	Ingénieurs 2000	
	Projet Corba	

Projet

Java Mobile Data Security

Cahier des Charges

Fiche d'approbation

Version	Date	Rédacteurs			
0.5	17/03/05	J. GUERS	S. GUINCHARD	M. GEFFROY	D. BARDIN

Fiche de révision

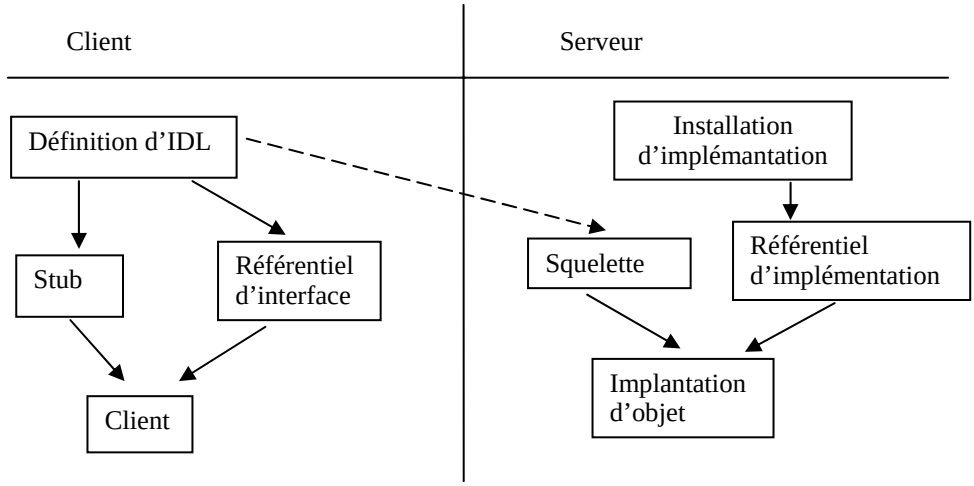
Version	Date	Objet	Auteur
0.1	08/02/05	Création du document	D. BARDIN
0.2	08/02/05	Rédaction de la partie 3.1 & 3.4	J. GUERS
0.3	08/02/05	Rédaction de la partie 3.3	D. BARDIN
0.4	09/02/05	Rédaction de la partie 3.2	M.GEFFROY
0.5	9/02/05	Rédaction de la partie 2 et 3.1	S. GUINCHARD

SOMMAIRE

1 - Architecture Client-Serveur CORBA

1.1 - Architecture Corba

La relation entre objets distribués est une relation client-serveur. Le serveur fournit une interface distante et le client invoque une interface distante.



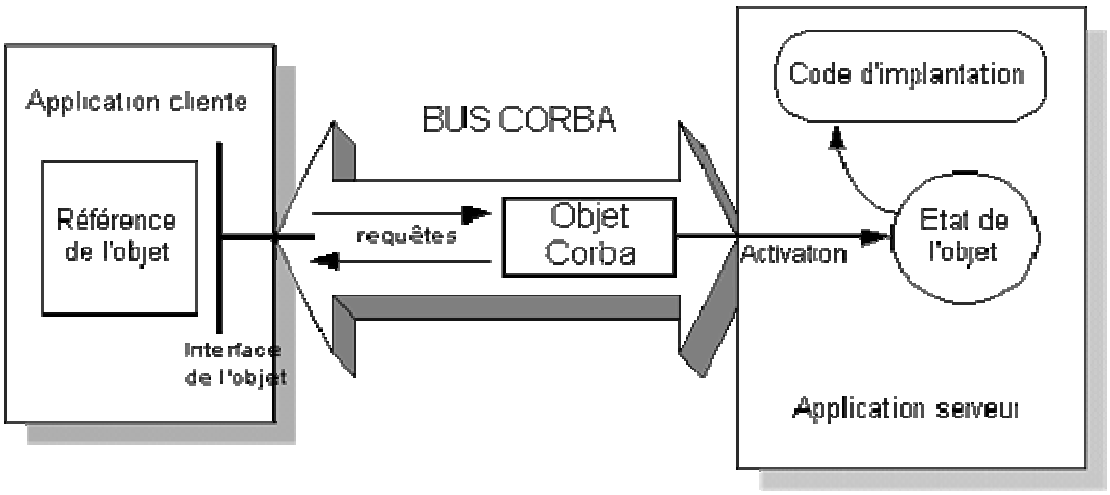
1.1.1 - Coté client

Le client connaît l'interface d'un objet spécifique (il dispose de l'IDL). Il possède une représentation de l'objet distant appelée **stub** générée par l'IDL. Le **stub** transmet la requête à l'objet distant via le bus logiciel (**ORB**). Il s'agit d'une représentation de l'objet distant responsable de la transmission d'une invocation de méthode du talon vers l'objet distant. Il ne s'agit pas d'une copie de l'objet distant.
A travers le **stub** le client voit l'interface de l'objet sans savoir où est l'objet et sans connaître le langage de programmation utilisé pour son implantation.

1.1.2 - Coté Serveur

L'ORB utilise un **squelette** de code pour traduire l'invocation distante en un appel de méthode sur l'objet local. Le **squelette** traduit l'appel et les paramètres dans le format de l'implantation spécifique et appelle la méthode invoquée.
Au retour de la méthode, les résultats (ou erreurs) sont traduits par le squelette et renvoyés au client via l'ORB. Intégration des intercepteurs

1.2 - Le modèle objet client / serveur



L'**application cliente** est un programme qui invoque les méthodes des objets à travers le bus CORBA.
La **référence d'objet** est une structure désignant l'objet CORBA etcontenant l'information nécessaire pour le localiser sur le bus (Interoperable Object Reference IOR).
Interface OMG-IDL, Adresse IP, port, Clé de format libre.
L**'interface de l'objet** est le type abstrait de l'objet CORBA définissant ses opérations et attributs. Celle-ci se définit par l'intermédiaire du langage OMG-IDL.
La **requête** est le mécanisme d'invocation d'une opération ou d'accès à un attribut de l'objet.
Le **bus CORBA** achemine les requêtes de l'application cliente vers l'objet en masquant tous les problèmes d'hétérogénéité (langages, systèmes d'exploitation, matériels, réseaux).

1.3 - Intégration des intercepteurs

Les intercepteurs Corba permettent d'écouter sur l'ORB tous les messages échangés entre le client et le serveur. Chacun possède une implémentation de l'intercepteur. Lorsqu'une trame est acheminée sur l'ORB, l'intercepteur récupère le message, le code et le transmet à son destinataire. L'intercepteur coté serveur récupère ce message et le décode.

Dans notre cas, dès qu'un message est envoyé par le client, l'intercepteur récupère les informations de l'émetteur et les ajoute au message, lorsque le serveur le récupère, il est en mesure d'authentifier le client et de l'autoriser ou non à appeler l'objet distant.

2 - Objectifs

2.1 - Présentation générale des intercepteurs

Un point d'interception est un point d'accrochage dans l'ORB par lequel les services de l'ORB peuvent intercepter le flux normal d'exécution de l'ORB [OMG00]. Le support des opérations déconnectées est donc ici vu comme un service, au même titre que le support des transactions par exemple. La portabilité, au sens où le service est utilisable dans tous les ORB, est obtenue par la définition de trois types d'intercepteurs avec, pour chacun d'entre eux, un ensemble défini de points d'interception.

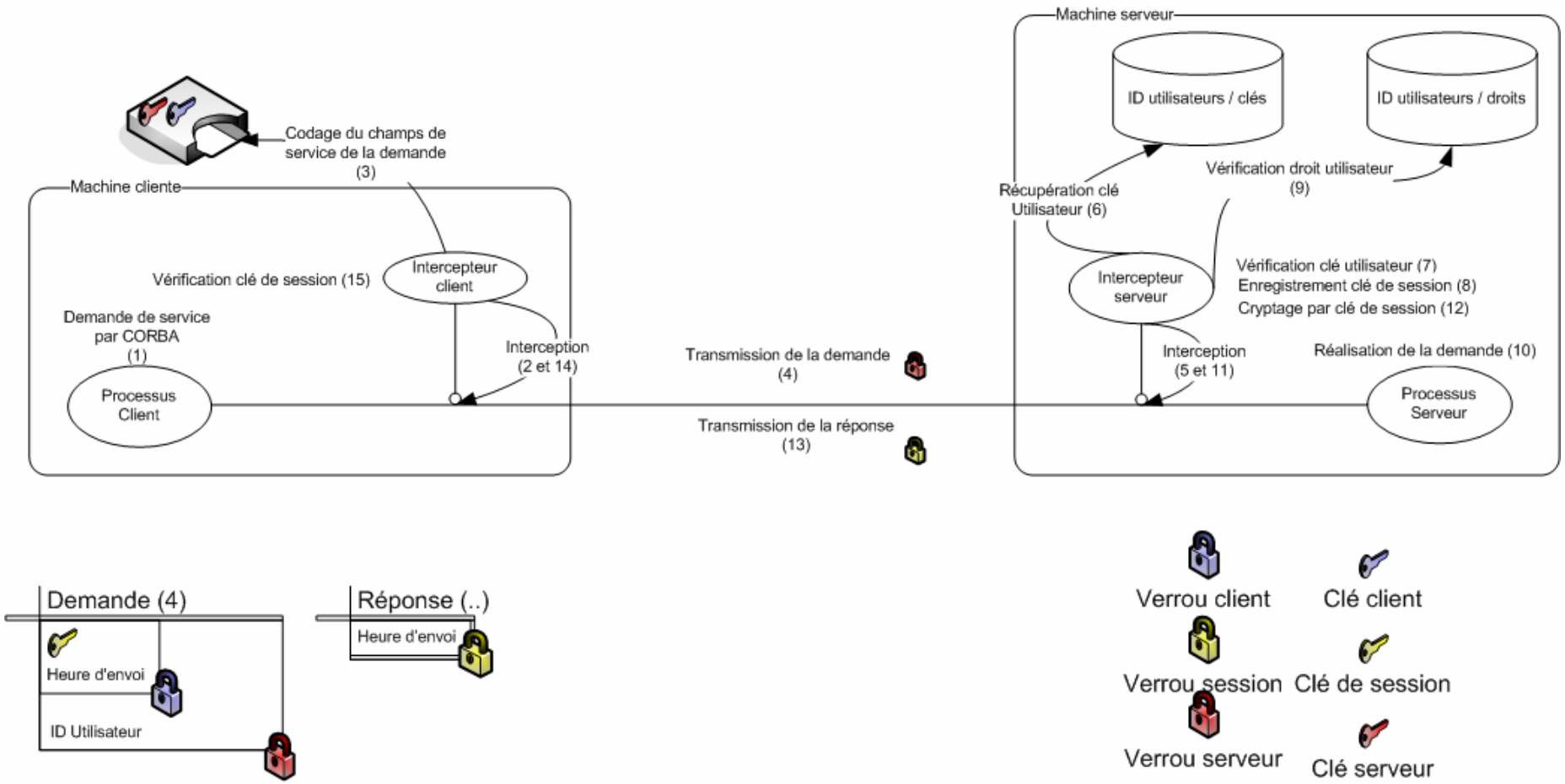
Un intercepteur est construit indépendamment de l'ORB qu'il intercepte. Les premiers points d'interception interviennent lors de la création des références des objets CORBA, nommées en CORBA "IOR". Ces points d'interception définis dans les intercepteurs d'IOR captent en fait les créations d'adaptateurs d'objets (POA) qui créent les références. Ils permettent d'ajouter de nouveaux composants aux IOR pour indiquer que tel objet supporte le mode déconnecté, et pour certains de ces objets, qu'il est possible de créer une copie locale sur le terminal mobile. Ces informations contenues dans les références servent aux deux autres types d'intercepteurs pour déterminer le type de traitement à effectuer.

Les deux autres types d'intercepteurs introduisent des points d'interception lors des envois et des réceptions des requêtes et des réponses par les clients et les serveurs. Les points d'interception créés du côté du client sont définis par les intercepteurs dits "côté client" et ceux créés du côté du serveur par les intercepteurs dits "côté serveur".

La définition d'un tel intercepteur implique que toutes les requêtes et les réponses passant par cet ORB sont interceptées. Chacun de ces points d'interception analyse l'IOR de l'objet correspondant à la requête ou à la réponse. Si cette IOR possède le composant de la politique du mode déconnecté, un traitement spécifique est appliqué. Ces intercepteurs s'échangent des informations en ajoutant un contexte de service aux requêtes et aux réponses.

En outre, un intercepteur côté client est ajouté à l'ORB de l'application sur le terminal mobile. Ces intercepteurs servent à effectuer la création de la copie locale sur le terminal mobile et à commuter de façon transparente entre l'objet distant et la copie locale.

2.2 - Authentification



Les étapes :

L'authentification et l'autorisation d'un utilisateur se dérouleront en 16 étapes :

1. Demande de service par le client
2. Interception par l'intercepteur client
3. Création du service contexte
4. Transmission de la demande
5. Interception par l'intercepteur serveur
6. Récupération de la clé utilisateur
7. Vérification de la clé utilisateur
8. Enregistrement de la clé de session
9. Vérification des droits utilisateurs
10. Réalisation de la demande par le serveur
11. Interception de la réponse
12. Génération du service contexte
13. Transmission de la réponse
14. Interception par l'intercepteur client
15. Vérification de la clé de session
16. Livraison de la réponse

Le cryptage des messages

- Le service contexte de la demande contient un message crypté par un code déchiffrable à l'aide de la clé serveur. Ce message est constitué de l'ID utilisateur et d'un sous-message crypté par un autre code déchiffrable à l'aide de la clé utilisateur. Ce sous-message contient l'heure d'envoi du message et une clé de session.

- Le service contexte de la réponse contient un message crypté par un troisième code déchiffrable à l'aide de la clé de session. Ce message contient l'heure d'envoi de la réponse.

Comment client et serveur sont-ils authentifiés ?

Auprès du serveur, le client peut être identifié lors du déchiffrage du sous message du service contexte. En effet, avec l'ID, le serveur peut récupérer la clé client et donc déchiffrer ce sous message. Une fois celui-ci déchiffré, il est possible de vérifier que l'heure enregistrée correspond à l'heure d'envoie du message et ainsi s'assurer que la discussion n'est pas rejouée. Une fois cette vérification effectuée, le serveur conserve la clé de session pour pouvoir répondre à la demande.

Auprès du client, le serveur est authentifié par le déchiffrage du service contexte de la réponse. En effet, seul le serveur a pu normalement déchiffré le sous message de la demande et donc récupérer la clé de session. Si l'heure de réponse est correctement encodée, cela signifie que le serveur est bien à l'origine de la réponse.

Le fait de transmettre l'heure dans la requête et la réponse permet de s'assurer que la discussion n'est pas rejouée puisque cette variable évolue au cours du temps.

2.3 - Gestion des permissions

Il a était convenu que les services offerts par le serveur seront régis par des droits d'accès. Ainsi ses droits seront stockés dans un fichier XML organisé autour d'une notion de groupe. Voici présenté ci-dessous un exemple de fichier de permissions présent sur le serveur :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<jmds>
   <server key = "serverKey" />
   <clients>
      <client id = "id1" key = "clinetKey1" groupe = "groupe 1"/>
      <client id = "id2" key = "clinetKey2" groupe = "groupe 2"/>
      <client id = "id3" key = "clinetKey3" groupe = "groupe 1"/>
      <client id = "id4" key = "clinetKey4" groupe = "groupe 2"/>
      <client id = "id5" key = "clinetKey5" groupe = "groupe 1"/>
   </ clients >
   <groups>
      <group id = "groupe 1">
         <access>
            <object id = "object 1" fullAscess = "false">
               <fonction id = "fonction 1"/>
               < fonction id = "fonction 3"/>
            </ object >
            < object id = "object 2" fullAscess = "true"/>
         </ access >
      </ group >
      < group id = "groupe 2">
         < access >
            < object id = "object 1" fullAscess = "true">
            < object id = "object 2" fullAscess = "false">
               < fonction id = "fonction 2"/>
               < fonction id = "fonction 3"/>
            </ object >
         </ access >
      </ group >
   </ groups >
</jmds>
```

Ce fichier contient trois grandes parties:

- d'abord la clé du server permettant le décryptage des messages reçcus,

- ensuite une liste de clients auxquels sont associés une clé et un groupe,

- enfin une liste de groupes auxquels sont associés une liste de droits, soit un l'objet (fullAccess = "true") ou soit une (ou plusieurs) méthodes d'un objet (fullAccess = "false")

Lorsqu'un message est reçu par le serveur est que celui-ci l'a décodé à l'aide de la clé serveur, il connaît l'identifiant du client concerné. Il peut donc grâce au fichier des permissions associer le client à un groupe. Ensuite dans ce même fichier le serveur peut associer le groupe trouvé à une liste de droits. Ainsi, lors d'une demande de service par un client, le serveur est en mesure de répondre favorablement ou non à cette demande.

Le fait d'instaurer une notion de groupe permet de limiter les redondances dans la définition des accès. En effet, il apparaît fort probable que plusieurs clients puisse avoir les mêmes droits. Ainsi, la gestion d'un fichier contenant une liste d'identifiant client associée à une liste de droits d'accès est trop lourde en maintenance et en volume (fichier de permission important). Dans le principe présenté ci-dessus, la maintenance du fichier est beaucoup plus simple. Lors de l'ajout d'un nouvel utilisateur soit on le rattache à un groupe existant, soit on créé un nouveau groupe qui lui sera rattaché.

Notons également que le fichier de permissions permet d'affiner les droits d'accès jusqu'aux fonctions d'un objet. Or cette gestion au niveau CORBA n'est pas encore assurée ce qui pourra donc le faire évoluer. Si la gestion des accès ne peut se faire au niveau le plus fin (les fonctions) alors dans le fichier de permissions n'apparaîtrons plus la définition des fonctions pour chacun des objets. De même, l'attribut « fullAccess » de OBJECT viendra à disparaître.

2.4 - Générateur de cartes

2.4.1 - Principe de la modification de carte

D'après le fonctionnement d'authentification que nous avons choisi, il semble difficile de gérer une nouvelle carte par utilisateur. Nous fixons donc que l'ID utilisateur ainsi que sa clé et celle du serveur seront accessibles en écriture. Toutes les cartes seront donc identiques et la création d'une nouvelle carte reviendra à modifier l'ID et la clé utilisateur conservée en interne.

En revanche, pour des raisons de sécurité, l'ID et les clés ne seront accessibles qu'en écriture afin de s'affranchir de toute possibilité de récupérer ces informations par lecture de la carte.

Par ailleurs, les clés seront générées de façon aléatoire afin que seul le serveur soit détenteur de ces informations.

2.4.2 - Enregistrement d'un nouvel utilisateur sur le serveur

Pour créer un nouvel utilisateur sur le serveur, il suffira de lui donner un nom d'utilisateur et de connecter une carte sur le serveur. Celui-ci ira alors écrire l'ID utilisateur ainsi que la clé utilisateur que le serveur aura généré et la clé serveur permettant à l'utilisateur de communiquer avec ce serveur. De la même manière, le serveur ajoutera le couple ID-clé à sa liste d'utilisateur connu et lui donnera les droits d'accès aux applications par défaut.