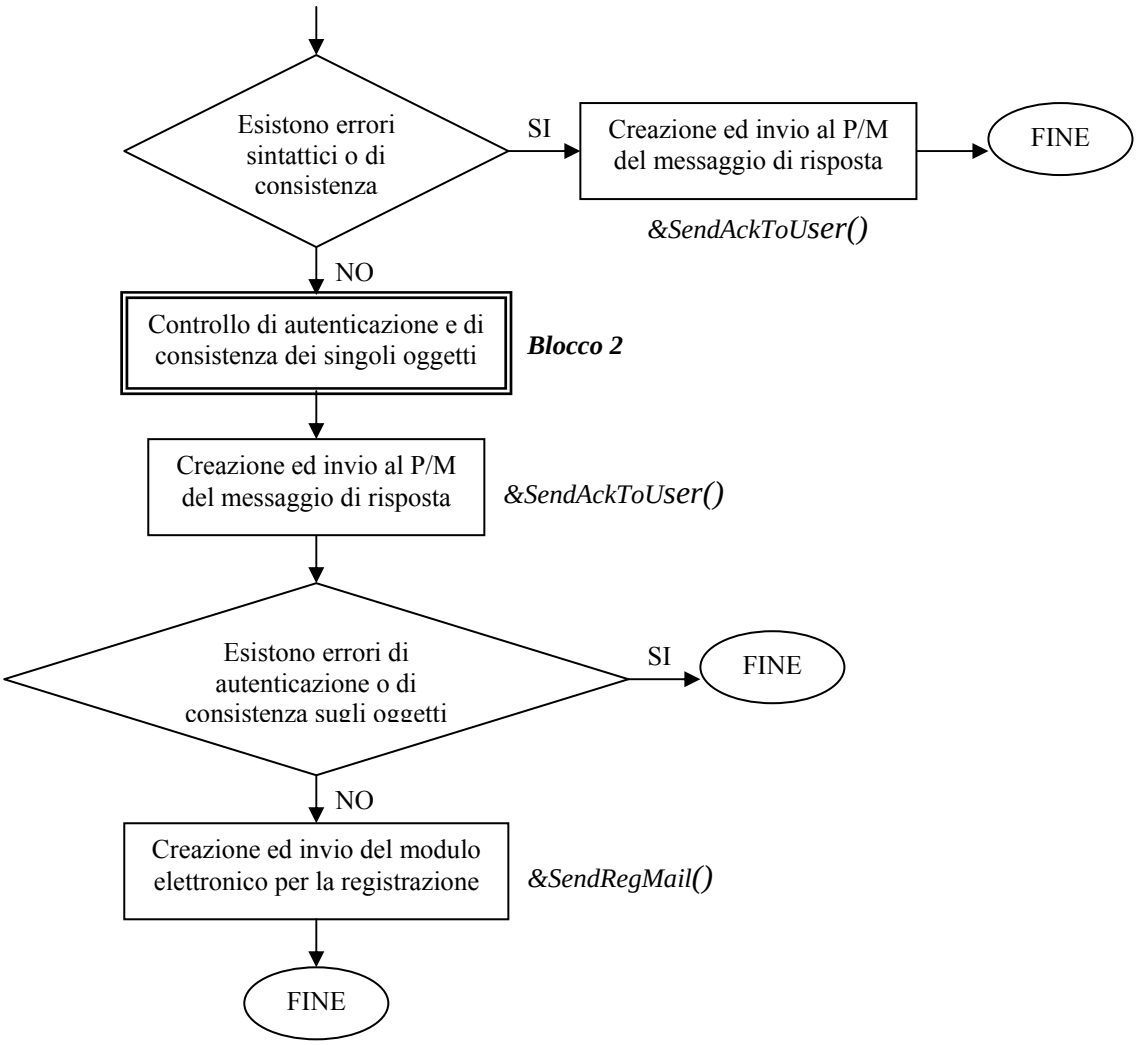


figura n.3

Prima di procedere all'analisi del modulo elettronico, il sistema esegue due fasi di inzializzazione che gli consentono di mettersi in condizione di eseguire i vari controlli. In una prima fase, si memorizza tutte le sezioni di testo standard necessarie per la creazione dei messaggi di risposta utilizzando la funzione *&SetMsgs::GetMessages()*, successivamente chiama la funzione *&Format::FormatMail()* che formatta l'intero messaggio ricevuto al fine di rendere il contenuto del corpo del messaggio in un formato il più possibile simile al formato standard in cui è descritto un modulo elettronico.

Il primo controllo del sistema viene effettuato dalla funzione *&parserfc822()* definita nel file *GetMailHeader.pl* la quale verifica che il modulo elettronico sia stato inviato all'interno del corpo del messaggio e in modalità testo. In caso contrario il programma crea ed invia al P/M il messaggio di posta elettronica contenente la segnalazione dell'errore.

Il secondo controllo verifica che all'interno del modulo elettronico ci sia almeno un oggetto (*domain*, *mntner*, *role*, *person*) e che non ci sia più di un oggetto *domain* o *mntner* oppure oggetti

domain e *mntner* contemporaneamente; questo controllo viene effettuato utilizzando la funzione *GetObjects()* che conta gli oggetti e li memorizza all'interno di appositi vettori hash dai quali verranno letti i dati per tutti i controlli futuri; nel caso in cui i controlli non vadano a buon fine, il programma genera ed invia la risposta al P/M e termina.

Seguono poi i controlli sintattici di tutti i singoli oggetti contenuti nel modulo e i controlli di consistenza dei dati inseriti nell'oggetto *domain*. Questi controlli, eseguiti nel *Blocco 1*, sono descritti in modo dettagliato più avanti in questa sezione; a conclusione dei controlli, se si presenta almeno un errore, il programma termina con l'invio della risposta al P/M.

Infine, nel *Blocco 2*, sono effettuati i controlli di consistenza di tutti gli oggetti *person* e *role* nonché i controlli di autenticazione di ogni oggetto contenuto nel modulo, tali controlli sono descritti di seguito in questa sezione.

Al termine di questa fase viene comunque creato ed inviato al P/M il messaggio di posta contenente la risposta che può essere sia positiva che negativa. Nel caso in cui la risposta sia positiva, il programma chiama la funzione *SendRegMail()* che genera il messaggio contenente il modulo per la registrazione che invia al controllo del DNS o meno a seconda che il modulo contenga o meno un oggetto *domain* o no. Tutti i messaggi contenenti l'esito dei controlli sono creati ed inviati dalla funzione *SendAckToUser().*

È da segnalare che, poiché i controlli di autenticazione e di consistenza dei dati sono legati al contenuto dei database del Registro, il programma può terminare all'interno dei Blocchi 1 e 2 nel caso in cui si verifichi un errore di connessione.

Segue la descrizione dettagliata del *Blocco 1* e del *Blocco 2*.

- **Blocco 1** In questo blocco vengono eseguiti i controlli sintattici di ogni oggetto componente il modulo elettronico come pure i controlli di consistenza di tutti i *nic-handle* menzionati e dei valori inseriti in un eventuale oggetto *domain*.

  Per ogni oggetto, componente il modulo elettronico, viene chiamata la funzione *&CheckObjects::CheckObject()* che controlla la sintassi del valore di ogni singolo attributo inserito nell'oggetto, la presenza degli attributi obbligatori e la non molteplicità degli attributi univoci.

  Se il modulo contiene un oggetto *domain* e se possibile (cioè se non si sono verificati errori sintattici sugli attributi da controllare), vengono eseguiti tutti i controlli di consistenza previsti sui suoi attributi; questi controlli sono:

- la corretta configurazione della tipologia del dominio verificando una presenza congruente degli attributi *nserver* e *zone-c* oppure *mailgate* e *gate-c*;
- unicità dei domini assegnati a persone fisiche controllando l'assenza nel database WHOIS di un codice *pin* associato ad un oggetto *domain* diverso da quello analizzato – solo se è presente l'attributo *pin* e non si è presentato un errore sintattico sul valore;
- la corretta compilazione nell'attributo *x400-domain* in relazione al valore inserito nell'attributo *domain*, solo se sono stati inseriti tutti gli attributi obbligatori; i valori confrontati vengono prima formattati trasformandoli in caratteri minuscoli ed inserendo il carattere ';' in fondo all'*x400-domain*, nel caso questo si stato omesso.

Tutti i controlli di consistenza sono eseguiti utilizzando le funzioni definite nel modulo Perl *CheckCons.pm.*

Inoltre (vedi *figura n.3*, *e figura n.3.1.2*), per ogni *nic-handle* menzionato in un eventuale oggetto *domain* o *mntner*, viene verificato che sia stato precedentemente registrato nel database HANDLE. Questo controllo è effettuato eseguendo delle interrogazioni attraverso le funzioni definite nel modulo *WhoisDBinterface.pm*. In questa fase viene memorizzato anche il nome della persona associata al *nic-handle* all'interno del database WHOIS del Registro, se l'oggetto *person* è già registrato, oppure all'interno del database HANDLE.

Nel caso in cui si verifichi un errore di connessione a uno dei due database, il programma utilizza le funzioni *&AdviseTechnicalError()* e *&SendWhoisErrorToUser()* per creare ed inviare due messaggi di posta elettronica segnalando l'errore sia allo staff tecnico del Registro, sia al P/M (vedi *sezioni 5.6.2.4 e 5.6.2.6*) e termina la sua esecuzione.

Di seguito è mostrato il diagramma di flusso del Blocco 1 e di alcuni suoi sottoblocchi.

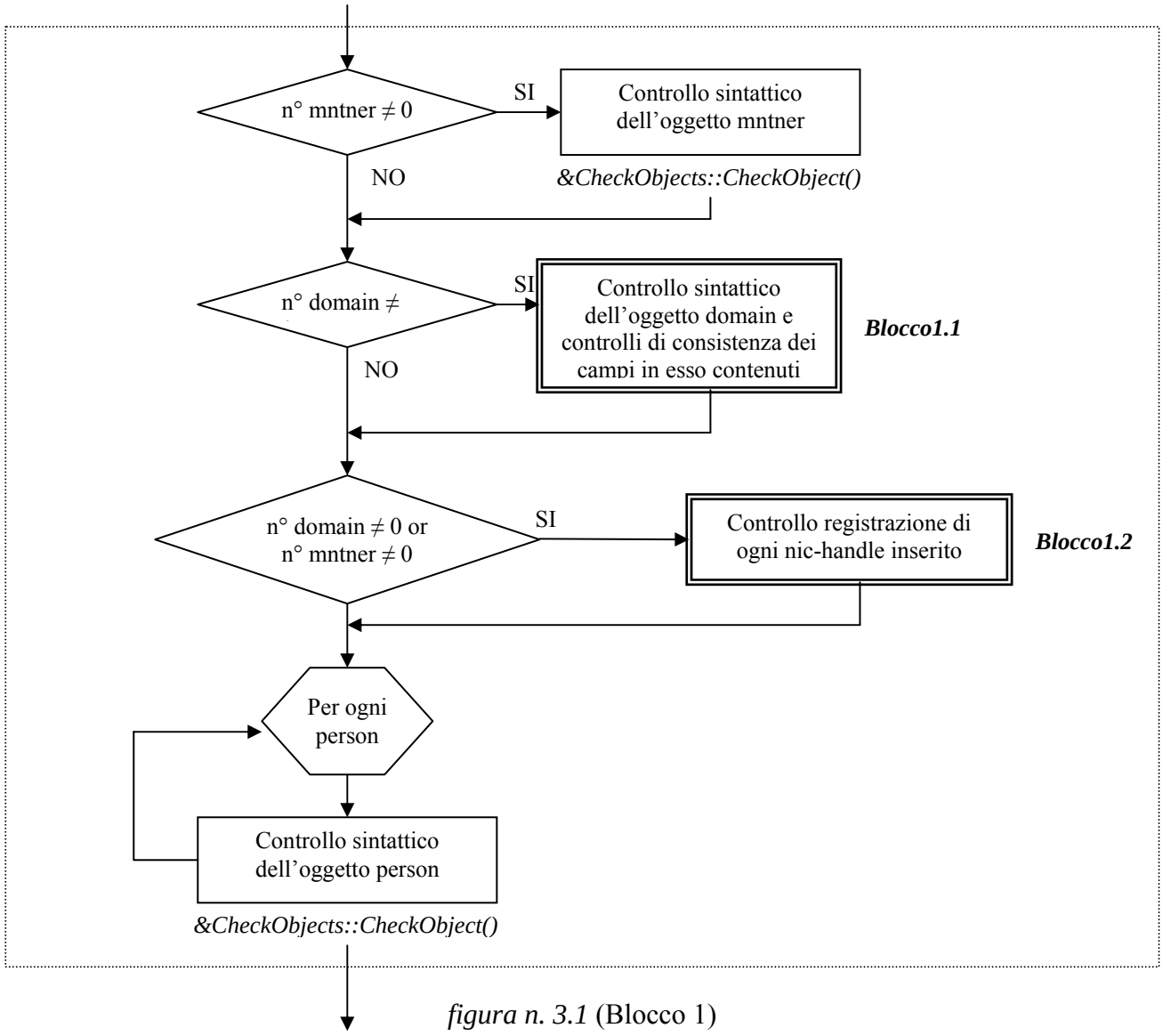


figura n. 3.1 (Blocco 1)

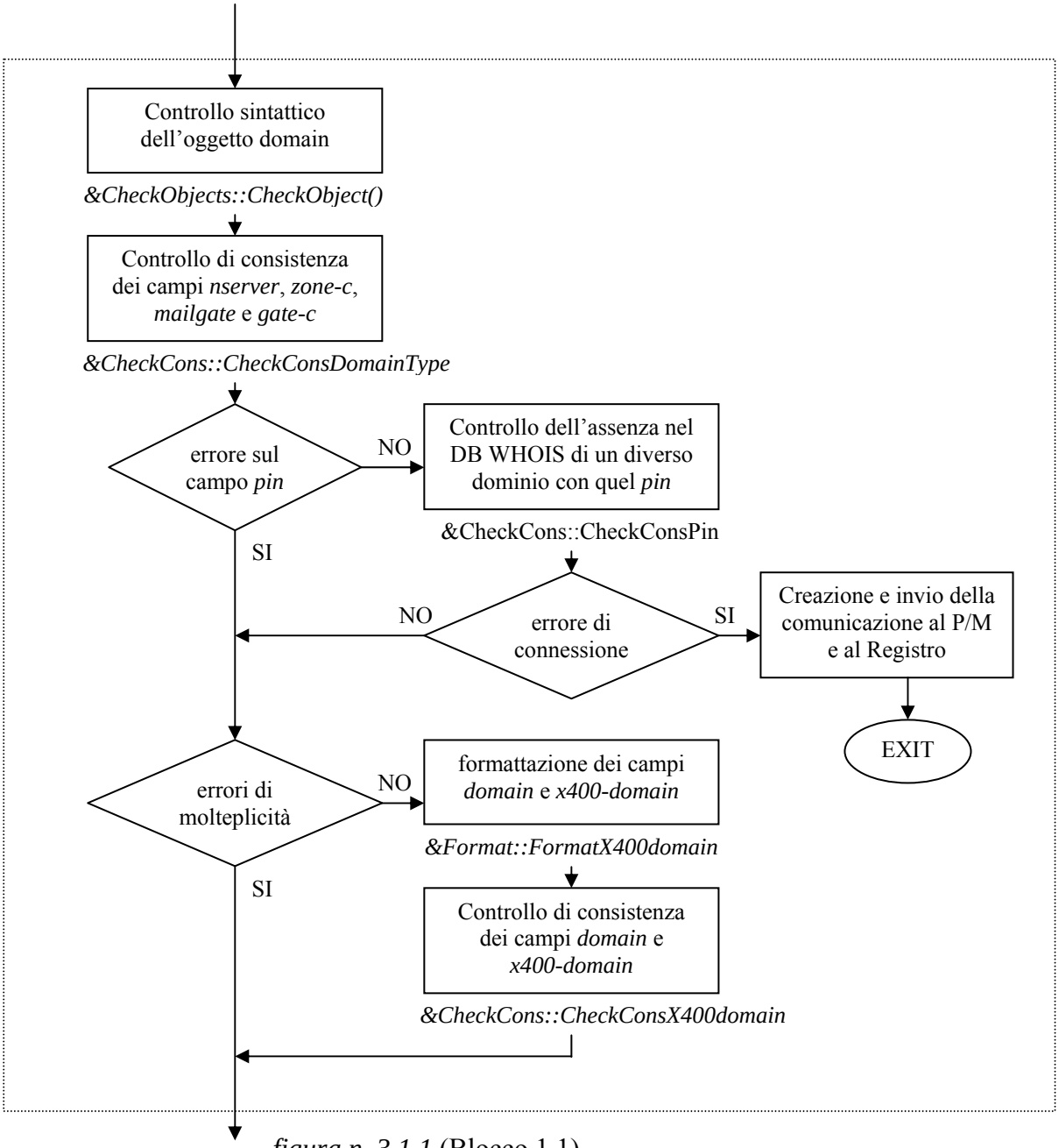


figura n. 3.1.1 (Blocco 1.1)

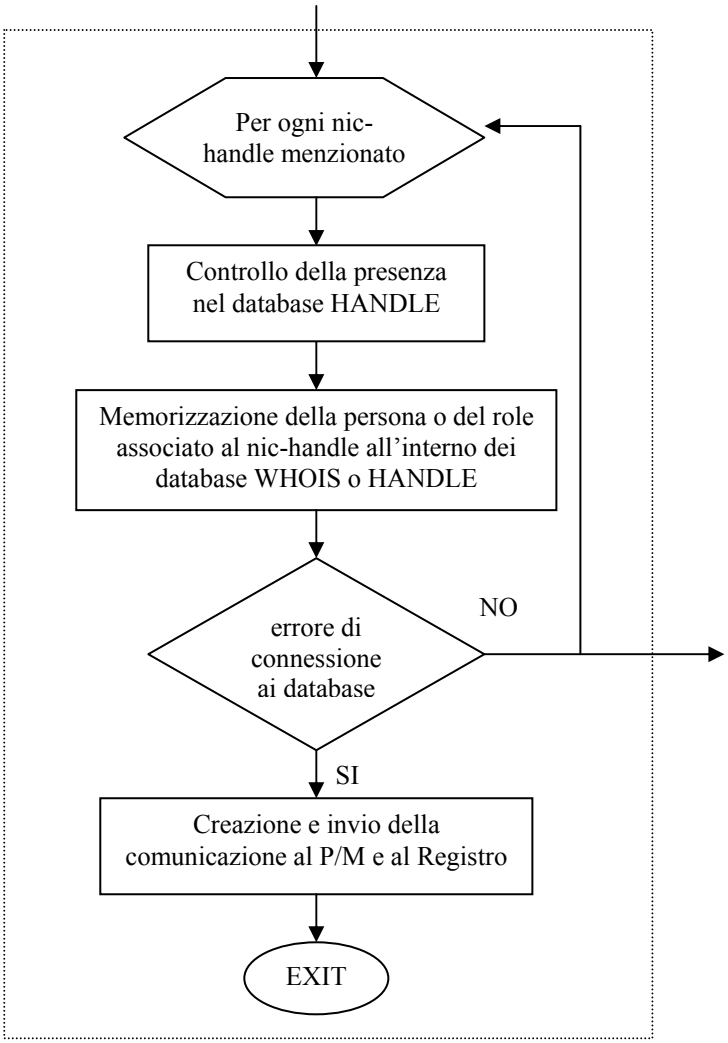


figura n. 3.1.2 (Blocco 1.2)

- **Blocco 2** In questo blocco sono eseguiti i controlli di autenticazione su tutti gli oggetti contenuti nel modulo nonché i controlli di consistenza di ogni contatto tecnico e amministrativo (oggetto *person* o *role*) in relazione al contenuto dei database WHOIS e HANDLE e contatti referenziati all'interno di un eventuale oggetto *domain* o *mntner*. Nella *figura n.3.2* è riportato il diagramma a blocchi che ne specifica i principali controlli.

  I controlli di autenticazione sui singoli oggetti sono realizzati con la chiamata alle funzioni definite nel modulo Perl *CheckAuth.pm.*

  Prima di effettuare i controlli sugli oggetti *person*, il programma controlla se nel modulo elettronico è presente un oggetto *domain* o *mntner*. Se il modulo contiene solo oggetti *person*, il programma esegue subito il controllo di autenticazione su ogni oggetto, altrimenti, esegue il controllo su un oggetto solo se:

  - l'oggetto è referenziato nell'eventuale oggetto *domain* o *mntner*;
  - il *nic-handle* ad esso associato non è già stato assegnato ad un'altra persona all'interno dei database del Registro;
  - il controllo di autenticazione non coincide con quello già eseguito per l'oggetto *domain* o *mntner*, ovvero se non è stato rilevato un errore di maintainer non autoritativo, oppure il contenuto degli attributi *mnt-by* è diverso ed è stata inserita una password nella *person* diversa da quella inserita per l'oggetto che lo referenzia.

  Al termine di ogni controllo di autenticazione sono, presi in particolare considerazione l'errore di connessione e quello di maintainer non autoritativo a seguito dei quali il programma genera ed invia appropriati messaggi di posta elettronica:

  - nel primo caso, vengono chiamate le funzioni *&AdviseTechnicalError()* e *&SendWhoisErrorToUser()* per segnalare l'errore rispettivamente allo staff tecnico del Registro ed al P/M mittente, ed il programma termina;
  - nel secondo caso, viene chiamata la funzione *&SendFwdToMntner()* la quale avvisa il P/M gestore dell'oggetto 'violato' del tentativo di corruzione.

Segue il diagramma di flusso dettagliato.

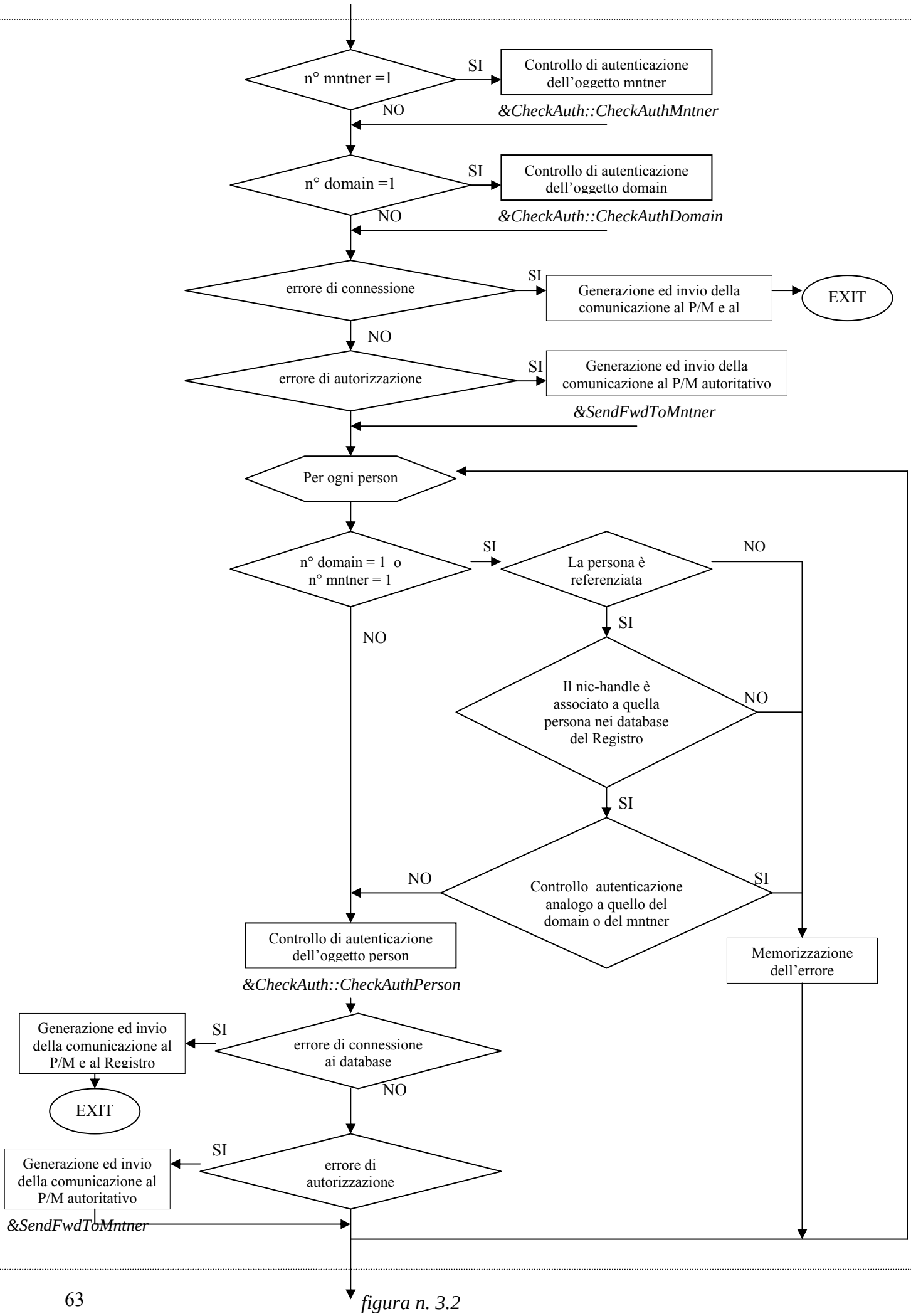

n° mntner =1
SI
Controllo di autenticazione dell'oggetto mntner
NO
&CheckAuth::CheckAuthMntner
n° domain =1
SI
Controllo di autenticazione dell'oggetto domain
NO
&CheckAuth::CheckAuthDomain
errore di connessione
SI
Generazione ed invio della comunicazione al P/M e al
EXIT
NO
errore di autorizzazione
SI
Generazione ed invio della comunicazione al P/M autoritativo
&SendFwdToMntner
Per ogni person
n° domain = 1 o n° mntner = 1
SI
La persona è referenziata
NO
SI
NO
Il nic-handle è associato a quella persona nei database del Registro
NO
SI
Controllo autenticazione analogo a quello del domain o del mntner
NO
SI
Controllo di autenticazione dell'oggetto person
&CheckAuth::CheckAuthPerson
Memorizzazione dell'errore
Generazione ed invio della comunicazione al P/M e al Registro
SI
errore di connessione ai database
NO
EXIT
Generazione ed invio della comunicazione al P/M autoritativo
SI
errore di autorizzazione
&SendFwdToMntner

figura n. 3.2

5.6.1 Variabili globali definite all'interno del file

All'interno del file main.pl sono definite delle variabili principali che, nel corso dell'esecuzione del programma contengono, in forma strutturata, tutti i dati che costituiscono il messaggio da analizzare come pure i dati sugli errori via via rilevati.

Queste variabili sono utilizzate dal programma principale e da tutte le funzioni definite nel file *main.pl*. Di seguito è riportato l'elenco:

%DOMAIN: Contiene gli attributi dell'oggetto *domain* nel formato (*etichetta, valore*);

%MNTNER: Contiene gli attributi dell'oggetto *mntner* nel formato (*etichetta, valore*);

@PERSONS: Contiene gli attributi di ogni oggetto *person* contenuto nel modulo nel formato *$PERSONS[i]=[%PERSON]* dove *%PERSON* contiene gli attributi di un oggetto nel formato (*etichetta, valore*);

$numdomain: Contiene il numero degli oggetti *domain* contenuti nel modulo;

$nummntner: Contiene il numero degli oggetti *mntner* contenuti nel modulo;

$numperson: Contiene il numero degli oggetti *person* contenuti nel modulo;

%DOMAINSYNTAXERROR: Contiene gli errori sintattici rilevati dell'oggetto *domain* nel formato (*etichetta, codice_errore*);

%MNTNERSYNTAXERROR: Contiene gli errori sintattici rilevati dell'oggetto *mntner* nel formato (*etichetta, codice_errore*);

@PERSONSYNTAXERROR: Contiene gli errori sintattici rilevati negli oggetti *person* nel formato *$PERSONSYNTAXERROR[i]=[%error]* dove *%error* contiene gli errori di un singolo oggetto *person* nel formato (*etichetta, codice_errore*);

%NICHDLERROR: Contiene i *nic-handle* che non sono stati trovati nel database HANDLE nel formato (*etichetta, nichdl*). È utilizzato per il controllo di consistenza dei *nic-handle* contenuti negli oggetti *domain* e *mntner*;

$PINERROR: Contiene il codice di errore rilevato durante il controllo di

consistenza sull'attributo *pin*.

$numnserver, $numzonec, $nummailgate, $numgatec:	Numero di attributi *nserver, zone-c*, *mailgate* e *gate-c* presenti nell'oggetto *domain*; Vengono utilizzate nel controllo di consistenza della configurazione del nome a dominio;
%DOMAINCONSISTENTERROR:	Contiene gli errori di consistenza rilevati per l'oggetto *domain* nel formato (*nome campo, codice_errore*);
$DOMAINAUTHERROR:	Contiene il codice corrispondente all'errore di autenticazione rilevato per l'oggetto *domain*;
$MNTNERAUTHERROR:	Contiene il codice corrispondente all'errore di autenticazione rilevato per l'oggetto *mntner*;
@PERSONAUTHERROR:	Contiene i codici degli errori di autenticazione rilevati negli oggetti *person* nell'ordine di presenza degli oggetti nel modulo;
%NICPERSON:	Contiene le associazioni tra i *nic-handle* menzionati negli oggetti *domain* o *mntner* e le persone a questi associati nei database del Registro nel formato (*nic-handle*, *nome persona*). È utilizzato per il controllo di consistenza degli oggetti *person* contenuti nel modulo;
%NICHDLEDATA:	Contiene i dati degli oggetti *person* che dovranno essere aggiunti al modulo elettronico prima di inviarlo per la registrazione. I dati sono nel formato *(nic-handle, [@person])* dove *nic-handle* è il codice del *nic-handle* menzionato nell'oggetto *domain* o *mntner* e *@person* contiene i dati dell'oggetto *person* più aggiornato associato ad esso suddiviso per righe;
$domain:	Nome del dominio inserito nell'oggetto *domain* contenuto nel modulo;
$mntner:	Nome del maintainer inserito nell'oggetto *mntner* contenuto nel modulo;
$FROM, $SUBJECT, $MDATE,	Valore dei campi *From*, *Subject*, *Date* e *Message-Id*

$MSGID: contenuti nell'intestazione del messaggio di posta elettronica esaminato. Sono utilizzate per la ricostruzione dell'intestazione del messaggio analizzato all'interno dei messaggi restituiti in uscita;

$logFile: File di log in cui sono registrati i moduli elettronici in uscita dal sistema di controllo sintattico. Il file è definito come *$GlobalCheck::ACKDIR."$date"* dove *$date* contiene la data odierna nel formato YYYYMMDD;

$logErrorFile: File di log in cui sono registrati i messaggi di segnalazione inviati allo staff tecnico del Registro, in seguito ad un errore di connessione ai database. Il file è definito come *$GlobalCheck::ERRORDIR."$date"* dove *$date* contiene la data odierna nel formato YYYYMMDD.

5.6.2 Funzioni definite all'interno del file

All'interno del file *main.pl* sono definite delle procedure la cui definizione ha permesso di rendere il programma principale più modulare. L'inclusione delle procedure all'interno del file ha permesso di tener conto dello stato dei controlli effettuati e del contenuto dei vari oggetti utilizzando le variabili globali, descritte nella sezione precedente, visibili all'interno di tutto il file.

Le procedure principali sono:

- ***GetObjects():*** Memorizza il numero degli oggetti contenuti nel modulo e gli attributi che li compongono in apposite variabili globali che verranno utilizzate dal programma principale e dalle procedure definite nel file;
- ***SendAckToUser():*** Genera il messaggio di posta elettronica contenente l'esito dei controlli e lo invia al P/M mittente;
- ***SendFwdToMntner():*** Genera il messaggio di posta elettronica contenente una comunicazione di tentata corruzione di un oggetto e lo invia al P/M gestore dell'oggetto violato;
- ***SendWhoisErrorToUser():*** Genera un messaggio di posta elettronica contenente la comunicazione di avvenuta interruzione dei controlli causata da un errore di connessione a uno dei database del Registro e lo invia al P/M mittente;
- ***SendRegMail():*** Genera un messaggio di posta elettronica contenente il modulo elettronico inviato dal P/M opportunamente modificato e lo invia alle successive procedure di controllo (controllo della configurazione dei nameserver, ecc.);
- ***LogSyntaxCheck():*** Esegue il log dei messaggi di posta inviati al P/M contenenti le risposte dei controlli;
- ***LogErrorSyntaxCheck():*** Esegue il log dei messaggi di errore che vengono inviati allo staff tecnico del Registro.

All'interno del file sono inoltre definite le seguenti procedure utilizzate dalle procedure sopra elencate:

- ***CreateAckMail():*** Genera il messaggio contenente il risultato dei controlli (vedi *sezione 5.6.2.9*). Questa procedura si avvale delle funzioni ***PrintAckHeader()***, ***PrintAckBoby()*** e ***PrintSyntaxError(),*** definite anch'esse all'interno del file, per la creazione dell'intestazione e del corpo della mail con la segnalazione di tutti gli errori rilevati;
- ***CreateRegMail():*** Genera il messaggio di posta contenente il modulo elettronico da inviare per la registrazione (vedi *sezione 5.6.2.10*);
- ***CreateFwdMail():*** Genera il messaggio di posta contenente la comunicazione di tentata corruzione che verrà inviata al P/M gestore dell'oggetto incriminato (vedi *sezione 5.6.2.11*);
- ***CreateErrorMail():*** Genera il messaggio di posta che verrà inviato allo staff tecnico del Registro, contenente la descrizione dettagliata di un errore avvenuto durante l'esecuzione del programma (vedi *sezione 5.6.2.12*);
- ***CreateUsrErrorMail():*** Genera il messaggio di posta, che verrà inviato al P/M mittente, contenente la comunicazione di avvenuto errore durante la connessione del database (vedi *sezione 5.6.2.13*);
- ***SendMail():*** Invia il messaggio di posta elettronica passato come parametro.

Le seguenti sezioni descrivono le procedure sopra elencate.

5.6.2.1 GetObjects()

Questa funzione esamina il messaggio di posta elettronica ricevuto in ingresso dal programma principale, memorizza gli oggetti contenuti nel modulo in apposite variabili globali e restituisce i codici di errore relativi alla consistenza del modulo elettronico.

IN

- **@form:** vettore contenente il modulo elettronico suddiviso per righe;

OUT

- **@MailError:** vettore contenente i codici di errore eventualmente rilevati:

  - ERR_TOO_MANY_DOMAIN, se il modulo elettronico contiene piu' di un oggetto *domain*,

  - ERR_TOO_MANY_MNTNER, se il modulo elettronico contiene piu' di un oggetto *mntner*,

  - ERR_INCONSISTENT_MESSAGE, se il modulo elettronico contiene sia oggetti *domain* che *mntner* contemporaneamente;

  - ERR_NONE, altrimenti.

GLOBALI INIZIALIZZATE

- **%DOMAIN:** contiene gli attributi presenti nell'oggetto *domain*;
- **%MNTNER:** contiene gli attributi presenti nell'oggetto *mntner*;
- **@PERSONS:** vettore di puntatori ad hash in cui vengono inseriti gli attributi degli oggetti *person*;
- **@ROLES:** vettore di puntatori ad hash in cui vengono inseriti gli attributi degli oggetti *role*;
- **$numdomain:** contiene il numero degli oggetti *domain* contenuti nel modulo;
- **$nummntner:** contiene il numero degli oggetti *mntner* contenuti nel modulo;
- **$numperson:** contiene il numero degli oggetti *person* contenuti nel modulo;
- **$numrole:** contiene il numero degli oggetti *role* contenuti nel modulo;
- **$numnserver:** contiene il numero di attributi *nserver* trovati nell'oggetto *domain*;

- **$numzonec:** contiene il numero di attributi *zone-c* trovati nell'oggetto *domain*;
- **$nummailgate:** contiene il numero di attributi *mailgate* trovati nell'oggetto *domain*;
- **$numgatec:** contiene il numero di attributi *gate-c* trovati nell'oggetto *domain*;

DESCRIZIONE:

La funzione esamina tutte le righe del modulo, suddivide l'etichetta dal valore in base al carattere ':' e inizializza le variabili globali impiegate come contatori in base alle etichette trovate.

I vettori hash contenenti gli oggetti sono riempiti ogni volta che la funzione trova una riga vuota, in base all'ultima etichetta (*domain*, *mntner* e *person*) esaminata. Il vettore contenente gli oggetti *person* viene riempito nell'ordine in cui sono stati inseriti gli oggetti nel modulo.

Per ogni campo contenente un *nic-handle* (*admin-c*, *tech-c*, *postmaster*, *zone-c*, *nic-hdl, ecc.*), la funzione trasforma il valore in lettere maiuscole.

Una volta esaminate tutte le righe del modulo, la funzione legge i contatori degli oggetti esaminati e in base a questi inserisce i relativi codici di errore nel vettore *@MailError*.

5.6.2.2 SendAckToUser()

Questa funzione crea il messaggio di posta elettronica contenente la risposta del controllo sintattico del modulo esaminato, lo registra in un file di log ed infine lo invia al mittente.

Nel caso in cui l'invio del messaggio non vada a buon fine, viene avvisato lo staff tecnico del Registro dell'avvenuto malfunzionamento tramite un messaggio di posta elettronica.

IN

- **nessuno**

OUT

- **$send :** indica se i controlli sintattici sono andati a buon fine (1) o meno (0);

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::ResultCheckMail:** nome del file in cui viene creato il messaggio;

FUNZIONI UTILIZZATE

- **&CreateAckMail():** creazione del messaggio;
- **&LogSyntaxCheck():** registrazione del messaggio nel file di log;
- **&SendMail():** invio del messaggio;
- **&AdviseTechnicalError():** invio della comunicazione di malfunzionamento della funzione *&SendMail* allo staff tecnico del Registro.

DESCRIZIONE:

La funzione chiama la procedura *&CreateAckMail()* che crea il messaggio di posta contenente la risposta e segnala se i controlli sono andati a buon fine; il messaggio viene registrato in un determinato file di log dalla funzione *&LogSyntaxCheck()* ed infine inviato dalla funzione *&SendMail()*.

Nel caso in cui la funzione *&SendMail()* restituisca un errore, la funzione invia un messaggio di segnalazione allo staff tecnico del Registro chiamando la *&AdviseTechnicalError* ed il programma termina; altrimenti restituisce il valore dato in uscita dalla *&CreateAckMail()* che indica l'esito complessivo dei controlli effettuati.

5.6.2.3 SendFwdToMntner()

Questa funzione crea e spedisce tutti i messaggi di posta elettronica necessari nel caso in cui si verifichi un errore di tentata corruzione di un oggetto da parte di un P/M non autoritativo.

Nel caso in cui l'invio di un messaggio non vada a buon fine, viene avvisato lo staff tecnico del Registro dell'avvenuto malfunzionamento tramite un messaggio di posta elettronica ed il programma termina.

IN

- **$id:** contiene il nome dell'oggetto che ha subito il tentativo di corruzione;
- **$updPtr:** puntatore ad un vettore contenente gli indirizzi e-mail dei destinatari dei messaggi;
- **$objectPtr:** puntatore ad un vettore contenente l'oggetto che ha subito il tentativo di corruzione;

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::ResultCheckMail:** nome del file in cui viene creato il messaggio;

FUNZIONI UTILIZZATE

- **&CreateFwdMail():** creazione dei messaggi da inviare;
- **&SendMail():** invio del messaggio;
- **&AdviseTechnicalError():** invio della comunicazione di malfunzionamento della funzione *&SendMail()* allo staff tecnico del Registro.

DESCRIZIONE:

La funzione crea tutti i messaggi che devono essere inviati chiamando la funzione *&CreateFwdMail()*; ogni file creato viene inviato utilizzando la funzione *&SendMail()*.

Nel caso in cui la funzione *&SendMail()* restituisca un errore, la funzione invia un messaggio di segnalazione allo staff tecnico del Registro chiamando la *&AdviseTechnicalError*() ed il programma termina.

5.6.2.4 SendWhoisErrorToUser()

Questa funzione crea il messaggio di posta elettronica contenente una segnalazione di malfunzionamento verificatosi durante il collegamento al database WHOIS o HANDLE, lo registra in un file di log e lo spedisce al mittente del modulo esaminato.

Nel caso in cui l'invio di un messaggio non vada a buon fine, viene avvisato lo staff tecnico del Registro dell'avvenuto malfunzionamento tramite un messaggio di posta elettronica ed il programma termina.

IN

- **nessuno**

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::UsrErrorMail:** nome del file in cui viene creato il messaggio da inviare;

FUNZIONI UTILIZZATE

- **&CreateUsrErrorMail():** creazione del messaggio da inviare;
- **&LogErrorSyntaxCheck():** registrazione del messaggio nel file di log;
- **&SendMail():** invio del messaggio;
- **&AdviseTechnicalError():** invio della comunicazione di malfunzionamento della funzione *&SendMail()* allo staff tecnico del Registro.

DESCRIZIONE:

La funzione crea il messaggio contenente la segnalazione di errore chiamando la funzione *&CreateUsrErrorMail*(), il messaggio viene poi salvato in un determinato file di log con la funzione *&LogErrorSyntaxCheck()* e spedito utilizzando la funzione *&SendMail()*.

Nel caso in cui la funzione *&SendMail()* restituisca un errore, la funzione invia un messaggio di segnalazione allo staff tecnico del Registro chiamando la *&AdviseTechnicalError()* ed il programma termina.

5.6.2.5 SendRegMail()

Questa funzione crea il messaggio di posta elettronica contenente il modulo elettronico da inviare per procedere alla registrazione.

Nel caso in cui l'invio di un messaggio non vada a buon fine, viene avvisato lo staff tecnico del Registro dell'avvenuto malfunzionamento tramite un messaggio di posta elettronica ed il programma, in questo caso, termina.

IN

- **nessuno**

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::RACheckMail:** nome del file temporaneo in cui viene creato il messaggio

FUNZIONI UTILIZZATE

- **& CreateRegMail():** creazione del messaggio da inviare;
- **&SendMail():** invio del messaggio;
- **&AdviseTechnicalError():** invio della comunicazione di malfunzionamento della funzione *&SendMail()* allo staff tecnico del Registro.

DESCRIZIONE

La procedura crea il messaggio contenente il modulo elettronico da registrare con la funzione *&CreateRegMail()* e lo spedisce utilizzando la funzione *&SendMail()*.

Nel caso in cui la funzione *&SendMail()* restituisca un errore, la funzione invia un messaggio di segnalazione allo staff tecnico del Registro chiamando la *&AdviseTechnicalError()* ed il programma termina.

5.6.2.6 AdviseTechnicalError()

La procedura invia allo staff tecnico del Registro un messaggio di posta elettronica contenente una segnalazione di errore che ha causato l'interruzione del processo in corso.

IN

- **$message** stringa contenente la descrizione dell'errore verificatosi;

OUT

- **nessuno**

FUNZIONI UTILIZZATE

- **&CreateErrorMail():** utilizzata per la creazione del corpo del messaggio da inviare;
- **&GlobalCheck::fastmail:** programma client utilizzato per l'invio del messaggio.

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::RAErrorMail:** nome del file contenente il corpo del messaggio da inviare;
- **$GlobalCheck::ERRORMAIL:** contiene gli indirizzi di posta elettronica ai quali viene inviato il messaggio.

DESCRIZIONE

La procedura crea il corpo del messaggio chiamando la funzione &CreateErrorMail() e passandogli come parametro la variabile di ingresso *$message*.

Il messaggio creato è inviato agli indirizzi contenuti nella variabile globale *$GlobalCheck::ERRORMAIL* utilizzando il comando *fastmail* contenuto nella variabile *&GlobalCheck::fastmail*. Il comando eseguito è il seguente:

$GlobalCheck::fastmail *-f "Syntax Check detecting error"*
-F hostmaster@nic.it
-s "Error during the new syntax check execution"
$GlobalCheck::RAErrorMail
GlobalCheck::ERRORMAIL

5.6.2.7 LogSyntaxCheck()

Registra il messaggio contenente il modulo elettronico in uscita dal sistema in un apposito file di log.

IN

- **$file:** nome del file contenente il messaggio da registrare.

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$logFile:** nome del file di log in cui viene registrato il messaggio.

FUNZIONI UTILIZZATE

- **&utility::log():** registrazione del file.

DESCRIZIONE

La procedura aggiunge in testa al file *$file*, ricevuto in ingresso, la stringa di delimitazione '>>> MAIL ACK <<<' seguita da due linee bianche e, successivamente, registra il risultato nel file di log *$logFile* utilizzando la procedura *&utility::log()*.

5.6.2.8 LogErrorSyntaxCheck()

La procedura registra in un apposito file di log il messaggio contenente la segnalazione di errore inviato allo staff tecnico del Registro.

IN

- **nessuno**

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::UsrErrorMail:** nome del file contenente il messaggio da registrare;
- **$logErrorFile:** nome del file di log in cui viene registrato il messaggio.

FUNZIONI UTILIZZATE

- **&utility::log():** registrazione del file.

DESCRIZIONE:

La procedura aggiunge in testa al file *$GlobalCheck::UsrErrorMail* una stringa riga vuota di delimitazione e successivamente registra il risultato nel file di log *$logErrorFile* utilizzando la procedura *&utility::log()*.

5.6.2.9 CreateAckMail()

Crea il messaggio di posta elettronica contenente la risposta del sistema sintattico che viene inviato al P/M mittente.

IN

- **nessuno**

OUT

- **$register:** indica se il modulo inviato dall'utente deve proseguire i controlli per la registrazione (1) oppure no (0).

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::ResultCheckMail:** nome del file in cui viene creato l'intero messaggio;
- **$GlobalCheck::BodyCheckMail:** nome del file in cui viene creato il corpo del messaggio;
- **$CHECKRESULT:** variabile inserita nel 'subject' del messaggio dalla funzione *&PrintAckHeader(),* per indicare se i controlli sono andati a buon fine o meno;
- **$GlobalCheck::cat:** comando Unix *cat*.

FUNZIONI UTILIZZATE

- **&PrintAckHeader():** stampa l'intestazione del messaggio;
- **&PrintAckBoby():** stampa il corpo del messaggio.

DESCRIZIONE:

La procedura apre il file *$GlobalCheck::BodyCheckMail* e vi crea l'intestazione del messaggio utilizzando la funzione *&PrintAckBoby()*. Sulla base del valore restituito da quest'ultima assegna il valore SUCCEDED o FAILED alla variabile *$CHECKRESULT* e successivamente crea l'interstazione del messaggio nel file *$GlobalCheck::ResultCheckMail* utilizzando la procedura *&PrintAckHeader()*.

Infine, l'intestazione ed il corpo del messaggio vengono uniti nel file *$GlobalCheck::ResultCheckMail* utilizzando il comando *$GlobalCheck::cat*.

La funzione restituisce in uscita il valore della funzione *&PrintAckBoby()*.

5.6.2.10 CreateRegMail()

Questa procedura crea il messaggio di posta elettronica contenente il modulo elettronico inviato dal sistema ai successivi controlli previsti dal processo di registrazione (vedi *sezione 4.2*).

Il corpo del messaggio contiene gli stessi oggetti contenuti nel modulo elettronico processato con l'eventuale aggiunta degli oggetti *person* o *role* necessari alle successive fasi di verifica.

Se il modulo contiene un oggetto *domain* o *mntner*, l'oggetto *person* aggiunto è eventualmente quello corrispondente all' *admin-c*, se non presente nel modulo, oppure gli oggetti *person* o *role* riferiti negli oggetti del modulo ma che non sono né registrati nel database WHOIS né presenti all'interno del modulo stesso.

Il destinatario del messaggio è impostato ad un indirizzo diverso a seconda che contenga o meno un oggetto *domain*.

IN

- **nessuno**

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::RACheckMail:** nome del file in cui viene creato l'intero messaggio;
- **$GlobalCheck::$DNSCHECK:** indirizzo di posta elettronica corrispondente al programma per il controllo dei nameserver;
- **$GlobalCheck::$DBDOMAIN:** indirizzo di posta dal quale verranno prelevati i moduli per i controlli successivi sui *mntner* e sulle *person*;
- **$USRHEADER:** contenuta nel modulo *GlobalCheck.pm,* contiene l'intestazione del messaggio;
- **$X400WARNING:** indica se si è presentato un warning sull'attributo *x400-domain*;
- **%SyntaxError::ERROR_MESSAGES:** hash contenente tutti i possibili messaggi di errore nel formato *(codice_errore,*

messaggio);

- **$numdomain, $nummntner, $numrole, $numperson:** numero di oggetti *domain*, *mntner* e *person* inseriti nel modulo elettronico;
- **%DOMAIN, %MNTNER, @ROLES, @PERSONS:** contengono i dati degli oggetti *domain*, *mntner* e *person* presenti nel modulo elettronico;
- **%NICHDLEDATA:** Hash contenente i dati di tutti gli oggetti *person* o *role* riferiti nell'oggetto *domain* o *mntner* che non sono stati ancora registrati o che sono associati a attributi admin-c. I dati sono nel formato *(nic-handle, [@person])* dove *@person* contiene i dati di un oggetto divisi per righe;
- **@ CONTACTNICHDL:** contiene i *nic-handle* associati ad ogni oggetto *person* o *role* presente nel modulo elettronico.

FUNZIONI UTILIZZATE

- **&WriteObject::WriteObjects():** stampa di un singolo oggetto.

DESCRIZIONE:

La procedura crea il messaggio nel file $GlobalCheck::RACheckMail.

L'intestazione del messaggio viene creata stampando il campo To: con l'indirizzo *$GlobalCheck::DNSCHECK* o *$GlobalCheck::DBDOMAIN* a seconda che nel modulo sia presente un oggetto *domain* o meno e completata con il contenuto della variabile *$USRHEADER*.

Se la variabile *$X400WARNING* segnala un errore, il corpo del messaggio inizia con un warning agli operatori definito nel campo *%SyntaxError::ERROR_MESSAGES[$X400WARNING]*; successivamente vengono stampati gli oggetti contenuti nel modulo elettronico esaminato partendo dagli oggetti di tipo *domain* o *mntner* seguiti dagli oggetti *role* e *person*; infine il modulo viene completato aggiungendo gli oggetti *person* i cui *nic-handle* sono contenuti nell'hash *%NICHDLEDATA* ma non nel vettore *@CONTACTNICHDL*.

Tutti gli oggetti sono stampati appositamente formattati utilizzando la funzione *&WriteObject::WriteObjects().*

5.6.2.11 CreateFwdMail()

Questa procedura crea i messaggi di posta elettronica inviati dal sistema nel caso in cui si verifichi un tentativo di modifica di un oggetto da parte di un P/M non autorizzato. Questi messaggi sono destinati a tutti gli indirizzi di posta elettronica indicati negli attributi *upd-to* dell'oggetto *mntner* gestore degli oggetti violati.

Il corpo del messaggio (*vedi sezione 4.3.4*) è composto da una prima sezione standard seguita dalla stampa dell'oggetto violato, così come contenuto nel modulo elettronico analizzato, e da una sezione finale fissa.

IN

- **$id:** tipo di oggetto violato (*domain*, *mntner*, *role*, *person*);
- **$updPtr:** puntatore ad un vettore contenente gli indirizzi e-mail dei destinatari del messaggio;
- **%object:** hash contenente l'oggetto da stampare nel messaggio nel formato *(attributo, valore)*.

OUT

- **$FileToSend:** puntatore ad un vettore contenente il nome dei file con i messaggi creati.

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::FwdMessage:** parte iniziale del nome dei file in cui verranno creati i messaggi;
- **$FWHEADER:** contenuta nel modulo *GlobalCheck.pm,* contiene l'intestazione dei messaggi ad esclusione del campo 'To:';
- **$FWTXT:** contenuta nel modulo *GlobalCheck.pm,* contiene la sezione di testo standard che compone la prima parte di ogni messaggio;
- **$ACKSIG:** contenuta nel modulo *GlobalCheck.pm,* contiene la sezione di testo standard per la chiusura di ogni messaggio.

FUNZIONI UTILIZZATE

- **&WriteObject::WriteObjects():** utilizzata per la stampa dell'oggetto all'interno del messaggio.

DESCRIZIONE

La procedura scorre il vettore puntato da *$updPtr* e, per ogni indirizzo di posta elettronica in esso contenuto, esegue le seguenti operazioni:

- crea un file il cui nome viene costituito appendendo alla stringa *$GlobalCheck::FwdMessage* la stringa contenuta nel parametro *$id* seguita da un numero progressivo;
- stampa l'intestazione del messaggio, nel file creato, impostando il campo 'To:' con l'indirizzo di posta contenuto nel campo corrente del vettore ed, il resto degli attributi, valutando e stampando la variabile globale *$FWHEADER*;
- stampa l'oggetto contenuto nel parametro *%object* utilizzando la funzione *&WriteObject::WriteObjects()*;
- stampa la chiusura del messaggio stampando la variabile globale *$ACKSIG*;
- inserisce il nome del file contenente il messaggio creato nel vettore *@FileToSend*.

Al termine restituisce il puntatore al vettore *@FileToSend* che sarà utilizzato dal programma principale per l'invio dei messaggi.

5.6.2.12 CreateErrorMail()

Crea il corpo del messaggio di posta elettronica contenente un messaggio di errore che viene inviato allo staff tecnico del Registro nel caso in cui si verifichi un errore durante l'esecuzione del sistema tale da causare l'interruzione del processo per il modulo correntemente esaminato.

Il corpo del messaggio è composto da una sezione iniziale standard contenente la causa dell'errore e il nome della procedura tecnica all'interno della quale si è verificato l'errore stesso e una seconda sezione contenente il modulo correntemente esaminato dal sistema sintattico.

IN

- **$errortype:** stringa descrittiva contenente il tipo di errore accorso ed il nome della procedura nel quale si è verificato l'errore;

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::RAErrorMail:** nome del file in cui viene creato il messaggio;
- **$NICTEXT:** contenuta nel modulo *GlobalCheck.pm,* contiene il testo standard che compone la prima parte del messaggio; da notare che il testo è parametrico rispetto ad un variabile $TYPE nella quale deve essere dettagliato il messaggio;
- **@mail:** vettore contenente il modulo correntemente esaminato dal sistema.

DESCRIZIONE

La procedura crea il corpo del messaggio nel file *$GlobalCheck::RAErrorMail*. La parte iniziale del messaggio è standard ed è creata assegnando il valore del parametro in ingresso *$errortype* al parametro *$TYPE* e stampando la variabile globale *$NICTEXT*. Successivamente, viene creata la seconda parte del messaggio stampando il vettore *@mail*.

5.6.2.13 CreateUsrErrorMail()

La procedura crea un messaggio di posta elettronica contenente la segnalazione di un errore che il sistema invia al mittente del modulo correntemente esaminato nel caso in cui sia verificato un errore nell'accesso ad uno dei database del Registro.

Il corpo del messaggio è formato da tre sezioni di testo standard, una di apertura, una contenente il messaggio e una di chiusura.

IN

- **nessuno**

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::UsrErrorMail:** nome del file in cui viene creato il messaggio;
- **$FROM:** indirizzo di posta elettronica del mittente del modulo correntemente esaminato;
- **$USRERRORHEADER:** dichiarata nel file *GlobalCheck.pm* e definita nel file *Message*, contiene l'intestazione del messaggio ad esclusione del campo 'To:';
- **$USRERRORTEXT** dichiarata nel file *GlobalCheck.pm* e definita nel file *Message*, contiene il testo del messaggio;
- **$ACKSIG** dichiarata nel file *GlobalCheck.pm* e definita nel file *Message*, contiene la chiusura del messaggio.

DESCRIZIONE

La procedura crea il messaggio nel file *$GlobalCheck::UsrErrorMail*. L'intestazione del messaggio è creata impostando il campo 'To:' con il valore contenuto nella variabile globale *$FROM* e valutando e stampando la variabile *$USRERRORHEADER*.

Il corpo e la chiusura del messaggio sono creati stampando rispettivamente le variabili globali *$USRERRORTEXT* e *$ACKSIG*.

5.6.2.14 SendMail()

Invia il messaggio di posta elettronica contenuto nel file passato come parametro utilizzando il comando Unix *sendmail*.

IN

- **$mail:** Nome del file contenente il messaggio di posta elettronica da inviare;

OUT

- **codice di errore** - ERR_SENDMAIL se si è verificato un errore nell'esecuzione del comando sendmail;
  - ERR_OPENFILE se si è verificato un errore nell'apertura del file da inviare;
  - ERR_NONE se non si sono verificati errori;

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::sendmail:** comando *sendmail* Unix utilizzato per inviare il messaggio.

DESCRIZIONE

La funzione invia via posta elettronica il file denominato dal parametro in ingresso *$mail* utilizzando il comando di sistema *sendmail* contenuto nella variabile *$GlobalCheck::sendmail*. L'invio del file è monitorato al fine di restituire in uscita un codice di errore che rilevi l'esito dell'invio del file.

5.6.2.15 PrintAckHeader()

Questa procedura stampa l'intestazione del messaggio di posta elettronica contenente la risposta dei controlli effettuati dal sistema.

IN

- **$GlobalResult:** indica se i controlli effettuati dal sistema sul messaggio correntemente esaminato sono andati a buon fine (1) o meno (0);

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::TEST_TOPM** indirizzo di posta elettronica con cui viene impostato il campo 'To:' dell'intestazione nel caso in cui il sistema è eseguito in modalità TEST;
- **$FROM** indirizzo di posta elettronica del mittente del messaggio esaminato;
- **$MHEADER** dichiarata nel file *GlobalCheck.pm* e definita nel file *Message*, contiene l'intestazione del messaggio ad eccezione del campo 'To:'; il testo è parametrico rispetto alla variabile *$CHECKRESULT* che indica nel campo 'Subject' il risultato complessivo dei controlli.
- **$GlobalCheck::TESTMODE** indica se il sistema è eseguito in modalità TEST (1) o meno (0);

DESCRIZIONE

La procedura stampa il campo 'To:' dell'intestazione impostando il valore con il contenuto delle variabili *$GlobalCheck::TEST_TOPM* o *$FROM* a seconda del valore impostato nella variabile *$GlobalCheck::TESTMODE*.

Prima completare la stampa dell'intestazione, viene assegnato ad un parametro *$CHECKRESULT* il valore SUCCEDED o FAILED dipendentemente dal valore del parametro in ingresso (1 o 0 rispettivamente) ed infine viene valutata e stampata la variabile globale *$MHEADER*.

5.6.2.16 PrintAckBoby()

Costruisce il corpo del messaggio contenente la risposta del controllo sintattico sul modulo inviato dal P/M.

Il messaggio è formato da un'introduzione ed una chiusura standard e contiene il risultato dei controlli effettuati per ogni oggetto contenuto nel modulo elettronico esaminato oppure i messaggi di errore eventualmente rilevati sul formato del modulo o del messaggio come descritto nella *sezione 4.3* di questo documento.

IN

- **nessuno**

OUT

- **$IsOK:** indica se i controlli effettuati sono andati tutti a buon fine (1) oppure se si è verificato almeno un errore sui dati (0);

VARIABILI GLOBALI UTILIZZATE

- **$MAILTXT:** dichiarata nel file *GlobalCheck.pm* e definita nel file *Message*, contiene il testo standard con cui inizia il messaggio di risposta;
- **$ACKERR:** dichiarata nel file *GlobalCheck.pm* e definita nel file *Message*, contiene il testo standard con cui viene chiuso il messaggio nel caso in cui si sia presentato almeno un warning o un errore;
- **$ACKOK:** dichiarata nel file *GlobalCheck.pm* e definita nel file *Message*, contiene il testo standard con cui viene chiuso il messaggio nel caso in cui tutti i controlli abbiano dato esito positivo;
- **$ACKSIG:** dichiarata nel file *GlobalCheck.pm* e definita nel file *Message*, contiene la firma standard del messaggio;
- **%SyntaxError::ERROR_MESSAGES:** contiene la definizione di tutti i possibili messaggi di errore e di warning nel formato *(codice_errore, messaggio)*;

- **$nummntner:** numero di oggetti *mntner* contenuti nel modulo;
- **$numdomain:** numero di oggetti *domain* contenuti nel modulo;
- **%DOMAIN:** contiene gli attributi dell'oggetto *domain*, utilizzato come parametro della funzione *&PrintSyntaxError()*.
- **%MNTNER:** contiene gli attributi dell'oggetto *mntner*, utilizzato come parametro della funzione *&PrintSyntaxError()*.
- **@PERSONS:** contiene i puntatori agli hash che contengono gli attributi degli oggetti *person* presenti nel modulo, utilizzati come parametro della funzione *&PrintSyntaxError()*;
- **@ROLES:** contiene i puntatori agli hash che contengono gli attributi degli oggetti *role* presenti nel modulo, utilizzati come parametro della funzione *&PrintSyntaxError()*;
- **@PERSONNAME:** contiene i nomi delle persone associate agli oggetti *person*;
- **%DOMAINSYNTAXERROR:** contiene gli errori sintattici dell'oggetto *domain*, utilizzati come parametro della funzione *&PrintSyntaxError()*;
- **%MNTNERSYNTAXERROR:** contiene gli errori sintattici dell'oggetto *mntner*, utilizzati come parametro della funzione *&PrintSyntaxError()*;
- **@PERSONSYNTAXERROR:** contiene i puntatori agli errori sintattici degli oggetti *person*, utilizzati come parametro della funzione *&PrintSyntaxError()*;
- **@ROLESSYNTAXERROR:** contiene i puntatori agli errori sintattici degli oggetti *role*, utilizzati come parametro della

funzione *&PrintSyntaxError()*;

- **$DOMAINAUTHERROR:** contiene l'errore di autenticazione rilevato sul *domain*, utilizzato come parametro della funzione *&PrintSyntaxError()*;
- **$MNTNERAUTHERROR:** contiene l'errore di autenticazione rilevato sul *mntner*, utilizzato come parametro della funzione *&PrintSyntaxError()*;
- **@PERSONAUTHERROR:** contiene gli errori di autenticazione rilevati sulle persone, utilizzati come parametro della funzione *&PrintSyntaxError()*;
- **@ROLEAUTHERROR:** contiene gli errori di autenticazione rilevati sui *role*, utilizzati come parametro della funzione *&PrintSyntaxError()*;
- **$PINERROR:** contiene l'errore di consistenza sull'attributo *pin*;
- **@MailError:** contiene i codici di errore eventualmente rilevati sul formato del messaggio o del modulo esaminato;
- **$numDomainSyntaxError:** numero di errori sintattici rilevati sull'oggetto *domain*, utilizzato come parametro della funzione *&PrintSyntaxError()* nel caso in cui il modulo contenga un oggetto *domain*;
- **$numMntnerSyntaxError:** numero di errori sintattici rilevati sull'oggetto *mntner*, utilizzati come parametro della funzione *&PrintSyntaxError()* nel caso in cui il modulo contenga un oggetto *mntner*;
- **@numPersonSyntaxError:** numero di errori sintattici rilevati su ogni oggetto *person* presente nel modulo, nell'ordine di presenza; utilizzato come parametro della funzione *&PrintSyntaxError()*;

- **@numRoleSyntaxError:** numero di errori sintattici rilevati su ogni oggetto *role* presente nel modulo, nell'ordine di presenza; utilizzato come parametro della funzione *&PrintSyntaxError()*;
- **$numDomainSyntaxWarning:** numero di warning rilevati sull'oggetto *domain*, utilizzato come parametro della funzione *&PrintSyntaxError()* nel caso in cui il modulo contenga un oggetto *domain*;
- **$numMntnerSyntaxWarning:** numero di warning rilevati sull'oggetto *mntner*, utilizzato come parametro della funzione *&PrintSyntaxError()* nel caso in cui il modulo contenga un oggetto *mntner*;
- **@numPersonSyntaxWarning:** numero di warning rilevati su ogni oggetto *person* presente nel modulo, nell'ordine di presenza; utilizzati come parametro della funzione *&PrintSyntaxError()*;
- **$numDomainConsistentError:** numero di errori di consistenza rilevati sull'oggetto *domain*;
- **$numMailError:** numero di errori rilevati inerenti alla tipologia del modulo o dell'intero messaggio analizzato.

FUNZIONI UTILIZZATE

- **&PrintSyntaxError():** utilizzata per la stampa dei messaggi di errore rilevati su un singolo oggetto contenuto nel modulo analizzato;

DESCRIZIONE

La funzione crea la prima sezione standard del messaggio valutando e stampando la variabile globale *$MAILTXT*.

Successivamente, controlla, tramite la variabile *$numMailError*, se si sono verificati errori sul formato del modulo o del messaggio ed in questo caso:

- stampa i messaggi di errore contenuti nell'hash *%SyntaxError::ERROR_MESSAGES* per ogni codice contenuto nel vettore *@MailError*;
- stampa la firma del messaggio contenuta nella variabile *$ACKSIG* e

- esce restituendo in uscita il valore ‘1’, senza prendere il considerazione gli errori rilevati sugli oggetti.

Se non si sono verificati errori di formato, la funzione controlla tramite le variabili *$nummntner* e *$numdomain* se devono essere stampati gli errori relativi ad un oggetto *domain* o *mntner* ed eventualmente li stampa chiamando la funzione *&PrintSyntaxError()* con i relativi parametri. Inoltre, prende in esame tutti gli oggetti *person e role* scorrendo i vettori *@PERSONS e @ROLES* e, per ogni oggetto, stampa il risultato dei controlli utilizzando la funzione *&PrintSyntaxError()* con i parametri relativi contatti.

Infine, stampa la chiusura del messaggio contenuta nella variabile globale *$ACKOK* o *$ACKERR* a seconda che i controlli siano andati tutti a buon fine o meno e la firma contenuta in *$ACKSIG*.

5.6.2.17 PrintSyntaxError()

La funzione stampa il risultato dei controlli effettuati su un singolo oggetto, all'interno del messaggio di risposta inviato al mittente del modulo analizzato.

Viene stampata una frase standard che riassume il risultato globale sull'oggetto e, nel caso si sia verificato un errore o un warning, viene stampato l'oggetto seguito dai messaggi di errore e/o di warning rilevati.

IN

- **$objecttype:** tipo di oggetto correntemente analizzato (*domain*, *mntner*, o *person*);
- **$objectname:** nome dell'oggetto correntemente trattato (valore degli attributi: *domain*, *mntner* o *person* a seconda della classe dell'oggetto);
- **$numerror:** numero di errori sintattici rilevati sull'oggetto;
- **$numwarning:** numero di warning sintattici rilevati sull'oggetto;
- **$AuthError:** errore di autenticazione rilevato sull'oggetto;
- **$SYNTAXERROR_P:** puntatore ad un vettore hash contenente gli errori sintattici rilevati nel formato (*etichetta, codice_errore*);
- **$OBJECT_P:** puntatore ad un vettore hash contenente gli attributi dell'oggetto trattato nel formato (*etichetta, valore*).

OUT

- **$IsError:** '1' se ha stampato almeno un errore o un warning, '0' altrimenti;

VARIABILI GLOBALI UTILIZZATE

- **$CHECKFAILED:** messaggio standard stampato per ogni classe di oggetto nel caso in cui almeno un controllo effettuato su di esso sia fallito; la variabile è parametrica nelle variabili *$OBJECT* e *$NAME* che devono contenere il tipo di oggetto ed il nome rispettivamente;
- **$CHECKOK:** dichiarato nel file *Globalcheck.pm* e definito nel file *Message*, contiene il messaggio

standard stampato per ogni classe di oggetto nel caso in cui tutti i controlli effettuati su di esso siano andati a buon fine oppure si siano verificati dei warning; la variabile è parametrica nelle variabili *$OBJECT* e *$NAME* che devono contenere il tipo di oggetto ed il nome rispettivamente;

- **%SyntaxError::ERROR_MESSAGES:** contiene la definizione di tutti i messaggi di errore e di warning nel formato *(codice_errore, messaggio);*
- **$numNichdlError:** numero di errori di consistenza rilevati sui *nic-handle* se l'oggetto corrente è un *domain* o un *mntner*;
- **$PersonNicdlError:** indica se si è verificato un errore di registrazione sul *nic-handle* associato all'oggetto se è di tipo *person*;
- **$PINERROR:** indica se si è presentato un errore di consistenza sull'attributo *pin* se l'oggetto è di tipo *domain*;
- **%DOMAINCONSISTENTERROR:** contiene tutti gli errori di consistenza degli attributi contenuti nell'oggetto *domain* nel formato *(etichetta, codice_errore)*;

FUNZIONI UTILIZZATE

- **&WriteObject::WriteObjects():** utilizzata per la stampa formattata dell'oggetto corrente;

DESCRIZIONE

La funzione associa ai parametri *$OBJECT* e *$NAME*, contenuti nel messaggio standard (vedi *sezione 5.1*), il tipo e il nome dell'oggetto leggendoli rispettivamente dai parametri di ingresso *$objecttype* e *$objectname*.

Successivamente, valuta le variabili *$numerror*, *$AuthError*, *$PersonNicdlError*, *$numNichdlError* e *$PINERROR* e, nel caso in cui non si siano rilevati errori sintattici, di autenticazione, di

consistenza sui *nic-handle* e sul codice fiscale**,** stampa il messaggio standard contenuto nella variabile *$CHECKOK*, altrimenti stampa *$CHECKFAILED* che segnala il fallimento dei controlli sull'oggetto.

Nel caso in cui si sia presentato almeno un errore o un warning, viene stampato l'oggetto utilizzando la funzione *&WriteObject::WriteObjects($objecttype, '0', %OBJECT)* e a seguire vengono stampati tutti i messaggi di errore e di warning nel seguente ordine:

- se l'oggetto è di tipo *domain* e si è verificato un errore sul codice fiscale *($PINERROR* ≠ ERR_NONE) la funzione stampa il messaggio corrispondente nel formato:

  ERROR: *$SyntaxError::ERROR_MESSAGES{$PINERROR}*

  e ritorna 1 senza prendere in considerazione gli altri errori;

- se l'oggetto è di tipo *person* o *role* e la persona non è referenziata, oppure è associata ad un *nic-handle* già assegnato ad un'altra persona ($AuthError uguale a 'ERR_UNREFERENCED_OBJECT' oppure 'ERR_ASSIGNED_PERSON'), la funzione stampa il messaggio di errore corrispondente nel formato:

  ERROR: *$SyntaxError::ERROR_MESSAGES{$AuthError}*

  e ritorna 1 senza prendere in considerazione gli altri errori;

- stampa tutti i messaggi degli errori sintattici rilevati sugli attributi analizzando ogni coppia *(etichetta, codice_errore)* contenuta nell'hash puntato dal parametro in ingresso *$SYNTAXERROR_P* e stampando i messaggi corrispondenti nel formato:

  ERROR syntax error in *'etichetta'* value: *$SyntaxError::ERROR_MESSAGES{'codice_errore' }*

oppure

  WARNING in *'etichetta'* value: *$SyntaxError::ERROR_MESSAGES{'codice_errore' }*

- stampa i messaggi di errore inerenti la consistenza dei *nic-handle* valutando e stampando il *messaggio* di errore associato al codice ERR_NICHDL_NOT_FOUND per ogni coppia (*etichetta*, *nic-handle*) contenuta nella variabile globale *%NICHDLERROR* oppure per il *nic-handle* contenuto nella variabile *$PersonNichdlError* o *$RoleNichdlError*, se l'oggetto è di tipo *person* o *role*; i messaggi sono stampati nel formato:

  ERROR syntax error in *'etichetta'* value:*' messaggio'*

Da notare che il messaggio contenuto nella globale $SyntaxError::ERROR_MESSAGES{‘ ERR_NICHDL_NOT_FOUND’} è parametrico rispetto al codice del *nic-handle* (vedi *sezione 5.3*).

- se l’oggetto è di tipo *domain*, stampa i messaggi corrispondenti agli errori di consistenza contenuti nell’hash *%DOMAINCONSISTENTERROR*; questi messaggi vengono gestiti diversamente a seconda del tipo di errore e nell’attributo di riferimento *(etichetta, codice_errore)*:
  - se il codice di errore corrisponde al warning sull’attributo *x400-domain* (*codice_errore* = WARN_LOOK_X400DOMAIN), questo non viene segnalato all’utente ma viene assegnato il codice alla variabile globale *$X400WARNING* che verrà analizzata dalla procedura &xxxxx() la quale eventualmente segnalerà il warning agli operatori sul modulo elettronico inviato per la registrazione;
  - se l’errore è inerente il campo *x400-domain* (*etichetta* ≠ ‘’), il messaggio viene stampato nel formato:

    ERROR syntax error in '*x400-domain*' value: ‘*messaggio’*

    dove ‘*messaggio’* è parametrico rispetto al nome del sottocampo (*etichetta*) in cui si è verificato l’errore e corrisponde alla stringa *$SyntaxError::ERROR_MESSAGES{‘ codice_errore’}*;
  - per tutti gli altri tipi di errore i messaggi vengono stampati nel formato

    ERROR: *$SyntaxError::ERROR_MESSAGES{‘codice_errore’}*
- stampa l’errore di autenticazione nel formato:

  ERROR: *$SyntaxError::ERROR_MESSAGES{$AuthError}*

Infine ritorna ‘1’ (uno) se ha stampato almeno un messaggio di errore o di warning, altrimenti ritorna ‘0’ (zero).

5.7 Format.pm

Questo modulo contiene le funzioni utilizzate dal sistema di controllo sintattico per la formattazione di alcuni degli oggetti trattati. In particolare, contiene le seguenti funzioni, descritte nelle sezioni successive:

- **& FormatMail():** formatta il messaggio di posta elettronica contenente il modulo elettronico inviato dall'utente;
- **FormatX400domain():** formatta il campo *x400-domain* contenuto nell'oggetto *domain*.

5.7.1 FormatMail()

Questa funzione esegue la formattazione del messaggio di posta elettronica contenente il modulo elettronico inviato dall'utente.

In particolare, la funzione riconduce il corpo del messaggio alla sintassi del modulo elettronico standard suddividendo i vari oggetti in esso contenuti ed eliminando tutte le righe che non sono nel formato *'label:<spazi bianchi>valore'*. La funzione, per motivi di sicurezza, sostituisce il carattere "`" con il "'" in tutto il messaggio.

IN

- **@mail:** vettore contenente il messaggio di posta elettronica da formattare suddiviso per righe;

OUT

- **@newmail:** vettore contenente il messaggio di posta elettronica formattato suddiviso per righe;

DESCRIZIONE

La funzione prende in esame il messaggio, scorrendo il vettore *@mail,* passato come parametro in ingresso e per ogni riga analizzata esegue le seguenti operazioni:

- toglie gli spazi bianchi agli estremi della riga e sostituisce tutti i caratteri <tab> con spazi bianchi;
- sostituisce tutte le occorrenze del carattere "`" con il carattere "'";
- prende tutte le righe che compongono l'intestazione del messaggio ad eccezione delle righe contenenti i campi 'Cc:' o 'Bcc:' che riportano come valore l'indirizzo domain@nic.it;

- esclude:
    - le righe bianche;
    - le righe di commento;
    - le righe che iniziano con il carattere ‘:’ o con uno spazio;
    - le righe che non contengono il carattere ‘:’;
    - le righe che potrebbero descrivere un campo di un oggetto che ha un’etichetta ma non un valore, ovvero le righe nel formato “*<etichetta>:*<spazi>*” o “*<etichetta>:*<spazi>*”;
- suddivide gli oggetti del modulo inserendo una riga bianca prima di una riga che inizia con la parola ‘password’, oppure prima di una riga che inizia con una delle parole ‘domain’, ‘mntner’ o ‘person’, se la riga precedentemente esaminata non iniziava con la parola ‘password’;
- formatta le righe che potrebbero descrivere il campo di un oggetto:
    - inserisce uno spazio bianco dopo il carattere ‘:’,
    - attacca il carattere ‘:’ all’etichetta (parola precedente),
    - trasforma l’etichetta (parola a sinistra dei ‘:’) in caratteri minuscoli,
    - trasforma gli spazi multipli in spazi singoli.

Infine, aggiunge una riga bianca in fondo al modulo.

Ogni riga del messaggio in ingresso non scartata è inserita nel vettore *@newmail* che restituisce in uscita.

5.7.2 FormatX400domain

Formatta il valore inserito nel campo '*x400-domain*'di un oggetto *domain*.

IN

- **$x400domain:** stringa contenente il valore da formattare;

OUT

- **$x400domain:** stringa contenuta nel parametro in ingresso formattata.

DESCRIZIONE

La funzione legge il parametro in ingresso e cerca di ricondurlo al formato corrispondente al valore nell'attributo '*x400-domain*' eseguendo le seguenti trasformazioni:

- trasforma tutte le lettere in minuscolo;
- compatta i sotto-campi togliendo gli spazi prima e dopo ogni carattere '=' e prima del carattere ';';
- suddivide i sotto-campi inserendo uno spazio subito dopo il carattere ';' ;
- inserisce il carattere ';' in fondo alla stringa, se non inserito.

La stringa trasformata è restituita in uscita nel parametro *$x400domain*.

5.8 GetMailHeader.pl

Questo file contiene la funzione **&parserfc822()** che viene utilizzata dal programma principale al fine di ottenere il valore degli attributi contenuti nell'intestazione del messaggio di posta elettronica esaminato.

Segue la descrizione dettagliata della funzione.

5.8.1 &parserfc822()

Questa funzione esamina un file contenente un messaggio di posta elettronica scritto secondo le specifiche dell'RFC822 e fornisce in uscita il valore dei vari campi contenuti nell'intestazione.

In particolare, restituisce i valori dei campi *From*, *Subject*, *Message-Id*, *Reply-To*, Date e *Content-Type* e restituisce un errore nel caso in cui il messaggio non rispetti le specifiche previste dall'RFC822 o il messaggio sia stato firmato e non provenga dal sistema di verifica dei moduli firmati previsti dal Registro (message verify).

IN

- **@file:** vettore contenente il file da esaminare suddiviso per righe.

OUT

- **%result:** hash contenente il valore dei campi contenuti nell'intestazione nel formato *(etichetta, valore)*, dove le etichette corrispondono alla parola che contraddistingue il campo nell'intestazione (*From*, *Subject*, *Message-Id*, *Reply-To*, *Date, MIME-Version* e *Content-Type*);
- **$error:** codice di errore eventualmente rilevato:
  - ERR_FROM, se il campo 'From' e' vuoto,
  - ERR_CONTENT_TYPE, se il messaggio non rispetta le specifiche RFC822,
  - ERR_SIGNED_MAIL, se se il Content-Type non e' di tipo 'multipart/signed' ma la mail non proviene dal 'message verify';
  - ERR_NONE, altrimenti.

DESCRIZIONE

La funzione scorre tutte le righe del file contenute nel parametro in ingresso *@file* ed esamina l'intestazione del messaggio identificandolo con le prime righe di testo delimitate da una riga bianca.

Per ogni riga nel formato

'<*etichetta*>:<*spazi*><*valore*>'

con

<*etichetta*> = *From*, *Subject*, *Message-Id*, *Reply-To*, *Date*, *MIME-Version* o *Content-Type*,

registra il valore del campo nel parametro di uscita *%result*.

Una volta memorizzato il valore dei capi, questi sono esaminati dalla funzione che assegna il codice di errore appropriato al parametro di uscita *$error* e termina. In particolare sono eseguiti i seguenti controlli:

- *$error* = ERR_FROM, se il valore del campo *From* è una stringa vuota;
- $*error* = ERR_CONTENT_TYPE, se
  - il valore del campo *Content-Type* e di tipo *multipart/alternative*, oppure
  - esistono nell'intestazione il campo *MIME-Version* ed il campo *Content-Type* e quest'ultimo non è di tipo *text/plain*;
- $*error* = ERR_SIGNED_MAIL, se il campo *Content-Type* è di tipo *multipart/signed* e il campo *Sender* non indica il message verify;
- $error = ERR-NONE, altrimenti.

5.9 CheckObjects.pm

Questo modulo contiene la funzione **&CheckObject()** utilizzata dal programma principale per eseguire il controllo sintattico degli attributi contenuti in ogni oggetto che compone il modulo elettronico esaminato.

Nel file è incluso il modulo **CONF/SyntaxConf.pm** che contiene le specifiche di tutte le tipologie di oggetti ed il modulo **CheckFields.pm** dal quale preleva le funzioni atte al controllo sintattico del valore dei singoli attributi.

Segue la descrizione dettagliata della funzione.

5.9.1 &CheckObject()

Questa funzione effettua il controllo sintattico sul valore di tutti gli attributi contenuti in un oggetto, di qualsiasi tipologia, contenuto in un modulo elettronico. Inoltre, esegue i controlli sull'inserimento degli attributi obbligatori e degli attributi multipli.

Da notare che, per facilitare i controlli seguenti che sono effettuati dal sistema su alcuni attributi dell'oggetto, la funzione elimina il valore numerico in testa alle etichette corrispondenti.

IN

- **$type:** definisce la tipologia dell'oggetto da analizzare (*domain*, *mntner*, *role*, *person*);
- **$object:** puntatore ad un hash contenente gli attributi dell'oggetto da analizzare nel formato *(etichetta, valore)*;

  N.B. le etichette iniziano con un valore numerico inserito per consentire gli attributi multipli (vedi la funzione *&GetObjects*()).

OUT

- **$nerror:** numero di errori rilevati;
- **$nwarning:** numero di warning rilevati;
- **$IsMulError:** indica se si sono verificati errori di molteplicità;
- **%ERROR:** hash contenente tutti gli errori riscontrati durante l'analisi dell'oggetto nel formato *(etichetta, codice_errore)*.

  N.B. l'etichetta e' preceduta da un valore numerico per consentire la

segnalazione di errori multipli associati ad un singolo campo.

VARIABILI GLOBALI UTILIZZATE

- **$GlobalConf::etichetta:** variabili inizializzate dinamicamente dalla funzione e denominate in base alle etichette delle sezioni;

FUNZIONI UTILIZZATE

- **&SyntaxConf::GetEmptyOBJECT_FIELD():** utilizzata per registrare la struttura dell'oggetto esaminato;
- **&SyntaxConf::GetOBJECT_FIELD():** utilizzata per ottenere le caratteristiche di obbligatorietà e di molteplicità dell'oggetto da esaminare;
- **&SyntaxConf::GetOBJECT_FUNCTION():** utilizzata per ottenere le caratteristiche sintattiche degli attributi contenuti nell'oggetto.

DESCRIZIONE

La funzione legge la struttura di un oggetto di tipo *$type* e le caratteristiche degli attributi in esso contenuti chiamando le funzioni *&SyntaxConf::GetEmptyOBJECT_FIELD()*, *&SyntaxConf::GetOBJECT_FIELD()* e *&SyntaxConf::GetOBJECT_FUNCTION()* con parametro *$type*.

Successivamente, confronta le caratteristiche lette con quelle dell'oggetto passato nel parametro in ingresso *$object* eseguendo anche i controlli sintattici sui singoli attributi.

Per ogni elemento *(etichetta, valore)* del vettore hash *%$object:*

- toglie il valore numerico in testa alle etichette *domain*, *mntner*, *person*, *mnt-by*, *nic-hdl*, *pin*, e *x400-domain*;
- controlla l'appartenenza del campo nell'oggetto di tipo *$type* - cerca l'etichetta nell'oggetto *%OBJECT* restituito dalla funzione *&SyntaxConf::GetOBJECT_FIELD(),* se il campo non esiste, registra in *%ERRORS* l'errore ERR_UNKNOWN_ATTRIBUTE associato a quell'etichetta;
- marca la presenza dell'attributo nell'oggetto *%EMPTY_OBJECT* restituito da *&SyntaxConf::GetEmptyOBJECT_FIELD()* - setta il flag di presenza e incrementa il flag di molteplicità associati all'etichetta*;*

- esegue i controlli sintattici sul valore degli attributi:
    - legge il nome della funzione associata all'attributo dall'oggetto restituito dalla funzione *&SyntaxConf::GetOBJECT_FUNCTION(),*
    - esegue la funzione omonima definita nel modulo *CheckFields.pm* sul valore dell'attributo,
    - registra nel vettore *%ERRORS* i codici di errore e di warning restituiti dalla funzione associandoli all'etichetta;

Una volta analizzati tutti gli attributi dell'oggetto, la funzione controlla la presenza degli attributi obbligatori e la non molteplicità degli attributi univoci confrontando il vettore %OBJECT, che contiene la struttura teorica dell'oggetto, con il vettore *%EMPTY_OBJECT,* che contiene la struttura dell'oggetto analizzato - per ogni campo:

- se il campo non è multiplo ma il suo contatore di molteplicità è maggiore di '1', registra in *%ERRORS* l'errore ERR_MULTIPLE_FIELD associato all'attributo;
- se il campo è obbligatorio ma il suo contatore di molteplicità è uguale a '0', registra in *%ERRORS* l'errore ERR_MISSING_FIELD associato all'attributo.

Per ogni codice di errore e di warning registrato in *%ERRORS*, la funzione incrementa opportunamente i contatori *$nerror* e *$nwarning*.

5.10 CheckFields.pm

Questo modulo contiene la definizione di tutte le funzioni che eseguono i controlli sintattici previsti sui vari attributi presenti negli oggetti.

Per ogni tipologia di campo (*nic-handle* , numeri telefonici, nomi di maintainer, ecc), è definita una funzione che assicura la correttezza sintattica del valore dell'attributo contenuto in un oggetto.

Le associazioni tra i singoli attributi e le funzioni che realizzano i controlli sintattici sono definite nel modulo *GlobalCheck.pm* dalle strutture dati *%DOMAIN_FUNCTION*, *%MNTNER_FUNCTION, %ROLE_FUNCTION* e *%PERSON_FUNCTION* a seconda della tipologia di oggetto.

Di seguito sono elencate le funzioni che effettuano i controlli sul valore degli attributi:

- **&DomainOK():** campo *domain* dell'oggetto *domain*;
- **&MntnerOK():** attributi contenenti il nome di un maintainer (*mntner* dell'oggetto *mntner* e *mnt-by* di tutti gli oggetti);
- **&PersonOK():** campo *person* dell'oggetto *person*;
- **&RoleOK():** campo *role* dell'oggetto *role*;
- **&SourceOK():** campo *source* di tutti gli oggetti;
- **&NicHandleOK():** attributi contenenti il codice di un *nic-handle* (*admin-c* e *tech-c* degli oggetti *domain* e *mntner*, *zone-c*, *gate-c*, e *postmaster* dell'oggetto *domain*, *nic-hdl* dell'oggetto *person*);
- **&ChangedOK():** attributi *changed* di tutti gli oggetti;
- **&EmailOK():** attributi *notify* di tutti gli oggetti, *upd-to* e *mnt-nfy* dell'oggetto *mntner* e *e-mail* dell'oggetto *person*;
- **&X400DomainOK():** campo *x400-domain* dell'oggetto *domain*;
- **&PinOK():** campo *pin* dell'oggetto *domain*;
- **&NserverOK():** attributi *nserver* e *mailgate* dell'oggetto *domain*;
- **&DomNetOK():** campo *dom-net* dell'oggetto *domain*;
- **&AuthOK():** campo *auth* dell'oggetto *mntner*;
- **&TelNumberOK():** attributi *fax-no* e *phone* dell'oggetto *person*.

Inoltre, sono definite le seguenti funzioni utilizzate nell'implementazione delle funzioni sopra elencate:

- **&DNSOK():** controlla che un stringa contenga un nome valido per un DNS;
- **&IPaddrOK():** controlla che una stringa sia un indirizzo IP valido;
- **&NETaddrOK():** controlla che un indirizzo IP sia un indirizzo di rete valido;
- **&DateOK():** controlla che una stringa contenga una data nel formato YYYYMMDD.

Il file include il modulo predefinito Perl *Time::ParseDate* che viene utilizzato dalla funzione *&DateOK()* per effettuare operazioni sulle date.

Le sezioni seguenti descrivono in dettaglio le funzioni definite in questo modulo.

5.10.1 DomainOK()

Questa funzione controlla che la stringa passata come parametro in ingresso abbia una sintassi conforme ad un nome a dominio:

$str_1.str_2.str_N.it$

con:

- 'str_i' stringa composta da caratteri alfabetici e numerici con l'aggiunta del segno '-' (meno);
- $N \geq 1$ (dominio almeno di secondo livello);
- $2 \leq$ lunghezza(str_i) ≤ 63;
- $2 \leq$ lunghezza($str_1.str_2.str_N.it$) ≤ 255;
- 'str_i' non contenga il carattere '-' (meno) all'inizio e in coda.

La funzione viene utilizzata per il controllo del valore inserito nel campo *domain* dell'oggetto *domain*.

IN

- **$domain:** stringa contenente il nome a dominio da controllare;

OUT

- **@return:** vettore contenente i codici di errore eventualmente rilevati:
  - ERR_NONE, se la sintassi del parametro *$domain* e' corretta;
  - ERR_EMPTY, se il parametro in ingresso *$domain* e' vuoto;
  - ERR_SYNTAX_DOMNAME, se la sintassi della stringa *$domain* non è conforme ad un nome a dominio;
  - WARN_SYNTAX_HYPHEN_DOMNAME, se la stringa *$domain* contiene due caratteri '-' consecutivi (es. aaaa.b--b.cccc.it);
  - WARN_SYNTAX_LOWERCASE_DOMNAME: se la stringa *$domain* contiene almeno un carattere maiuscolo.

DESCRIZIONE

La funzione effettua i controlli sintattici sopra elencati sulla stringa contenuta nel parametro in ingresso *$domain* e restituisce in *@result* un solo codice di errore con eventualmente i codici corrispondenti ai warning rilevati.

5.10.2 MntnerOK()

Questa funzione controlla che la stringa passata come parametro in ingresso abbia una sintassi conforme al nome di un P/M:

str_1-str_2

con:

- 'str_1' stringa composta da caratteri alfanumerici compreso il carattere '-' (meno);
- 'str_2' uguale a 'MNT' o 'ENT'.

Inoltre controlla che all'interno dell'intero nome non ci siano due caratteri '-' consecutivi.

IN

- **$maintainer:** stringa da controllare;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:
  - ERR_NONE, se la sintassi del parametro in ingresso è corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_ILLEGAL_MNT, se la stringa contenuta nel parametro $maintainer contiene due caratteri '-' consecutivi;
  - ERR_SYNTAX_MNT, se la sintassi della stringa contenuta nel parametro $maintainer non è conforme al nome di un P/M.

DESCRIZIONE

La funzione esegue i controlli sintattici sopra citati sulla stringa contenuta nel parametro in ingresso *$maintainer* facendo uso dei 'pattern matching' del Perl, inserisce gli eventuali codici di errore rilevati all'interno del vettore *@result* che infine, restituisce in uscita.

5.10.3 PersonOK()

Questa funzione controlla che la stringa passata come parametro in ingresso abbia una sintassi conforme al nome di una persona. In particolare controlla che:

- la stringa sia composta da almeno due sottostringhe;
- la stringa non contenga titoli di persona come prof., dr., dott., mr., mrs., sig., sr., ing., rag.;
- le sottostringhe siano composte da soli caratteri alfanumerici, compresi tutti i tipi di apici, il carattere '-' (meno) e '_' (sottolineato).

La funzione è eseguita per il controllo sintattico dell'attributo *person* degli oggetti *person.* Poiché attualmente l'oggetto *person* è ancora impropriamente utilizzato, al posto dell'oggetto *role,* anche per identificare ditte, società o gruppi di persone che ricoprono ruoli specifici per un nome a dominio o un maintainer, è stato necessario accettare anche i caratteri '-' e '_' e numerici che solitamente non fanno parte di un nome di persona.

IN

- **$person:** stringa contenente il nome da controllare;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:
  - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_PERS, se il parametro in ingresso contiene dei caratteri non ammessi;
  - ERR_SYNTAX_NAME, se la stringa contenuta nel parametro in ingresso non è composta da almeno due sottostringhe;
  - ERR_ILLEGAL_NAME, se la stringa contenuta nel parametro in ingresso contiene una sottostringa corrispondente ad un titolo personale.

DESCRIZIONE

La funzione esegue i controlli sintattici sopra citati sulla stringa contenuta nel parametro in ingresso *$person* facendo uso dei 'pattern matching' del Perl, inserisce gli eventuali codici di errore rilevati all'interno del vettore *@result* che infine, restituisce in uscita.

5.10.4 RoleOK()

Questa funzione controlla che la stringa passata come parametro in ingresso abbia una sintassi conforme al nome di una società o di un identificativo di un ruolo. Attualmente, fino a quando l'utilizzo dei role object non avrà eliminato dal database WHOIS oggetti person utilizzati impropriamente al posto dei role, il codice della funzione *RoleOK()* coincide con quello della funzione *PersonOK()* (Vedi *sezione 5.10.3*).

5.10.5 SourceOK()

Questa funzione controlla che la stringa contenuta nel parametro in ingresso sia uguale a 'IT-NIC'. La funzione è eseguita per il controllo sintattico dell'attributo *source* contenuto negli oggetti di ogni tipologia.

IN

- **$string:** stringa da controllare;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:
  - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_SOURCE, se la stringa contenuta nel parametro $string non contiene il valore 'IT-NIC'.

DESCRIZIONE

La funzione controlla che la stringa contenuta nel parametro in ingresso *$string* contenga il valore 'IT-NIC' ed inserisce il codice di errore rilevato nel vettore *@result* che restituisce in uscita.

5.10.6 NicHandleOK()

Questa funzione controlla che la stringa passata come parametro in ingresso abbia una sintassi conforme al codice di un *nic-handle* rilasciato dal Registro italiano:

str-ITNIC

con *'str'* composta da due sottostrighe consecutive str_1 e str_2 tali che

- 'str_1' composta di soli caratteri alfabetici e $2 \leq$ lunghezza('str_1') ≤ 4,
- 'str_2' composta da soli caratteri numerici con lunghezza('str_2') ≥ 1 e valore('str_2') < 9999999;

IN

- **$str:** stringa su cui devono essere effettuati i controlli;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:
  - ERR_NONE, se la sintassi della stringa contenuta nel parametro di in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_NICHDL, se la sintassi della stringa contenuta nel parametro in ingresso è errata.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$str* utilizzando i 'pattern matching' del Perl ed inserisce i codici di errore eventualmente rilevati nel vettore *@result* che restituisce in uscita.

5.10.7 ChangedOK()

Questa funzione controlla che la stringa passata come parametro abbia una sintassi del tipo:

‘*indirizzo-email’ ‘data’*

con:

- ‘*indirizzo-email’* stringa contenente un indirizzo di posta elettronica valido;
- *‘data’* stringa contenente una data non superiore alla data odierna, nel formato yyyymmdd.

La funzione è utilizzata per il controllo sintattico degli attributi *changed* contenuti in ogni tipologia di oggetto.

IN

- **$changed:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@return:** vettore contenente i codici di errore eventualmente rilevati:
  - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_IN_CHANGED, se il parametro in ingresso contiene più di due stringhe;
  - ERR_SYNTAX_EMAIL_IN_CHANGED, la stringa ‘*indirizzo-email’* non contiene un indirizzo di posta elettronica conforme;
  - ERR_SYNTAX_DATE_IN_CHANGED, la stringa ‘*data’* non è conforme ad una data nel formato yyyymmdd;
  - ERR_SYNTAX_NOEMAIL_IN_CHANGED, il parametro in ingresso non contiene una stringa del tipo ‘*indirizzo-email’*;
  - ERR_SYNTAX_NODATE_IN_CHANGED, il parametro in ingresso non contiene una stringa del tipo ‘*data’*;
  - ERR_SYNTAX_SWAP_EMAIL_DATE_IN_CHANGED, il parametro in ingresso contiene le stringhe ‘*indirizzo-email’* e ‘*data’* ma inserite nell’ordine inverso;
  - ERR_SYNTAX_OUT_OF_DATE_IN_CHANGED, la stringa ‘*data’*

contiene una data successiva alla data odierna.

FUNZIONI UTILIZZATE

- **&EmailOK():** utilizzata per il controllo sintattico della stringa '*indirizzo-email*';
- **&DateOK():** utilizzata per il controllo sintattico della stringa '*data*'.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$changed* utilizzando le funzioni *&EmailOK()* e *&DateOK()* (vedi sezioni 5.10.8 e 5.10.18) per i controlli sintattici sull'indirizzo di posta elettronica e sulla data. Ogni errore rilevato è inserito nel vettore *@result* che, infine, viene restituito in uscita.

5.10.8 EmailOK()

Questa funzione controlla che la stringa passata come parametro abbia una sintassi conforme ad un indirizzo di posta elettronica:

user_name@domain_name

con:

- '*user_name',* stringa non vuota composta da caratteri alfanumerici con l'aggiunta dei caratteri '-' (meno), '.' (punto), '_' (sottolineato), '/' (barra), '=' (uguale), '%' (percentuale), e '&' (e commerciale);
- '*domain_name',* stringa non vuota contenente un nome a dominio secondo la notazione internet RFC822 e la sintassi RFC1034, RFC1035, RFC1912 ovvero, stringa composta da caratteri alfanumerici con l'aggiunta dei caratteri '.' (punto) e '-' (meno).

La funzione viene utilizzata sia per il controllo degli attributi *e-mail* contenuti nell'oggetto *person*, che dalla funzione *ChangedOK()* per il controllo sintattico dell'attributo *changed*.

IN

- **$email:** stringa su cui vengono effettuati i controlli sintattici;

OUT

- **$errore:** codice di errore:

  ERR_NONE, se la sintassi del parametro in ingresso e' corretta;

  ERR_EMPTY, se il parametro in ingresso e' vuoto;

  ERR_SYNTAX_MAIL, se la sintassi del parametro in ingresso non contiene un indirizzo di posta elettronica valido.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$email* utilizzando i 'pattern matching' del Perl ed inserisce i codici di errore eventualmente rilevati nel vettore *@result* che restituisce in uscita.

5.10.9 X400DomainOK()

Questa funzione controlla che la stringa passata come parametro in ingresso abbia la sintassi di nome a dominio secondo la notazione ISO X400:

c=it; admd=str$_1$; prmd=str$_2$; o=str$_3$; ou=str$_4$;

con:

- '*str$_i$*', stringa non vuota composta da soli caratteri alfanumerici con l'aggiunta del carattere '-' (meno);
- '*str$_1$*', uguale a '0' (zero), 'garr' o ' ' (composta da spazi).

IN

- **$x400domain:** stringa sulla quale sono effettuati i controlli;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:
  - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_C, se il sottocampo '*c*' non ha come valore "it";
  - ERR_SYNTAX_ADMD, se il sottocampo '*admd*' ha come valore una stringa diversa da 0 (zero), 'garr' o formata da soli spazi bianchi;
  - ERR_SYNTAX_PRMD_LONG, se la lunghezza(*prmd*) > 16 caratteri;
  - ERR_SYNTAX_PRMD, se il sottocampo *prmd* non è stato inserito o contiene caratteri non consentiti;
  - ERR_SYNTAX_O, se il sottocampo *o* non è stato inserito o contiene caratteri non consentiti;
  - ERR_SYNTAX_Ou, se il sottocampo *ou* non è stato inserito o contiene caratteri non consentiti;
  - ERR_SYNTAX_X4001, se esiste un sottocampo non compilato numero(';') > numero('=');
  - WARN_LOWERCASE_X400, se la stringa contenuta nel parametro *$x400domain* contiene almeno un carattere minuscolo;

- WARN_SEMICOLON_X400, se è stato omesso l'ultimo carattere ';'.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$x400domain* utilizzando i 'pattern matching' del Perl ed inserisce i codici di errore eventualmente rilevati nel vettore *@result* che restituisce in uscita. Da notare che ogni controllo sulle stringhe viene effettuato indipendentemente dal formato, maiuscolo o minuscolo, delle lettere che lo compongono.

5.10.10 PinOK()

Questa funzione esegue alcuni controlli sintattici sul valore dell'attributo *pin* inserito in un oggetto *domain*.

In particolare, controlla che la stringa passata nel parametro in ingresso contenga solo caratteri alfanumerici con l'aggiunta del carattere '-' (meno) ma non inizi e finisca con quest'ultimo carattere.

Poiché un dominio può essere assegnato a qualsiasi persona fisica con cittadinanza o residenza in uno dei paesi della Comunità Economica europea, non è stato possibile implementare in questa funzione, controlli sintattici più mirati e stringenti.

IN

- **$pin:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@result:** vettore contenente gli eventuali codici di errore rilevati:
  - ERR_NONE, se i controlli effettuati vanno tutti a buon fine;
  - ERR_SYNTAX_PIN, altrimenti;

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$pin* utilizzando i 'pattern matching' del Perl ed inserisce i codici di errore eventualmente rilevati nel vettore *@result* che restituisce in uscita.

5.10.11 NserverOK()

Questa funzione controlla che la sintassi della stringa passata come parametro in ingresso sia conforme ad un indirizzo di nameserver:

indirizzo_IP domain_name

con:

- '*indirizzo_IP*', stringa contenente un indirizzo IP valido;
- '*domain_name*', stringa contenente un nome a dominio valido.

La funzione è utilizzata per il controllo sintattico degli attributi *nserver* contenuti in un oggetto *domain*.

IN

- **$nserver:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:
    - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
    - ERR_EMPTY, se il parametro in ingresso e' vuoto;
    - ERR_SYNTAX_IN_NSERVER, se il parametro in ingresso è formato da più di due stringhe;
    - ERR_SYNTAX_IP_IN_NSERVER, sintassi della stringa '*indirizzo_IP*' non conforme ad un indirizzo IP;
    - ERR_SYNTAX_FQDN_IN_NSERVER, sintassi della stringa '*domain_name*' non conforme ad un nome a dominio;
    - WARN_SYNTAX_FQDN_IN_NSERVER, se la stringa '*domain_name*' contiene due caratteri '-' consecutivi;
    - ERR_SYNTAX_NOIP_IN_NSERVER, la stringa passata come parametro in ingresso non contiene una stringa del tipo '*indirizzo_IP*';
    - ERR_SYNTAX_NOFQDN_IN_NSERVER, la stringa passata come parametro in ingresso non contiene una stringa del tipo '*domain_name*';
    - ERR_SYNTAX_SWAP_IP_FQDN_IN_NSERVER, il parametro in

ingresso contiene le due stringhe '*domain_name*' e '*indirizzo_IP*' in ordine inverso.

FUNZIONI UTILIZZATE

- **&IpaddrOK():** utilizzata per il controllo sintattico della stringa '*indirizzo_IP*' contenuta nel parametro in ingresso;
- **&DNSOK():** utilizzata per il controllo sintattico della stringa '*domain_name*' contenuta nel parametro in ingresso;

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$nserver* utilizzando le funzioni *&IpaddrOK()* e *&DNSOK()* per i controlli sintattici sull'indirizzo IP e sul nome a dominio. Ogni errore rilevato è inserito nel vettore *@result* che, infine, viene restituito in uscita.

5.10.12 DomNetOK()

Questa funzione controlla la correttezza sintattica del valore di un campo *dom-net* contenuto in un oggetto *domain*, ovvero controlla che la stringa passata come parametro in ingresso sia del tipo:

$$indirizzo\_rete_1\ indirizzo\_rete_2 \ldots indirizzo\_rete_N$$

con:

- $N \geq 1$;
- $indirizzo\_rete_i$, stringa contenente un indirizzo IP di rete valido;

IN

- **$domnet:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:
  - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_DOMNET, se la sintassi del parametro in ingresso non è corretta.

FUNZIONI UTILIZZATE

- **&NETaddrOK():** utilizzata per il controllo sintattico delle singole stringhe $indirizzo\_IP_i$.

DESCRIZIONE

La funzione individua nella stringa contenuta nel parametro in ingresso *$domnet* 'N' sottostringhe divise da spazi bianchi e controlla che ognuna di esse rappresenti un indirizzo IP di rete valido utilizzando la funzione *&NETaddrOK()*. Se il controllo sintattico di almeno una delle sottostringhe fallisce, fallisce anche il controllo su tutto il parametro in ingresso. Il codice di errore individuato è inserito nel vettore *@result* che viene restituito in uscita.

5.10.13 AuthOK()

Questa funzione esegue il controllo sintattico del valore di un campo *auth* inserito in un oggetto *domain,* ovvero controlla che la stringa passata come parametro in ingresso abbia una sintassi del tipo:

'*CRYPT-PW password'* oppure '*MAIL-FROM espressione*'

con:

- '*password',* stringa non vuota di 13 caratteri;
- '*espressione*', stringa non vuota.

All'interno dell'attributo la stringa '*password'* contiene una password criptata con un algoritmo triplo DES, mentre '*espressione*' è un'espressione regolare che descrive una categoria di indirizzi di posta elettronica.

IN

- **$auth:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:
  - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_CRYPT, se lunghezza('*password'*) ≠13 caratteri;
  - ERR_SYNTAX_AUTH, se il parametro in ingresso non contiene né la stringa CRYPT-PW nè MAIL-FROM;
  - ERR_SYNAX_NOEMPTY, se la stringa '*password'* o *espressione*' è vuota.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$auth* utilizzando i 'pattern matching' del Perl ed inserisce i codici di errore eventualmente rilevati nel vettore *@result* che restituisce in uscita.

5.10.14 TelNumberOK()

Questa funzione controlla che la stringa passata in ingresso abbia la sintassi di un numero telefonico:

'$+N_1\ N_2 N_3$' oppure '$+N_1\ N_2 N_3\ ext\ N_4$' oppure '$+N_1\ N_2 N_3\ EXT\ N_4$'

con:

- N_i, stringa numerica non vuota;

Se il prefisso internazionale indica un numero telefonico italiano, allora controlla che il prefisso telefonico abbia la sintassi di un prefisso teleselettivo relativi a tutti gli operatori di telefonia fissa o mobile italiana, ovvero

- se, N_1 = '39', allora N_2 = '$0N$' con 'N' $\notin$ (333, 334, 335, 336, 337, 338, 339, 330, 360, 366, 368, 340, 347, 348, 349, 320, 328, 329, 380, 388, 389).

La funzione è utilizzata per il controllo sintattico degli attributi *phone* e *fax-no* dell'oggetto *person*.

IN

- **$phone:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:

  ERR_NONE, se la sintassi del parametro in ingresso e' corretta;

  ERR_EMPTY, se il parametro in ingresso e' vuoto;

  ERR_SYNTAX_PHONE, se la sintassi del parametro in ingresso è errata.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$phone* utilizzando i 'pattern matching' del Perl ed inserisce i codici di errore eventualmente rilevati nel vettore *@result* che restituisce in uscita.

5.10.15 DNSOK()

Questa funzione controlla che la sintassi della stringa passata come parametro in ingresso abbia una sintassi conforme ad un nome a dominio:

$str_1.str_2.str_N.tld$

con:

- str_i = <*c*><*str*><*c*> con 'str' stringa non vuota composta da soli caratteri alfanumerici con l'aggiunta del carattere '-' (meno) e '*c*' carattere alfanumerico;
- lunghezza($str_1.str_2.str_N.tld$) ≤ 255;
- 2 ≤ lunghezza(str_i) ≤ 63;
- *tld,* stringa non vuota corrispondente ad uno dei ccTLD o gTLD attualmente esistenti.

La funzione è utilizzata dalla funzione *NserverOK()* per il controllo sintattico della parte del nameserver contenente il nome a dominio del DNS.

IN

- **$domain:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@result:** vettore contenente i codici degli errori eventualmente rilevati:
  - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_DOMNAME, se la sintassi del parametro in ingresso non è conforme;
  - WARN_SYNTAX_HYPHEN_DOMNAME, se almeno una delle stringhe str_i , sopra descritte, contiene due caratteri '-' consecutivi.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$domain* utilizzando i 'pattern matching' del Perl ed inserisce i codici di errore eventualmente rilevati nel vettore *@result* che restituisce in uscita.

5.10.16 IPaddrOK()

Questa funzione controlla che la stringa passata come parametro in ingresso abbia la sintassi conforme ad un indirizzo IP valido secondo le specifiche IANA [RFC1466]:

$N_1.N_2.N_3.N_4$

con:

- N_i con $2 \leq i \leq 4$, valore numerico $0 \leq N_i \leq 255$;
- N_1, valore numerico $0 \leq N_i \leq 223$.

La funzione è utilizzata dalla funzione *NserverOK()* per il controllo sintattico della parte del nameserver contenente l'indirizzo IP del DNS.

IN

- **$ipaddr:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@result:**
  - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_IPADDR, se la sintassi del parametro in ingresso non è conforme.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$ipadrr* utilizzando i 'pattern matching' del Perl ed inserisce i codici di errore eventualmente rilevati nel vettore *@result* che restituisce in uscita.

5.10.17 NETaddrOK()

Questa funzione controlla che la stringa passata come parametro in ingresso abbia la sintassi conforme ad un indirizzo IP di rete valido [XXXX] tenendo conto anche della notazione CIDR (Classless Inter Domain Routing) [RFC1517]:

$N_1.N_2.N_3.N_4$ oppure $N_1.N_2.N_3.N_4/ N_5$

con:

- N_i con $2 \leq i \leq 4$, valore numerico $0 \leq N_i \leq 255$;
- N_1, valore numerico $0 \leq N_i \leq 223$;
- N_5, valore numerico $1 \leq N_i \leq 32$.

La funzione è utilizzata dalla funzione *NserverOK()* per il controllo sintattico della parte del nameserver contenente l'indirizzo IP del DNS.

IN

- **$ipaddr:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@result:**
  - ERR_NONE, se la sintassi del parametro in ingresso e' corretta;
  - ERR_EMPTY, se il parametro in ingresso e' vuoto;
  - ERR_SYNTAX_NETADDR, se la sintassi del parametro in ingresso non è conforme.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$ipadrr* utilizzando i 'pattern matching' del Perl ed inserisce i codici di errore eventualmente rilevati nel vettore *@result* che restituisce in uscita.

5.10.18 DateOK()

Questa funzione controlla che la stringa passata come parametro in ingresso abbia la sintassi di una data nel formato:

yyyymmdd

con:

- 1970 ≤ '*yyyy*' ≤ 'anno_corrente';
- 1 ≤ '*mm*' ≤ 12;
- 1 ≤ '*gg*' ≤ 31.

Inoltre, la funzione, durante i controlli, tiene conto degli anni bisestili.

La funzione viene utilizzata nella funzione *&ChangedOK()* per il controllo sintattico della parte dell'attributo *changed* contenente la data.

IN

- **$data:** stringa sulla quale vengono effettuati i controlli;

OUT

- **@result:** vettore contenente i codici di errore eventualmente rilevati:

  ERR_NONE, se la sintassi del parametro in ingresso e' corretta;

  ERR_EMPTY, se il parametro in ingresso e' vuoto;

  ERR_SYNTAX_DATE, se la sintassi del parametro in ingresso non è conforme;

  ERR_SYNTAX_OUT_OF_DATE, se il parametro in ingresso indica una data successiva alla data odierna.

FUNZIONI UTILIZZATE

- **localtime():** funzione predefinita Perl, utilizzata per il calcolo della data odierna;
- **parsedate():** contenuto nel modulo predefiniti Perl Time::ParseDate, è utilizzata per il confronto della data contenuta nel parametro in ingresso con la data odierna.

DESCRIZIONE

La funzione esegue i controlli sintattici sulla stringa contenuta nel parametro in ingresso *$data* utilizzando le funzioni *localtime()* e *parsedate()* per controllare che la data indicata dal parametro non sia successiva alla data odierna. La funzione inserisce i codici di errore relativi nel vettore *@result* il quale, terminati tutti i controlli viene restituito in uscita.

5.11 CheckAuth.pm

Questo modulo contiene le funzioni che effettuano i controlli di autenticazione sulle varie tipologie di oggetti contenuti nel modulo elettronico.

Nel file sono inclusi i moduli *WHOISDBinterface.pm* e *GlobalCheck.pm* utilizzati per la realizzazione dell'interfaccia con i database del Registro.

Le funzioni definite in questo modulo sono:

- **&CheckAuthDomain():** effettua i controlli di autenticazione su un oggetto *domain*;
- **&CheckAuthMntner():** effettua i controlli di autenticazione su un oggetto *mntner*;
- **&CheckAuthContact():** effettua i controlli di autenticazione su un contatto tecnico o amministrativo (oggetto *person* o *role)*;
- **&GetUpdto():** funzione utilizzata internamente al modulo, legge gli indirizzi di posta elettronica associati agli attributi *upd-to* di un dato maintainer.

Nelle sezioni seguenti descrivono le funzioni sopra elencate.

5.11.1 CheckAuthDomain()

Questa funzione effettua i controlli di autenticazione su un oggetto *domain*.

IN

- **$domain:** nome del dominio (valore dell'attributo *domain*);
- **$passwd:** password di autenticazione dell'oggetto (valore dell'attributo *password*);
- **$from:** indirizzo di posta elettronica inserito nel campo 'From:' del messaggio che contiene l'oggetto;
- **$mntby:** nome del maintainer gestore dell'oggetto (valore dell'attributo *mnt-by*)

OUT

- **$error-code:** codice di errore che indica il risultato dei controlli:
  - ERR_CONNECTION, se c'e' stato un errore di connettivita' nella connessione al database;
  - ERR_MNTNOTFOUND, se il maintainer dichiarato come gestore dell'oggetto non esiste nel database del Registro;
  - ERR_AUTH, se la password o l'indirizzo di posta elettronica utilizzate per l'autenticazione non coincidono a quelle dichiarate;
  - ERR_MNTNOTAUTH, se il maintainer non e' autorizzato alla variazione dei dati del dominio;
  - ERR_NONE, se i controlli di autenticazione sono andati a buon fine;
- **$updto:** puntatore ad un vettore contenente gli indirizzi di posta elettronica associati agli attributi *upd-to* del maintainer gestore nel caso in cui *$error_code* sia uguale a ERR_MNTNOTAUTH

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::whois:** comando necessario corrispondente all'interrogazione al database WHOIS;

FUNZIONI UTILIZZATE

- **&WHOISDBinterface::MakeQuery():** esegue le interrogazioni al database WHOIS del Registro;

- **&WHOISDBinterface::QueryError():** decifra la risposta fornita dall'interrogazione al database WHOIS;
- **&GetUpdto():** restituisce gli indirizzi di posta elettronica associati agli attributi *upd-to* di un determinato maintainer.

DESCRIZIONE

La funzione esegue i controlli di autenticazione di un oggetto *domain* in base ai parametri forniti in ingresso ed al contenuto del database WHOIS del Registro. In particolare, la funzione controlla:

- che il maintainer indicato come gestore dell'oggetto (*$mnt-by*) sia registrato nel database WHOIS del Registro;
- che il maintainer gestore dell'oggetto sia autorizzato a modificare l'oggetto stesso, nel caso in cui questo sia già stato registrato nel database WHOIS del Registro, in caso negativo fornisce gli indirizzi di posta elettronica indicati dal maintainer negli attributi *upd-to* (*$updto*) per consentire l'invio dei messaggi di notifica del tentativo di corruzione dell'oggetto;
- che la password di autenticazione inserita nell'oggetto (*$passwd*) o l'indirizzo di posta dal quale proviene il modulo elettronico (*$from*) coincidano con le autenticazioni dichiarate dal maintainer al momento della sua registrazione al Registro (attributi *auth* dell'oggetto *mntner* associato al maintainer gestore).

In *figura n.4* è mostrato il diagramma di flusso dei vari controlli:

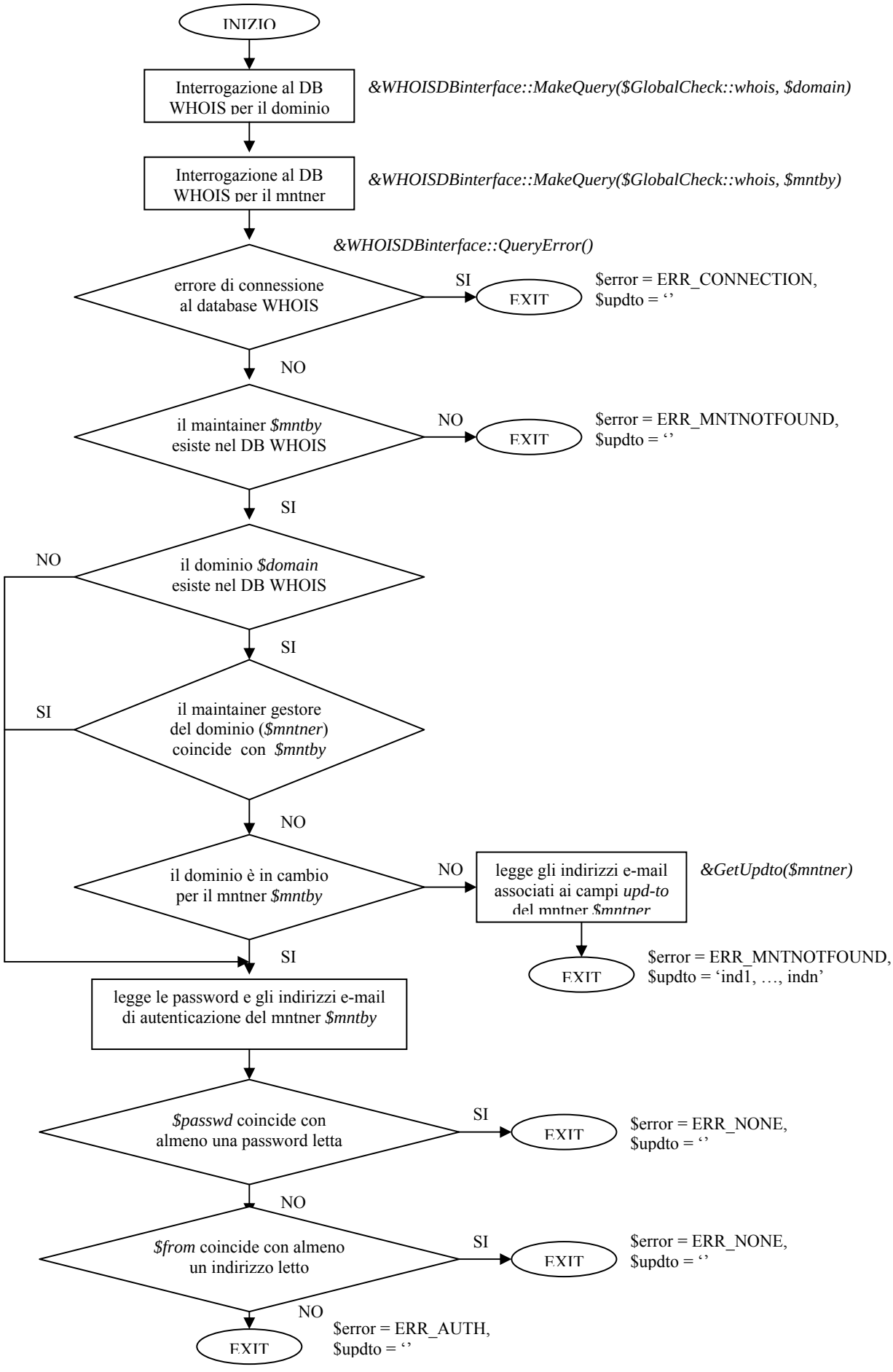

INIZIO
Interrogazione al DB WHOIS per il dominio
&WHOISDBinterface::MakeQuery($GlobalCheck::whois, $domain)
Interrogazione al DB WHOIS per il mntner
&WHOISDBinterface::MakeQuery($GlobalCheck::whois, $mntby)
&WHOISDBinterface::QueryError()
errore di connessione al database WHOIS
SI
EXIT
$error = ERR_CONNECTION,
$updto = ‘’
NO
il maintainer $mntby esiste nel DB WHOIS
NO
EXIT
$error = ERR_MNTNOTFOUND,
$updto = ‘’
SI
il dominio $domain esiste nel DB WHOIS
NO
SI
il maintainer gestore del dominio ($mntner) coincide con $mntby
SI
NO
il dominio è in cambio per il mntner $mntby
NO
legge gli indirizzi e-mail associati ai campi upd-to del mntner $mntner
&GetUpdto($mntner)
EXIT
$error = ERR_MNTNOTFOUND,
$updto = ‘ind1, …, indn’
SI
legge le password e gli indirizzi e-mail di autenticazione del mntner $mntby
$passwd coincide con almeno una password letta
SI
EXIT
$error = ERR_NONE,
$updto = ‘’
NO
$from coincide con almeno un indirizzo letto
SI
EXIT
$error = ERR_NONE,
$updto = ‘’
NO
EXIT
$error = ERR_AUTH,
$updto = ‘’

figura n.4

5.11.2 CheckAuthMntner()

Questa funzione effettua i controlli di autenticazione su un oggetto *mntner*.

IN

- **$mntner:** nome del maintainer (valore dell'attributo *mntner*);
- **$passwd:** password di autenticazione dell'oggetto (valore dell'attributo *password*);
- **$from:** indirizzo di posta elettronica inserito nel campo 'From:' del messaggio che contiene l'oggetto;

OUT

- **$error:**
  - ERR_CONNECTION, se si è presentato un errore di connettivita' nella connessione al database WHOIS;
  - ERR_AUTH, se c'e' un errore di autenticazione rispetto alle autenticazioni via password o posta elettronica dichiarate;
  - ERR_NONE, se non ci sono errori di autenticazione.

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::whois:** comando corrispondente all'interrogazione al database WHOIS;

FUNZIONI UTILIZZATE

- **&WHOISDBinterface::MakeQuery():** esegue le interrogazioni al database WHOIS del Registro;
- **&WHOISDBinterface::QueryError():** decifra la risposta fornita dall'interrogazione al database WHOIS;
- **&GetUpdto():** restituisce gli indirizzi di posta elettronica associati agli attributi *upd-to* di un determinato maintainer.

DESCRIZIONE

La funzione esegue i controlli di autenticazione di un oggetto *mntner* in base ai parametri forniti in ingresso ed al contenuto del database WHOIS del Registro. In particolare, nel caso in cui il maintainer *$mntner* sia gia' registrato nel database, controlla che la password di autenticazione dichiarata nell'oggetto (paramerto *$passwd*) o l'indirizzo di posta elettronica inserito nel campo

'From' del messaggio contenente il modulo elettronico (parametro *$from*) coincidano con una delle autenticazioni dichiarate al momento della registrazione del maintainer (attributi *auth* dell'oggetto *mntner* registrato).

In *figura n.5* è mostrato il diagramma di flusso dei vari controlli:

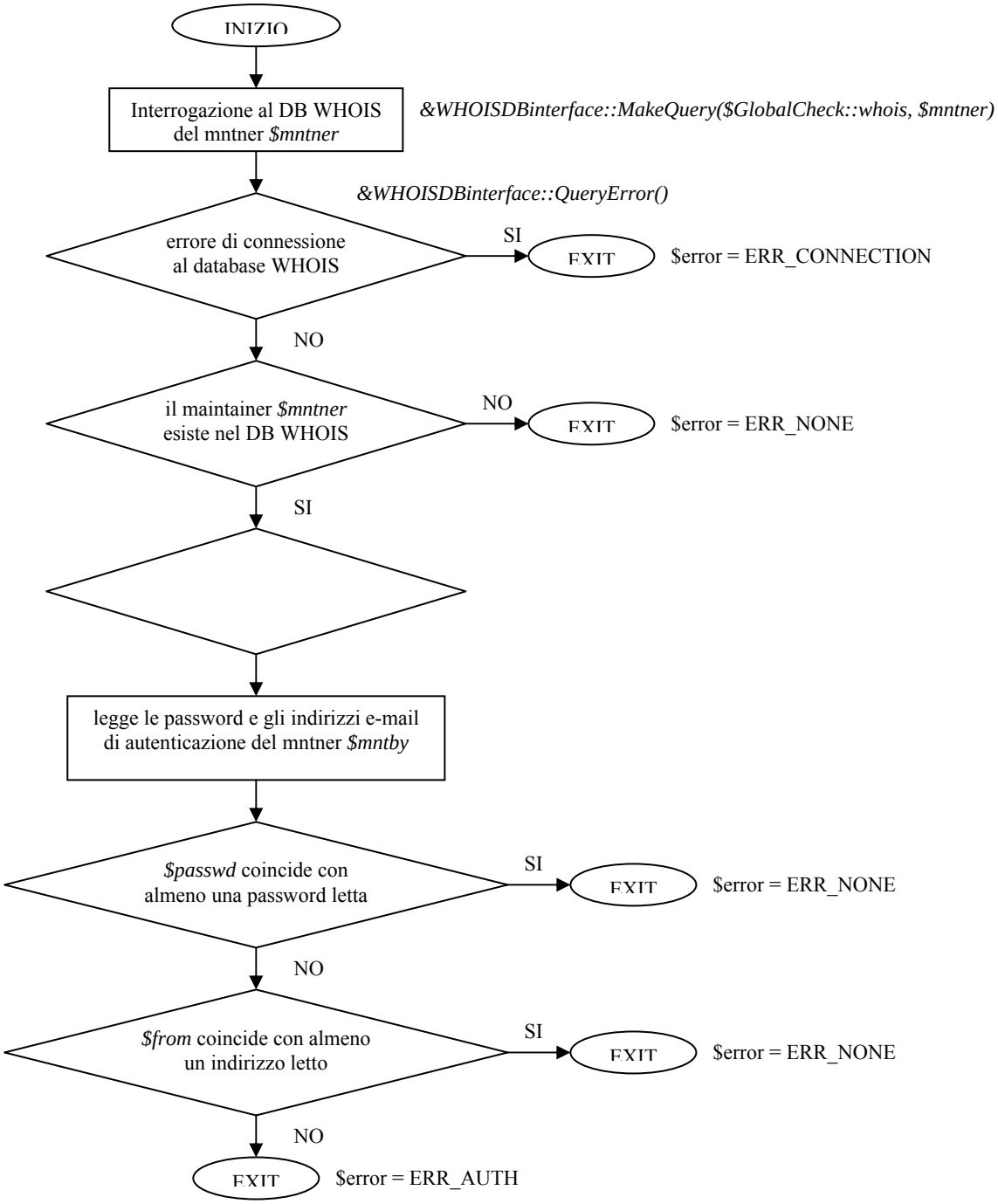


figura n.5

5.11.3 CheckAuthContact()

Questa funzione effettua i controlli di autenticazione su un oggetto *person*.

IN

- **$nichdl:** *nic-handle* associato al contatto analizzato (valore dell'attributo *nic-hdl*);
- **$mntby:** nome del maintainer gestore dell'oggetto (valore dell'attributo *mnt-by*)
- **$passwd:** password di autenticazione dell'oggetto (valore dell'attributo *password*);
- **$from:** indirizzo di posta elettronica inserito nel campo 'From:' del messaggio che contiene l'oggetto;

OUT

- **$error-code:** codice di errore che indica il risultato dei controlli:
  - ERR_CONNECTION, se c'e' stato un errore di connettivita' nella connessione al database;
  - ERR_MNTNOTFOUND, se il maintainer dichiarato come gestore dell'oggetto non esiste nel database del Registro;
  - ERR_AUTH, se la password o l'indirizzo di posta elettronica utilizzate per l'autenticazione non coincidono a quelle dichiarate;
  - ERR_MNTNOTAUTH, se il maintainer non e' autorizzato alla variazione dei dati del dominio;
  - ERR_NONE, se i controlli di autenticazione sono andati a buon fine;
- **$updto:** puntatore ad un vettore contenente gli indirizzi di posta elettronica associati agli attributi *upd-to* del maintainer gestore nel caso in cui *$error_code* sia uguale a *ERR_MNTNOTAUTH.*

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::whois:** comando corrispondente all'interrogazione al database WHOIS;
- **$GlobalCheck::whoisHandle:** comando corrispondente all'interrogazione al database HANDLE;

FUNZIONI UTILIZZATE

- **&WHOISDBinterface::MakeQuery():** esegue le interrogazioni ai database WHOIS e HANDLE del Registro;
- **&WHOISDBinterface::QueryError():** decifra la risposta fornita dall'interrogazione ai database WHOIS e HANDLE del Registro;
- **&GetUpdto():** restituisce gli indirizzi di posta elettronica associati agli attributi *upd-to* di un determinato maintainer.

DESCRIZIONE

La funzione esegue i controlli di autenticazione di un oggetto *person* in base ai parametri forniti in ingresso ed al contenuto dei database WHOIS e HANDLE del Registro. Per effettuare le verifiche sono presi in considerazione i dati registrati più aggiornati, ovvero quelli presenti nel database WHOIS oppure, se l'oggetto *person* o *role* non è stato ancora registrato, quelli presenti nel database HANDLE. La funzione prevede che il *nic-handle* associato al contatto sia già stato registrato e lo utilizza come chiave per la ricerca della persona nei database.

In particolare la funzione controlla:

- che il maintainer indicato come gestore dell'oggetto (*$mntby*) sia stato registrato nel database WHOIS del Registro;
- che il maintainer gestore dell'oggetto, se esistente, sia autorizzato a modificare l'oggetto stesso, nel caso in cui questo sia già stato registrato nel database WHOIS o HANDLE del Registro, in caso negativo fornisce gli indirizzi di posta elettronica indicati dal maintainer negli attributi *upd-to* per consentire l'invio dei messaggi di notifica del tentativo di corruzione (*$updto*);
- che la password di autenticazione inserita nell'oggetto (*$passwd*) o l'indirizzo di posta dal quale proviene il modulo elettronico (*$from*) coincidano con le autenticazioni dichiarate dal maintainer al momento della sua registrazione al Registro (attributi *auth* dell'oggetto *mntner* associato al maintainer gestore).

In *figura n.6* è mostrato il diagramma di flusso dei vari controlli:

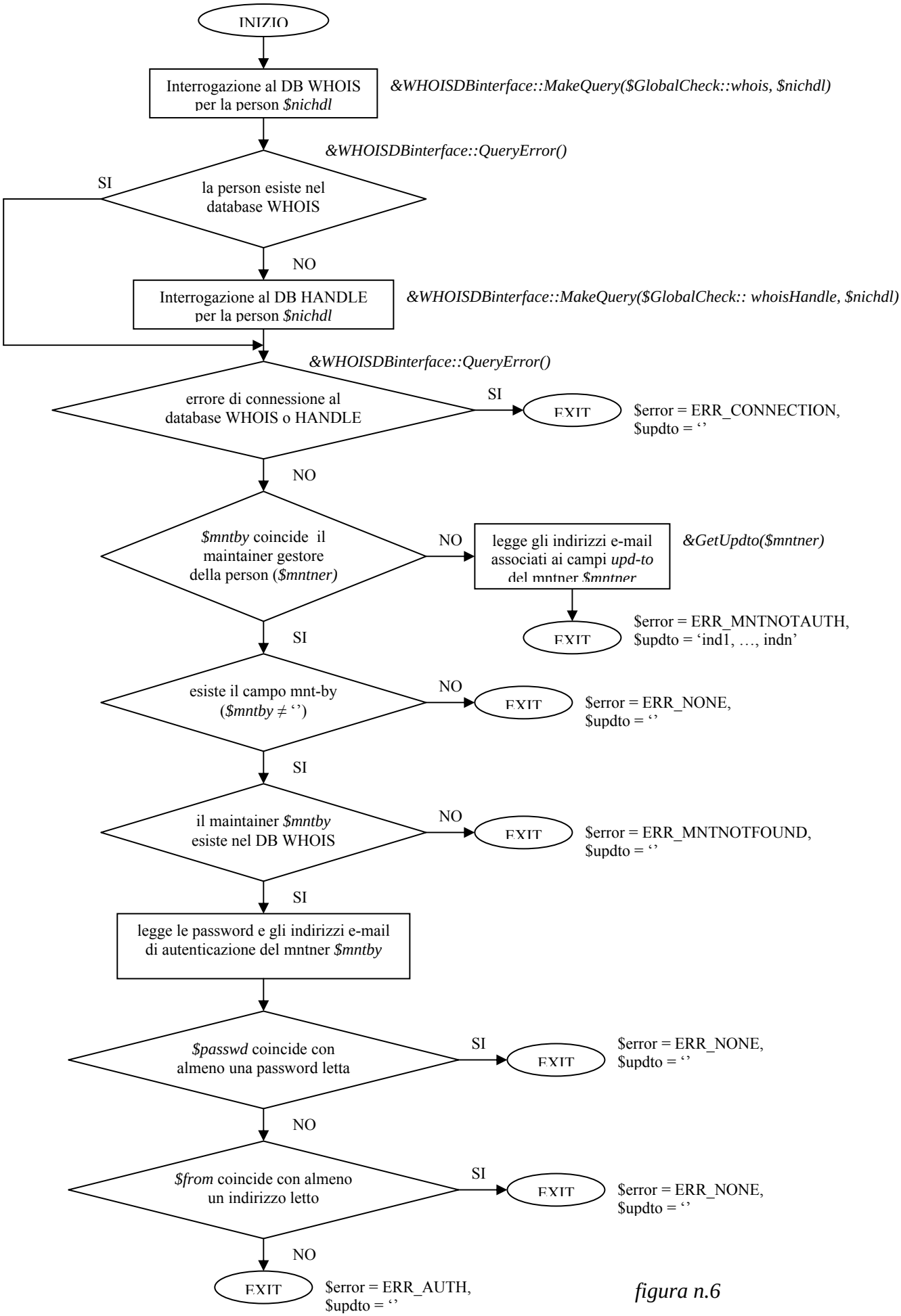

INIZIO
Interrogazione al DB WHOIS per la person $nichdl
&WHOISDBinterface::MakeQuery($GlobalCheck::whois, $nichdl)
&WHOISDBinterface::QueryError()
SI
la person esiste nel database WHOIS
NO
Interrogazione al DB HANDLE per la person $nichdl
&WHOISDBinterface::MakeQuery($GlobalCheck:: whoisHandle, $nichdl)
&WHOISDBinterface::QueryError()
errore di connessione al database WHOIS o HANDLE
SI
EXIT
$error = ERR_CONNECTION,
$updto = ‘’
NO
$mntby coincide il maintainer gestore della person ($mntner)
NO
legge gli indirizzi e-mail associati ai campi upd-to del mntner $mntner
&GetUpdto($mntner)
EXIT
$error = ERR_MNTNOTAUTH,
$updto = ‘ind1, …, indn’
SI
esiste il campo mnt-by ($mntby ≠ ‘’)
NO
EXIT
$error = ERR_NONE,
$updto = ‘’
SI
il maintainer $mntby esiste nel DB WHOIS
NO
EXIT
$error = ERR_MNTNOTFOUND,
$updto = ‘’
SI
legge le password e gli indirizzi e-mail di autenticazione del mntner $mntby
$passwd coincide con almeno una password letta
SI
EXIT
$error = ERR_NONE,
$updto = ‘’
NO
$from coincide con almeno un indirizzo letto
SI
EXIT
$error = ERR_NONE,
$updto = ‘’
NO
EXIT
$error = ERR_AUTH,
$updto = ‘’

figura n.6

5.11.4 GetUpdto()

Questa funzione restituisce gli indirizzi di posta elettronica associati agli attributi *upd-to* di un oggetto maintainer il cui nome viene passato come parametro in ingresso.

IN

- **$mntner:** nome del maintainer dal quale vengono letti i attributi *upd-to*;

OUT

- **$error-code:** codice di errore:
  - ERR_CONNECTION, se c'e' stato un errore di connettivita' durante la connessione al database WHOIS;
  - ERR_NONE, altrimenti;
- **$updto:** puntatore ad un vettore contenente gli indirizzi di posta elettronica associati agli attributi *upd-to* del maintainer *$mntner*.

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::whois:** comando necessario corrispondente all'interrogazione al database WHOIS;

FUNZIONI UTILIZZATE

- **&WHOISDBinterface::MakeQuery():** esegue le interrogazioni ai database WHOIS del Registro;
- **&WHOISDBinterface::QueryError():** decifra la risposta fornita dall'interrogazione al database WHOIS del Registro;

DESCRIZIONE

La funzione prende, dal database WHOIS, l'oggetto *mntner* associato al maintainer $*mntner* passato come parametro utilizzando la funzione *&WHOISDBinterface::MakeQuery($GlobalCheck::whois, $mntner)*. Se non si sono verificati errori di connettività, esamina tutti gli attributi dell'oggetto registrando il valore associato agli attributi *upd-to* in un vettore che infine restituisce in uscita assieme al codice di errore ERR_NONE. Altrimenti, restituisce il codice ERR_CONNECTION con il vettore vuoto.

5.12 CheckCons.pm

Questo modulo contiene le funzioni che effettuano i controlli di consistenza previsti dal sistema sugli attributi di un oggetto *domain*.

Le funzioni contenute nel file sono le seguenti:

&CheckConsX400domain(): controlla la consistenza dell'attributo *x400-domain* in relazione al campo *domain* contenuto nello stesso oggetto;

&CheckConsPin(): controlla la consistenza dell'attributo 'pin' in relazione al contenuto del database WHOIS;

&CheckConsDomainType(): Controlla la consistenza degli attributi *nserver*, *mailgate*, *zone-c* e *gate-c* in relazione alla tipologia del dominio (sola posta elettronica o un dominio *full delegated*);

&CheckConsPostmaster(): Controlla che l'oggetto associato al *postmaster* di un nome a dominio contenga il campo *e-mail*.

Il file include i moduli *WHOISDBinterface.pm* e *GlobalCheck.pm* dai quali utilizza le funzioni e le variabili globali necessarie per effettuare l'interrogazione al database WHOIS del Registro.

Le sezioni successive descrivono le funzioni contenute nel modulo.

5.12.1 CheckConsX400domain()

Questa funzione controlla la consistenza dei valori inseriti nei sotto-campi dell'attributo *x400-domain* in relazione al valore inserito nel campo *domain* di un oggetto *domain*. La funzione suppone che i valori confrontati siano sintatticamente corretti.

IN

- **$domain:** stringa corrispondente al valore inserito nel campo *domain*;
- **$x400domain:** stringa corrispondente al valore inserito nel campo *x400-domain*;

OUT

- **%ERROR:** hash contenente gli errori ed i warning rilevati nel formato (*etichetta, codice_errore*):
  - *(etichetta*, ERR_INCONSISTENT_X400DOMAIN), se il valore di un sottocampo dell'attributo *x400-domain* non è consistente con il valore del livello corrispondente dell'attributo *domain,*in questo caso l'etichetta corrisponde al nome del sottocampo corrente;
  - *(x400*, ERR_SYNTAX_CHECK_X400), se il numero dei sotto-campi non coincide col numero dei livelli del nome a dominio;
  - *(Nadmd*, WARN_LOOK_X400DOMAIN), se il valore del sottocampo 'admd' non e' uguale a zero;

  l'etichetta associata all'errore contiene un suffisso numerico 'N' per permettere la segnalazione di piu' errori associati allo stesso sottocampo.

DESCRIZIONE

La funzione suddivide il nome a dominio contenuto nel parametro in ingresso *$domain* secondo i livelli che lo compongono e li confronta con i sotto-campi contenuti nel parametro *$x400domain* inserendo nell'hash *%ERROR* gli eventuali codici di errore.

5.12.2 CheckConsPin()

Questa funzione controlla che non esistano, all'interno del database WHOIS, un altro dominio assegnato alla persona fisica individuata da un determinato codice fiscale.

IN

- **$pin:** stringa corrispondente al codice fiscale che individua la persona fisica;
- **$domain:** nome del dominio contenente il campo *pin* con valore *$pin*;

OUT

- **$error:**
  - ERR_REGISTERED_PIN, se esiste gia' un dominio, diverso da *$domain* registrato nel database WHOIS;
  - ERR_INVALID_PIN, se il valore *$pin* non è un codice valido;
  - ERR_CONNECTION, se c'e' stato un errore di connessione durante l'esecuzione dell'interrogazione al database WHOIS;
  - ERR_NONE, se non esiste nel database WHOIS un dominio diverso da *$domain* contenente il campo pin con valore *$pin*.

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::whois:** comando associato all'interrogazione del database WHOIS;

FUNZIONI UTILIZZATE

- **&WHOISDBinterface::MakeQuery():** utilizzata per effettuare l'interrogazione al database WHOIS del Registro;
- **&WHOISDBinterface::QueryError():** utilizzata per decifrare la risposta fornita dall'interrogazione al database WHOIS;

DESCRIZIONE

La funzione controlla l'esistenza del codice fiscale all'interno del database WHOIS del Registro utilizzando le funzioni *&WHOISDBinterface::MakeQuery()* e *&WHOISDBinterface::QueryError()* e restituisce in uscita un codice di errore dipendente dal risultato dell'interrogazione.

In *figura n.7* è riportato il diagramma che specifica la struttura del programma:

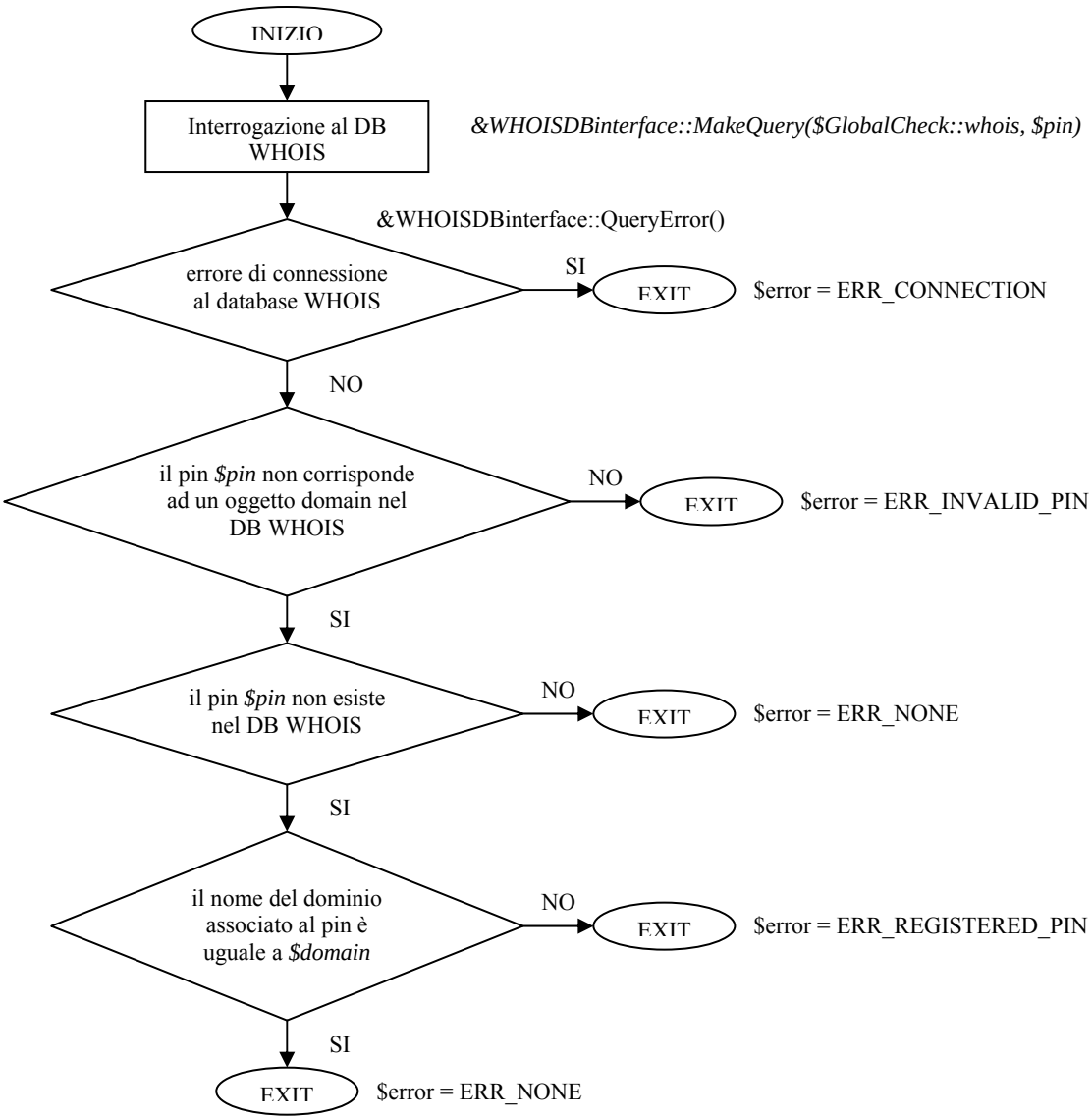

INIZIO
Interrogazione al DB WHOIS
&WHOISDBinterface::MakeQuery($GlobalCheck::whois, $pin)
&WHOISDBinterface::QueryError()
errore di connessione al database WHOIS
SI
EXIT
$error = ERR_CONNECTION
NO
il pin $pin non corrisponde ad un oggetto domain nel DB WHOIS
NO
EXIT
$error = ERR_INVALID_PIN
SI
il pin $pin non esiste nel DB WHOIS
NO
EXIT
$error = ERR_NONE
SI
il nome del dominio associato al pin è uguale a $domain
NO
EXIT
$error = ERR_REGISTERED_PIN
SI
EXIT
$error = ERR_NONE

figura n.7

5.12.3 CheckConsDomainType()

Questa funzione controlla che in un oggetto domain siano stati inseriti tutti e soli gli attributi necessari per la configurazione del dominio come di 'sola posta elettronica' (attributi mailgate e gate-c) o 'interamente delegato' (attributi nserver e zone-c) (Vedi *sezione 4.1.3*).

IN

- **$NumNserver:** numero di attributi *nserver* presenti nell'oggetto *domain*;
- **$NumGatec:** numero di attributi *gate-c* presenti nell'oggetto *domain*;
- **$NumMailgate:** numero di attributi *mailgate* presenti nell'oggetto *domain*;
- **$NumZonec:** numero di attributi *zone-c* presenti nell'oggetto *domain*;

OUT

- **@ERROR:** vettore contenente i codici degli errori eventualmente rilevati. I codici possono essere:
  - ERR_MISS_ZONEC, se è stato inserito almeno un campo *nserver* e non è stato inserito un campo *zone-c*;
  - ERR_MISS_GATEC, se e' stato inserito almeno un campo *mailgate* e non e' stato inserito un campo *gate-c*;
  - ERR_MISS_NSERVER, se e' stato inserito un solo campo *nserver*;
  - ERR_NO_CONF, se e' stato inserito ne' un campo *nserver*, ne' un campo *mailgate*;
  - ERR_BAD_CONF1, se è stato inserito almeno un campo *mailgate* e *zone-c* o *nserver*;
  - ERR_BAD_CONF2, se è stato inserito almeno un campo *nserver* e *gate-c*;
  - ERR_NONE, altrimenti.

DESCRIZIONE

La funzione legge il numero di attributi *nserver, mailgate, zone-c* e *gate-c* presenti nell'oggetto *domain* dai parametri di ingresso $NumNserver, $NumMailgate, $NumGatec e $NumZonec e li confronta tra loro aggiungendo al vettore *@error* i codici di errore secondo le condizioni sopra descritte.

5.12.4 CheckConsPostmaster()

Questa funzione è utilizzata dal sistema per il controllo si consistenza del postmaster di un nome a dominio. Essa controlla che nell'oggetto passato come parametro sia presente il campo *e-mail*.

IN

- **%OBJECT:** oggetto nel formato (label, valore); da notare che le label che indicano la tipologia dell'oggetto non devono contenere prefissi numerici.

OUT

- **$error:** codice di errore eventualmente rilevato:
  - ERR_NO_VALID_POSTMASTER, se l'oggetto passato come parametro non contiene il campo *e-mail*;
  - ERR_NONE, altrimenti;

DESCRIZIONE

Il programma esamina tutti gli attributi dell'oggetto *%OBJECT* passato come parametro e restituisce 'ERR_NONE' non appena trova una label di valore 'e-mail', altrimenti restituisce 'ERR_NO_VALID_POSTMASTER'.

5.13 WHOISDBinterface.pm

Questo modulo contiene le funzioni che implementano l'interfaccia con i database del Registro. Le funzioni, descritte nelle sezioni successive, sono le seguenti:

- **&MakeQuery():** restituisce il risultato di una interrogazione ad un database specificato;
- **&QueryError():** restituisce il codice di errore corrispondente ad una determinata risposta fornita dal database WHOIS o HANDLE dl Registro.

5.13.1 MakeQuery()

Restituisce il risultato di una interrogazione ai database WHOIS o HANDLE del Registro.

IN

- **$database:** comando che identifica l'interrogazione al database prescelto;
- **$param:** parametro sulla base del quale effettuare l'interrogazione.

OUT

- **@result:** vettore contenente il risultato dell'interrogazione al database suddiviso per righe.

DESCRIZIONE

La funzione esegue il comando descritto dal parametro in ingresso *$database* sul parametro *$param*, suddivide il risultato in righe e le assegna al vettore *@result* che restituisce in uscita.

5.13.2 QueryError()

Analizza il risultato di una interrogazione al database WHOIS o HANDLE del Registro e restituisce un codice di errore corrispondente al tipo di risposta analizzata.

IN

- **$QueryResult:** puntatore ad un vettore contenente il risultato dell'interrogazione al database suddiviso per righe;
- **$word:** parola chiave sulla base della quale viene interpretato il risultato dell'interrogazione (domain, mntner, role, person);

OUT

- **$error** codice di errore:
  - ERR_NOTFOUND, se il risultato dell'interrogazione è del tipo 'no entry found';
  - ERR_NONE, se il risultato dell'interrogazione contiene un oggetto che contiene un campo denominato *$word*;
  - ERR_CONNECTION, se si è verificato un errore di connessione;
  - ERR_DEFAULT: in tutti gli altri casi.

DESCRIZIONE

La funzione prende in esame tutte le righe di testo della risposta contenuta nel vettore puntato dal parametro in ingresso *$QueryResult* e restituisce il codice di errore appropriato a seconda del loro contenuto. In particolare, restituisce:

- ERR_NOTFOUND, se incontra una riga contenente la stringa 'no entries found';
- ERR_NONE, se analizza una riga contenente la parola contenuta nel parametro in ingresso *$word*;
- ERR_CONNECTION, se analizza una riga contenente la stringa 'connect';
- ERR_DEFAULT: in tutti gli altri casi.

5.14 WriteObject.pm

Questo modulo contiene la funzione **&WriteObjects()** utilizzata dal sistema per stampare gli oggetti nei messaggi di posta elettronica generati nel corso dell'esecuzione.

Il modulo è utilizzato nei file *main.pl* dalle funzioni

- *&CreateRegMail(),* per la creazione del modulo elettronico da inviare alle fasi successive della procedura di registrazione, nel caso in cui tutti i controlli effettuati vadano a buon fine;
- *&PrintSyntaxError(),* per la stampa degli oggetti in cui si è verificato un errore e/o un warning;
- *&CreateFwdMail(),* per la stampa dell'oggetto che ha subito il tentativo di modifica da parte di un P/M non autoritativo.

In questo file sono inclusi i moduli *SyntaxConf.pm* e *utility.pm* utilizzati rispettivamente per il reperimento della definizione degli oggetti da stampare e per l'esecuzione della funzione di utilità che genera la data odierna.

5.14.1 WriteObjects()

Questa funzione stampa gli attributi di un oggetto *domain*, *mntner, role* o *person* passato come parametro nell'ordine stabilito nel modulo Perl *CONF/SyntaxConf.pm*.

Nel caso in cui la stampa dell'oggetto sia destinata a ricreare un modulo per la sua registrazione, la funzione controlla che al momento della compilazione dell'oggetto, sia stato inserito un campo *changed* aggiornato e in caso contrario, aggiunge un campo '*changed: hostmaster@nic.it data_odierna*' in coda agli attributi *changed* già esistenti.

La stampa dell'oggetto formattato viene effettuata nel file correntemente selezionato.

IN

- **$type:** tipologia dell'oggetto da stampare (*domain*, *mntner*, *role*, *person*);
- **$reg:** flag che indica se l'oggetto da stampare verrà inviato per la registrazione (1/0);
- **%OBJECT:** hash contenente gli attributi dell'oggetto nel formato *(etichetta, valore)*,

  N.B. L'etichetta puo' avere un prefisso numerico per consentire gli attributi multipli;

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **@SyntaxConf::*OBJECT*_FIELD** con *OBJECT* = DOMAIN, MNTNER, ROLE o PERSON a seconda della tipologia dell'oggetto da stampare, contiene il nome degli attributi dell'oggetto nell'ordine di successione in cui devono essere stampati.

FUNZIONI UTILIZZATE

- **&utility::GetDate()** utilizzata per il calcolo della data odierna.

DESCRIZIONE

La funzione legge la tipologia dell'oggetto da stampare dal parametro in ingresso *$type* e, a seconda del suo contenuto, legge l'ordinamento degli attributi dell'oggetto dalla variabile globale *@DOMAIN_FIELD*, *@MNTNER_FIELD, @ROLE_FIELD* o *@PERSON_FIELD* definite nel modulo *SyntaxConf.pm*.

Successivamente, scorre tutti gli attributi dell'oggetto da stampare contenuti nell'hash *%OBJECT* e, per ogni campo, costruisce l'etichetta togliendo l'eventuale valore numerico inserito in testa, legge il valore e lo inserisce in un vettore omonimo all'etichetta creato dinamicamente in base al valore dell'etichetta stessa.

Una volta letti tutti i valori da stampare, la funzione stampa l'oggetto con gli attributi ordinati scorrendo il vettore @SyntaxConf::*OBJECT*_FIELD e stampando, per ogni *etichetta,* tutti gli attributi del vettore omonimo, riempito precedentemente e riferito dinamicamente come '*@etichetta*'.

Se l'oggetto deve essere inviato per la registrazione (*$reg*=1), la funzione controlla che la data inserita nell'ultimo campo *changed* corrisponda alla data odierna e, in caso contrario, aggiunge un ulteriore campo *changed* con valore '*hostmaster@nic.it data_odierna*': la *data_dierna* è calcolata dalla funzione *&utility::GetDate()*. Altrimenti (*$reg*=0), il campo '*changed'* non viene aggiunto e non viene neanche stampato il campo password eventualmente presente nell'oggetto.

5.15 utility.pm

Questo modulo contiene le funzioni usate dal sistema per operazioni di varia utilità. Nel modulo sono definite le seguenti funzioni:

&GetDate(): calcola la data odierna;

&lock(): esegue il lock di un file;

&unlock(): sblocca un file precedentemente lockato;

&log(): registra un stringa di testo in un file.

Segue la descrizione delle funzioni contenute nel modulo.

5.15.1 GetDate()

Genera la data odierna e la restituisce in uscita.

IN

- **nessuno**

OUT

- **$year:** anno in corso nel formato YYYY;
- **$mon:** mese corrente nel formato MM;
- **$mday:** giorno corrente nel formato GG.

FUNZIONI UTILIZZATE

- **localtime():** funzione predefinita del Perl utilizzata per il calcolo della data odierna;

DESCRIZIONE

La funzione calcola la data odierna chiamando la funzione predefinita localtime() e normalizza i dati aggiungendo:

- 1900 alla cifra restituito per l'anno in corso;
- 1 alla cifra restituita per il mese (0-11);
- 0 in testa alla cifra restituita per il giorno corrente, se minore di 9.

Infine, assegna i valori calcolati ai parametri *$year*, *$mon* e *$mday* e li restituisce in uscita.

5.15.2 lock()

Esegue il lock esclusivo del file passato come parametro in ingresso..

IN

- **$file:** handle del file su cui eseguire il lock;

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::LOCK_EX:** flag che indica la modalità di lock esclusivo;

FUNZIONI UTILIZZATE

- **flock():** funzione predefinita Perl utilizzata per la realizzazione del lock;

DESCRIZIONE

La funzione esegue il lock del file contenuto nel parametro in ingresso *$file* chiamando la funzione predefinita *flock()* con parametro *$GlobalCheck::LOCK_EX*.

5.15.3 unlock()

Sblocca da un lock precedente su un file passato come parametro in ingresso.

IN

- **$file:** handle del file da sbloccare;

OUT

- **nessuno**

VARIABILI GLOBALI UTILIZZATE

- **$GlobalCheck::LOCK_UN:** flag che indica la modalità di 'unlock' utilizzata dalla funzione Perl predefinita *flock()*;

FUNZIONI UTILIZZATE

- **flock():** funzione predefinita Perl per utilizzata per sbloccare il file.

DESCRIZIONE

La funzione sblocca il file definito dal parametro in ingresso *$file* chiamando la funzione predefinita Perl *flock()* con parametro $*GlobalCheck::LOCK_UN*.

5.15.4 log()

Registra una stringa in un file di log eseguendo un lock sul file in modalità esclusiva.

IN

- **$log:** file in cui viene registrata la stringa;
- **$string:** stringa da registrare;

OUT

- **$error:** codice di errore:
  - ERR_LOG, se non e' stato possibile aprire il file di log;
  - ERR_NONE, altrimenti.

FUNZIONI UTILIZZATE

- **&lock():** utilizzata per eseguire il lock sul file in cui viene registrata la stringa;
- **&unlock()** utilizzata per sbloccare il file una volta registrata la stringa.

DESCRIZIONE

La funzione apre il file contenuto nel parametro in ingresso *$log*; se non è possibile aprire il file, termina restituendo in uscita il codice ERR_LOG; altrimenti, esegue il lock sul file chiamando la funzione *&lock()*, stampa la stringa *$string* e, successivamente, sblocca il file chiamando la funzione *&unlock()*; infine restituisce il codice di errore ERR_NONE.

6 Bibliografia

[MM00] M. Martinelli, *La registrazione dei Nomi a Dominio sotto il Top Level Domain .IT: L'oggetto MNTNER*, 31 marzo 2000;

[MM001] M. Martinelli, *La registrazione dei Nomi a Dominio sotto il Top Level Domain .IT: Il Modulo tecnico*, 3 aprile 2000;

[MD00] M. Martinelli, A. Del Soldato, *La registrazione dei Nomi a Dominio sotto il Top Level Domain .IT: un'interfaccia Web per la compilazione di un Modulo Tecnico,* 27 aprile 2000;

[MD99] M. Martinelli, A. Del Soldato, *La registrazione dei Nomi a Dominio sotto il Top Level Domain .IT: un'interfaccia Web per la registrazione di un Provider/Maintainer,* 29 luglio 1999;

[MRTV99] *The evolution in the management of Top Level Domains: .it as a case study*. M. Martinelli, R. Rossi, S. trumpy, D. Vannozzi, IAT-CNR, Technical Report IAT-B4-1999-005, 30 dicembre 1999.

[PTR202] *Regolamento di Assegnazione e Gestione dei nomi a dominio sotto il ccTLD "it", versione 4.0, http://www.nic.it/RA/domini/regole/regolamento.pdf*;

[RFC822] Dept. of Electrical Engineering University of Delaware, Newark, DE 19711, *Standard for the format of ARPA Internet text messages*, August 13, 1982;

[RFC1034] P. Mockapetris ISI, *Domain Name – Concepts and Facilities*, November 1987;

[RFC1035] P. Mockapetris ISI, *Domain Name – Implementation and Specification*, November 1987;

[RFC1466] E. Gerich, Merit, *Guidelines for Management of IP Address Space*, May 1993;

[RFC1517] Internet Engineering Steering Group, R. Hinden, *Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)*, September 1993;

[RFC1912] D. Barr, The Pennsylvania State University, *Common DNS Operational and Configuration Errors*, February 1996;