Gestion des NDD-TOOLKIT EPP

RTK EPP Pour Java

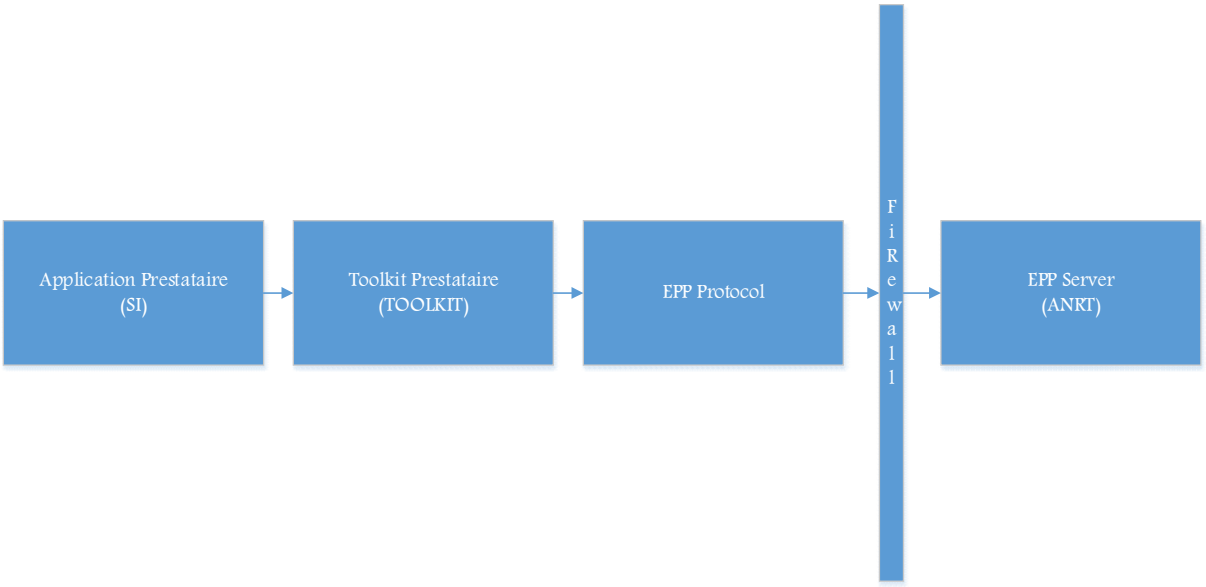
Table des matières

1. Introduction

Le Présent document fournit une description du toolkit EPP (Extensible Provisioning Protocol) qui se base sur l'API EPP RTK pour Java. La description comprend un aperçu général du protocole EPP, les prérequis nécessaires, une description générale des interfaces EPP (Interface Definition Language), les attributs nécessaires ainsi que les méthodes principales. En outre, un exemple complet d'utilisation de l'API est fourni en annexe.

2. Présentation EPP

Le protocole EPP (Extensible Provisioning Protocol) est un protocole qui fournit des informations complètes sur les objets contacts, domaines et hôtes stockés dans un référentiel central partagé. Au niveau de l'EPP, le client établit une connexion sécurisée avec le serveur puis échange les paramètres d'identification, l'authentification et toute information nécessaire à travers une série d'échanges initiée par le client.



Les commandes EPP opèrent en XML (Extensible Markup Language), un protocole qui permet aux prestataires d'utiliser plusieurs opérations en utilisant un référentiel central partagé.

EPP fournit sept services basics : hello/greeting, login, logout, poll, commands, responses et un framework extension.

Il fournit, également, sept commandes : check, info, create, update, delete, transfer, et renew. Ces commandes opèrent dans un contexte des trois objets de base dont domains, contacts et hosts (nameservers) selon le tableau ci-dessous.

Operation	Domain	Contact	Host
Create	X	X	X
Modify	X	X	X
Query	X	X	X
Delete	X	X	X
Check	X	X	X
Renew	X		
Transfer	X	X	

3. Les pré-requis de l'Api RTK EPP pour Java

Cette section décrit les packages, les APIs ainsi que les protocoles familiers nécessaires pour utiliser l'Api EPP RTK pour Java

3.1. Packages

- Java Standard Edition Software Development Kit (SDK) 1.6 : Il est utilisé pour le développement et la gestion des codes ainsi que sa compilation principalement dans un environnement de développement et de test.
- Java Runtime Environment (JRE) 1.6 ou plus : C'est l'environnement d'exécution d'un programme Java.
- Java Secure Socket Layer (SSL) : Il est utilisé pour assurer une connexion sécurisée avec le Serveur EPP.

3.2. Les packages Open

Les packages suivants sont nécessaires pour la compilation du toolkit à base du RTK EPP pour java. Les librairies comportent :

- Xerces-J 1.3.0
- Ant 1.3:

3.3. Protocoles Usuels

- Extensible Provisioning Protocol (EPP)
- Extensible Markup Language (XML)

4. Interfaces utilisées

4.1. Vue d'ensemble

L'Api EPP RTK est composé d'un ensemble d'interfaces (Interface Definition Language). Ces interfaces spécifient les données et les méthodes utilisées au niveau du protocole EPP et les interfaces nécessaires pour la communication. Elles permettent de s'assurer de l'indépendance de l'implémentation.
Ces interfaces sont compilées dans les packages suivants :

- org.openrtk.idl.epp
- org.openrtk.idl.epp.contact
- org.openrtk.idl.epp.domain
- org.openrtk.idl.epp.host

Les interfaces commencent par "epp_" et sont utilisées comme des requêtes et des réponses 'data holders'. Les classes du package com.tucows.oxrs.epp.rtk.xml, commencent par "EPP", et sont utilisés pour convertir les données en format XML lors des échanges avec le référentiel central.

Toutes les actions EPP comprennent trois classes : la première pour la demande des données (requête), la seconde pour la réponse et la troisième pour les conversions XML. Par exemple, pour l'action 'Domain Check', les trois classes sont respectivement epp_DomainCheckReq, epp_DomainCheckRsp and EPPDomainCheck.

Les attributs d'une classe epp_* sont publics et peuvent être affectés directement (exemple domain_create_req.m_name = "srs.ma"). Ces attributs ont également des méthodes 'setters' et 'getters' publiques. A travers ce document, les attributs sont énumérés et décrits. Quand nous rencontrons l'attribut "name", l'attribut au niveau de la classes est "m_name" et les deux méthodes sont "setName()" and "getName()". Pour un attribut comme "postal-code", l'attribut est m_postal_code et les deux méthodes sont "setPostalCode()" and "getPostalCode()".

4.2. Description des interfaces

4.2.1. Repository Object Identifiers (ROID)

Un ROID est un attribut qui identifie uniquement un objet à travers le référentiel. Les ROIDs sont créés pour tous les objets EPP (domain, contact ou host). Un ROID est généré en combinant un identifiant assigné par le serveur avec la valeur enregistrée qui permet une identification globale.
Les ROIDs sont formatés de la sorte : [1-80 characters] '-' [1-8 characters] (exemple ANRT01-SRS)

4.2.2. epp_Command

Cet attribut est utilisé pour stocker les données associées à toutes les demandes vers le serveur epp. Il est utilisé pour chaque commande check, info, create, update, query, transfer et renew pour les différents objets contact, domain ou host. Il est aussi utilisé pour le login et le pool.

4.2.3. epp_AuthInfo

Cet attribut est associé aux deux objets Domaine et Contact. Chacun des deux objets comporte un champ authcode qui est indépendant. Il contient l'information d'autorisation nécessaire pour autoriser les requêtes de transfert au niveau d'un serveur EPP.

- **type :** Type d'information d'autorisation
- **roid** : Facultatif. Correspond à l'identifiant de l'objet ROID.
- **valeur** : C'est la passephrase d'autorisation.

4.2.4. epp_TransferRequest

Cet attribut est utilisé pour les requêtes de transfert des objets. Il est utilisé pour chaque commande de transfert de domaine (changement prestataire) ou de contact (transfert de titulaire). Les éléments requis sont l'information d'autorisation et le type d'opération. Les attributs epp_TransferOpType permis sont : PENDING, APPROVE, CANCEL, QUERY, REJECT et REQUEST.

- **auth-info** : Il contient l'information sur l'autorisation requise pour permettre le transfert. C'est une instance de epp_AuthInfo.
- **op** : Facultatif. Il correspond au type d'opération de transfert. Il est spécifié par une valeur constante dans la classe epp_TransferOpType (exemple epp_TransferOpType.QUERY). Si cette valeur est nulle, la valeur par défaut vaut QUERY.

4.2.5. epp_TransferStatusType

Cet attribut répond avec l'état transfert de toutes les demandes de transfert. Six états sont possibles : PENDING, CLIENT_APROVED, CLIENT_CANCELLED, CLIENT_REJECTED, SERVER_APPROVED et SERVER_CANCELLED.

4.2.6. epp_Response

Cet attribut est un espace composé d'informations de réponses standard d'un serveur EPP (exemple code, message, etc.). Cet attribut est composé de :

- **résultats** : C'est un tableau des instances epp_Result.
- **qcount** : Facultatif. C'est la taille de la file d'attente au niveau du serveur. Il est présente pour une réponse de type Pool.
- **trans-id** : C'est l'identifiant retourné de la transaction (TRID), qui inclut l'identifiant client utilisé dans la requête et l'identifiant serveur.

4.2.7. epp_Result

Cet attribut est un espace composé d'informations qui constituent le résultat (exemple code, message, etc.).

- **code** : C'est le code de réponse à partir d'un serveur EPP.
- **Msg :** C'est le message de retour à partir du serveur EPP.
- **lang** : C'est le langage de la réponse.

- **Identifiant** : C'est l'identifiant du message. Il est présent pour les réponses sur une requête pool.
- **values** : Optionnel, c'est un tableau des messages descriptifs des erreurs rencontrés.

4.2.8. epp_CheckResult

Cet attribut correspond à la la réponse de la commande Check. Cet attribut est constitué de deux champs

- **valeur** : Le nom de l'objet en cours de vérification. Il doit être le ROID dans le cas de vérification d'un objet contact.
- **existe** : Indicateur booléen. Il est à "true" si l'objet existe ou à "false" si l'objet n'existe pas.

4.2.9. epp_Greeting

C'est une méthode de connexion qui statue sur l'habilité du serveur. Cette méthode liste le langage et la version supportés par le serveur. Elle liste également les services offerts par le serveur.

- **server-id** : C'est l'identifiant du serveur.
- **server-date** : Correspond à la date et heure actuelle du serveur en UTL.
- **versions** : Identifie la version supportée par le serveur.
- **langs** : Il identifie le langage de la réponse.
- **services :** Il identifie les objets que le serveur est capable de gérer.

4.2.10. epp_LoginReq

C'est une méthode qui permet la gestion des sessions et qui établit une session avec un serveur EPP.

- **cmd** : C'est la requête générique.
- **services** : Il identifie l'objet que le serveur est capable de gérer.

4.2.11. epp_LogoutReq

C'est une méthode de la gestion de session.

- **client-trid** : Facultatif. C'est l'identifiant de la transaction du client.

4.2.12. epp_PollReq

Cette méthode est utilisée pour découvrir et récupérer les messages du service client à partir du serveur EPP. Les deux opérations possibles sont : REQ (request) and ACK (acknowledge).

- **cmd** : C'est une instance de l'attribut epp_command.
- **op** : Facultatif. Correspond au type d'opération. Il est specifié par une valeur constante dans l'attribut epp_PollOpType (exemple epp_PollOpType.REQ). Cet attribut prend la valeur REQ comme valeur par défaut.

4.2.13. epp_PollResData

Cette structure contient les données associées à chaque message pool. Elle est composé de deux champs :'data' et 'type'.

4.2.14. epp_Session

Cette interface est implémentée dans la classe EPPClient, qui fournit au développeur l'accès pour utiliser les deux méthodes processAction() et processXML(). Cette interface définit la liste des codes de retour pour une session. Deux codes de retour additionnels sont déclarés et qui ne sont pas définis dans les spécifications EPP standards :

- 2600 RTK_COMMUNICATIONS_FAILURE
- 2601 RTK_UNEXPECTED_SERVER_DISCONNECT

5. Vue générale sur les classes de l'Api

5.1. Liste des classes

Les classes suivantes sont décrites dans le chapitre suivant.

- Les classes génériques
  - EPPClient
  - EPPGreeting
  - EPPLogin
  - EPPLogout
  - EPPPoll
  - RTKBase
  - EPPXMLBase
  - EPPTransport classes
- EPP Contact
  - EPPContactCheck
  - EPPContactInfo
  - EPPContactCreate
  - EPPContactUpdate
  - EPPContactDelete
  - EPPContactTransfer
- EPP Domain
  - EPPDomainCheck
  - EPPDomainInfo
  - EPPDomainCreate
  - EPPDomainUpdate
  - EPPDomainDelete
  - EPPDomainTranfer
  - EPPDomainRenew
- EPP Host
  - EPPHostCheck
  - EPPHostInfo
  - EPPHostCreate
  - EPPHostUpdate
  - EPPHostDelete

5.2. Présentation des objets

5.2.1. Object Domaine

Attribut	Type de données	Source
Domain Name	Chaine de caractère(64) + TLD/SLD suffix	Prestataire
ROID	Chaine de caractère(89)	SRS
Status	Chaine de caractère(50)	Prestataire or SRS

Sponsoring Prestataire	Chaine de caractère(16)	SRS
Registrant	Chaine de caractère(16)	Prestataire
Expiration Date	Date	SRS
Last Transfer Date	Date	SRS
Authorization Information (Password)	Chaine de caractère(16)	Prestataire
Technical Contact	Chaine de caractère(16)	Prestataire
Admin Contact	Chaine de caractère(16)	Prestataire
Billing Contact	Chaine de caractère(16)	Prestataire
Registration Date	Date	SRS
Registration Period	Numérique	Prestataire
Created Date	Date	SRS
Created By	Chaine de caractère(16)	SRS
Updated Date	Date	SRS
Updated By	Chaine de caractère(16)	SRS
Last Transfer Date	Date	SRS

5.2.2. Objet Contact

Attribut	Type de donnée	Source
ID	Chaine de caractère (16)	Prestataire
ROID	Chaine de caractère (89)	SRS
Status	Chaine de caractère (50)	Prestataire or SRS
Name	Chaine de caractère (255)	Prestataire
Organization	Chaine de caractère (255)	Prestataire
Address1	Chaine de caractère (255)	Prestataire
Address2	Chaine de caractère (255)	Prestataire
Address3	Chaine de caractère (255)	Prestataire
City	Chaine de caractère (255)	Prestataire
State/Province	Chaine de caractère (255)	Prestataire
Postal Code	Chaine de caractère (16)	Prestataire

Country Code	Chaine de caractère (2)	Prestataire
I15d Name	Chaine de caractère (255)	Prestataire
I15d Organization	Chaine de caractère (255)	Prestataire
I15d Address 1	Chaine de caractère (255)	Prestataire
I15d Address 2	Chaine de caractère (255)	Prestataire
I15d Address 3	Chaine de caractère (255)	Prestataire
I15d City	Chaine de caractère (255)	Prestataire
I15d State/Province	Chaine de caractère (255)	Prestataire
I15d Postal Code	Chaine de caractère (16)	Prestataire
I15d Country	Chaine de caractère (2)	Prestataire
Telephone Number	Chaine de caractère (17)	Prestataire
Telephone Extension	Chaine de caractère (10)	Prestataire
Fax Number	Chaine de caractère (17)	Prestataire
Fax Extension	Chaine de caractère (10)	Prestataire
Email Address	Chaine de caractère (80)	Prestataire
Sponsoring Prestataire	Chaine de caractère (16)	SRS
Authorization Information (type & roid)	Type – Chaine de caractère Roid _ Chaine de caractère (89)	Prestataire
Created Date	Date	SRS
Created By	Chaine de caractère(16)	SRS
Updated Date	Date	SRS
Updated By	Chaine de caractère(16)	SRS
Last Transfer Date	Date	SRS

5.2.3. Objet Host

Attribut	Type de données	Source
Host Name	Chaine de caractère(255)	Prestataire

ROID	Chaine de caractère(89)	SRS
Status	Chaine de caractère(50)	Prestataire or SRS
IP Address (Type & Address)	Type – Chaine de caractère(2) Address- Chaine de caractère(45)	Prestataire
Sponsoring Prestataire	Chaine de caractère(16)	SRS
Created Date	Date	SRS
Created By	Chaine de caractère(16)	SRS
Updated Date	Date	SRS
Updated By	Chaine de caractère(16)	SRS
Last Transfer Date	Date	SRS

6. Description détaillée des classes

Pour développer un client EPP, il faut suivre les étapes suivantes :

- Initialiser une connexion vers le serveur (EPP Session)
- Etablir un canal de communication entre le client et le serveur (EPP Channel)
- Etablir une session avec le serveur (EPP Login)
- Envoyer une requête au serveur (EPP Request). Cette requete peut être unitaire ou composer de plusieurs.
- Traiter la réponse (EPP Response)
- Fermer la session
- Fermer le canal de communication
- Terminer la connexion avec le serveur.

Cette section décrit l'ensemble des classes listées dans le chapitre précèdent et qui sont nécessaire au développement d'un client EPP. Ces classes sont utilisées pour fournir une relation entre le client et le serveur.

6.1. Gestion des sessions

6.1.1. EPPClient

Description

Cette classe encapsule toutes les connexions et les communications avec le serveur. Il permet d'établir une connexin ssl avec le serveur sur le port 700. Le choix de la langue doit être fixé avant d'établir une connexion. L'api supporte seulement la langue 'en'. Les méthodes principales sont décrites ci-après :

- connect() ,
- connectAndGetGreeting() :
- login(Chaine de caractère)
- logout(Chaine de caractère)
- disconnect () :
- setLang(Chaine de caractère)
- processAction(epp_Action)
- processXML(Chaine de caractère)
- setSocketToEPPServer(Socket)
- setTransport(EPPTransportBase) setVersionSentOnLogin(boolean) isVersionSentOnLogin()
- hello()

Usage Syntax

```
EPPClient epp_client = new EPPClient(epp_host_name,
                                     epp_host_port,
                                     epp_client_id,
                                     epp_password);
```

```
epp_client.setLang("en");
// Establish an SSL-based socket with a third-party package.
// Completed establishing a connection. At this point, the EPP Server has sent a greeting for the Prestataire to pick up.

epp_client.setSocketToEPPServer(the_socket);
epp_Greeting greeting = epp_client.getGreeting();

// Now the EPP Server is ready to proceed with login.
```

6.1.2. EPPGreeting

Description

Cette classe représente un retour envoyé du serveur. L'instanciantion de cette méthode est récupérée des trois méthodes getGreeting(), connectAndGetGreeting() or hello() de la classe EPPClient.

6.1.3. EPPLogin

Description

Cette classe permet au prestataire d'établir une session avec un serveur EPP après avoir reçu un greeting du serveur.

6.1.4. EPPLogout

Description

Cette classe est utilisée pour fermer une session avec un serveur EPP.

6.2. Gestion des échanges

6.2.1. EPPPoll

Description

Cette méthode est utilisée pour découvrir et récupérer les messages en cours de traitement à partir d'un serveur. Les prestataires doivent exécuter des poll périodiques à partir du serveur. Cette action a deux objetifs : maintenir une session avec le serveur et avoir les notifications en attente

Usage Syntax

```
epp_PollReq poll_request = new epp_PollReq();
epp_Command command_data = new epp_Command();
command_data.setClientTrid( client_trid );

poll_request.setCmd( command_data );

// Set the Poll "op" to REQ (ie request).  query the message queue for the first message available.
// Note that the REQ op does not dequeue the message. The ACK op is required to acknowledge messages and  remove them from the waiting queue.

// Note that the Poll op type of REQ is the default, so it could have been left null here.
poll_request.setOp( epp_PollOpType.REQ );

EPPPoll poll = new EPPPoll();
poll.setRequestData(poll_request);
```

```
// Now ask the EPPClient to process the request and retrieve a response from the server.
poll = (EPPPoll) epp_client.processAction(poll);

// or, alternatively, this method can be used...
//poll.fromXML(epp_client.processXML(poll.toXML()));

epp_PollRsp poll_response = poll.getResponseData();
epp_Response response = poll_response.getRsp();

// This is the number of messages in the SRS  queue, including the currently REQ'd message short
message_count = response.getQcount();

epp_Result[] results = response.getResults();

// In a subsequent Poll ACK, this is the message ID that would be used.
String message_id = results[0].getId();
```

6.2.2. RTKBase

Description

Cette méthode abstraite est la base du toolkit abstrait (RTK). L'ensemble des classes hérite à partir de celle-ci. Cette classe définit les méthodes de debug, les contraintes et le format de la date

Static Member Summary

- **DateFormat UTC_FMT = new DateFormat("yyyy-MM-dd'T'hh:mm:ss.0'Z'")**
- **DateFormat DATE_FMT = new DateFormat("yyyy-MM-dd")**

Static Method Summary

- **setDebugLevel(int)** : Quatre paramètres de debug DEBUG_NONE, DEBUG_LEVEL_ONE, DEBUG_LEVEL_TWO, DEBUG_LEVEL_THREE.

Usage syntax

```
import com.oxrs.epp.rtk.RTKBase;

// Setting debug level to ONE to see exception reporting in
// System.err.
RTKBase.setDebugLevel(RTKBase.DEBUG_LEVEL_ONE);
```

6.2.3. EPPXMLBase

Description

Cette classe abstraite est la classe de l'analyse XML

Static Method Summary

- Object[] convertListToArray(java.lang.Class, java.util.List)

Usage Syntax

```
List ip_list = (List)new ArrayList();
ip_list.add(new epp_HostAddress(epp_HostAddressType.IPV4, "100.103.44.151"));
ip_list.add(new epp_HostAddress(epp_HostAddressType.IPV4, "100.103.45.151"));
host_create_request.setAddresses( (epp_HostAddress[])
        EPPXMLBase.convertListToArray((new epp_HostAddress()).getClass(),
                                ip_list) );
```

6.2.4. EPPTransport Classes

Description
L'ensemble des classes qui commence par "EPPTransport" permet la gestion des couches. La classe principale est EPPTransportBase. Les sous-classes incluent :

- EPPTransportTCP
- EPPTransportTCPTLS
- EPPTransportSMTP

Les méthodes de la classes EPPTransportBase sont

- setEPPHostName(Chaine de caractère)
- getEPPHostName()
- setEPPHostPort(int)
- getEPPHostPort()
- setTimeout(int)
- getTimeout()
- initialize(Chaine de caractère,int,int)
- connect()
- disconnect ()
- writeToServer(Chaine de caractère)
- Chaine de caractère readFromServer()

6.3. EPP Domain

Un domaine représente un espace de nom dans internet. Les domaines peuvent avoir des contacts et des hôtes associés. Le domaine peut avoir des champs qui peuvent être consultés ou modifies par un prestataire. Les commandes possibles sur un domaine sont check, info, create, update delete, transfer and renew.

6.3.1. Les données communes

epp_DomainPeriodUnitType : ANNEE . L'implementation doit fixer une périodicité annuelle.

epp_DomainPeriod : une structure composée de

- **unit**
- **value**

epp_DomainContactType ADMIN, TECH

epp_DomainContact une strcture composé de

- **type**
- **Identifiant**

epp_DomainStatusType : CLIENT_DELETE_PROHIBITED, CLIENT_HOLD, CLIENT_RENEW_PROHIBITED, CLIENT_TRANSFER_PROHIBITED, CLIENT_UPDATE_PROHIBITED, INACTIVE, ACTIVE , PENDING_DELETE, PENDING_TRANSFER, PENDING_VERIFICATION, SERVER_DELETE_PROHIBITED, SERVER_HOLD, SERVER_RENEW_PROHIBITED, SERVER_TRANSFER_PROHIBITED, SERVER_UPDATE_PROHIBITED

epp_DomainStatus une strcture

- **type**
- **lang**
- **value**

6.3.2. EPPDomainCreate

Description
Cette action permet la création d'un nouveau domaine.

Attributs requis

- Domain Name
- Admin Contact ID
- Technical Contact ID
- Registrant ID
- Name Servers

Attributs optionnel

- Registration Period
- Billing Contact ID
- Registrar name/value pairs

Attributs implicites

- Status
- Expiration Date
- Registration Date
- Sponsoring Registrar
- Created By
- Created Date
- Updated By
- Updated Date

Données de retour

- Domain Name
- Expiration Date

Usage syntax

```
epp_DomainCreateReq domain_create_request = new epp_DomainCreateReq();
command_data = new epp_Command();

// The client trid is optional. It's main use is for Prestataire tracking and logging of requests, especially for data creation or modification requests.
command_data.setClientTrid( client_trid;
domain_create_request.setCmd( command_data );
domain_create_request.setName( "domain.info" );
// The domain's period is optional. It is specified with an object that contains the unit of measurement (years or
```

```
// months) and the actual period value.
epp_DomainPeriod period = new epp_DomainPeriod();
period.setUnit( epp_DomainPeriodUnitType.YEAR );
period.setValue( 2 );
domain_create_request.setPeriod( period );

// From an EPP perspective, nameserver associations are optional for a domain, so we're not
specifying them
 / / Ici. We will add them later in the domain update.

EPPDomainCreate domain_create = new EPPDomainCreate();
domain_create.setRequestData(domain_create_request);

// Now ask the EPPClient to process the request and retrieve a response from the server.
domain_create = (EPPDomainCreate) epp_client.processAction(domain_create);

// or, alternatively, this method can be used...
//domain_create.fromXML(epp_client.processXML(domain_create.toXML()));

epp_DomainCreateRsp domain_create_response = domain_create.getResponseData();
epp_Response response = domain_create_response.getRsp();

epp_Result[] results = response.getResults();

// The domain's ROID and expiration date are returned on a successful domain creation.
String domain_roid = domain_create_response.getRoid();

// Save the expiration date for a renew command later
Date domain_exp_date = RTKBase.UTC_FMT.parse(domain_create_response.getExpirationDate());
```

6.3.3. EPPDomainCheck

Description

Cette action permet de vérifier la présence ou non d'un domaine.

Attributs requis

- Domain Name

Données de retour

- Vrai ou faux selon la présence ou non du domaine

Usage syntax

```
epp_DomainCheckReq domain_check_request = new epp_DomainCheckReq();

command_data = new epp_Command();

// The client trid is optional. It's main use is for Prestataire tracking and logging of requests, especially
for data creation or modification requests.
command_data.setClientTrid( client_trid );
domain_check_request.setCmd( command_data );

// The Domain Check request can accept an array of domain names. In this example, an ArrayList is
used to dynamically create the List of domain names and then EPPXMLBase's utility method
convertListToChaine de caractèreArray() is used to convert the List to a Chaine de caractère array.
List domain_list = (List)new ArrayList();
domain_list.add("domain1.info");
```

```
domain_list.add("domain2.info");
domain_check_request.setNames( EPPXMLBase.convertListToChaine de
caractèreArray(domain_list) );

EPPDomainCheck domain_check = new EPPDomainCheck();
domain_check.setRequestData(domain_check_request);

// Now ask the EPPClient to process the request and retrieve a response from the server.
domain_check = (EPPDomainCheck) epp_client.processAction(domain_check);

// or, alternatively, this method can be used...
//domain_check.fromXML(epp_client.processXML(domain_check.toXML()));

epp_DomainCheckRsp domain_check_response = domain_check.getResponseData();
epp_Response response = domain_check_response.getRsp();

epp_CheckResult[] check_results = domain_check_response.getResults();

Boolean exists = EPPXMLBase.getCheckResultFor(check_results,
                                    "domain.info");
```

6.3.4. EPPDomainInfo

Description

Cette action permet de récupérer les informations associées à un domaine.

Attributs requis

- Domain Name

Données de retour

- Tous les attributs de l'objet domaine

Usage syntax

```
epp_DomainInfoReq domain_info_request = new epp_DomainInfoReq();
command_data = new epp_Command();

// The client trid is optional. It's main use is for Prestataire tracking and logging of requests, especially
for data creation or modification requests.
command_data.setClientTrid( client_trid );
domain_info_request.setCmd( command_data );

// The only domain-specific parameter is the domain name itself.
domain_info_request.setName( "domain.info" );

EPPDomainInfo domain_info = new EPPDomainInfo();
domain_info.setRequestData(domain_info_request);

// Now ask the EPPClient to process the request and retrieve a response from the server.
domain_info = (EPPDomainInfo) epp_client.processAction(domain_info);

// or, alternatively, this method can be used...
//domain_info.fromXML(epp_client.processXML(domain_info.toXML()));

epp_DomainInfoRsp domain_info_response = domain_info.getResponseData();
epp_Response response = domain_info_response.getRsp();
```

```
// You can also save the auth info from an info where the calling Prestataire is the sponsoring client for the
// object.
epp_AuthInfo domain_auth_info = domain_info_response.getAuthInfo();

// The Info command returns some standard information like the current sponsoring client id, the creator client id, the create time and the last update time.
// For a Domain Info, the domain's nameservers, hosts, last transfer client id, last transfer date, and expiration date are returned.
Chaine de caractère client_id = domain_info_response.getClientId();
Chaine de caractère created_by = domain_info_response.getCreatedBy();
Chaine de caractère create_data_Chaine de caractère = domain_info_response.getCreatedDate();

// Save the expiration date for the renew command later.
Date domain_exp_date =
                        RTKBase.UTC_FMT.parse(domain_info_response.getExpirationDate());
```

6.3.5. EPPDomainUpdate

Description

Cette action permet la modification d'un domaine existant. Seul les deux contacts peuvent être modifié, le changement de prestataire ou de titulaire doit passer par une opération de changement de prestataire ou de transfert de domaine. Le workflow associé est présenté en annexe 4.

Attributs requis

- Domain Name

Attributs optionnels

- Contact Admin
- Contact Technique

Données de retour

- Aucun résultat

Usage syntax

```
epp_DomainUpdateReq domain_update_request = new epp_DomainUpdateReq();

command_data = new epp_Command();

// The client trid is optional. It's main use is for Prestataire tracking and logging of requests, especially for data creation or modification requests.
command_data.setClientTrid( client_trid;
domain_update_request.setCmd( command_data );
domain_update_request.setName( "domain.info" );

epp_DomainUpdateAddRemove add = new epp_DomainUpdateAddRemove();

// Here is a list of nameservers to add to the domain. An array is expected here, but to avoid using Java native
// arrays, we're using an ArrayList and then converting it to a String array.
List name_server_list = (List)new ArrayList();
name_server_list.add("new-ns1.domain.info");
name_server_list.add("new-ns2.domain.info");
```

```
add.setNameServers( EPPXMLBase.convertListToChaine de caractèreArray(name_server_list) );

// We also want to add the CLIENT-HOLD status to the domain. This time we'll create an array of
epp_DomainStatus. EPP allows for Prestataire notes in the status field. EPP even allows for a
language specifier.
epp_DomainStatus status = new epp_DomainStatus[1];
status[0] = new epp_DomainStatus();
status[0].setType( epp_DomainStatusType.CLIENT_HOLD );
status[0].setLang( "fr" );
status[0].setValue( "Le client n'as pas envoyé de l'argent" );

add.setStatus( status );

// Set the add information.
domain_update_request.setAdd( add );

epp_DomainUpdateAddRemove remove = new epp_DomainUpdateAddRemove();

// Now to remove 1 nameserver.
name_server_list = (List)new ArrayList();
name_server_list.add("old-ns1.domain.info");
remove.setNameServers( EPPXMLBase.convertListToChaine de caractèreArray(name_server_list) );

// Set the remove information.
domain_update_request.setRemove( remove );

// We're also specifying the information to change. Only the registrant information can be changed.
epp_DomainUpdateChange change = new epp_DomainUpdateChange();
change.setRegistrant( "contact1" );
domain_update_request.setChange( new epp_DomainUpdateChange() );

EPPDomainUpdate domain_update = new EPPDomainUpdate();
domain_update.setRequestData(domain_update_request);

// Now ask the EPPClient to process the request and retrieve a response from the server.
domain_update = (EPPDomainUpdate) epp_client.processAction(domain_update);

// or, alternatively, this method can be used...
//domain_update.fromXML(epp_client.processXML(domain_update.toXML()));

epp_DomainUpdateRsp domain_update_response = domain_update.getResponseData();
epp_Response response = domain_update_response.getRsp();
```

6.3.6. EPPDomainDelete

Description

Cette action permet la résiliation d’un domaine.

Attributs requis

- Domain Name

Données de retour

- Aucun retour

Syntaxe d'utilisation

```
epp_DomainDeleteReq domain_delete_request = new epp_DomainDeleteReq ();
```

```
command_data = new epp_Command ();
/ / Le tridéo client est facultative.  Son utilisation principale
/ / Est pour le suivi des registres et l'enregistrement des requêtes,
/ / En particulier pour la création ou la modification de données des demandes.
command_data.setClientTrid (client_trid);
domain_delete_request.setCmd (command_data);
domain_delete_request.setName ("domain.info");

EPPDomainDelete domain_delete = new EPPDomainDelete ();
domain_delete.setRequestData (domain_delete_request);

/ / Maintenant, demandez le EPPClient pour traiter la demande et récupérer une  réponse du serveur.
domain_delete = (EPPDomainDelete) epp_client.processAction (domain_delete);

/ / Ou, en variante, ce procédé peut être utilisé ...
/ / Domain_delete.fromXML (epp_client.processXML (domain_delete.toXML ()));
```

6.3.7. EPPDomainTransfer

Description

Cette action permet de transférer un domaine vers un nouveau prestataire. Cinq opérations de transfert sont autorisées : approuver, annuler, requête, rejeter et demande. Si aucun n'est spécifié, le RTK par défaut à requête.

Six états de transfert existent : en attente, autorisé annulé, rejeté, auto-approuvé, auto-annulation.

Une requête de transfert permet au prestataire d'interroger l'état d'un transfert en attente actuelle ou de la dernière approuvé, d'annulation ou de transfert rejetée. Une requête de transfert sur un objet qui n'a jamais encore été transferé va provoquer une levée de l'exception par processAction de EPPClient () ou fromXML de EPPDomainTranfer () car le code d'erreur étant renvoyé par le registre.

Une demande de transfert déclenche un processus de transfert dans le registre. Le serveur notifie le prestataire sortant du transfert en attente. Le prestataire sortant a alors la possibilité d'approuver ou de rejeter explicitement le transfert. Le prestataire entrant a également la possibilité d'annuler le transfert avant qu'il ne soit validé.

Attributs requis

- Domain Name
- Operation Type (REQUEST, REJECT, QUERY, APPROVE, CANCEL)

Attributs optionnels

- Registration Period (pour les requêtes de type ‘transfer request’)

Données de retour

- Domain name
- transfer Status
- Requested Registrar id
- Request date

- Sponsoring Registrar id
- Action Requested date from Sponsoring Registrar (action not necessary for Neustar SRS)
- Domain expiration date

Syntaxe d'utilisation

```
epp_DomainTransferReq domain_transfer_request = new epp_DomainTransferReq ();

command_data = new epp_Command ();

/ / Le tridéo client est facultative.  Son utilisation principale est pour le suivi des registres et
l'enregistrement des requêtes,  En particulier pour la création ou la modification de données des
demandes.
command_data.setClientTrid (client_trid);
domain_transfer_request.setCmd (command_data);

/ / La demande de transfert de domaine est un «transfert» demande PPE,
/ / Ce qui signifie qu'il nécessite une valeur "op" et l'objet de
/ / Informations auth actuelle pour le traitement réussi.
epp_TransferRequest transfer_request = new epp_TransferRequest ();

/ / Une requête de transfert ne interroger l'état du courant
/ / En attente de transfert ou de la dernière complété / annulation / rejeté
/ / Transfert.  Pour demander un transfert, le "op" doit être réglé sur
/ / DEMANDE.
/ /
/ / QUERY est le type de transfert de op par défaut, de sorte qu'il
/ / Aurait pu être laissé nulle ici.
transfer_request.setOp (epp_TransferOpType.QUERY);
/ /
transfer_request.setAuthInfo (domain_auth_info);
domain_transfer_request.setTrans (transfer_request);

domain_transfer_request.setName ("domain.info");

/ / À la demande de transfert, PPE permet un renouvellement de domaine Pour être regroupés dans
un transfert de domaine.  Si la période n'est pas spécifié, 1 année est la valeur par défaut.
/ / Donc, si c'était une demande de transfert, nous pourrions faire cela.
/ *
epp_DomainPeriod période = new epp_DomainPeriod ();
period.setUnit (epp_DomainPeriodUnitType.YEAR);
period.setValue ((court) 2);
domain_transfer_request.setPeriod (période);
* /

EPPDomainTransfer domain_transfer = new EPPDomainTransfer ();
domain_transfer.setRequestData (domain_transfer_request);

/ / Maintenant, demandez le EPPClient pour traiter la demande et récupérer une réponse du serveur.
domain_transfer = (EPPDomainTransfer) epp_client.processAction (domain_transfer);

/ / Ou, en variante, ce procédé peut être utilisé ...
/ / Domain_transfer.fromXML (epp_client.processXML (domain_transfer.toXML ()));

epp_DomainTransferRsp domain_transfer_response = domain_transfer.getResponseData ();
```

```
réponse de epp_Response = domain_transfer_response.getRsp ();

Chaîne transfer_status = EPPXMLBase.transferStatusToChaine de caractère
(domain_transfer_response.getTransferStatus ());
```

6.3.8. EPPDomainRenew

Description

Cette action permet à un prestataire de prolonger la période de validité d'un domaine.

Attributs requis

- Domain Name
- Expiration Date – Current expiration date of the domain
- Registration Renewal Period (optional). Le nombre maximal d'année est 5 ans. Si la période n'est pas enregistré, elle est par défaut à une année.
- Expiration year – L'année d'expiration avant renouvellement.

Données de retour

- Aucun retour

syntaxe d'utilisation

```
epp_DomainRenewReq domain_renew_request = new epp_DomainRenewReq ();
command_data = new epp_Command ();

/ / Le identifiant client est facultative. Son utilisation principale
/ / Est pour le suivi des registres et l'enregistrement des requêtes,
/ / En particulier pour la création ou la modification de données des demandes.
command_data.setClientTrid (client_trid);
domain_renew_request.setCmd (command_data);
domain_renew_request.setName ("domain.info");

/ / Comme dans le domaine de créer le fonctionnement, la période de domaine
/ / Peut être précisée ici aussi.
epp_DomainPeriod période = new epp_DomainPeriod ();
period.setUnit (epp_DomainPeriodUnitType.YEAR);
period.setValue ((court) 2);
domain_renew_request.setPeriod (période);

/ / Expiration actuelle du domaine doit également être spécifié
/ / Pour éviter multiples involontaire renouveler la demande de réussir.
/ / Le format de la date d'expiration doit être «aaaa-mm-jj"
/ / Ici, domain_exp_date est une instance de java.util.Date.
domain_renew_request.setCurrentExpriationDate (RTKBase.DATE_FMT.format (domain_exp_date));

EPPDomainRenew domain_renew = new EPPDomainRenew ();
domain_renew.setRequestData (domain_renew_request);

/ / Maintenant, demandez le EPPClient pour traiter la demande et récupérer
/ / Une réponse du serveur.
domain_renew = (EPPDomainRenew) epp_client.processAction (domain_renew);

/ / Ou, en variante, ce procédé peut être utilisé ...
/ / Domain_renew.fromXML (epp_client.processXML (domain_renew.toXML ()));
```

epp_DomainRenewRsp domain_renew_response = domain_renew.getResponseData ();
réponse de epp_Response = domain_renew_response.getRsp ();

6.4. EPP Contact

Un contact EPP représente l'information d'un titulaire (qu'il soit personne morale ou physique), d'un contact adminitratif ou d'un contact technique. Un objet contact peut être consulté ou modifié par un prestataire. Un contact peut être associé à un domaine. Les commandes possibles sont check, info, create, update, delete and transfer. Cette dernière commande ne pas va être utilisé, comme elle ne correspond pas aux exigence du projet .ma

6.4.1. Les attributs commun d'un contact

- **epp_ContactNameAddress** : contient l'information sur une adresse complète pour un contact. Cette information est codée en ASCII (7-bit ASCII as defined in [US-ASCII]) ou UTF-8 data (UTF-8 as defined in [RFC2279]).
- **nom** : contient le nom ou le rôle exercé par le contact
- **Org** : contient le nom de l'organisation
- **Address** : C'est une structure eppcontact adress
- **epp_ContactAddress** : une structure pour matérialiser l'information d'une adresse postale
    - ***street1***
    - ***street2***
    - ***street3***
    - ***city***
    - ***state-province***
    - ***postal-code***
    - ***country-code***
- **epp_ContactPhone** : une strcture réservée pour construire un numéro de telephone. Cette strcture est utilisée pour les téléphones et les fax
    - ***extension***
    - ***value***
- **epp_ContactStatusType** : Les états possibles sont SERVER_DELETE_PROHIBITED, CLIENT_DELETE_PROHIBITED, SERVER_TRANSFER_PROHIBITED, CLIENT_TRANSFER_PROHIBITED, SERVER_UPDATE_PROHIBITED, CLIENT_UPDATE_PROHIBITED, LINKED, OK, PENDING_DELETE, and PENDING_TRANSFER.
- **epp_ContactStatus** : cette structure est composée des informations suivantes :
    - ***type***
    - ***lang***
    - ***value***

6.4.2. EPPContactCreate

Description
Cette action permet à un prestataire de créer un nouveau contact.

Attributs requis

- Name
- Organization
- Address1
- City
- Country
- Email
- Phone
- NID

Attributs Optionnels

- Address2
- Address3
- State/Province
- Postal Code
- Fax
- Registrar name/value pairs

Attributs implicites

- Status
- Sponsoring Registrar
- Created By
- Created Date
- Updated By
- Updated Date

Données de retour

- Contact ID

Les valeurs seront validées par le serveur. Il est recommandé qu'une vérification supplémentaire et un formatage se fait aussi au niveau du client

Usage syntax

```
epp_ContactCreateReq contact_create_request = new epp_ContactCreateReq();

command_data = new epp_Command();

// The client trid is optional. It's main use is for Prestataire tracking and logging of requests, especially for data creation or modification requests.
command_data.setClientTrid( client_trid );
contact_create_request.setCmd( command_data );

// The ID used here should have been shown as available in a contact:check issued immediately before the create.
contact_create_request.setId( "contact2953" );
```

```
epp_ContactNameAddress name_address = new epp_ContactNameAddress();
name_address.setName( "John Doe" );
name_address.setOrg( "ACME Solutions" );

epp_ContactAddress address = new epp_ContactAddress();

// Up to three street values may be specified.
address.setStreet1( "100 Centre St" );
address.setCity( "Townsville" );
address.setStateProvince( "County Derry" );
address.setPostalCode( "Z1Z1Z1" );

// The country code must be a valid ISO country code.
address.setCountryCode( "CA" );

name_address.setAddress( address );

// Now assign that name/address set to the contact's ASCII address. If you wish to also use UTF-8
characters in the name/address data, set the i15d-address data member.
contact_create_request.setAsciiAddress( name_address );

// The voice, fax and the email values can only be in ASCII.
// The voice value may contain an optional extension.

// Ceate a voice value with an extenstion.
contact_create_request.setVoice( new epp_ContactPhone("1234", "+1.4165559999") );
contact_create_request.setFax( new epp_ContactPhone(null, "+1.4165558888") );
contact_create_request.setEmail( "jdoe@acmesolutions.info" );

EPPContactCreate contact_create = new EPPContactCreate();
contact_create.setRequestData(contact_create_request);

contact_create = (EPPContactCreate) epp_client.processAction(contact_create);

// or, alternatively, this method can be used...
//contact_create.fromXML(epp_client.processXML(contact_create.toXML()));

epp_ContactCreateRsp contact_create_response = contact_create.getResponseData();
epp_Response response = contact_create_response.getRsp();
```

La requête XML de création d'un contact avec un NID est la suivant

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
   <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
        <command>
            <create>
                  <contact:create
       xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
                        <contact:id>123456t</contact:id>
                        <contact:postalInfo type="int">
                              <contact:name>Test101</contact:name>
                              <contact:org>Test101</contact:org>
                              <contact:addr>
                                    <contact:street>Nairobi</contact:street>
                                    <contact:street>Example</contact:street>
                                    <contact:city>Example</contact:city>
                                    <contact:sp>Example</contact:sp>
```

```
                        <contact:pc>00100</contact:pc>
                        <contact:cc>KE</contact:cc>
                    </contact:addr>
                </contact:postalInfo>
                <contact:voice x="">254.713445690</contact:voice>
                <contact:fax></contact:fax>
                <contact:email>test@example.com</contact:email>
                <contact:authInfo>
                    <contact:pw>CoCCA12345</contact:pw>
                </contact:authInfo>
            </contact:create>
        </create>
    <extension>
     <contact:create xmlns:contact="urn:ietf:params:xml:ns:contact-id-1.0">
       <contact:person>
         <contact:NID>44589329</contact:NID>
       </contact:person>
       </contact:create>
    </extension>

       <clTRID>1999357013978501676</clTRID>
  </command>
</epp>
```

6.4.3. EPPContactCheck

Description

Cette action est utilisée pour déterminer si un ensemble de contacts sont connus par le serveur. Le resultat de cette requête est un tableau d'epp_CheckResult.
Attributs requis

- Contact Id

Données de retour

- Vrai ou faux selon la présence ou non d'un contact

Usage syntax

```
epp_ContactCheckReq contact_check_request = new epp_ContactCheckReq();
command_data = new epp_Command();

// The client trid is optional. It's main use is for Prestataire tracking and logging of requests, especially
for data creation or modification requests.
command_data.setClientTrid( client_trid );
contact_check_request.setCmd( command_data );

// The Contact Check request can accept an array of contact IDs. In this example, an ArrayList is used
to dynamically create the List of IDs and then EPPXMLBase's utility method convertListToChaine de
caractèreArray() is used to convert the List to a Chaine de caractère array.
List contact_list = (List)new ArrayList();
contact_list.add("contact1");
contact_list.add("contact2");
contact_list.add("contact3");
contact_check_request.setIds( EPPXMLBase.convertListToChaine de caractèreArray(contact_list) );
```

```
EPPContactCheck contact_check = new EPPContactCheck();
contact_check.setRequestData(contact_check_request);

contact_check = (EPPContactCheck) epp_client.processAction(contact_check);

// or, alternatively, this method can be used...
//contact_check.fromXML(epp_client.processXML(contact_check.toXML()));

epp_ContactCheckRsp contact_check_response = contact_check.getResponseData();
epp_Response response = contact_check_response.getRsp();
epp_Result[] results = response.getResults();

epp_CheckResult[] check_results = contact_check_response.getResults();

Boolean exists = EPPXMLBase.getCheckResultFor(check_results,
                                    "contact1");
```

6.4.4. EPPContactInfo

Description

Cette action est utilisée pour récupérer l'information détaillée associée à un contact.

Attributs requis

- Contact Id

Données de retour

- Tous les attributs d'un contact.

Usage syntax

```
epp_ContactInfoReq contact_info_request = new epp_ContactInfoReq();

command_data = new epp_Command();

// The client trid is optional. It's main use is for Prestataire tracking and logging of requests, especially
for data creation or modification requests.
command_data.m_client_trid = client_trid;
contact_info_request.setCmd( command_data );
contact_info_request.setId( "contact1" );
EPPContactInfo contact_info = new EPPContactInfo();
contact_info.setRequestData(contact_info_request);

contact_info = (EPPContactInfo) epp_client.processAction(contact_info);

// or, alternatively, this method can be used...
//contact_info.fromXML(epp_client.processXML(contact_info.toXML()));

epp_ContactInfoRsp contact_info_response = contact_info.getResponseData();
epp_Response response = contact_info_response.getRsp();
epp_Result[] results = response.getResults();

String client_id = contact_info_response.getClientId();
String created_by = contact_info_response.getCreatedBy();
String create_data_Chaine de caractère = contact_info_response.getCreatedDate();
```

```
// You can also save the auth info from an info where the calling Prestataire is the sponsoring client for the
// object.
epp_AuthInfo contact_auth_info = contact_info_response.getAuthInfo();
```

6.4.5. EPPContactUpdate

Description

Cette action permet à un prestataire de modifier les champs associés à un contact. Le nom et le prénom ne doivent pas être modifié dans le cadre de cette commande selon les exigences du projet .ma

Attributs requis

- Contact Id

Attributs optionnels

- Liste des attributs à modifier (voir attributs pour créer un contact)

Données de retour

- aucun

Usage syntax

```
epp_ContactUpdateReq contact_update_request = new epp_ContactUpdateReq();

command_data = new epp_Command();

// The client trid is optional. It's main use is for Prestataire tracking and logging of requests, especially for data creation or modification requests.
command_data.setClientTrid( client_trid );
contact_update_request.setCmd( command_data );
contact_update_request.setId( "contact2" );

epp_ContactUpdateChange contact_update_change = new epp_ContactUpdateChange();
epp_ContactNameAddress name_address = new epp_ContactNameAddress();

// In a contact change request, you only need to explicitly specify the values that are changing, leaving the remain values set to null (although, there is no harm in sending the complete contact information). If an optional value is  to be removed, then specify an empty Chaine de caractère for the new value.
name_address.setName( "Jane Doe" );
name_address.setOrg( "ACME Systems" );
epp_ContactAddress address = new epp_ContactAddress();

// Up to three street values may be specified.
address.setStreet1( "999 Front St" );
// The postal is optional, so could have been set to "" here.
address.setPostalCode( "A9A9A9" );

name_address.setAddres( address );

// The contact update operation allows changes to both the ASCII and "i15d" name/address information.
contact_update_change.setAsciiAddress( name_address );
```

```
// The voice and email values are not changing, but we want to remove the fax value. Can't just use
null here  because then it would not be present in the request XML, so the server would affect no
change to the fax
contact_update_change.setFax( new epp_ContactPhone("", "") );
contact_update_request.setChange( contact_update_change );
EPPContactUpdate contact_update = new EPPContactUpdate();
contact_update.setRequestData(contact_update_request);
contact_update = (EPPContactUpdate) epp_client.processAction(contact_update);

// or, alternatively, this method can be used...
//contact_update.fromXML(epp_client.processXML(contact_update.toXML()));

epp_ContactUpdateRsp contact_update_response = contact_update.getResponseData();
epp_Response response = contact_update_response.getRsp();
```

La requête XML pour la mise à jour est comme suit :

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
 <epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
   <update>
    <contact:update xmlns:contact="urn:ietf:params:xml:ns:contact-1.0">
     <contact:id>123456t</contact:id>
     <contact:chg>
      <contact:postalInfo type="int">
       <contact:name>My Test</contact:name>
       <contact:org>Example LTD</contact:org>
       <contact:addr>
        <contact:street>Street1</contact:street>
        <contact:street>Street2</contact:street>
        <contact:city>Example</contact:city>
        <contact:sp>Mombasa</contact:sp>
        <contact:pc>00200</contact:pc>
        <contact:cc>TZ</contact:cc>
       </contact:addr>
      </contact:postalInfo>
      <contact:voice>256.562778390</contact:voice>
      <contact:email>example@test.com</contact:email>
     </contact:chg>
    </contact:update>
    </update>
    <extension>
     <contact:update xmlns:contact="urn:ietf:params:xml:ns:contact-id-1.0">
       <contact:person>
        <contact:NID>99725364</contact:NID>
```

```
        </contact:person>
        </contact:update>
      </extension>
    <clTRID>17292851526674749</clTRID>
      </command>
</epp>
```

6.4.6. EPPContactDelete

Description

Cette action permet de supprimer un contact. La suppression se fait lorsque le contact n'a plus de dépendance (association à un domaine) ou son état le permet.

Attributs requis

- Contact ID

Données de retour

- Aucun retour

Usage syntax

```
epp_ContactDeleteReq contact_delete_request = new epp_ContactDeleteReq();
command_data = new epp_Command();

// The client trid is optional. It's main use
// is for Prestataire tracking and logging of requests,
// especially for data creation or modification requests.
command_data.setClientTrid( client_trid );
contact_delete_request.setCmd( command_data );

contact_delete_request.setRoid( "contact3" );

EPPContactDelete contact_delete = new EPPContactDelete();
contact_delete.setRequestData(contact_delete_request);

// Now ask the EPPClient to process the request and retrieve
// a response from the server.
contact_delete = (EPPContactDelete) epp_client.processAction(contact_delete);

// or, alternatively, this method can be used...
//contact_delete.fromXML(epp_client.processXML(contact_delete.toXML()));

epp_ContactDeleteRsp contact_delete_response = contact_delete.getResponseData();
epp_Response response = contact_delete_response.getRsp();
```

6.4.7. EPPContactTransfer

Description

Cette action permet à un prestataire de changer un titulaire. Les opérations possibles sont : approve, cancel, query, reject and request. Par defaut la valeur est "query". Les états de transfert sont : pending, approved, cancelled, rejected, auto-approved, auto-cancelled. Cette commande ne doit pas être utilsé dans le cadre de ce projet. Le transfert de prestataire doit se faire à travers la commande de suppression, création d'un domaine (voir les workflow préseté en annexe 4)

Usage syntax

```
epp_ContactTransferReq contact_transfer_request = new epp_ContactTransferReq();
command_data = new epp_Command();

// The client trid is optional. it's main use is for Prestataire tracking and logging of requests, especially for data creation or modification requests.
command_data.setClientTrid( client_trid );
contact_transfer_request.setCmd( command_data );

// The Contact Transfer request is an EPP "transfer" request, meaning it requires an "op" value and the object's  current auth info for successful processing.
epp_TransferRequest transfer_request = new epp_TransferRequest();

// A transfer query will only query the status of the current pending transfer or the last completed/cancelled/rejected transfer. To request a transfer, the "op" should be set to REQUEST.
// QUERY is the default transfer op type, so it could have been left null here.
transfer_request.setOp( epp_TransferOpType.QUERY );
transfer_request.setAuthInfo( contact_auth_info );
contact_transfer_request.setTrans( transfer_request );
contact_transfer_request.setId( "contact4" );

EPPContactTransfer contact_transfer = new EPPContactTransfer();
contact_transfer.setRequestData(contact_transfer_request);

// Now ask the EPPClient to process the request and retrieve a response from the server.
contact_transfer = (EPPContactTransfer) epp_client.processAction(contact_transfer);

// or, alternatively, this method can be used...
//contact_transfer.fromXML(epp_client.processXML(contact_transfer.toXML()));

epp_ContactTransferRsp contact_transfer_response = contact_transfer.getResponseData();
epp_Response response = contact_transfer_response.getRsp();

String transfer_status = EPPXMLBase.transferStatusTo
String(contact_transfer_response.getTransferStatus());
```

6.5. EPP hôte

Un hôte est un serveur sur le réseau qui recueille les requêtes associées à un domaine.

6.5.1. les données communes

epp_HostAddressType IPV4 , IPV6
epp_HostAddress une strcture composée de

- **type**

- **ip**

epp_HostStatusType : CLIENT_DELETE_PROHIBITED, CLIENT_UPDATE_PROHIBITED, LINKED, OK, PENDING_DELETE, PENDING_TRANSFER, SERVER_DELETE_PROHIBITED, SERVER_UPDATE_PROHIBITED

epp_HostStatus une strcture composé de

- **type**
- **lang**
- **valeur**

6.5.2. EPPHostCreate

Description

Cette action permet de créer un objet hôte. Lors de la création d'un domaine, une liste d'adresse IP peut être spécifiée. Les formats des adresses IP doivent être soit IPv4 ou IPv6.

Attributs requis

- Host Name
- IP Address

Attributs implicites

- Status
- Sponsoring Registrar
- Created By
- Created Date
- Updated By
- Updated Date

Données de retour

- Host Name

Syntaxe d'utilisation

```
epp_HostCreateReq host_create_request = new epp_HostCreateReq ();

command_data = new epp_Command ();

/ / Le tridéo client est facultative.  Son utilisation principale
/ / Est pour le suivi des registres et l'enregistrement des requêtes,
/ / En particulier pour la création ou la modification de données des demandes.
command_data.m_client_trid = client_trid;
host_create_request.m_cmd = command_data;

host_create_request.m_name = "ns1.domain.info";

/ / Lors de la création d'un hôte dans un TLD pour lequel le registre est autorisée,
/ / Au moins une adresse IP est nécessaire.  Un tableau est attendu ici,
/ / Mais pour éviter d'utiliser Java tableaux natifs, nous utilisons une ArrayList
/ / Puis le convertir en un tableau epp_HostAddress utilisant le EPPXMLBase.
Liste ip_list = (Liste) new ArrayList ();
```

```
ip_list.add (nouveau epp_HostAddress (epp_HostAddressType.IPV4, "100.103.44.151"));
ip_list.add (nouveau epp_HostAddress (epp_HostAddressType.IPV6,
                                                    "1080:0:0:0:8:800:200 C: 417A"));
host_create_request.m_addresses =
(Epp_HostAddress []) EPPXMLBase.convertListToArray (
       (Nouveau epp_HostAddress ()) getClass (), ip_list).;

EPPHostCreate host_create = new EPPHostCreate ();
host_create.setRequestData (host_create_request);

/ / Maintenant, demandez le EPPClient pour traiter la demande et récupérer
/ / Une réponse du serveur.
host_create = (EPPHostCreate) epp_client.processAction (host_create);

/ / Ou, en variante, ce procédé peut être utilisé ...
/ / Host_create.fromXML (epp_client.processXML (host_create.toXML ()));

epp_HostCreateRsp host_create_response = host_create.getResponseData ();
réponse de epp_Response = host_create_response.m_rsp;
```

6.5.3. EPPHostCheck

Description

Cette action est utilisée pour déterminer si un nom d'hôte est connu du serveur EPP.

Attributs requis

- Host Name

Données de retour

- Vrai ou faux

syntaxe d'utilisation

```
epp_HostCheckReq host_check_request = new epp_HostCheckReq ();

command_data = new epp_Command ();

/ / Le tridéo client est facultative.  Son utilisation principale Est pour le suivi des registres et
l'enregistrement des requêtes,
/ / En particulier pour la création ou la modification de données des demandes.
command_data.m_client_trid = client_trid;
host_check_request.m_cmd = command_data;

/ / La demande du vérificateur d'hôte peut accepter un tableau de l'hôte  noms.  Dans cet exemple,
une liste de tableaux est utilisé pour dynamiquement
/ / Créer la liste de noms d'hôte et de EPPXMLBase
/ / Méthode utilitaire convertListToChaine de caractèreArray () est utilisée
/ / Pour convertir la liste à un tableau Chaine de caractère.
Liste liste_hôtes = (Liste) new ArrayList ();
host_list.add ("ns1.domain.info");
host_list.add ("ns2.domain.info");
host_check_request.m_names = EPPXMLBase.convertListToChaine de caractèreArray (liste_hôtes);

EPPHostCheck host_check = new EPPHostCheck ();
host_check.setRequestData (host_check_request);

/ / Maintenant, demandez le EPPClient pour traiter la demande et récupérer
```

```
/ / Une réponse du serveur.
host_check = (EPPHostCheck) epp_client.processAction (host_check);

/ / Ou, en variante, ce procédé peut être utilisé ...
/ / Host_check.fromXML (epp_client.processXML (host_check.toXML ()));

epp_HostCheckRsp host_check_response = host_check.getResponseData ();
réponse de epp_Response = host_check_response.m_rsp;
epp_Result [] results = response.m_results;

epp_CheckResult [] = check_results host_check_response.m_results;

Boolean existe = EPPXMLBase.getCheckResultFor (check_results,
                                    "Ns2.domain.info");
```

6.5.4. EPPHostInfo

Description

Cette action est utilisée pour récupérer des informations détaillées associées à un hôte. La réponse à ces informations de commande retourne l'état actuel d'un hôte dans le Registre.

Attributs requis

- Host Name

Données de retour

- Tous les attributs d'un objet host

Syntaxe d'utilisation

```
epp_HostInfoReq host_info_request = new epp_HostInfoReq ();
command_data = new epp_Command ();

command_data.m_client_trid = client_trid;
host_info_request.m_cmd = command_data;

host_info_request.m_name = "ns1.domain.info";
EPPHostInfo host_info = new EPPHostInfo ();
host_info.setRequestData (host_info_request);

/ / Maintenant, demandez le EPPClient pour traiter la demande et récupérer une réponse du serveur.
host_info = (EPPHostInfo) epp_client.processAction (host_info);

/ / Ou, en variante, ce procédé peut être utilisé ...
/ / Host_info.fromXML (epp_client.processXML (host_info.toXML ()));

epp_HostInfoRsp host_info_response = host_info.getResponseData ();
réponse de epp_Response = host_info_response.m_rsp;

epp_Result [] results = response.m_results;

epp_HostAddress [] = host_info_response.m_addresses adresses;
```

6.5.5. EPPHostUpdate

Description

Cette action permet à un prestataire de modifier les champs d'un objet hôte.

Attributs requis

- Host Name

Données de retour

- Aucun

Syntaxe d'utilisation

```
epp_HostUpdateReq host_update_request = new epp_HostUpdateReq ();
command_data = new epp_Command ();

/ / Le tridéo client est facultative.  Son utilisation principale
/ / Est pour le suivi des registres et l'enregistrement des requêtes,
/ / En particulier pour la création ou la modification de données des demandes.
command_data.m_client_trid = client_trid;
host_update_request.m_cmd = command_data;

host_update_request.m_name = "ns1.domain.info";

epp_HostUpdateAddRemove ajouter = new epp_HostUpdateAddRemove ();

/ / Voici une liste d'adresses à ajouter à l'hôte.
/ / Un tableau est prévu, ici, comme dans l'hôte de créer,
/ / Nous utilisons une liste de tableaux, puis en convertissant à une
/ / Un tableau de chaînes.
Liste ip_list = (Liste) new ArrayList ();
ip_list.add (nouveau epp_HostAddress (epp_HostAddressType.IPV4, "101.22.55.99"));
add.m_addresses = (epp_HostAddress []) EPPXMLBase.convertListToArray (
            (Nouveau epp_HostAddress ()) getClass (), ip_list).;

/ / Définit les informations ajouter.
host_update_request.m_add = ajouter;

epp_HostUpdateAddRemove enlever = new epp_HostUpdateAddRemove ();

/ / Maintenant, pour supprimer une adresse.
ip_list = (Liste) new ArrayList ();
ip_list.add (nouveau epp_HostAddress
(Epp_HostAddressType.IPV6, "1080:0:0:0:8:800:200 C: 417A"));
remove.m_addresses = (epp_HostAddress []) EPPXMLBase.convertListToArray
                ((Nouvelle epp_HostAddress ()) getClass (), ip_list.);

/ / Définit les informations de suppression.
host_update_request.m_remove = enlever;

/ / Nous sommes également en spécifiant les informations à modifier.
/ / Seul le nom de l'hôte peut être modifiée.
host_update_request.m_change = new epp_HostUpdateChange ();
host_update_request.m_change.m_name = "ns99.domain.info";

EPPHostUpdate host_update = new EPPHostUpdate ();
host_update.setRequestData (host_update_request);

/ / Maintenant, demandez le EPPClient pour traiter la demande et récupérer
```

```
/ / Une réponse du serveur.
host_update = (EPPHostUpdate) epp_client.processAction (host_update);

/ / Ou, en variante, ce procédé peut être utilisé ...
/ / Host_update.fromXML (epp_client.processXML (host_update.toXML ()));

epp_HostUpdateRsp host_update_response = host_update.getResponseData ();
réponse de epp_Response = host_update_response.m_rsp;
```

6.5.6. EPPHostDelete

Description

Cette action permet à un prestataire de supprimer un objet hôte.
Un hôte ne peut pas être supprimé s'il est affecté à servir un ou plusieurs domaines dans le Registre.
Attributs requis

- Host Name

Données de retour

- Aucun retour

Syntaxe d'utilisation

```
epp_HostDeleteReq host_delete_request = new epp_HostDeleteReq ();

command_data = new epp_Command ();

/ / Le identifiant client est facultative.  Son utilisation principale
/ / Est pour le suivi des registres et l'enregistrement des requêtes,
/ / En particulier pour la création ou la modification de données des demandes.
command_data.m_client_trid = client_trid;
host_delete_request.m_cmd = command_data;

host_delete_request.m_name = "ns1.domain.info";

EPPHostDelete host_delete = new EPPHostDelete ();
host_delete.setRequestData (host_delete_request);

/ / Maintenant, demandez le EPPClient pour traiter la demande et récupérer
/ / Une réponse du serveur.
host_delete = (EPPHostDelete) epp_client.processAction (host_delete);

/ / Ou, en variante, ce procédé peut être utilisé ...
/ / Host_delete.fromXML (epp_client.processXML (host_delete.toXML ()));
```

7. Annexe 1 : Code de retour

Code	Désignation
1000	Command completed successfully
1001	Command completed successfully; action pending
1300	Command completed successfully; no messages
1301	Command completed successfully; ack to dequeue
1500	Command completed successfully; ending session
2000	Unknown command
2001	Command syntax error
2002	Command use error
2003	Required parameter missing
2004	Parameter value range error
2005	Parameter value syntax error
2100	Unimplemented protocol version
2101	Unimplemented command
2102	Unimplemented option
2103	Unimplemented extension
2104	Billing failure
2105	Object is not eligible for renewal
2106	Object is not eligible for transfer
2200	Authentication error
2201	Authorization error
2202	Invalid authorization information
2300	Object pending transfer
2301	Object not pending transfer
2302	Object exists
2303	Object does not exist
2304	Object status prohibits operation
2305	Object association prohibits operation
2306	Parameter value policy error
2307	Unimplemented object service
2308	Data management policy violation
2400	Command failed
2500	Command failed; server closing connection
2501	Authentication error; server closing connection
2502	Session limit exceeded; server closing connection

8. Annexe 2 : Exemple d'implémentation

```
/*
**
** EPP RTK Java
** Copyright (C) 2001-2002, Tucows, Inc.
** Copyright (C) 2003, Liberty RMS
**
**
** This library is free software; you can redistribute it and/or
** modify it under the terms of the GNU Lesser General Public
** License as published by the Free Software Foundation; either
** version 2.1 of the License, or (at your option) any later version.
**
** This library is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
** Lesser General Public License for more details.
**
** You should have received a copy of the GNU Lesser General Public
** License along with this library; if not, write to the Free Software
** Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
**
*/

/*
 * $Header: /cvsroot/epp-rtk/epp-
rtk/java/src/com/tucows/oxrs/epprtk/rtk/example/SessionExample.java,v 1.1 2004/12/07
15:53:26 ewang2004 Exp $
 * $Revision: 1.1 $
 * $Date: 2004/12/07 15:53:26 $
 */

package com.tucows.oxrs.epprtk.rtk.example;

import java.util.*;
import java.io.*;

import com.tucows.oxrs.epprtk.rtk.*;
import com.tucows.oxrs.epprtk.rtk.xml.*;

import org.openrtk.idl.epprtk.*;
import org.openrtk.idl.epprtk.domain.*;
import org.openrtk.idl.epprtk.host.*;
import org.openrtk.idl.epprtk.contact.*;


/**
 * Example code for a typical logical EPP sessions.
 * For more information on the creation of Domain, Host and Contact
 * objects, please see their respective example Java source files.
 *
```

```
 * @author Daniel Manley
 * @version $Revision: 1.1 $ $Date: 2004/12/07 15:53:26 $
 * @see com.tucows.oxrs.epprtk.rtk.example.DomainExample
 * @see com.tucows.oxrs.epprtk.rtk.example.HostExample
 * @see com.tucows.oxrs.epprtk.rtk.example.ContactExample
**/
public class SessionExample
{

   private static Chaine de caractère USAGE = "Usage:
com.tucows.oxrs.epprtk.rtk.example.SessionExample epp_host_name epp_host_port
epp_client_id epp_password domain_name [contact_id1] [contact_id2]";

   /**
    * Main of the example.  Performs typical Domain, Host and Contact operations
    * in a logical order.
    */
   public static void main(Chaine de caractère args[])
   {

      System.out.println("Start of the Session example");

      Date domain_exp_date = null;
      epp_AuthInfo domain_auth_info = null;
      epp_AuthInfo contact1_auth_info = null;
      epp_AuthInfo contact2_auth_info = null;
      // This date will be used in the client trid
      // because the .biz SRS  requires unique
      // trid's per client session.
         Date current_time = new Date();

      try
      {

         Chaine de caractère[] domain_nameservers = null;
         epp_Command command_data = null;
         epp_TransferRequest transfer_request = null;
         epp_CheckResult[] check_results = null;

         if (args.length < 5)
         {
            System.err.println(USAGE);
            System.exit(1);
         }

         Chaine de caractère epp_host_name = args[0];
         Chaine de caractère epp_host_port_Chaine de caractère = args[1];
         Chaine de caractère epp_client_id = args[2];
         Chaine de caractère epp_password  = args[3];
         Chaine de caractère domain_name  = args[4];
         Chaine de caractère contact_id1 = null;
```

```
Chaine de caractère contact_id2 = null;
if ( args.length > 5 )
{
   contact_id1 = args[5];
}
if ( args.length > 6 )
{
   contact_id2 = args[6];
}

if ( contact_id1 == null ) contact_id1 = epp_client_id + "001";
if ( contact_id2 == null ) contact_id2 = epp_client_id + "002";

int epp_host_port = Integer.parseInt(epp_host_port_Chaine de caractère);

EPPClient epp_client = new EPPClient(epp_host_name,
                                     epp_host_port,
                                     epp_client_id,
                                     epp_password);

epp_client.setLang("en");

System.out.println("Connecting to the EPP Server and getting the greeting");

/*
 * Uncomment following line if you don't want to send RTK version
 * number on Login. Although Liberty RTK recomends to use this extension
 * tag on Login request.
 */
//epp_client.setVersionSentOnLogin( false );

epp_Greeting greeting = epp_client.connectAndGetGreeting();

System.out.println("greeting's server: ["+greeting.getServerId()+"]");
System.out.println("greeting's server-date: ["+greeting.getServerDate()+"]");
System.out.println("greeting's service menu: ["+greeting.getSvcMenu()+"]");

// The .biz SRS  requires unique client trid's for
// a session, so we're using the date here to keep it unique
Chaine de caractère client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();

command_data = new epp_Command();
command_data.setClientTrid( client_trid );

System.out.println("Logging into the EPP Server");
epp_client.login(client_trid);

try
{
   // **************************
   // Poll (for waiting messages)
```

```
            // ***************************
            System.out.println("Polling the server...");
            current_time = new Date();
            client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
            epp_PollRsp poll_response = epp_client.poll(client_trid);

            epp_Response response = poll_response.getRsp();
            System.out.println("Poll results: "+response);
         }
         catch ( epp_XMLException xcp )
         {
            // Either the request was missing some required data in
            // validation before sending to the server, or the server's
            // response was either unparsable or missing some required data.
            System.err.println("epp_XMLException! ["+xcp.getErrorMessage()+"]");
         }
         catch ( epp_Exception xcp )
         {
            // The EPP Server has responded with an error code with
            // some optional messages to describe the error.
            System.err.println("epp_Exception!");
            epp_Result[] results = xcp.getDetails();
            System.err.println("\tcode: ["+results[0].getCode()+"] lang:
["+results[0].getLang()+"] msg: ["+results[0].getMsg()+"]");
            if ( results[0].getValues() != null && results[0].getValues().length > 0 )
            {
               System.err.println("\tvalue: ["+results[0].getValues()[0]+"]");
            }
         }
         catch ( Exception xcp )
         {
            // Other unexpected exceptions
            System.err.println("EPP Poll failed! ["+xcp.getClass().getName()+"]
["+xcp.getMessage()+"]");
            xcp.printStackTrace();
         }


         try
         {
            // ***************************
            // Domain Check
            //
            // First, the Prestataire should check if the given domain
            // is available in the SRS .  If it does not, we'll skip the domain
            // create step.
            //
            // ***************************
            System.out.println("Creating the Domain Check command");
            epp_DomainCheckReq domain_check_request = new epp_DomainCheckReq();
```

```
            // The .biz SRS  requires unique client trid's for
            // a session, so we're using the date here to keep it unique
            current_time = new Date();
            client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
            command_data.setClientTrid( client_trid );
            domain_check_request.setCmd( command_data );

            List domain_list = (List)new ArrayList();
            domain_list.add(domain_name);
            domain_check_request.setNames( EPPXMLBase.convertListToChaine de
caractèreArray(domain_list) );

            EPPDomainCheck domain_check = new EPPDomainCheck();
            domain_check.setRequestData(domain_check_request);

            domain_check = (EPPDomainCheck) epp_client.processAction(domain_check);

            epp_DomainCheckRsp domain_check_response =
domain_check.getResponseData();
            check_results = domain_check_response.getResults();
            System.out.println("DomainCheck results: domain ["+domain_name+"] available?
["+EPPXMLBase.getAvailResultFor(check_results, domain_name)+"]");

            if ( EPPXMLBase.getAvailResultFor(check_results, domain_name) != null &&
                 EPPXMLBase.getAvailResultFor(check_results, domain_name).booleanValue()
)
            {

               // We're going to be creating the domain in the SRS .
               // Let's see if the user gave us a contact_id to use
               // in the domain creation, or if we have to create a contact
               // as well.

               boolean contact1_avail = false;
               boolean contact2_avail = false;

               if ( contact_id1 != null )
               {
                  // ***************************
                  // Contact Check
                  //
                  // Make sure the contact_roid we were given is not available
                  // in the SRS .
                  //
                  // ***************************
                  System.out.println("Creating the Contact Check command for
["+contact_id1+"]");
                  epp_ContactCheckReq contact_check_request = new
epp_ContactCheckReq();

                  current_time = new Date();
```

```
                client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
                command_data.setClientTrid( client_trid );
                contact_check_request.setCmd( command_data );

                List contact_list = (List)new ArrayList();
                contact_list.add(contact_id1);
                if ( contact_id2 != null )
                {
                   contact_list.add(contact_id2);
                }
                contact_check_request.setIds( EPPXMLBase.convertListToChaine de
caractèreArray(contact_list) );

                EPPContactCheck contact_check = new EPPContactCheck();
                contact_check.setRequestData(contact_check_request);

                contact_check = (EPPContactCheck)
epp_client.processAction(contact_check);

                epp_ContactCheckRsp contact_check_response =
contact_check.getResponseData();
                check_results = contact_check_response.getResults();
                System.out.println("ContactCheck results: contact ["+contact_id1+"]
available? ["+EPPXMLBase.getAvailResultFor(check_results, contact_id1)+"]");
                System.out.println("ContactCheck results: contact ["+contact_id2+"]
available? ["+EPPXMLBase.getAvailResultFor(check_results, contact_id2)+"]");
                if ( EPPXMLBase.getAvailResultFor(check_results, contact_id1) != null )
                {
                   contact1_avail = EPPXMLBase.getAvailResultFor(check_results,
contact_id1).booleanValue();
                }
                if ( contact_id2 != null &&
                     EPPXMLBase.getAvailResultFor(check_results, contact_id2) != null )
                {
                   contact2_avail = EPPXMLBase.getAvailResultFor(check_results,
contact_id2).booleanValue();
                }
             }

             // id 1 will be used as the registrant for the domain.
             if ( contact_id1 == null || contact1_avail )
             {
                // ***************************
                // Contact Create
                //
                // The given contact_id1 is available in the SRS
                // or there was no contact_id1 specified, so let's
                // create one now.
                //
                // ***************************
                System.out.println("Creating the Contact Create command");
```

```
                epp_ContactCreateReq contact_create_request = new
epp_ContactCreateReq();

                current_time = new Date();
                client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
                command_data.setClientTrid( client_trid );
                contact_create_request.setCmd( command_data );
                contact_create_request.setId( contact_id1 );

                BufferedReader buffed_reader = new BufferedReader(new
InputStreamReader(System.in));
                contact1_auth_info = new epp_AuthInfo();
                System.out.print("Dear registrant, please enter a passphrase for the new
registrant contact(min 6, max 16): ");
                while ( contact1_auth_info.getValue() == null ||
                      contact1_auth_info.getValue().length() == 0 )
                {
                   contact1_auth_info.setValue( buffed_reader.readLine() );
                }
                contact1_auth_info.setType( epp_AuthInfoType.PW );
                contact_create_request.setAuthInfo( contact1_auth_info );

                epp_ContactNameAddress[] name_address = new
epp_ContactNameAddress[1];
                name_address[0] = new epp_ContactNameAddress();
                name_address[0].setType( epp_ContactPostalInfoType.INT );
                name_address[0].setName( "John Doe" );
                name_address[0].setOrg( "ACME Solutions" );
                epp_ContactAddress address = new epp_ContactAddress();
                address.setStreet1( "100 Centre St" );
                address.setCity( "Townsville" );
                address.setStateProvince( "County Derry" );
                address.setPostalCode( "Z1Z1Z1" );
                address.setCountryCode( "CA" );
                name_address[0].setAddress( address );

                contact_create_request.setAddresses( name_address );
                contact_create_request.setVoice( new epp_ContactPhone("1234",
"+1.4165559999") );
                contact_create_request.setFax( new epp_ContactPhone("9876",
"+1.4165558888") );
                contact_create_request.setEmail( "john.doe@company.info" );

                EPPContactCreate contact_create = new EPPContactCreate();
                contact_create.setRequestData(contact_create_request);

                contact_create = (EPPContactCreate)
epp_client.processAction(contact_create);

                epp_ContactCreateRsp contact_create_response =
contact_create.getResponseData();
```

```
                System.out.println("ContactCreate results: contact id
["+contact_create_response.getId()+"]");

            }

            // id 2 will be used as the "tech" contact for the domain
            // we'll be creating later.
            if ( contact_id2 == null || contact2_avail )
            {
                // ***************************
                // Contact Create
                //
                // The given contact_id2 is available in the SRS
                // or there was no contact_id2 specified, so let's
                // create one now.
                //
                // ***************************
                System.out.println("Creating the Contact Create command");
                epp_ContactCreateReq contact_create_request = new
epp_ContactCreateReq();

                current_time = new Date();
                client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
                command_data.setClientTrid( client_trid );
                contact_create_request.setCmd( command_data );
                contact_create_request.setId( contact_id2 );

                BufferedReader buffed_reader = new BufferedReader(new
InputStreamReader(System.in));
                contact2_auth_info = new epp_AuthInfo();
                System.out.print("Dear registrant, please enter a passphrase for the new tech
contact:(min 6, max 16) ");
                while ( contact2_auth_info.getValue() == null ||
                      contact2_auth_info.getValue().length() == 0 )
                {
                   contact2_auth_info.setValue( buffed_reader.readLine() );
                }
                contact2_auth_info.setType( epp_AuthInfoType.PW );
                contact_create_request.setAuthInfo( contact2_auth_info );

                epp_ContactNameAddress[] name_address = new
epp_ContactNameAddress[1];
                name_address[0] = new epp_ContactNameAddress();
                name_address[0].setType( epp_ContactPostalInfoType.INT );
                name_address[0].setName( "Jane Doe" );
                name_address[0].setOrg( "ACME Technicians" );
                epp_ContactAddress address = new epp_ContactAddress();
                address.setStreet1( "101 Centre St" );
                address.setCity( "Townsville" );
                address.setStateProvince( "County Derry" );
                address.setPostalCode( "Z1Z1Z1" );
```

```
                address.setCountryCode( "CA" );
                name_address[0].setAddress( address );

                contact_create_request.setAddresses( name_address );
                contact_create_request.setVoice( new epp_ContactPhone("1234",
"+1.4165551111") );
                contact_create_request.setFax( new epp_ContactPhone("9876",
"+1.4165552222") );
                contact_create_request.setEmail( "jane.doe@company.info" );

                EPPContactCreate contact_create = new EPPContactCreate();
                contact_create.setRequestData(contact_create_request);

                contact_create = (EPPContactCreate)
epp_client.processAction(contact_create);

                epp_ContactCreateRsp contact_create_response =
contact_create.getResponseData();
                System.out.println("ContactCreate results: contact id
["+contact_create_response.getId()+"]");

            }


            // Since we're going to create the domain,
            // we have to ask the registrant for
            // authorization information (a secret password,
            // or something similar).
            BufferedReader buffed_reader = new BufferedReader(new
InputStreamReader(System.in));
            domain_auth_info = new epp_AuthInfo();
            System.out.print("Dear registrant, please enter a passphrase for your new
domain:(min 6, max 16) ");
            while ( domain_auth_info.getValue() == null ||
                  domain_auth_info.getValue().length() == 0 )
            {
               domain_auth_info.setValue( buffed_reader.readLine() );
            }
            domain_auth_info.setType( epp_AuthInfoType.PW );

            // ***************************
            // Domain Create
            //
            // Domain is available in the SRS , so create it now.
            //
            // ***************************
            System.out.println("Creating the Domain Create command");
            epp_DomainCreateReq domain_create_request = new
epp_DomainCreateReq();

            current_time = new Date();
```

```
                client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
                command_data.setClientTrid( client_trid );
                domain_create_request.setCmd( command_data );

                domain_create_request.setName( domain_name );
                epp_DomainPeriod period = new epp_DomainPeriod();
                // Note that some openrtk might not accept registration
                // periods by months.
                period.setUnit( epp_DomainPeriodUnitType.YEAR );
                period.setValue( (short) 2 );
                domain_create_request.setPeriod( period );

                domain_create_request.setRegistrant( contact_id1 );
                List domain_contacts = new ArrayList();
                // EPP Domain registries often require at least one
                // of each type of contact.
                domain_contacts.add( new epp_DomainContact(
epp_DomainContactType.TECH, contact_id2 ) );
                domain_contacts.add( new epp_DomainContact(
epp_DomainContactType.ADMIN, contact_id1 ) );
                domain_contacts.add( new epp_DomainContact(
epp_DomainContactType.BILLING, contact_id2 ) );
                domain_create_request.setContacts( (epp_DomainContact[])
EPPXMLBase.convertListToArray((new epp_DomainContact()).getClass(), domain_contacts)
);

                domain_create_request.setAuthInfo( domain_auth_info );

                // From an EPP perspective, nameserver associations are
                // optional for a domain, so we're not specifying them
                // here.  We will add them later in the domain update.

                EPPDomainCreate domain_create = new EPPDomainCreate();
                domain_create.setRequestData(domain_create_request);

                domain_create = (EPPDomainCreate)
epp_client.processAction(domain_create);

                // We don't particularily care about the response here.
                // As long as an expection was not thrown, then the
                // creation was successful.  We'll get the expiration
                // date later in a domain info.

            } // end if for domain is available in SRS .



            // OK, before trying to do anything to this domain,
            // we should check to see if we have a situation where
            // the domain already existed and it's not owned by us.
            // If we're not the sponsoring Prestataire then we can't do
```

```
            // anything to it and the session end here.

            // ***************************
            // Domain Info
            //
            // Info will return to us a list of nameservers and
            // the auth Info if we're the owner.
            //
            // ***************************
            System.out.println("Creating the Domain Info command");
            epp_DomainInfoReq domain_info_request = new epp_DomainInfoReq();

            current_time = new Date();
            client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
            command_data.setClientTrid( client_trid );
            domain_info_request.setCmd( command_data );

            domain_info_request.setName( domain_name );

            EPPDomainInfo domain_info = new EPPDomainInfo();
            domain_info.setRequestData(domain_info_request);

            domain_info = (EPPDomainInfo) epp_client.processAction(domain_info);

            epp_DomainInfoRsp domain_info_response = domain_info.getResponseData();

            System.out.println("DomainInfo Results: registrant
["+domain_info_response.getRegistrant()+"]");
            System.out.println("DomainInfo Results: status count
["+domain_info_response.getStatus().length+"]");
            for ( int i = 0; i < domain_info_response.getStatus().length; i++ )
            {
               System.out.println("\tstatus["+i+"] Chaine de caractère
["+EPPDomainBase.domainStatusToChaine de caractère(
domain_info_response.getStatus()[i].getType() )+"]");
               System.out.println("\tstatus["+i+"] note
["+domain_info_response.getStatus()[i].getValue()+"]");
            }
            // Save the expiration date for the renew command later
            domain_exp_date =
RTKBase.UTC_FMT.parse(domain_info_response.getExpirationDate());
            // Save the list of nameservers
            domain_nameservers = domain_info_response.getNameServers();

            // Save the auth ID.
            domain_auth_info = domain_info_response.getAuthInfo();
            if ( domain_info_response.getAuthInfo() == null )
            {
               // We're out of luck, this domain is owned by another
               // Prestataire.  The session ends.
```

```
                System.out.println("Domain belongs to another Prestataire, building transfer
command.");

                // ***************************
                // Domain Transfer (Request)
                //
                // Ok, so the domain is not owned by us, so let's try to transfer it
                //
                // ***************************

                // First we have to know the auth info, so let's ask the registrant
                BufferedReader buffed_reader = new BufferedReader(new
InputStreamReader(System.in));
                epp_AuthInfo other_domain_auth_info = new epp_AuthInfo();
                System.out.print("Dear registrant, please enter the passphrase for the domain
you wish to transfer:(min 6, max 16) ");
                while ( other_domain_auth_info.getValue() == null ||
                      other_domain_auth_info.getValue().length() == 0 )
                {
                   other_domain_auth_info.setValue( buffed_reader.readLine() );
                }
                other_domain_auth_info.setType( epp_AuthInfoType.PW );

                System.out.println("Creating the Domain Transfer command");
                epp_DomainTransferReq domain_transfer_request = new
epp_DomainTransferReq();

                current_time = new Date();
                client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
                command_data.setClientTrid( client_trid );
                domain_transfer_request.setCmd( command_data );

                transfer_request = new epp_TransferRequest();
                transfer_request.setOp( epp_TransferOpType.REQUEST );
                // we just asked for the auth info, so let's use it here.
                transfer_request.setAuthInfo( other_domain_auth_info );
                domain_transfer_request.setTrans( transfer_request );

                domain_transfer_request.setName( domain_name );

                EPPDomainTransfer domain_transfer = new EPPDomainTransfer();
                domain_transfer.setRequestData(domain_transfer_request);

                domain_transfer = (EPPDomainTransfer)
epp_client.processAction(domain_transfer);

                epp_DomainTransferRsp domain_transfer_response =
domain_transfer.getResponseData();
                System.out.println("DomainTransfer Results: transfer status
["+EPPXMLBase.transferStatusToChaine de caractère(
domain_transfer_response.getTrnData().getTransferStatus() )+"]");
```

```
                // If an exception was thrown to this command, then probably
                // the auth ID we used was wrong, so maybe someone
                // transfered the domain away from us.

                System.out.println("Logging out from the EPP Server");
                epp_client.logout(client_trid);
                System.out.println("Disconnecting from the EPP Server");
                epp_client.disconnect();
                System.exit(1);
            }


            // The domain is ours.
            // Now let's create some hosts in this domain.
            // If the domain had the possibility of existing before and it
            // did than maybe the hosts exists in the SRS  too.
            // Let's check....

            // ***************************
            // Host Check
            //
            // Check for the existance of two hosts, ns1 and ns2
            // in the domain given to us by the user.
            //
            // ***************************
            System.out.println("Creating the Host Check command");
            epp_HostCheckReq host_check_request = new epp_HostCheckReq();

            current_time = new Date();
            client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
            command_data.setClientTrid( client_trid );
            host_check_request.setCmd( command_data );

            List host_list = (List)new ArrayList();
            host_list.add("ns1."+domain_name);
            host_list.add("ns2."+domain_name);
            host_check_request.setNames( EPPXMLBase.convertListToChaine de
caractèreArray(host_list) );

            EPPHostCheck host_check = new EPPHostCheck();
            host_check.setRequestData(host_check_request);

            host_check = (EPPHostCheck) epp_client.processAction(host_check);

            epp_HostCheckRsp host_check_response = host_check.getResponseData();
            check_results = host_check_response.getResults();
            System.out.println("HostCheck results: host [ns1."+domain_name+"] avail?
["+EPPXMLBase.getAvailResultFor(check_results, "ns1."+domain_name)+"]");
            System.out.println("HostCheck results: host [ns2."+domain_name+"] avail?
["+EPPXMLBase.getAvailResultFor(check_results, "ns2."+domain_name)+"]");
```

```
            if ( EPPXMLBase.getAvailResultFor(check_results, "ns1."+domain_name) == null
||
                 EPPXMLBase.getAvailResultFor(check_results,
"ns1."+domain_name).booleanValue() == true )
            {
               // ***************************
               // Host Create
               //
               // Host ns1."domain_name" is available, so let's create it.
               //
               // ***************************
               System.out.println("Creating the Host Create command");
               epp_HostCreateReq host_create_request = new epp_HostCreateReq();

               current_time = new Date();
               client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
               command_data.setClientTrid( client_trid );
               host_create_request.setCmd( command_data );

               host_create_request.setName( "ns1."+domain_name );

               List ip_list = (List)new ArrayList();
               // Some registries restrict the number of IPs per address type to 1,
               // so, we'll only use 1 in this example.  Also, some registries
               // restrict the number of times an IP address may be used to 1,
               // so we'll ask the user for a unique value.
               BufferedReader buffed_reader = new BufferedReader(new
InputStreamReader(System.in));
               System.out.print("Dear registrant, please enter an IPv4 address for the
nameserver "+host_create_request.getName()+"\n(it must not already be used and must not
be a restricted address): ");
               Chaine de caractère ipAddr = null;
               while ( ipAddr == null || ipAddr.length() == 0 )
               {
                  ipAddr = buffed_reader.readLine();
               }
               ip_list.add(new epp_HostAddress(epp_HostAddressType.IPV4, ipAddr));
               host_create_request.setAddresses(
(epp_HostAddress[])EPPXMLBase.convertListToArray((new epp_HostAddress()).getClass(),
ip_list) );

               EPPHostCreate host_create = new EPPHostCreate();
               host_create.setRequestData(host_create_request);

               host_create = (EPPHostCreate) epp_client.processAction(host_create);

               // As long as an exception is not thrown than the host
               // create succeeded.

            }
```

```
            if ( EPPXMLBase.getAvailResultFor(check_results, "ns2."+domain_name) == null
||
                 EPPXMLBase.getAvailResultFor(check_results,
"ns2."+domain_name).booleanValue() == true )
            {
               // ***************************
               // Host Create
               //
               // Host ns2."domain_name" is available, so let's create it.
               //
               // ***************************
               System.out.println("Creating the Host Create command");
               epp_HostCreateReq host_create_request = new epp_HostCreateReq();

               current_time = new Date();
               client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
               command_data.setClientTrid( client_trid );
               host_create_request.setCmd( command_data );

               host_create_request.setName( "ns2."+domain_name );

               List ip_list = (List)new ArrayList();
               // Like in the creation of the first host, we'll ask the
               // registrant for a valid IPv4 address
               BufferedReader buffed_reader = new BufferedReader(new
InputStreamReader(System.in));
               System.out.print("Dear registrant, please enter an IPv4 address for the
nameserver "+host_create_request.getName()+"\n(it must not already be used and must not
be a restricted address): ");
               Chaine de caractère ipAddr = null;
               while ( ipAddr == null || ipAddr.length() == 0 )
               {
                  ipAddr = buffed_reader.readLine();
               }
               ip_list.add(new epp_HostAddress(epp_HostAddressType.IPV4, ipAddr));
               host_create_request.setAddresses(
(epp_HostAddress[])EPPXMLBase.convertListToArray((new epp_HostAddress()).getClass(),
ip_list) );

               EPPHostCreate host_create = new EPPHostCreate();
               host_create.setRequestData(host_create_request);

               host_create = (EPPHostCreate) epp_client.processAction(host_create);

               // As long as an exception is not thrown than the host
               // create succeeded.

            }

            // ***************************
            // Let's do an info on one of the hosts to find its owner
```

```
            // and its status
            // **************************
            System.out.println("Creating the Host Info command");
            epp_HostInfoReq host_info_request = new epp_HostInfoReq();

            current_time = new Date();
            client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
            command_data.setClientTrid( client_trid );
            host_info_request.setCmd( command_data );

            host_info_request.setName( "ns2."+domain_name );

            EPPHostInfo host_info = new EPPHostInfo();
            host_info.setRequestData(host_info_request);

            host_info = (EPPHostInfo) epp_client.processAction(host_info);

            epp_HostInfoRsp host_info_response = host_info.getResponseData();

            System.out.println("HostInfo results: clID ["+host_info_response.getClientId()+"]
crID ["+host_info_response.getCreatedBy()+"]");
            System.out.println("HostInfo results: crDate
["+host_info_response.getCreatedDate()+"] upDate
["+host_info_response.getUpdatedDate()+"]");
            System.out.println("HostInfo results: number of ipaddresses ["+(
host_info_response.getAddresses() == null ? 0 : host_info_response.getAddresses().length
)+"]");
            for ( int i = 0; i < host_info_response.getAddresses().length; i++ )
            {
               System.out.println("\taddress["+i+"] type
["+EPPHostBase.hostAddressTypeToChaine de
caractère(host_info_response.getAddresses()[i].getType())+"] value
["+host_info_response.getAddresses()[i].getIp()+"]");
            }

            System.out.println("HostInfo Results: status count
["+host_info_response.getStatus().length+"]");
            for ( int i = 0; i < host_info_response.getStatus().length; i++ )
            {
               System.out.println("\tstatus["+i+"] Chaine de caractère
["+EPPHostBase.hostStatusToChaine de caractère(
host_info_response.getStatus()[i].getType() )+"]");
               System.out.println("\tstatus["+i+"] note
["+host_info_response.getStatus()[i].getValue()+"]");
            }


            epp_DomainUpdateAddRemove add = null;
            epp_DomainUpdateAddRemove remove = null;

            // OK, the domain exists, the hosts exists.  Are
```

```
            // they already assigned to the domain as nameservers?

            if ( domain_nameservers == null )
            {
               // No nameservers serving the domain
               // so let's add the two we created (or which already existed)
               List add_list = (List) new ArrayList();
               System.out.println("adding both ns1 and ns2 to domain");
               add_list.add("ns1."+domain_name);
               add_list.add("ns2."+domain_name);
               add = new epp_DomainUpdateAddRemove();
               add.setNameServers( EPPXMLBase.convertListToChaine de
caractèreArray(add_list) );
            }
            else
            {
               // Already nameservers for this domain,
               // so let's see if the two we want are there and
               // add or remove accordingly.
               int nameserver_count = domain_nameservers.length;
               List remove_list = (List) new ArrayList();
               List add_list = (List) new ArrayList();
               boolean found_ns1 = false;
               boolean found_ns2 = false;

               for ( int index = 0; index < nameserver_count; index++ )
               {
                  if ( domain_nameservers[index].equalsIgnoreCase("ns1."+domain_name) )
                  {
                     System.out.println("removing ns1 from domain");
                     remove_list.add("ns1."+domain_name);
                     found_ns1 = true;
                  }
                  else if (
domain_nameservers[index].equalsIgnoreCase("ns2."+domain_name) )
                  {
                     System.out.println("removing ns2 from domain");
                     remove_list.add("ns2."+domain_name);
                     found_ns2 = true;
                  }

               }

               if ( found_ns1 == false )
               {
                  System.out.println("adding ns1 to domain");
                  add_list.add("ns1."+domain_name);
               }
               if ( found_ns2 == false )
               {
                  System.out.println("adding ns2 to domain");
```

```
                add_list.add("ns2."+domain_name);
            }

            if ( add_list.size() > 0 )
            {
                add = new epp_DomainUpdateAddRemove();
                add.setNameServers( EPPXMLBase.convertListToChaine de
caractèreArray(add_list) );
            }
            if ( remove_list.size() > 0 )
            {
                remove = new epp_DomainUpdateAddRemove();
                remove.setNameServers( EPPXMLBase.convertListToChaine de
caractèreArray(remove_list) );
            }
        }

        // Let's modify the domain to have these hosts act
        // as its nameservers.

        // ***************************
        // Domain Update
        //
        // Adding two nameservers to this domain
        //
        // ***************************
        System.out.println("Creating the Domain Update command");
        epp_DomainUpdateReq domain_update_request = new epp_DomainUpdateReq();

        current_time = new Date();
        client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
        command_data.setClientTrid( client_trid );
        domain_update_request.setCmd( command_data );

        domain_update_request.setName( domain_name );

        // We determined a little earlier which operations to perform.
        if ( add != null )
        {
            domain_update_request.setAdd( add );
        }
        if ( remove != null )
        {
            domain_update_request.setRemove( remove );
        }

        EPPDomainUpdate domain_update = new EPPDomainUpdate();
        domain_update.setRequestData(domain_update_request);

        domain_update = (EPPDomainUpdate) epp_client.processAction(domain_update);
```

```
            // As long as no exception was thrown, the update was a success


            // ***************************
            // Let's do an info on one of the contacts to find its owner
            // and its status
            // ***************************
            System.out.println("Creating the Contact Info command");
            epp_ContactInfoReq contact_info_request = new epp_ContactInfoReq();

            current_time = new Date();
            client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
            command_data.setClientTrid( client_trid );
            contact_info_request.setCmd( command_data );

            contact_info_request.setId( domain_info_response.m_registrant );

            EPPContactInfo contact_info = new EPPContactInfo();
            contact_info.setRequestData(contact_info_request);

            contact_info = (EPPContactInfo) epp_client.processAction(contact_info);

            epp_ContactInfoRsp contact_info_response = contact_info.getResponseData();

            System.out.println("ContactInfo results: clID
["+contact_info_response.getClientId()+"] crID ["+contact_info_response.getCreatedBy()+"]");
            System.out.println("ContactInfo results: crDate
["+contact_info_response.getCreatedDate()+"] upDate
["+contact_info_response.getUpdatedDate()+"]");
            System.out.println("ContactInfo results: address street 1
["+contact_info_response.getAddresses()[0].getAddress().getStreet1()+"]");
            System.out.println("ContactInfo results: address street 2
["+contact_info_response.getAddresses()[0].getAddress().getStreet2()+"]");
            System.out.println("ContactInfo results: address street 3
["+contact_info_response.getAddresses()[0].getAddress().getStreet3()+"]");
            System.out.println("ContactInfo results: fax ["+contact_info_response.getFax()+"]");

            System.out.println("ContactInfo Results: status count
["+contact_info_response.getStatus().length+"]");
            for ( int i = 0; i < contact_info_response.getStatus().length; i++ )
            {
               System.out.println("\tstatus["+i+"] Chaine de caractère
["+EPPContactBase.contactStatusToChaine de caractère(
contact_info_response.getStatus()[i].getType() )+"]");
               System.out.println("\tstatus["+i+"] note
["+contact_info_response.getStatus()[i].getValue()+"]");
            }


            // ***************************
            // Domain Transfer (Query)
```

```
            //
            // Now, let's pretent that some time has passed...
            // Let's check up on our domain to see
            // if anyone happens to be requesting a transfer on it.
            //
            // ***************************
            try
            {
               System.out.println("Creating the Domain Transfer command");
               epp_DomainTransferReq domain_transfer_request = new
epp_DomainTransferReq();

               current_time = new Date();
               client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
               command_data.setClientTrid( client_trid );
               domain_transfer_request.setCmd( command_data );

               transfer_request = new epp_TransferRequest();
               transfer_request.setOp( epp_TransferOpType.QUERY );
               // Use the auth info from the creation of the domain
               transfer_request.setAuthInfo( domain_auth_info );
               domain_transfer_request.setTrans( transfer_request );

               domain_transfer_request.setName( domain_name );

               EPPDomainTransfer domain_transfer = new EPPDomainTransfer();
               domain_transfer.setRequestData(domain_transfer_request);

               domain_transfer = (EPPDomainTransfer)
epp_client.processAction(domain_transfer);

               epp_DomainTransferRsp domain_transfer_response =
domain_transfer.getResponseData();
               System.out.println("DomainTransfer Results: transfer status
["+EPPXMLBase.transferStatusToChaine de caractère(
domain_transfer_response.getTrnData().getTransferStatus() )+"]");

               if ( domain_transfer_response.getTrnData().getTransferStatus() ==
epp_TransferStatusType.PENDING )
               {
                  // hmmm... there's a transfer pending on this domain,

                  System.out.println("Creating the Domain Transfer command");
                  domain_transfer_request = new epp_DomainTransferReq();

                  current_time = new Date();
                  client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
                  command_data.setClientTrid( client_trid );
                  domain_transfer_request.setCmd( command_data );

                  transfer_request = new epp_TransferRequest();
```

```
                // Let's find out from the registrant/Prestataire if they want
                // the transfer approved.
                BufferedReader buffed_reader = new BufferedReader(new
InputStreamReader(System.in));
                System.out.print("Do you wish to approve the domain's transfer [y]? ");
                Chaine de caractère answer = buffed_reader.readLine();

                while ( ( answer != null ) &&
                      ( answer.length() != 0 ) &&
                      ( ! answer.equalsIgnoreCase("y") ) &&
                      ( ! answer.equalsIgnoreCase("n") ) )
                {
                   answer = buffed_reader.readLine();
                }

                if ( ! answer.equalsIgnoreCase("n") )
                {
                   System.out.println("Going to approve the transfer");
                   transfer_request.setOp( epp_TransferOpType.APPROVE );
                }
                else
                {
                   System.out.println("Going to reject the transfer");
                   transfer_request.setOp( epp_TransferOpType.REJECT );
                }


                // Use the auth info from the creation of the domain
                transfer_request.setAuthInfo( domain_auth_info );
                domain_transfer_request.setTrans( transfer_request );

                domain_transfer_request.setName( domain_name );

                domain_transfer = new EPPDomainTransfer();
                domain_transfer.setRequestData(domain_transfer_request);

                domain_transfer = (EPPDomainTransfer)
epp_client.processAction(domain_transfer);

                domain_transfer_response = domain_transfer.getResponseData();
                System.out.println("DomainTransfer Results: transfer status
["+EPPXMLBase.transferStatusToChaine de caractère(
domain_transfer_response.getTrnData().getTransferStatus() )+"]");

                if ( transfer_request.getOp() == epp_TransferOpType.APPROVE )
                {
                   // We've approved the domain's transfer, so
                   // since we don't own it anymore, we can't
                   // continue working on it.
                   System.out.println("Logging out from the EPP Server");
```

```
                epp_client.logout(client_trid);
                System.out.println("Disconnecting from the EPP Server");
                epp_client.disconnect();
                System.exit(1);
            }

         }
      }
      catch ( epp_Exception xcp )
      {
         // If an exception was thrown to this command, then maybe
         // the auth info we used was wrong, or maybe someone
         // transfered the domain away from us, or maybe
         // there is not transfer information to report on.
         epp_Result[] results = xcp.getDetails();
         if ( results[0].getCode() ==
epp_Session.EPP_OBJECT_NOT_PENDING_TRANSFER )
         {
            System.out.println("The domain is not currently in pending transfer state");
         }
         else if ( results[0].getCode() ==
epp_Session.EPP_UNIMPLEMENTED_OPTION )
         {
            System.out.println("This EPP command option has not been implemented in
the SRS  yet.  That's OK, let's continue...");
         }
         else
         {
            // Something else unexpected happended, so throw the exception up
            throw xcp;
         }
      }


      // Let's ask to see if the user wants to renew the domain
      BufferedReader buffed_reader = new BufferedReader(new
InputStreamReader(System.in));
      System.out.print("Do you wish to renew your domain [y]? ");
      Chaine de caractère answer = buffed_reader.readLine();

      while ( ( answer != null ) &&
            ( answer.length() != 0 ) &&
            ( ! answer.equalsIgnoreCase("n") ) &&
            ( ! answer.equalsIgnoreCase("y") ) )
      {
         answer = buffed_reader.readLine();
      }

      if ( ! answer.equalsIgnoreCase("n") )
      {
```

```
                // ***************************
                // Domain Renew
                //
                // Now, assuming no exception was thrown from the transfer
                // query request, we can probably try to renew the
                // domain.
                //
                // ***************************
                System.out.println("Creating the Domain Renew command");
                epp_DomainRenewReq domain_renew_request = new
epp_DomainRenewReq();

                current_time = new Date();
                client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
                command_data.setClientTrid( client_trid );
                domain_renew_request.setCmd( command_data );

                domain_renew_request.setName( domain_name );
                // How about for another 6 years?
                // Note that some openrtk might not accept renewal
                // periods by months.
                epp_DomainPeriod period = new epp_DomainPeriod();
                period.setUnit( epp_DomainPeriodUnitType.YEAR );
                period.setValue( (short) 6 );
                domain_renew_request.setPeriod( period );
                // The domain's current expiration must also be specified
                // to unintentional multiple renew request from succeeding.
                // The format of the expiration date must be "YYYY-MM-DD"
                domain_renew_request.setCurrentExpirationDate(
RTKBase.DATE_FMT.format(domain_exp_date) );

                EPPDomainRenew domain_renew = new EPPDomainRenew();
                domain_renew.setRequestData(domain_renew_request);

                domain_renew = (EPPDomainRenew)
epp_client.processAction(domain_renew);

                epp_DomainRenewRsp domain_renew_response =
domain_renew.getResponseData();
                // The domain renew action returns the domain's new expiration
                // date if the request was successful
                System.out.println("DomainRenew results: new exDate
["+domain_renew_response.getExpirationDate()+"]");
                domain_exp_date =
RTKBase.UTC_FMT.parse(domain_renew_response.getExpirationDate());

            }

            // Let's ask to see if the user wants to delete the domain.
            // You would not want to delete the domain if you want to
            // see domain transfer in action.
```

```
            buffed_reader = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Do you wish to delete your domain [y]? ");
            answer = buffed_reader.readLine();

            while ( ( answer != null ) &&
                  ( answer.length() != 0 ) &&
                  ( ! answer.equalsIgnoreCase("n") ) &&
                  ( ! answer.equalsIgnoreCase("y") ) )
            {
               answer = buffed_reader.readLine();
            }

            if ( ! answer.equalsIgnoreCase("n") )
            {

               // ***************************
               // Domain Delete
               //
               // Finally, let's end the session by deleting the domain
               //
               // Recent tests with the .biz SRS  show that this
               // command will fail because the domain has nameservers
               // that are associated with a domain.  Oddly enough, they
               // are associated with their own domain.
               //
               // ***************************
               System.out.println("Creating the Domain Delete command");
               epp_DomainDeleteReq domain_delete_request = new
epp_DomainDeleteReq();

               current_time = new Date();
               client_trid = "ABC:"+epp_client_id+":"+current_time.getTime();
               command_data.setClientTrid( client_trid );
               domain_delete_request.setCmd( command_data );

               domain_delete_request.setName( domain_name );

               EPPDomainDelete domain_delete = new EPPDomainDelete();
               domain_delete.setRequestData(domain_delete_request);

               domain_delete = (EPPDomainDelete) epp_client.processAction(domain_delete);

            }


         }
         catch ( epp_XMLException xcp )
         {
            // Either the request was missing some required data in
            // validation before sending to the server, or the server's
            // response was either unparsable or missing some required data.
```

```
            System.err.println("epp_XMLException! ["+xcp.getErrorMessage()+"]");
         }
         catch ( epp_Exception xcp )
         {
            // The EPP Server has responded with an error code with
            // some optional messages to describe the error.
            System.err.println("epp_Exception!");
            epp_Result[] results = xcp.getDetails();
            System.err.println("\t"+results[0]);
         }
         catch ( Exception xcp )
         {
            // Other unexpected exceptions
            System.err.println("EPP Action failed! ["+xcp.getClass().getName()+"]
["+xcp.getMessage()+"]");
            xcp.printStackTrace();
         }


         // All done with this session, so let's log out...
         System.out.println("Logging out from the EPP Server");
         epp_client.logout(client_trid);

         // ... and disconnect
         System.out.println("Disconnecting from the EPP Server");
         epp_client.disconnect();

      }
      catch ( epp_XMLException xcp )
      {
         System.err.println("epp_XMLException! ["+xcp.getErrorMessage()+"]");
      }
      catch ( epp_Exception xcp )
      {
         System.err.println("epp_Exception!");
         epp_Result[] results = xcp.getDetails();
         // We're taking advantage epp_Result's toChaine de caractère() here
         // for debugging.  Take a look at the javadocs for
         // the full list of attributes in the class.
         System.err.println("\tresult: ["+results[0]+"]");
      }
      catch ( Exception xcp )
      {
         System.err.println("Exception! ["+xcp.getClass().getName()+"]
["+xcp.getMessage()+"]");
         xcp.printStackTrace();
      }

   }
}
```

9. Annexe 3: Liste des Etats

9.1. Etats des domaines

1. ClientDeleteProhibited [EppDomain.STATUS_CLIENT_DELETE_PROHIBITED]: A Registrar request to delete the object must be rejected. May only be set by the sponsoring Registrar.

2. ServerDeleteProhibited [EppDomain.STATUS_SERVER_DELETE_PROHIBITED]: Any request to delete the object must be rejected. May only be set by Registry

3. ClientHold [EppDomain.STATUS_CLIENT_HOLD]: The domain must not appear in the TLD zone file. May only be set by the sponsoring Registrar.

4. ServerHold [EppDomain.STATUS_SERVER_HOLD]: The domain must not appear in the TLD zone file. May only be set by Registry.

5. ClientRenewProhibited [EppDomain.STATUS_CLIENT_RENEW_PROHIBITED]: A Registrar request to renew the object must be rejected. May only be set by the sponsoring Registrar.

6. ServerRenewProhibited EppDomain.STATUS_SERVER_RENEW_PROHIBITED]: Any request to renew the object must be rejected. May only be set by Registry.

7. ClientTransferProhibited [EppDomain.STATUS_CLIENT_TRANSFER_PROHIBITED]: A Registrar request to transfer the object must be rejected. May only be set by the sponsoring Registrar.

8. ServerTransferProhibited [EppDomain.STATUS_SERVER_TRANSFER_PROHIBITED]: Any request to transfer the object must be rejected. May only be set by Registry.

9. ClientUpdateProhibited [EppDomain.STATUS_CLIENT_UPDATE_PROHIBITED]: A Registrar request to update the object must be rejected. May only be set by the sponsoring Registrar. The only update allowed is to remove the ClientUpdateProhibited status.

10. ServerUpdateProhibited [EppDomain.STATUS_SERVER_UPDATE_PROHIBITED]: Any request to update the object must be rejected. May only be set by Registry. The only update allowed is to remove the ServerUpdateProhibited status.

11. Inactive [EppDomain.STATUS_INACTIVE]: The domain must not appear in the zone because it lacks proper delegation information (lacks nameserver information). May only be set by Registry.

12. OK [EppDomain.STATUS_OK]: No other statuses set. May only be set by Registry.

13. PendingCreate [EppDomain.STATUS_PENDING_CREATE]: A create request is successful, but the formal create is pending in the system. May only be set by Registry.

14. PendingUpdate [EppDomain.STATUS_PENDING_UPDATE]: An update request is successful, but the formal update is pending in the system. May only be set by Registry.

15. PendingDelete [EppDomain.STATUS_PENDING_DELETE]: A delete request is successful, but the formal delete is pending in the system. The domain must not be published in the zone. All requests must be rejected. May only be set by Registry.

16. PendingTransfer [EppDomain.STATUS_PENDING_TRANSFER]: A transfer request has been received for the object, and completion of the request is pending. The delete, update, and renew commands must be rejected. May only be set by Registry.
Prohibited status combinations are:
1. The OK status must not be combined with any other status.

2. The PendingDelete status must not be combined with either ClientDeleteProhibited or ServerDeleteProhibited.

3. The PendingTransfer status must not be combined with either ClientTransferProhibited or ServerTransferProhibited.

Statut ANRT	Statut Cocca web	Statut EPP	Description
Actif	Delegated	Actif INACTIVE, , PENDING_TRANSFER: ces deux existent sur le web PENDING_VERIFICATION: est ce que c'est pending create?	Domaine actif, publié sur la zone, et visible sur le whois avec un état « Actif »
Bloqué	Suspended		Domaine publié sur la zone, et visible sur le whois avec un état « Bloqué » et les noms d'hôtes associés sont visibles sur le Whois. L'accès au site est redirigé vers une page personalisable qui indique que le domaine est bloqué pour contacter le registre
Gelé	Locked	SERVER_DELETE_PROHIBITED, SERVER_RENEW_PROHIBITED, SERVER_TRANSFER_PROHIBITED, SERVER_UPDATE_PROHIBITED CLIENT_DELETE_PROHIBITED, , CLIENT_RENEW_PROHIBITED, CLIENT_TRANSFER_PROHIBITED, CLIENT_UPDATE_PROHIBITED	Domaine publié sur la zone, et visible sur le whois avec un état « Gelé » la modification par le prestatataire n'est pas possible
Expiré	Expired		Etat domaine arrivé à sa date d'expiration. Domaine publié sur la zone, et visible sur le whois avec un état « Expiré ». Aucun changement n'est possible sur le domaine
Résilié	Pending delete	PENDING_DELETE,	Domaine supprimé par le prestataire et

			attend l'approbation du registre durant la période de grace de résiliation
Supprimé	Deleted		Domaine libre et peut être enregistré
	Excluded		Domaine exclu de la zone, ne peut être changé que par le registre en mentionnant la raison. Il est visible sur le whois avec un état inactif, et les noms d'hotes associés ne sont pas visibles. L'accès au site web retourne une page de éserver not found ». Il est généralement utilisé quand le registre juge que ce domaine ne doit pas être enregistré et qui sera supprimé par la suite
	Hold	CLIENT_HOLD SERVER_HOLD,	Domaine exclu de la zone par le prestataire ou par le registre, s'il veut donner la main au prestataire de supprimer cette suspension et réactiver son domaine.

9.2. Etats des hosts

1. ClientDeleteProhibited [EppHost.STATUS_CLIENT_DELETE_PROHIBITED]: A Registrar request to delete the object must be rejected. May only be set by the sponsoring Registrar.

2. ServerDeleteProhibited [EppHost.STATUS_SERVER_DELETE_PROHIBITED]: Any request to delete the object must be rejected. May only be set by Registry

3. ClientUpdateProhibited [EppHost.STATUS_CLIENT_UPDATE_PROHIBITED]: A Registrar request to update the object must be rejected. May only be set by the sponsoring Registrar. The only update allowed is to remove the ClientUpdateProhibited status.

4. ServerUpdateProhibited [EppHost.STATUS_SERVER_UPDATE_PROHIBITED]: Any request to update the object must be rejected. May only be set by Registry. The only update allowed is to remove the ServerUpdateProhibited status.

5. Linked [EppHost.STATUS_LINKED]: The Object has at least one active association with another object, such as a Domain object. May only be set by Registry.

6. OK [EppHost.STATUS_OK]: No other statuses set. May only be set by Registry.

7. PendingCreate [EppHost.STATUS_PENDING_CREATE]: A create request is successful, but the formal create is pending in the system. May only be set by Registry.

8. PendingUpdate [EppHost.STATUS_PENDING_UPDATE]: An update request is successful, but the formal update is pending in the system. May only be set by Registry.

9. PendingDelete [EppHost.STATUS_PENDING_DELETE]: A delete request is successful, but the formal delete is pending in the system. The host must not be published in the zone. All requests must be rejected. May only be set by Registry.

10. PendingTransfer [EppHost.STATUS_PENDING_TRANSFER]: the subordinate domain object for this nameserver is in PendingTransfer status and the request transfer is pending.

Prohibited status combinations are:
1. The OK status must not be combined with any other status.

2. The PendingDelete status must not be combined with either ClientDeleteProhibited or ServerDeleteProhibited.

9.3. Etats des contacts

1. ClientDeleteProhibited [EppContact.STATUS_CLIENT_DELETE_PROHIBITED]: A Registrar request to delete the object must be rejected. May only be set by the sponsoring Registrar.

2. ServerDeleteProhibited [EppContact.STATUS_SERVER_DELETE_PROHIBITED]: Any request to delete the object must be rejected. May only be set by Registry

3. ClientTransferProhibited [EppContact.STATUS_CLIENT_TRANSFER_PROHIBITED]

4. A Registrar request to transfer the object must be rejected. May only be set by the sponsoring Registrar.

5. ServerTransferProhibited [EppContact.STATUS_SERVER_TRANSFER_PROHIBITED]: Any request to transfer the object must be rejected. May only be set by Registry.

6. ClientUpdateProhibited [EppContact .STATUS_CLIENT_UPDATE_PROHIBITED]: A Registrar request to update the object must be rejected. May only be set by the sponsoring Registrar. The only update allowed is to remove the ClientUpdateProhibited status.

7. ServerUpdateProhibited [EppContact.STATUS_SERVER_UPDATE_PROHIBITED]: Any request to update the object must be rejected. May only be set by Registry. The only update allowed is to remove the ServerUpdateProhibited status.

8. Linked [EppContact.STATUS_LINKED]: The Object has at least one active association with another object, such as a Domain object. May only be set by Registry

9. OK [EppContact.STATUS_OK]: No other statuses set. May only be set by Registry.

10. PendingCreate [EppContact.STATUS_PENDING_CREATE]: A create request is successful, but the formal create is pending in the system. May only be set by Registry.

11. PendingUpdate [EppContact.STATUS_PENDING_UPDATE]: An update request is successful, but the formal update is pending in the system. May only be set by Registry.

12. PendingDelete [EppContact.STATUS_PENDING_DELETE]: A delete request is successful, but the formal delete is pending in the system. All requests must be rejected. May only be set by Registry.

13. PendingTransfer [EppContact.STATUS_PENDING_TRANSFER]: A transfer request has been received for the object, and completion of the request is pending. The delete, update, and renew commands must be rejected. May only be set by Registry.

Prohibited status combinations are:

1. The OK status may only be combined with the Linked status. It must not be combined with any other status.

2. The PendingDelete status must not be combined with either ClientDeleteProhibited or ServerDeleteProhibited.

3. The PendingTransfer status must not be combined with either ClientTransferProhibited or ServerTransferProhibited

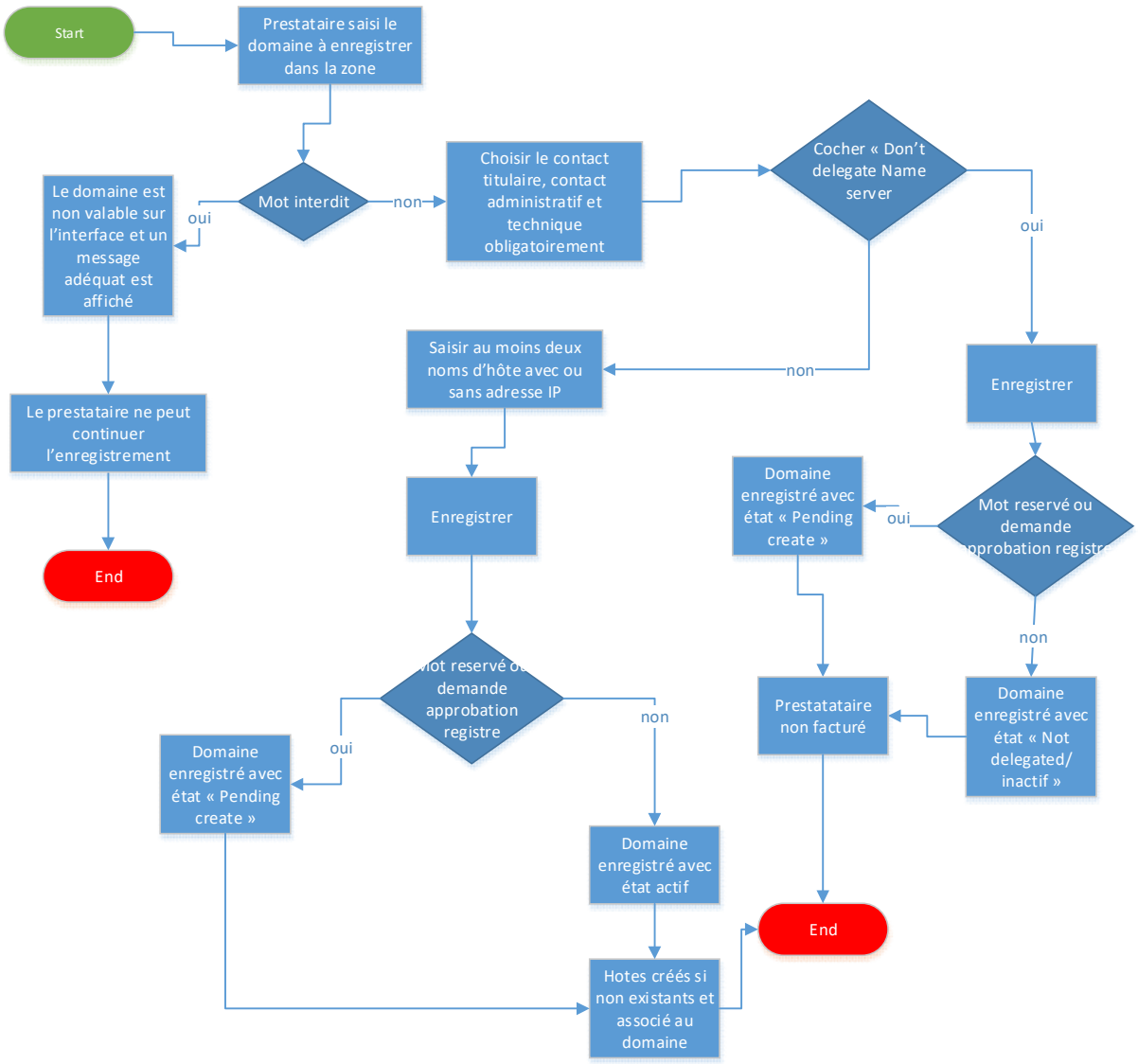
10. Annexe 4: Liste des workflow

10.1. Enregistrement Domaine

La plateforme offre différentes manières d'enregistrer un domaine selon le besoin du client. Cette partie décrit quatre scénarios possibles d'enregistrement qui dépendent du paramétrage au préalable effectué par le registre à savoir le paramétre état domaine à l'enregistrement sur l'interface de création zone et la restriction de noms appliquée à une zone (termes interdits, termes qui demandent une approbation)

10.1.1. Cas1 : Etat domaine à l'enregistrement : Actif

Le scénario par défaut d'enregistrement d'un domaine correspond au schéma ci-dessous. Cette procédure requière un paramétrage de la zone par le registre. Exemple : Créer une zone « net.ma » avec l'état du domaine à l'enregistrement « Actif ».



Ce cas de figure peut être appliqué aux zones .ma , net.ma et co.ma et org.ma

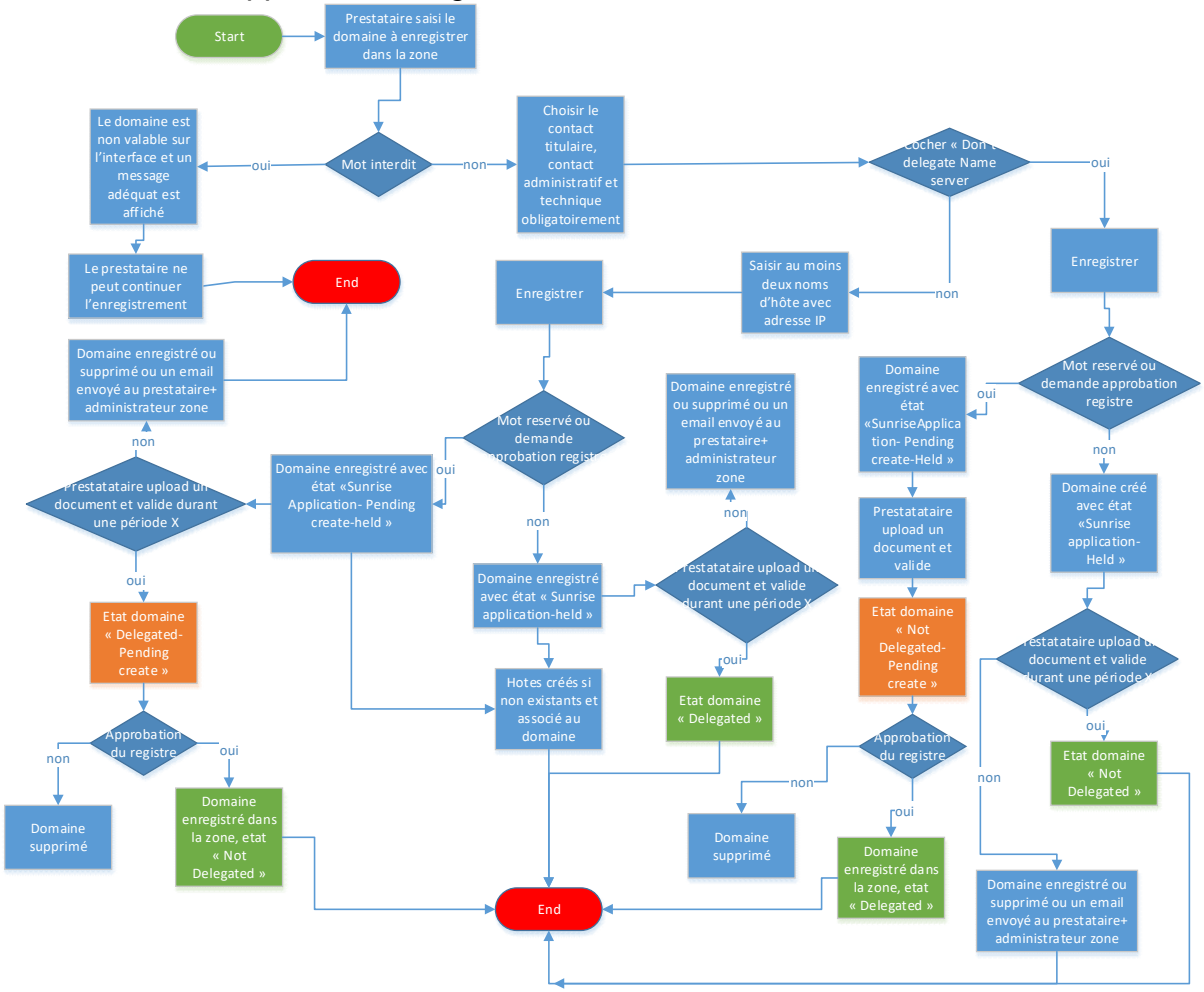
10.1.2. Cas2 : Etat domaine à l'enregistrement : Rolling Sunrise avec approbation prestataire et présentation document obligatoire

Le deuxième scénario d'enregistrement d'un domaine correspond au schéma ci-dessous. Cette procédure requière un paramétrage de la zone par le registre. Exemple : Créer une zone « .ma » avec l'état du domaine à l'enregistrement « Rolling sunrise ». Ce type d'état domaine engendre une configuration supplémentaire de deux paramètres :

1. Approbation prestataire : donne la possibilité au prestataire d'approuver ses propres domaines créés
2. Présentation de document obligatoire : oblige l'approbateur configuré (prestataire ou registre) de joindre un document justificatif pour pouvoir valider le domaine.

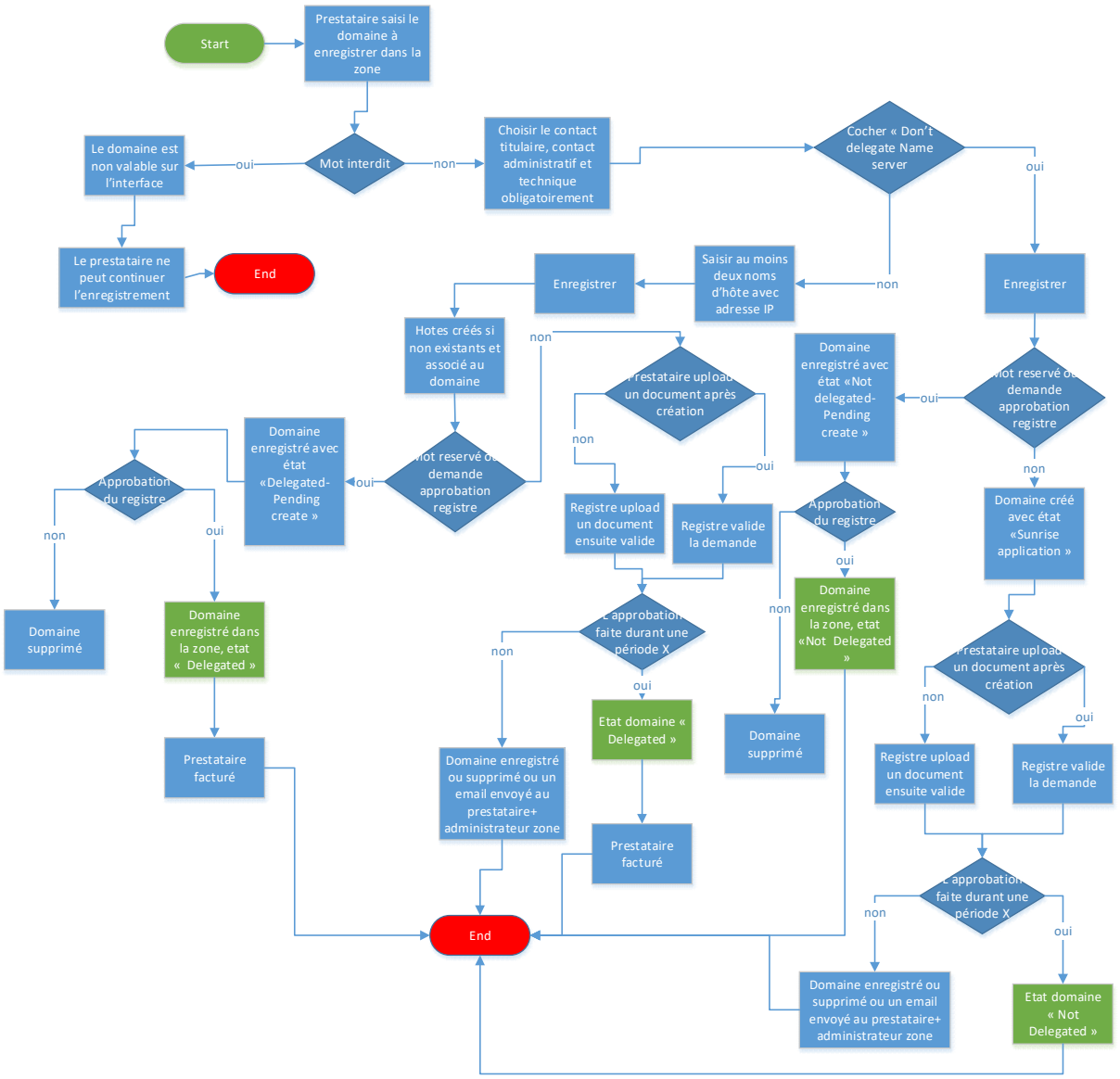
Les combinaisons possibles de ces paramètres font l'objet des trois scénarios présentés dans les paragraphes suivants.

Le schéma ci-dessous correspond au cas où le prestataire présente obligatoirement un document et approuve l'enregistrement



10.1.3. Cas3 : Etat domaine à l'enregistrement : Rolling sunrise sans approbation prestataire et présentation document obligatoire

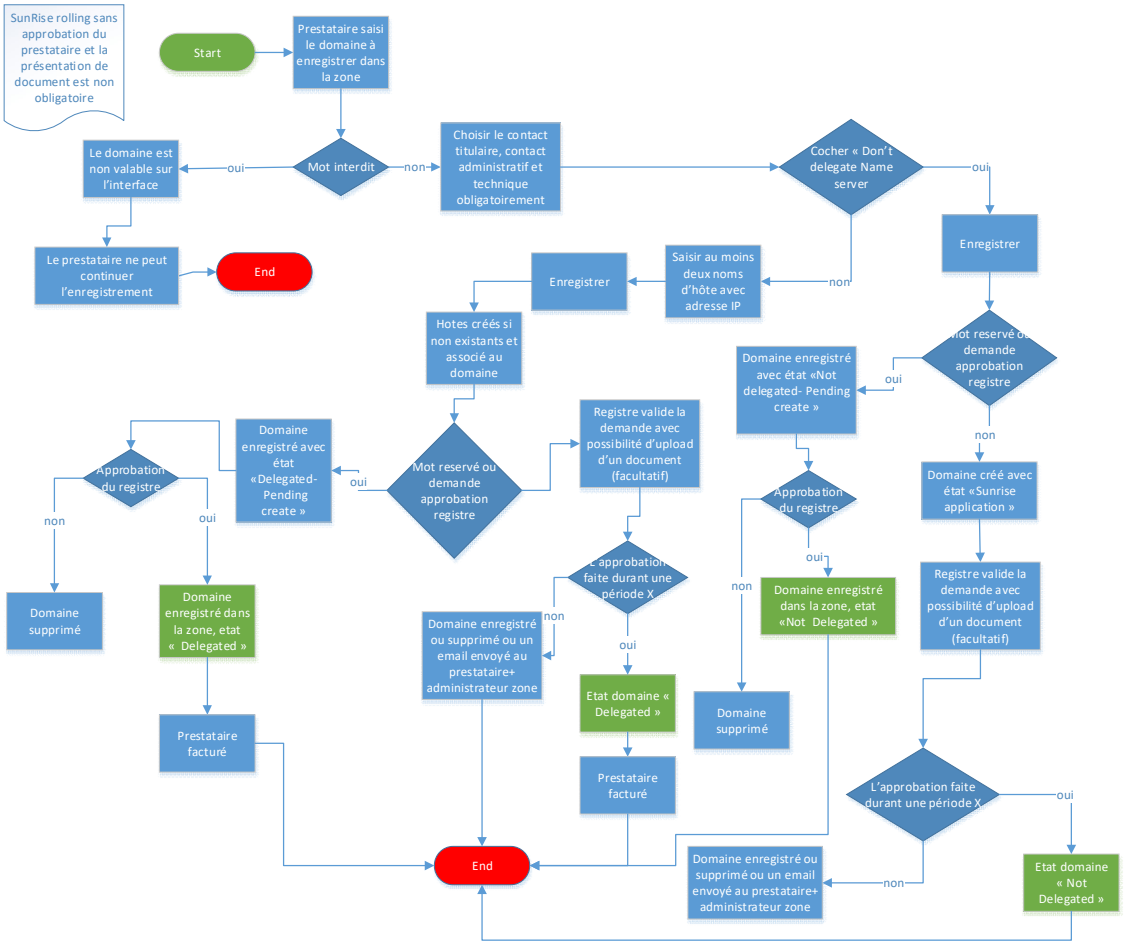
Le schéma ci-dessous correspond au cas où le registre présente obligatoirement un document avant d'approuver l'enregistrement



Ce cas de figure pourra être implémenté pour les zones gov.ma, press.ma et ac.ma qui nécessite toujours une approbation de la part du registre

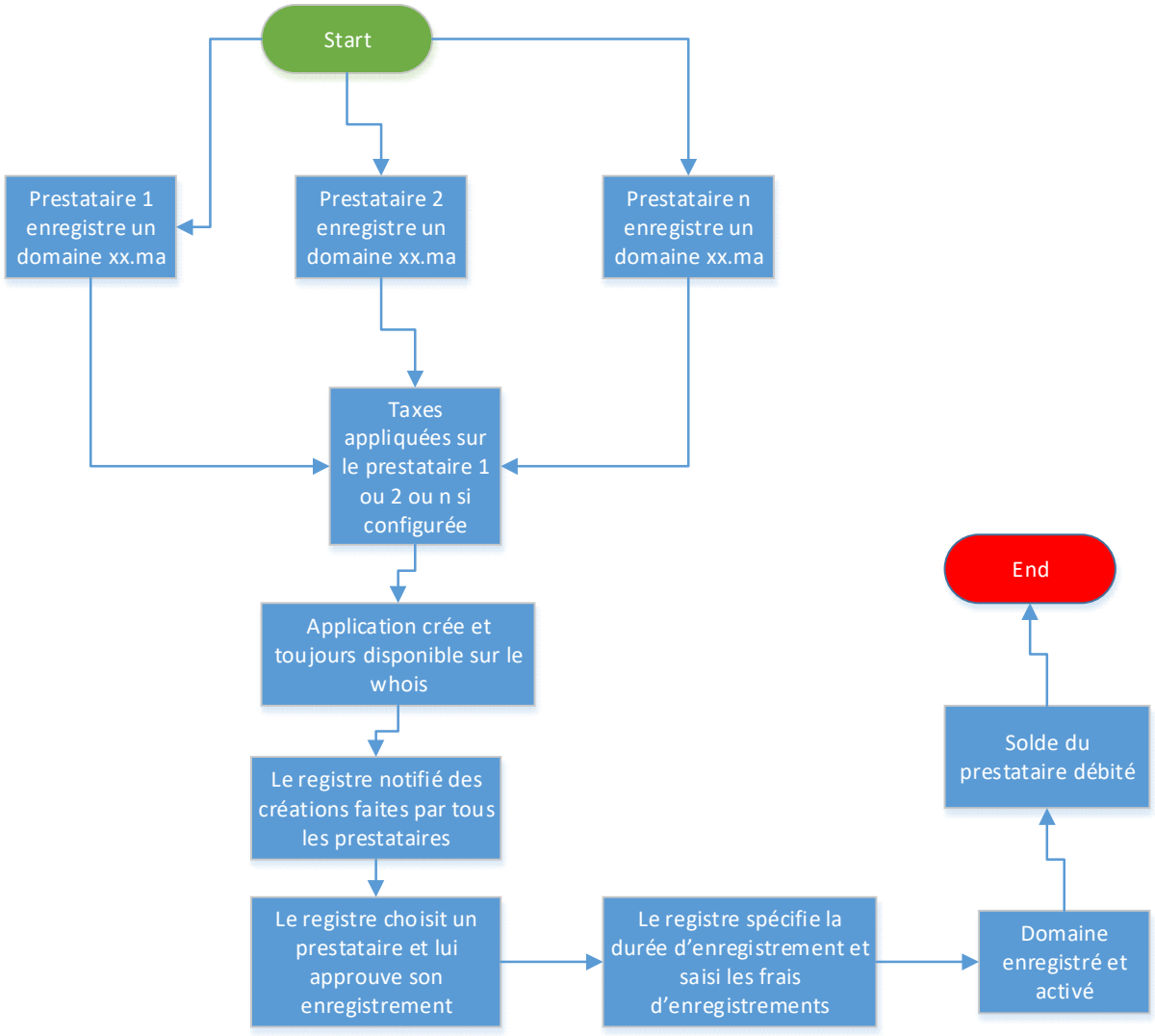
10.1.4. Cas4 : Etat domaine à l'enregistrement : Rolling sunrise sans approbation prestataire et présentation document optionnel :

Le schéma ci-dessous correspond au cas où le registre approuve l'enregistrement de domaine sans obligatoirement présenter un document



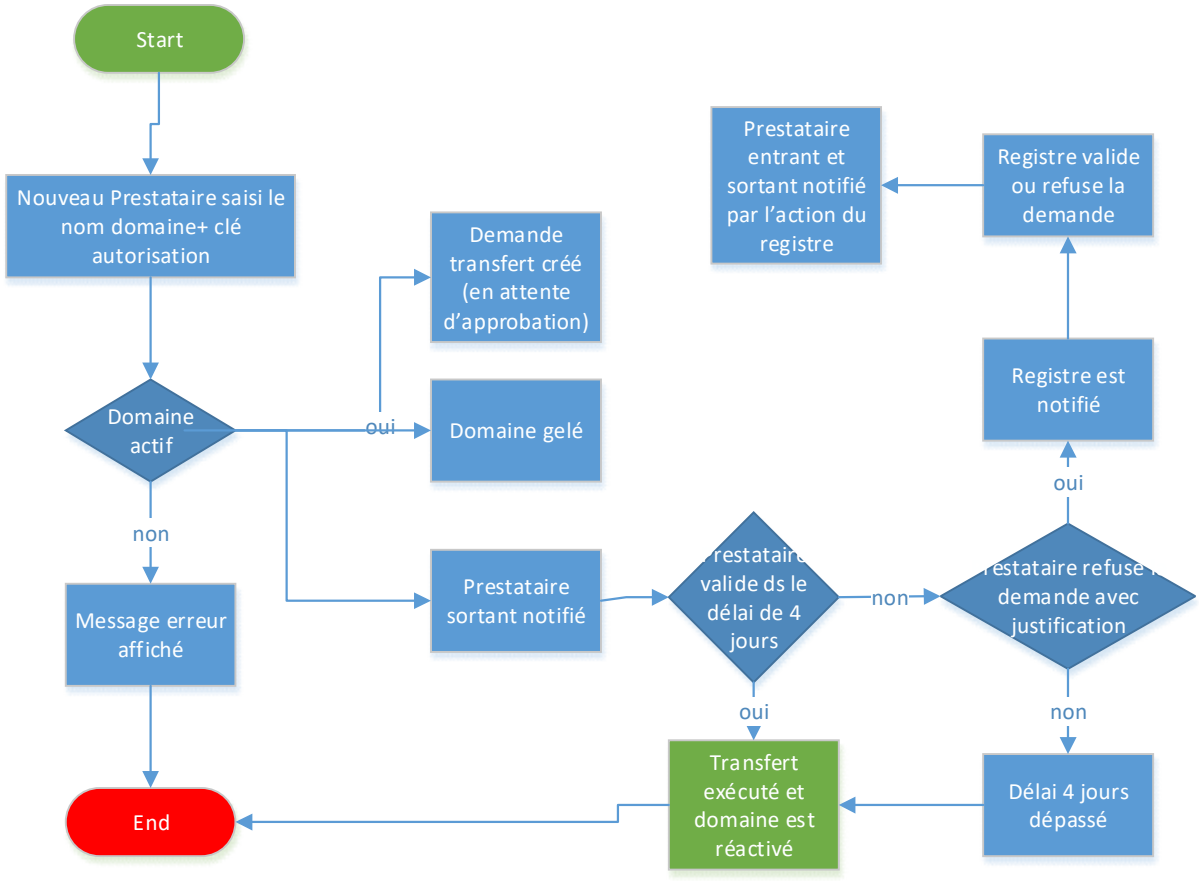
10.1.5. Cas 5 : Block Sunrise

Le block Sunrise est utilisé généralement au lancement d'une nouvelle zone, pour permettre à un ensemble de personnes de réserver un nom de domaine. Plusieurs prestataires peuvent réserver à la fois le même nom de domaine, le registre choisit par la suite lequel sera bénéficiaire du nom de domaine et active le. Ce cas peut être intéressant à appliquer sur « .maghreb » au lancement du projet.
Le schéma ci-dessous décrit les étapes d'enregistrement



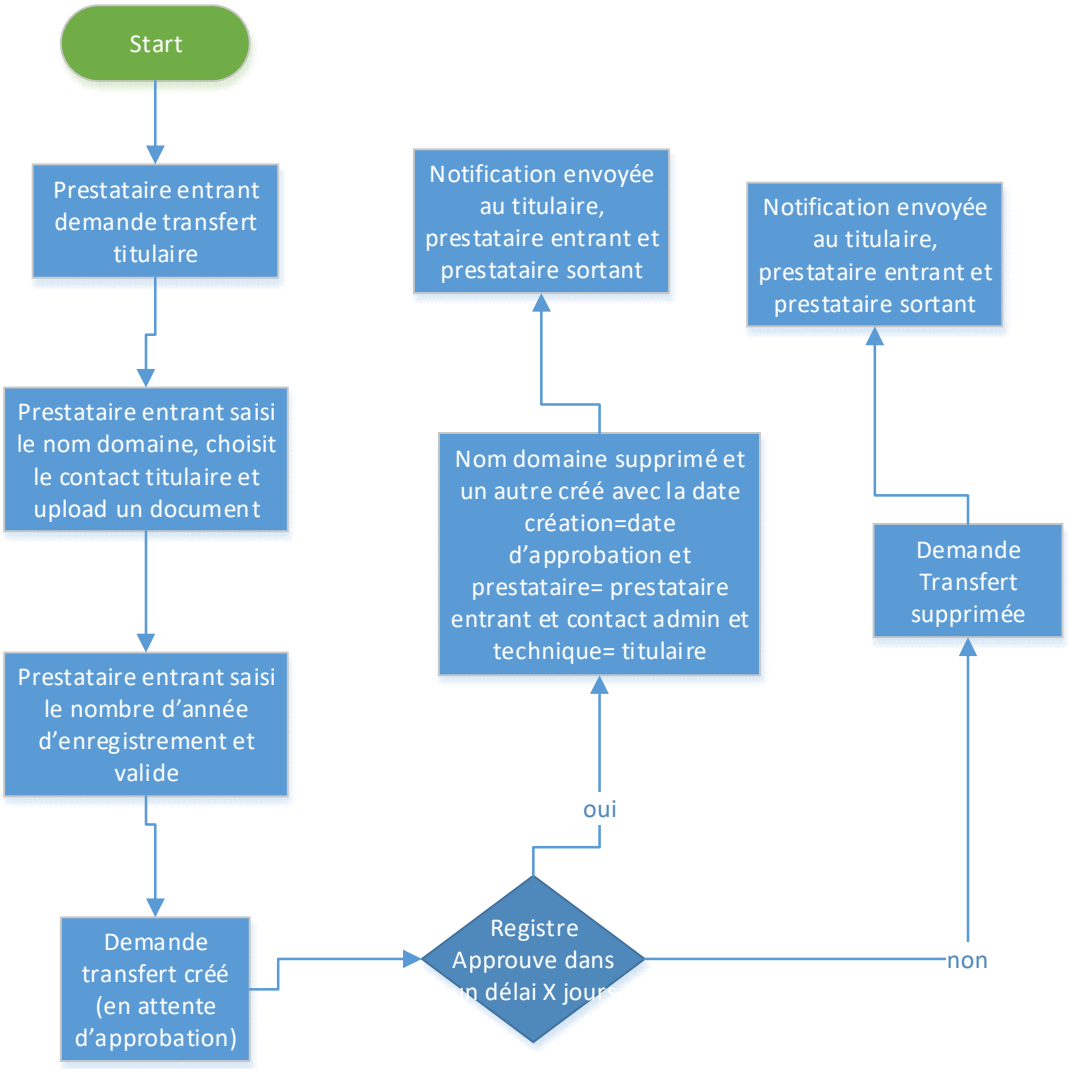
10.2. Changement prestataire

Le processus de changement de prestataire est initié par le nouveau prestataire voulant acquérir un domaine suite à la demande du titulaire. Cette demande est sujet de validation par l'ancien prestataire et le cas échéant par le registre.
Le schéma ci-dessous illustre les étapes de ce processus

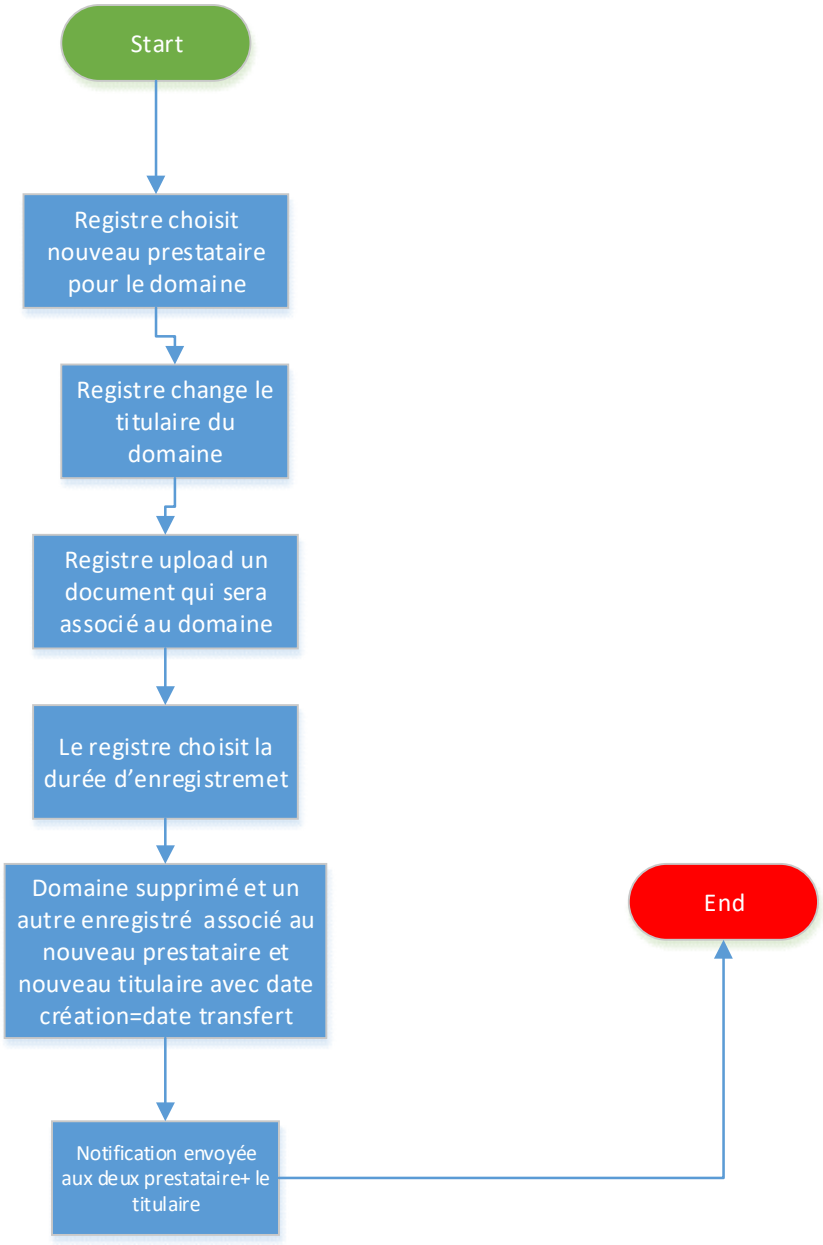


10.3. Transfert titulaire

Les noms de domaine actifs peuvent faire l'objet d'un transfert entre titulaire à la demande du prestataire du titulaire initiale. Cette opération inclut deux processus, le premier concerne le changement de titulaire et le deuxième concerne le changement de prestataire expliqué dans le paragraphe précédent.
Le schéma ci-dessous décrit les échanges entre les différents intervenants du processus de transfert.



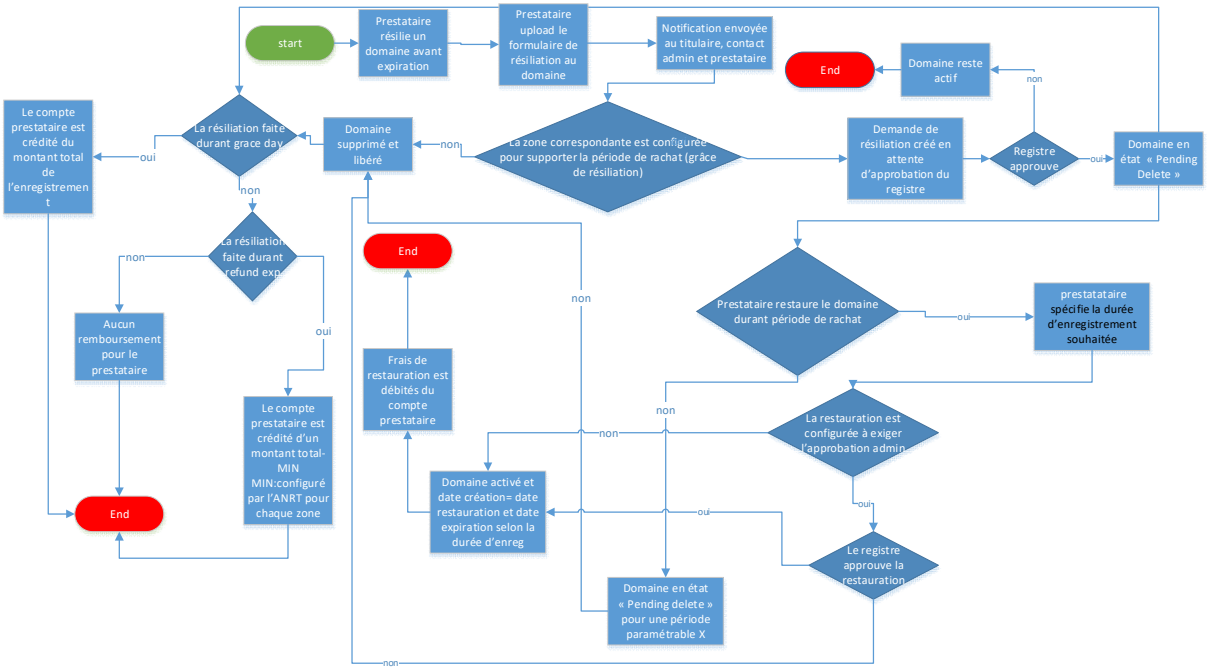
La solution permet de forcer la demande de transfert par le registre. Ci-dessous un schéma qui décrit ce scénario



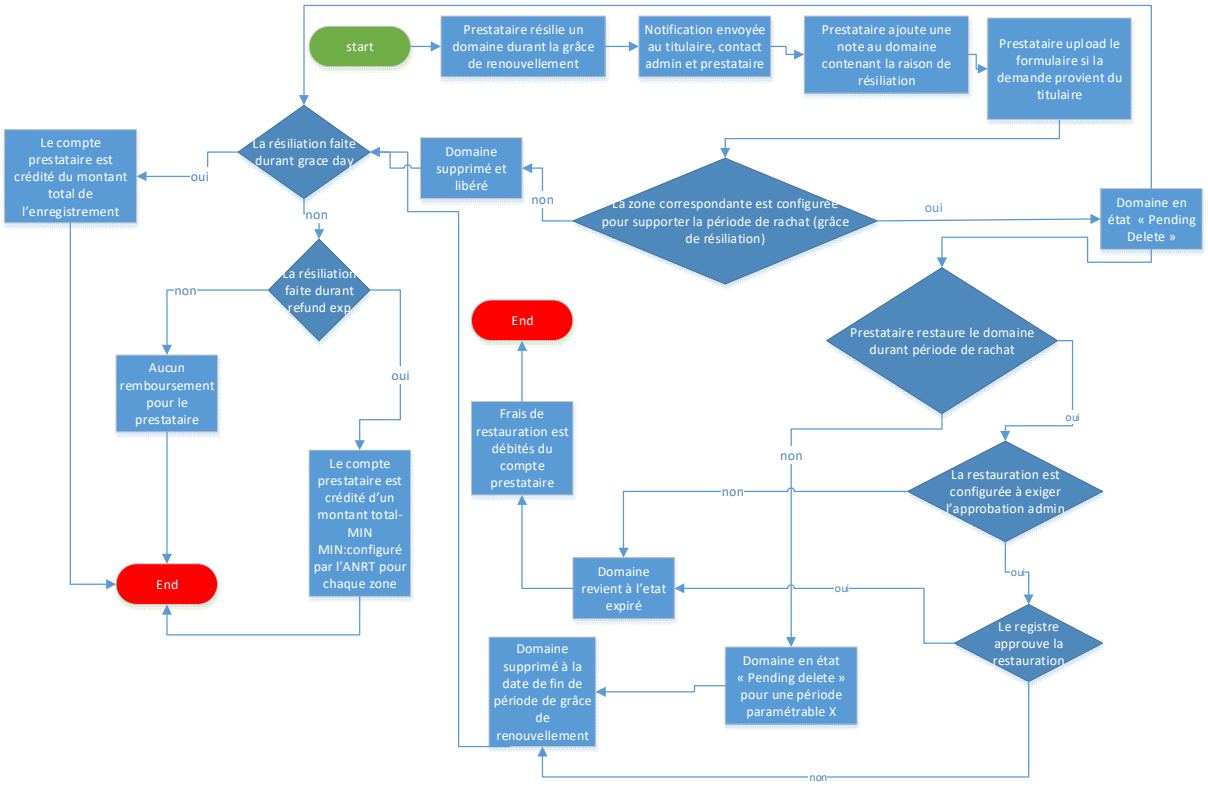
10.4. Résiliation Domaine par le prestataire

La solution SRS offre au prestataire la possibilité de résilier un domaine à tout moment après enregistrement et avant expiration. Différents scénarios sont possibles concernant la validation de la résiliation par le registre, la restauration après résiliation ainsi que les remboursements possibles aux prestataires qui dépendent de la date de résiliation par rapport à celle d'enregistrement.
La résiliation est traitée de deux manières différentes selon le moment de résiliation. Deux cas de figures se présentent :

- Cas 1 : le prestataire résilie son domaine avant expiration. Le schéma ci-dessous présente le workflow adéquat



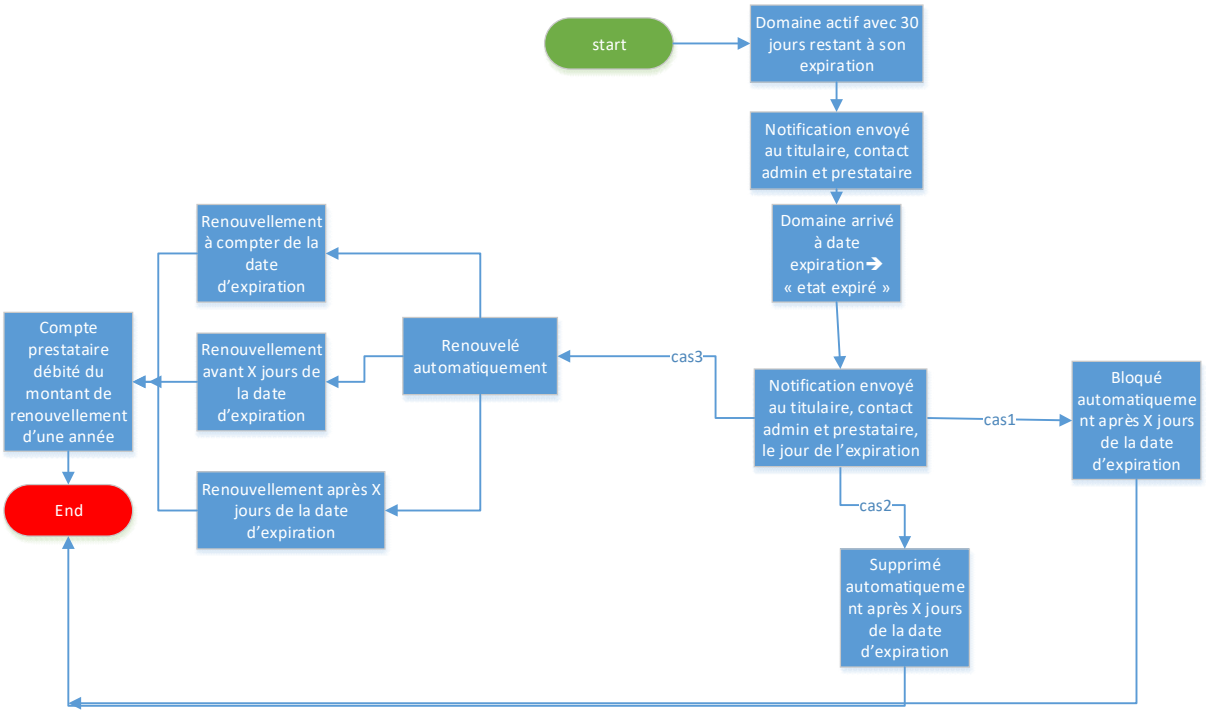
Cas 2 : le prestataire résilie son domaine durant la période de grâce. Le schéma ci-dessous présente le workflow adéquat



10.5. Expiration Domaine

Le registre peut configurer à travers l'interface l'état du domaine après son expiration. La solution permet trois cas de figure, le blocage, la suppression ou le renouvellement

automatique après l'expiration. Ces différentes possibilités sont décrites dans le schéma ci-dessous.

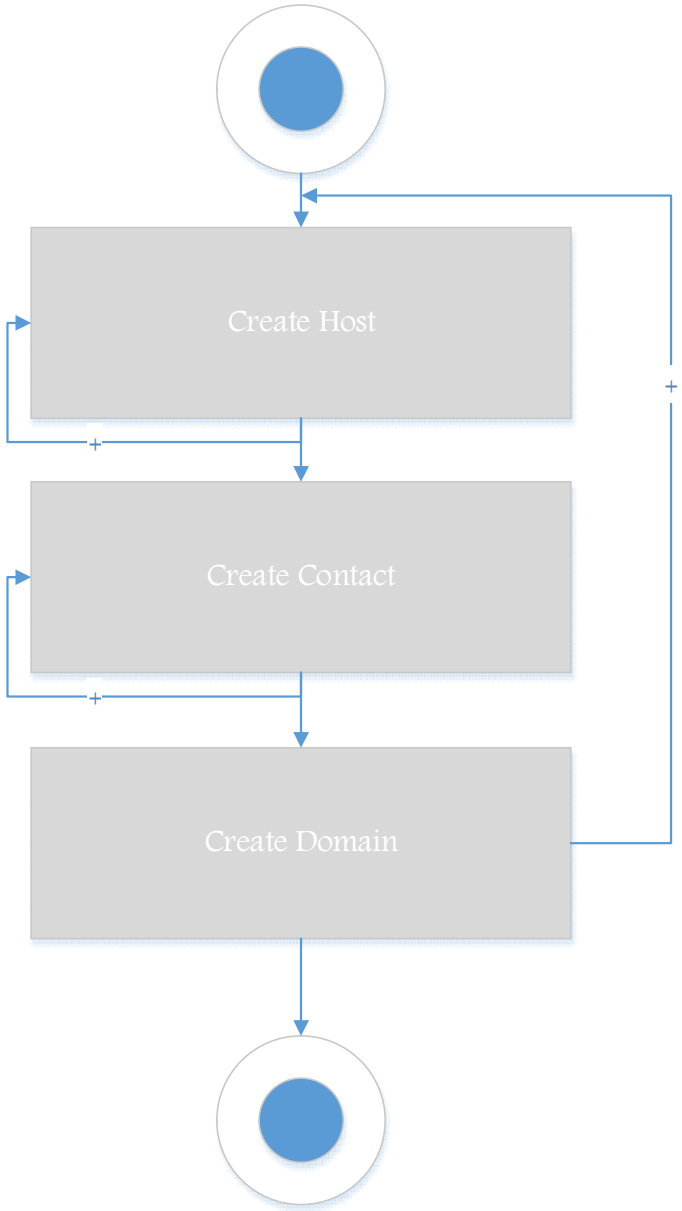


Dans le cadre de notre projet, les cas1 et cas 3 seront implémentés. La troisième option du cas 3 (30 jours après la date d'expiration) sera implémentée.

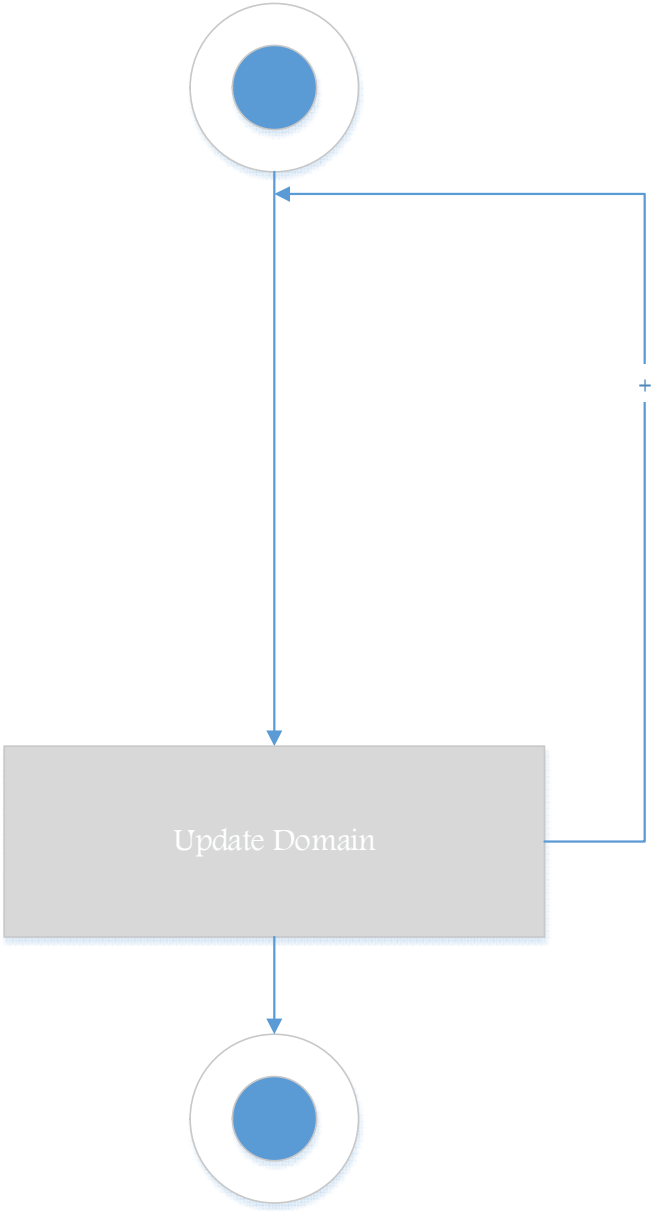
11. Annexe 5: Liste des workflow EPP

11.1. domaine

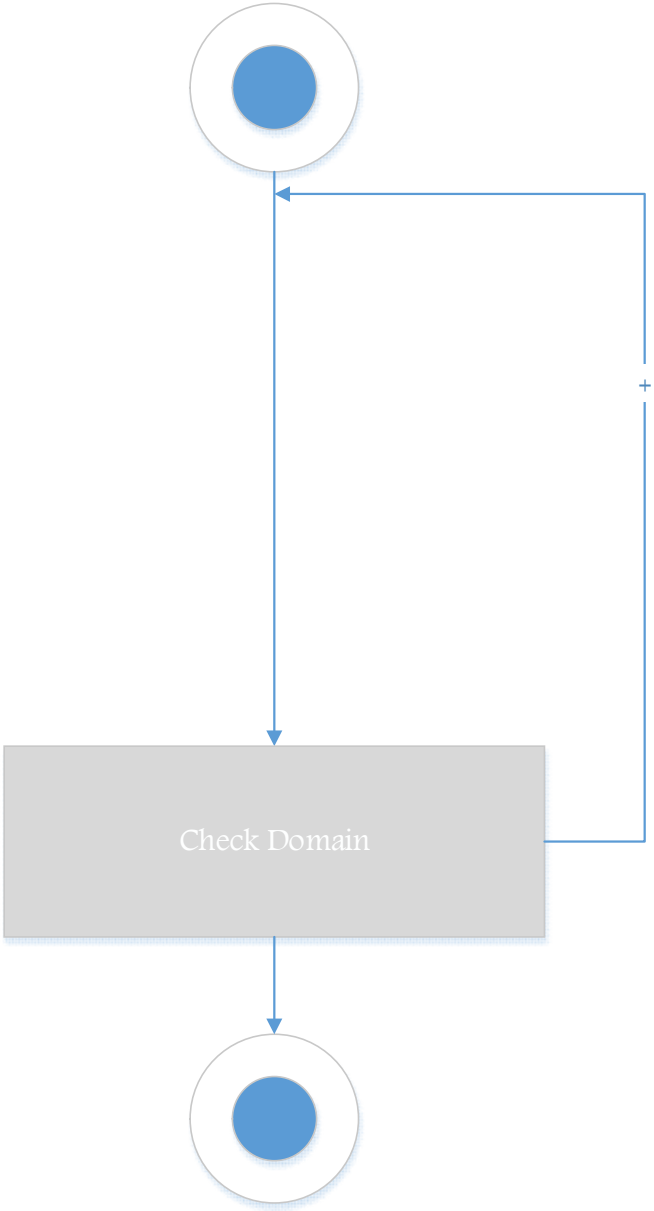
11.1.1. Create domaine



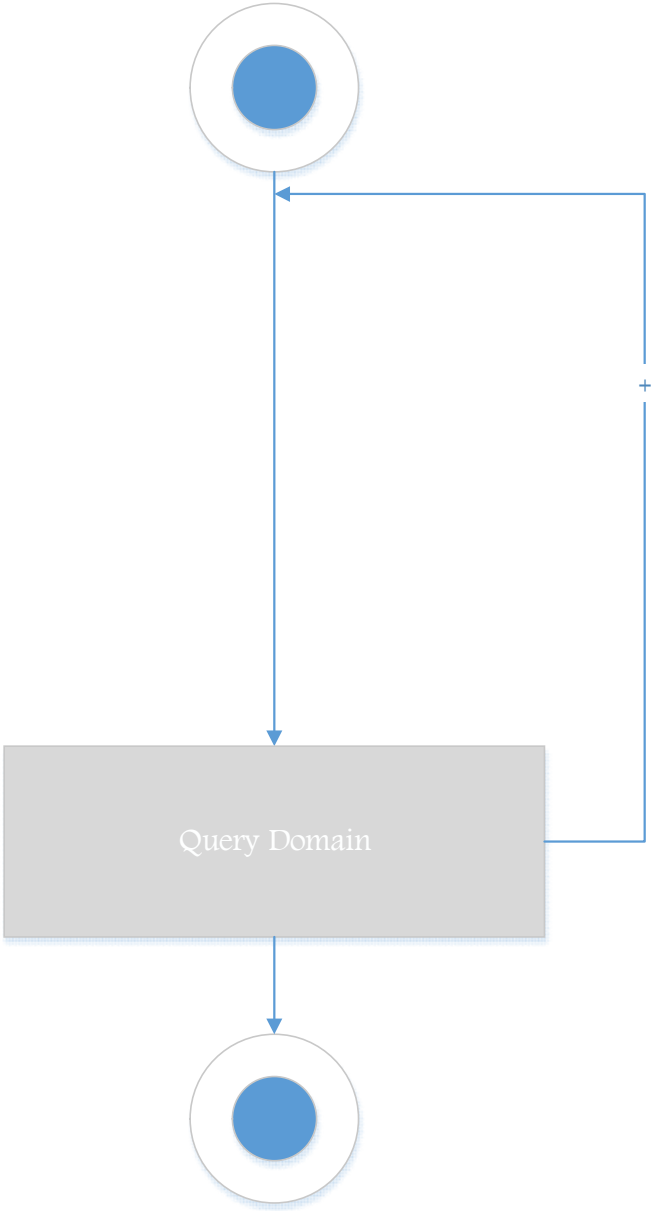
11.1.2. Update domaine



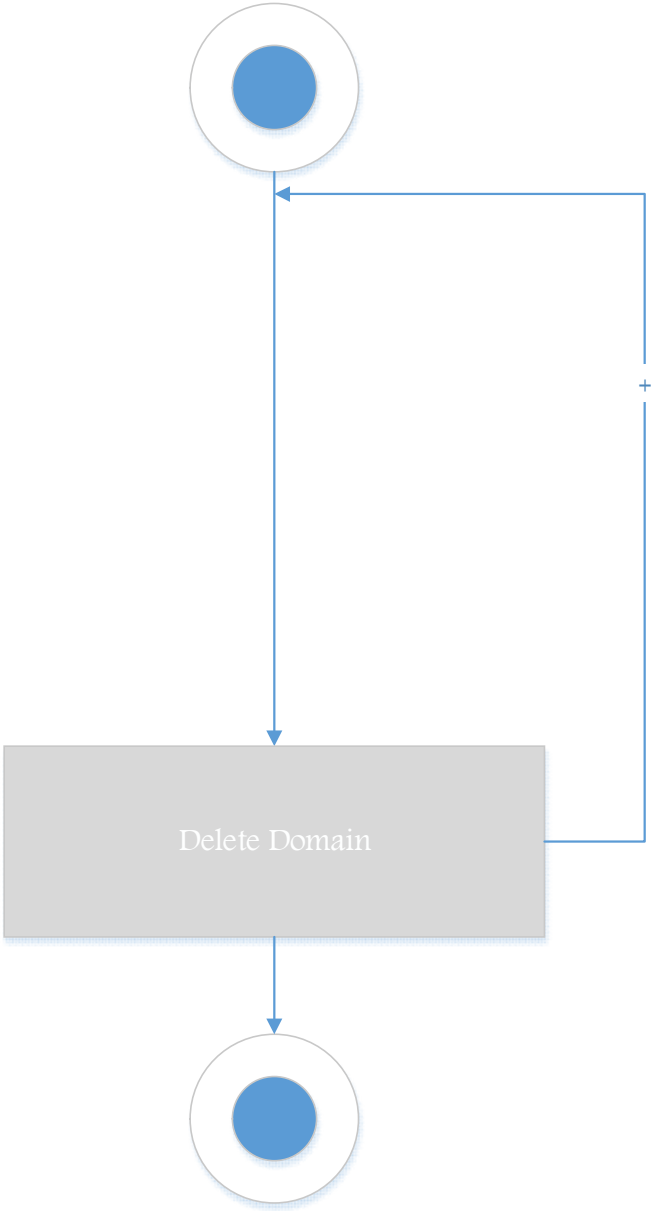
11.1.3. Check domaine



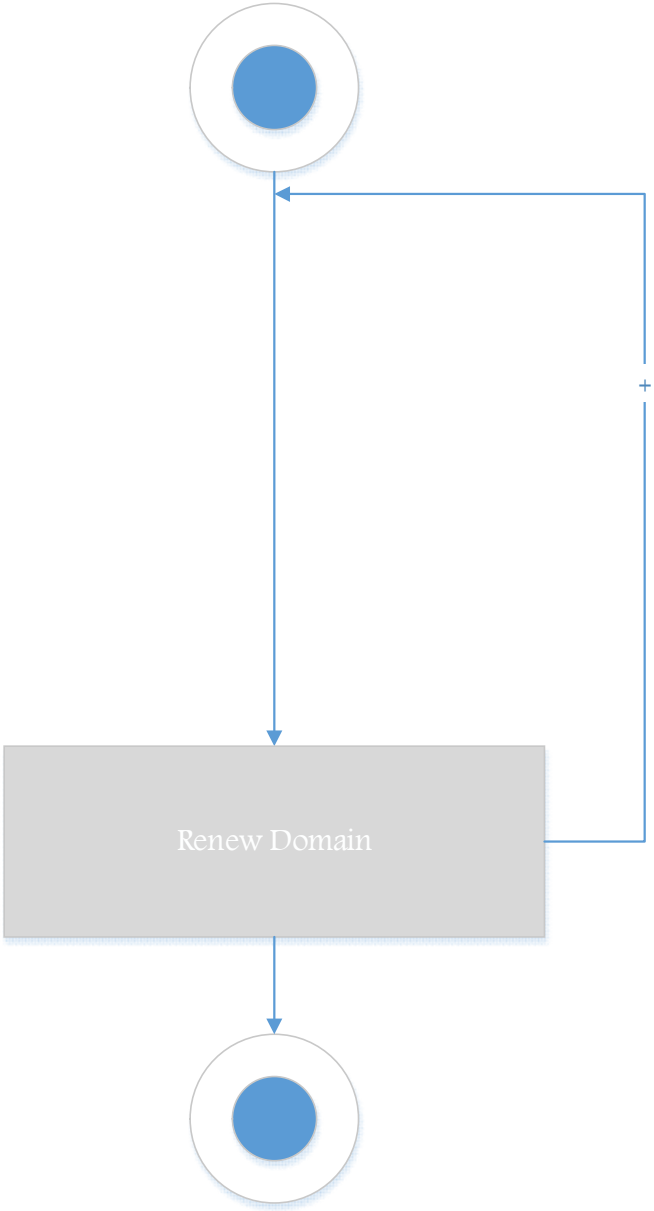
11.1.4. Info domaine



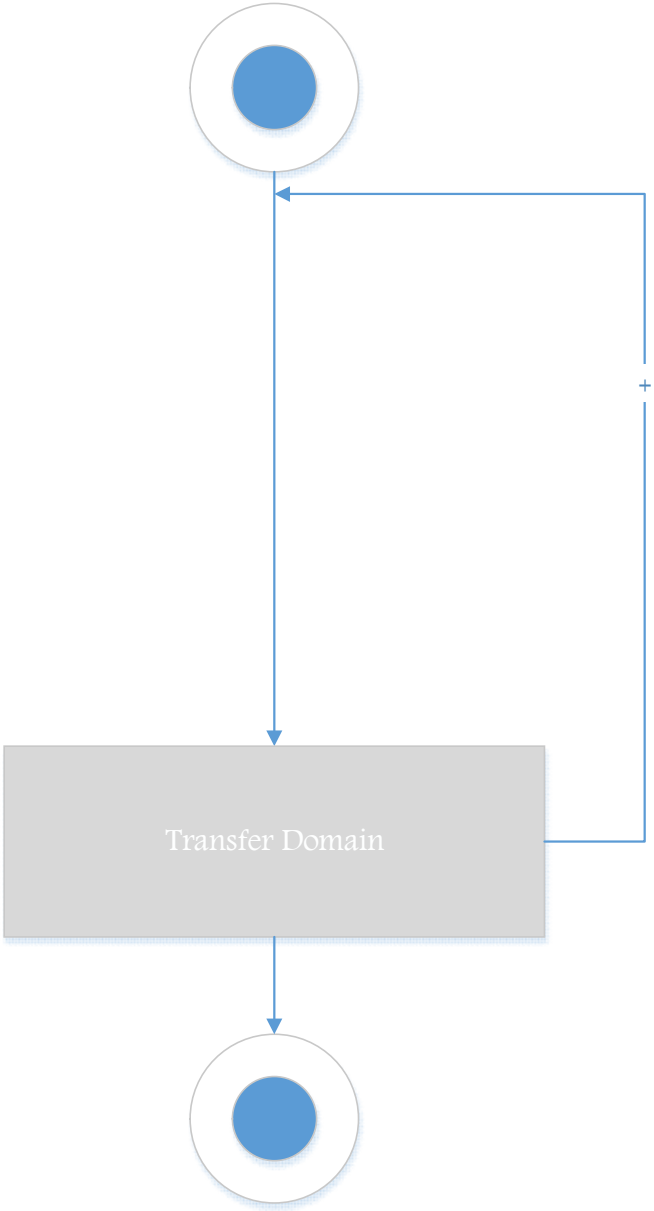
11.1.5. Delete domaine



11.1.6. Renew domaine

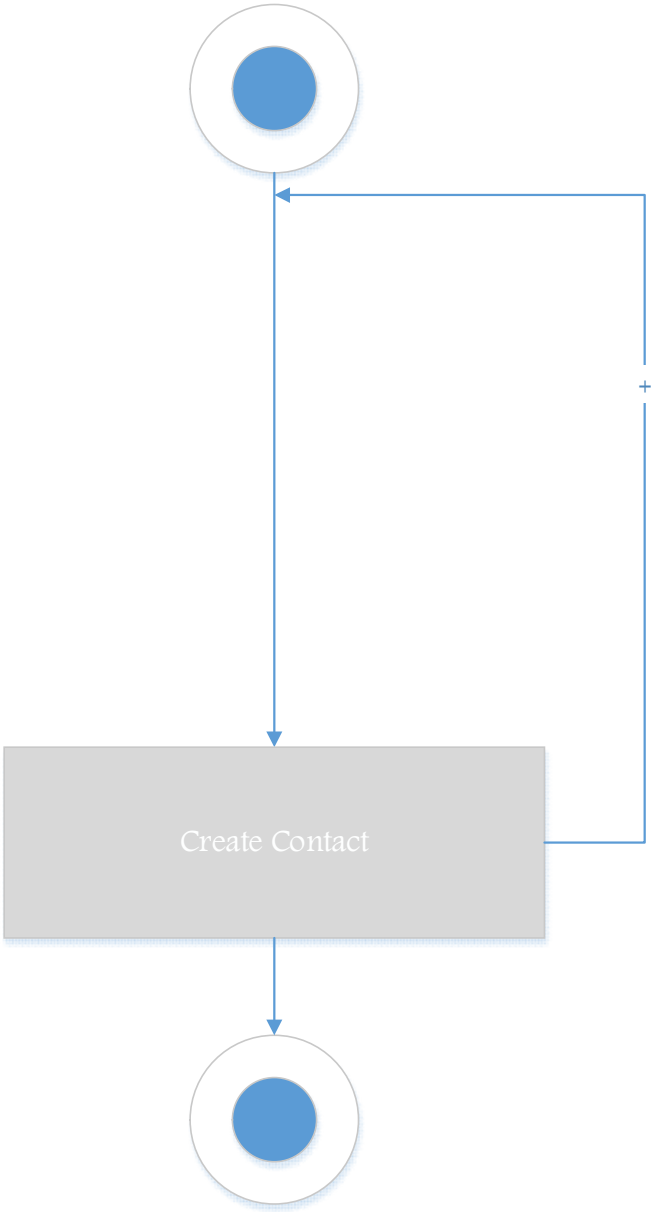


11.1.7. Transfert domaine

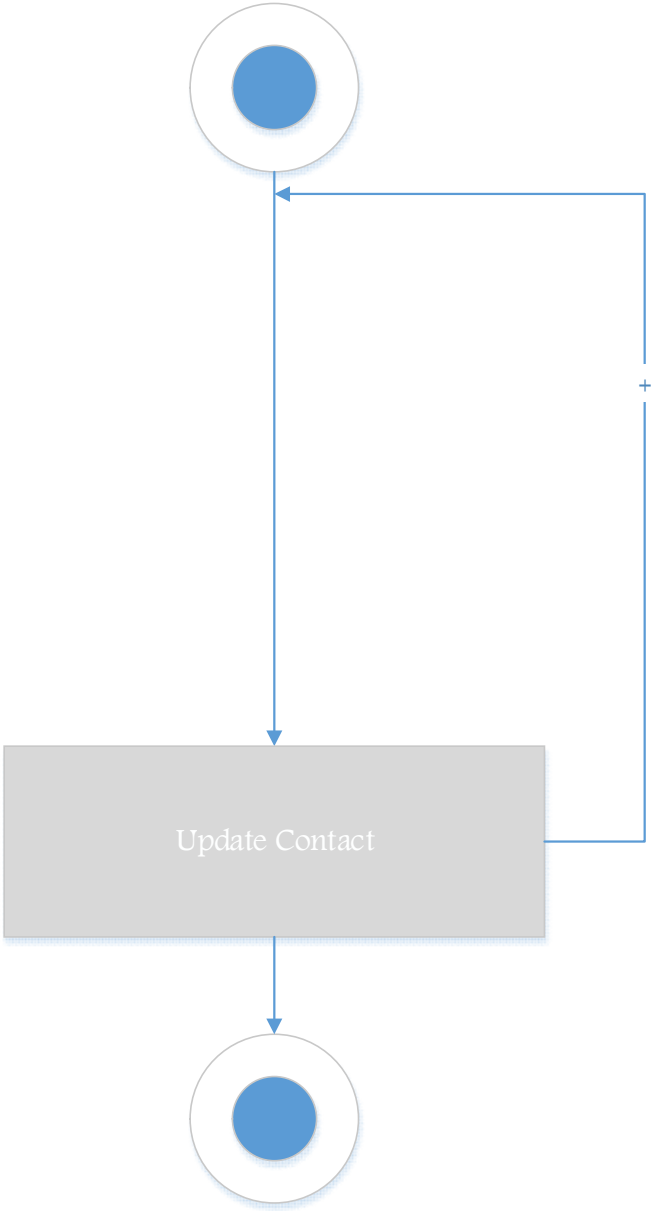


11.2. Création contact

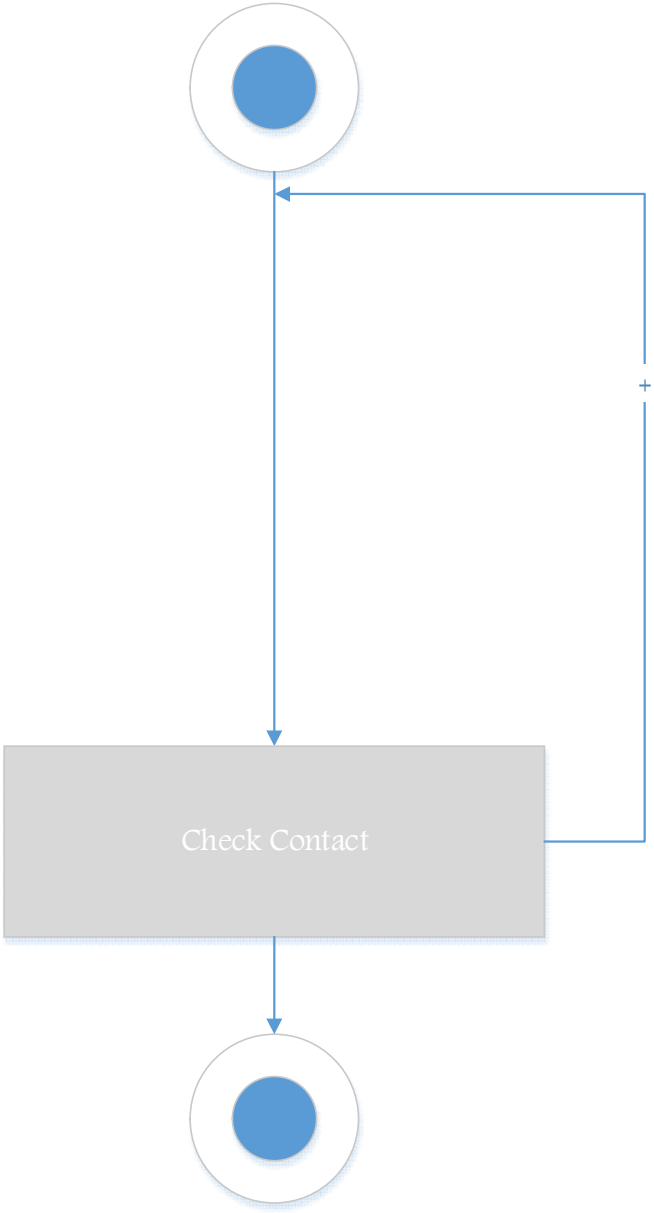
11.2.1. Create contact



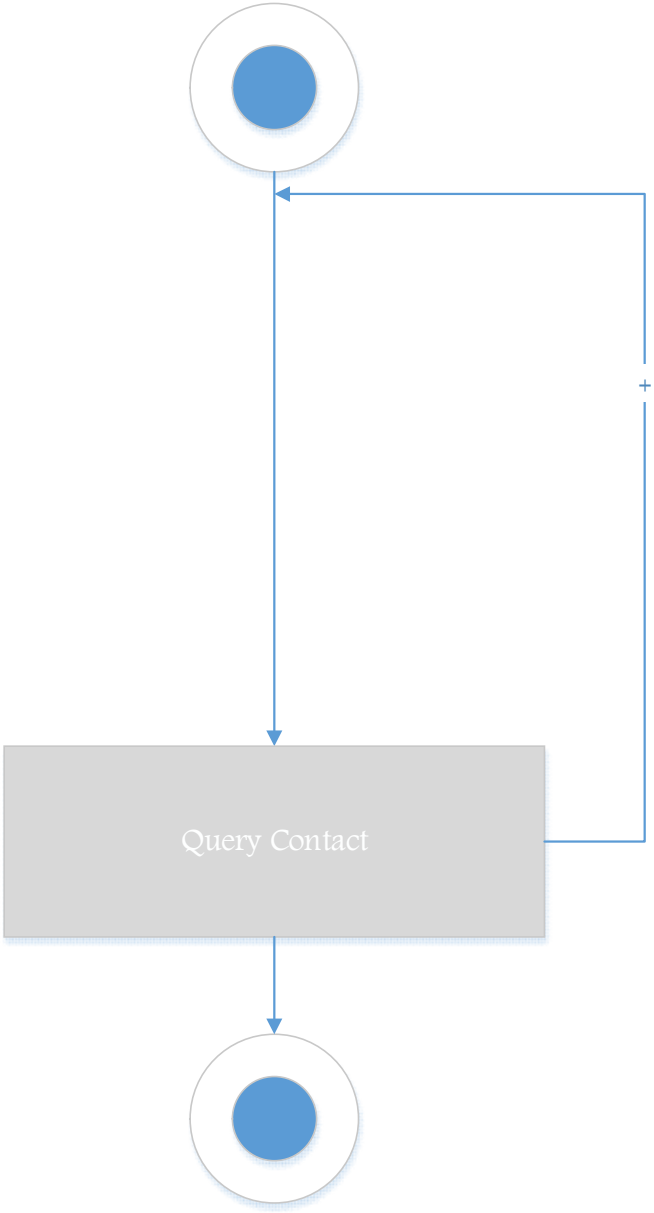
11.2.2. Update contact



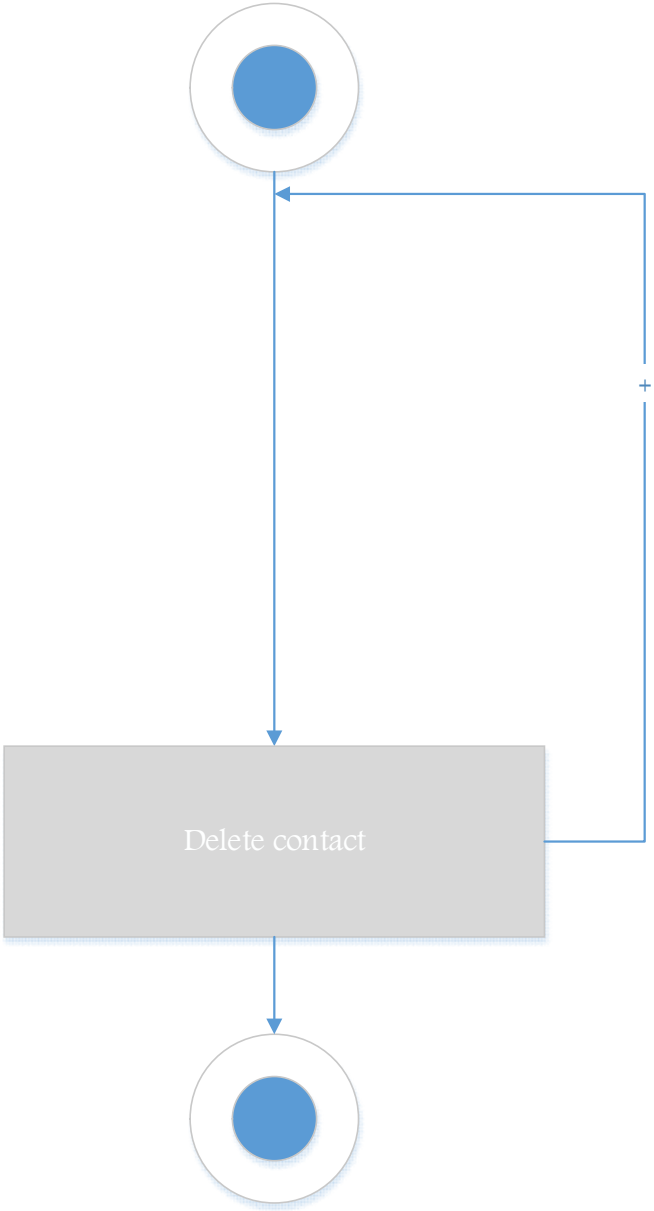
11.2.3. Check contact



11.2.4. Info contact

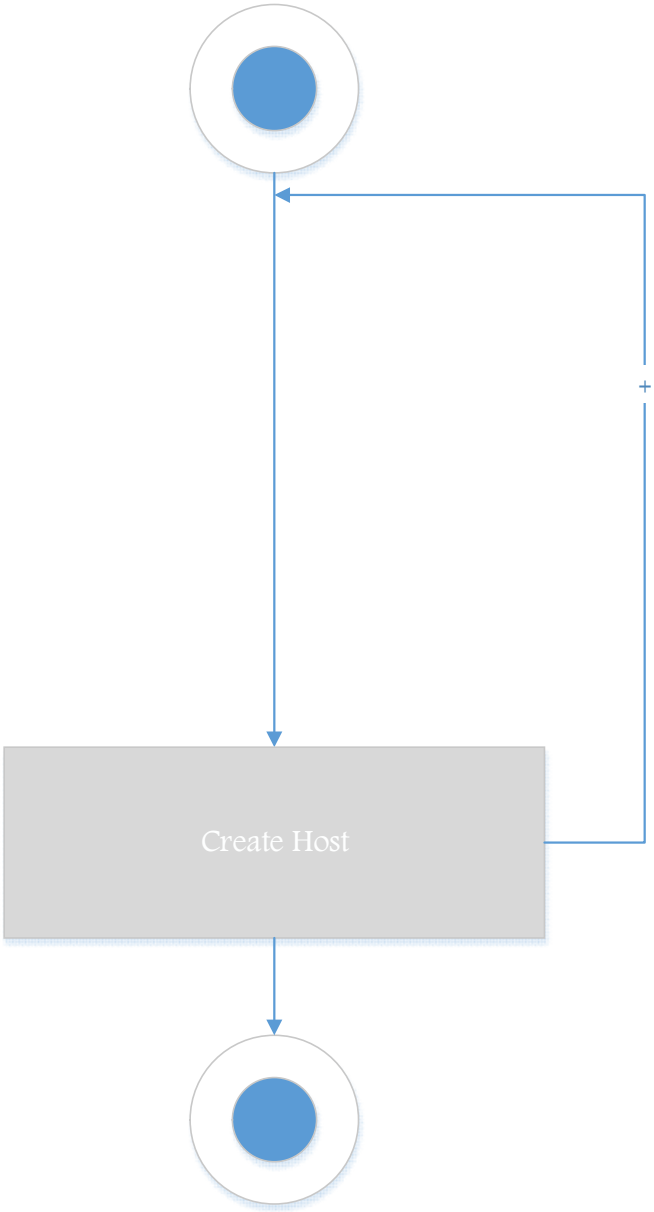


11.2.5. Delete contact

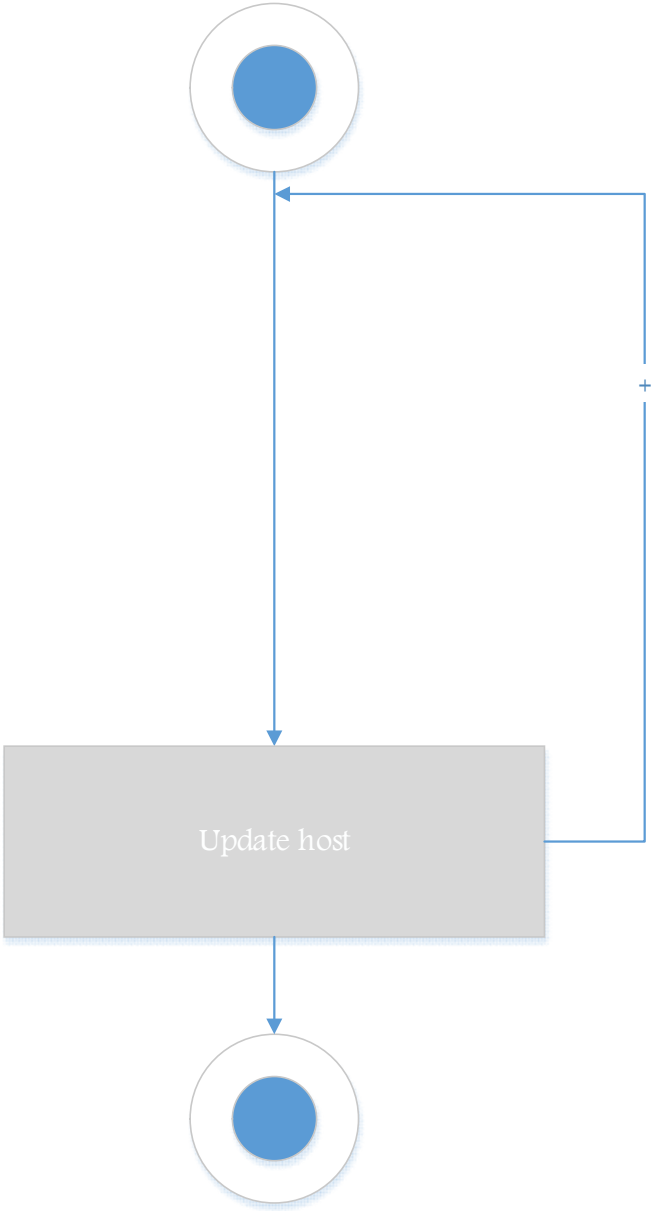


11.3. Création host

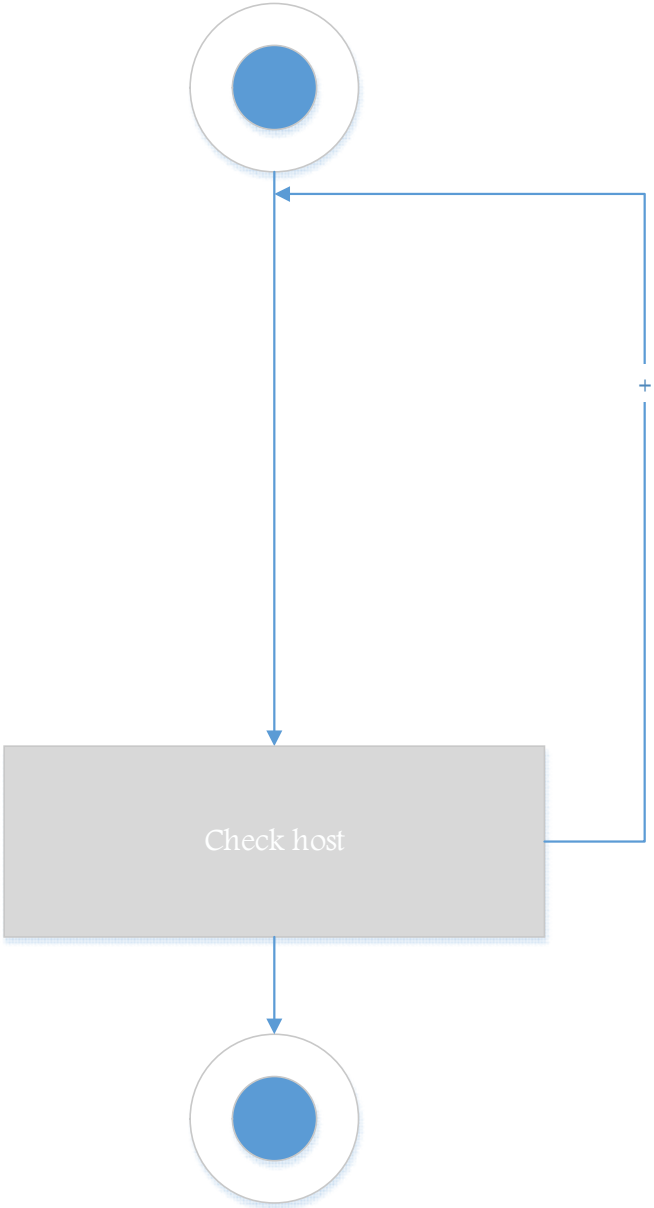
11.3.1. Create host



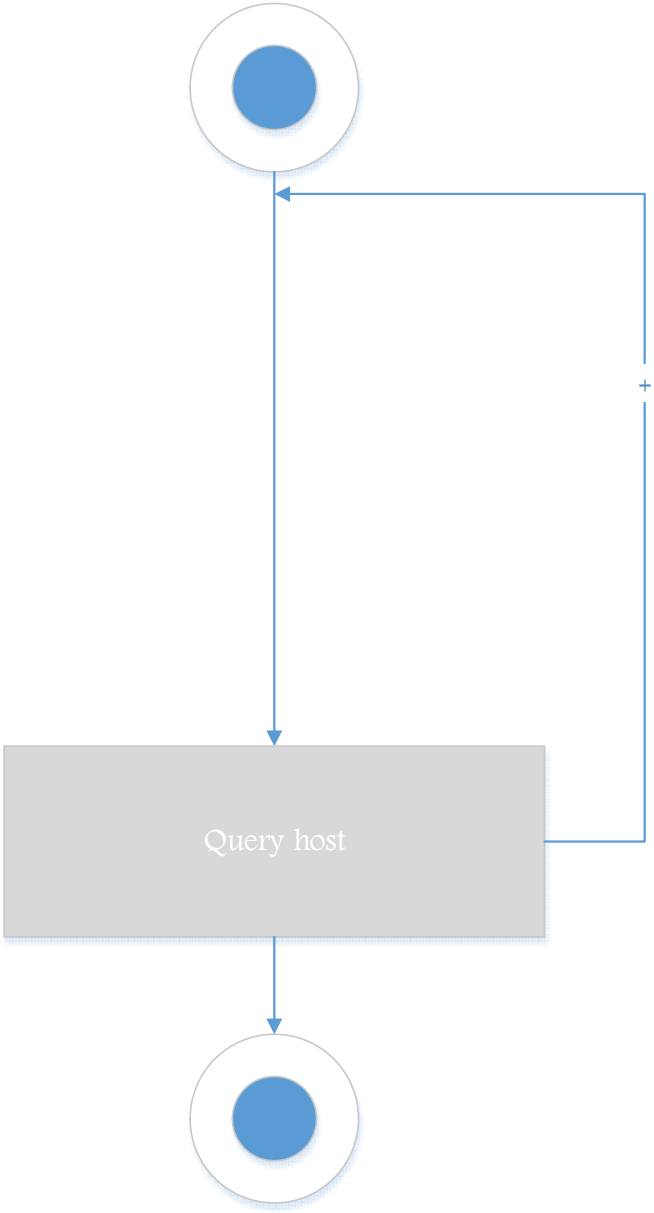
11.3.2. Update host



11.3.3. Check host



11.3.4. Info domaine



11.3.5. Delete domaine