Annexe n°7 Contrat SMS+

Règles de développement des Contenus exécutables Java

Version 3.12 décembre 2016

Orange
Société Anonyme au capital de 10 595 541 532 euros - RCS Paris 380 129 866
78 rue Olivier de Serres, 75015 Paris

SOMMAIRE

1 OBJET DU DOCUMENT 3

2 REGLES DE DEVELOPPEMENT 3

2.1 DEFINITION D'UNE PSEUDO-CONSTANTE 3
2.2 ENVOI DE SMS 4
2.3 RECEPTION DE SMS 6
2.4 FONCTIONNALITES MULTIMEDIA (MMAPI : MOBILE MULTIMEDIA API) 7
2.5 PROTECTION DES DONNEES PRIVEES DES RMS 7
2.6 PROPRIETES DU TERMINAL 8
2.7 CAS DE LA METHODE PLATFORMREQUEST() 8
2.8 CAS DE LA METHODE FORNAME() 8
2.9 MODIFICATION DU FICHIER JAD 8

3 REFERENCES 9

1 Objet du document

Ce document a pour but de définir les obligations en terme de développement des contenus exécutables Java MIDP 1.0 et MIDP 2.

2 Règles de développement

2.1 Définition d'une pseudo-constante

Le terme pseudo constante est utilisé dans la suite du document. Ce paragraphe précise la définition technique de ce terme.

Une **pseudo-constante** est un objet de type String qui est soit :

- un littéral ou un champ déclaré avec les modifieurs `final static` et initialisé (constante) ; ou bien,
- un champ `v` de classe (`static`) ou d'instance définie dans le code source de la midlet dont la valeur est définie par des affectations de la forme `v = c`; où `c` est une pseudo-constante ; ou bien,
- une variable `v` locale à la méthode dont la valeur est définie par des affectations de la forme `v = c`; où `c` est une pseudo-constante ; ou bien,
- un paramètre de la méthode, auquel cas l'argument correspondant dans tous les appels de cette méthode dans le code source de la midlet doit être une pseudo-constante ; ou bien,
- `new String(c)` où `c` est une pseudo-constante.

Les exemples qui suivent ne remplacent pas la définition précédente mais en donnent une idée plus intuitive. Les arguments passés à la méthode `legal` sont des pseudo-constantes respectant la définition :

```
/* littéral */
legal("constante")

/* constante Java */
final static c0 = "constante";
legal(c0);

/* champ d'une classe */
/* Note : la déclaration, l'affectation et l'utilisation du champs
peuvent être dans des classes différentes. Les affectations peuvent
être multiples */

static String c1;
String c2;
…
c1 = "constante";
c2 = c1;
…
legal(c1);
legal(c2);

/* paramètre d'une méthode et ses utilisations */
/* Note : la déclaration de la méthode et ses utilisations peuvent être
dans des méthodes différentes */
public void m(String p) {
      legal(p);
}

… m(c1); … m("constante")…;
```

```
/* variable locale d'une méthode */
public void m() {
      String l = "constante";
      legal(l);
}

/* new */
legal(new String("constante"));

/* Utilisation récursive des cas précédents */
static String c = new String("constante");

public m(String p) {
      local l = "pseudo";
      if (…) l = p;
      legal(l);
}
…
m(c3);
```

Les arguments de la méthode `illegal` ne sont pas des pseudo-constantes légales :

```
/* résultats de calculs */
illegal("sms://" + c1);
illegal(midlet.getAppProperty("sms-dans-le-jad"));

/* retour de fonction en général même si c'est une pseudo-constante */
public String calcul() { return "constante"; }
illegal(calcul());

/* définition de variable, de champ ou de paramètre qui ne sont pas des
pseudo-constantes légales */
String c4;

c4 = calcul();

public m(String p) {
  String l;
  illegal(c4);
  p = c4;
  l = c4;
  illegal(p);
  illegal(l);
}
```

Conséquence en terme de développement:
Une donnée passée à un argument qui doit être de type pseudo-variable ne peut provenir du fichier JAD.

2.2 Envoi de SMS

2.2.1 Introduction

Le numéro de destinataire d'un SMS doit être le numéro à 5 chiffres d'un serveur dit *Large Account (LA).* Le numéro de destinataire d'un SMS ne peut donc pas être celui d'un autre client mobile (à 10 chiffres au format national). Il y a 3 manières de spécifier le numéro de LA.

2.2.1.1 *Création d'un objet `MessageConnection`*

```
MessageConnection msgCon;
TextMessage msg;
msgCon = (MessageConnection) Connector.open("sms://12345");
msg = (TextMessage) msgCon.newMessage(MessageConnection.TEXT_MESSAGE);
msg.setPayloadText("Hello World!");
msgCon.send(msg);
msgCon.close();
```

2.2.1.2 *Écrasement lors de la création d'un objet Message*

```
MessageConnection msgCon;
TextMessage msg;
messCon = (MessageConnection) Connector.open("sms://12345");
msg = (TextMessage) msgCon.newMessage(MessageConnection.TEXT_MESSAGE, "54321");
msg.setPayloadText("Hello World!");
messCon.send(msg);
messCon.close();
```

2.2.1.3 *Écrasement, par appel de la méthode setAddress*

```
MessageConnection msgCon;
TextMessage msg;
msgCon = (MessageConnection) Connector.open("sms://12345");
msg = (TextMessage) msgCon.newMessage(MessageConnection.TEXT_MESSAGE);
msg.setAddress("54321");
msg.setPayloadText("Hello World!");
msgCon.send(msg);
msgCon.close();
```

2.2.2 Exigences relatives à l'envoi de SMS

2.2.2.1 *Cas standard*

REQ-SMSMO-1	Lors de appel aux méthodes:
	`Connector.open(String name)`
	`Connector.open(String name, int mode)`
	`Connector.open(String name, int mode, boolean timeouts)`
	l'argument `name` doit être une pseudo-constante (*c.f.* Annexe 1).

REQ-SMSMO-2	Lors de l'appel d'une de ces trois méthodes :
	`Connector.open(String name)`
	`Connector.open(String name, int mode)`
	`Connector.open(String name, int mode, boolean timeouts)`
	l'argument `name` doit être le numéro à 5 chiffres d'un serveur dit *Large Account (LA).* Le format requis pour `name` est un des suivants :
	`"sms://"`
	`"sms://`<numéro de LA>"
	`"sms://`<numéro de LA>`:`<numéro de port>"

REQ-SMSMO-3	Lors de l'appel de la méthode `newMessage(String type, String address)` l'argument `address` doit être une pseudo-constante.
REQ-SMSMO-4	Lors de l'appel de la méthode `newMessage(String type, String address)` l'argument `address` doit être le numéro à 5 chiffres d'un serveur dit *Large Account (LA).* Le format requis pour address sont les suivants : "" "<numéro de LA>" "<numéro de LA>:<numéro de port>"

REQ-SMSMO-5	Lors de l'appel de la méthode `setAddress(String addr)` l'argument `addr` doit être une pseudo-constante.
REQ-SMSMO-6	Lors de l'appel de la méthode `setAddress(String addr)` l'argument `addr` doit être le numéro à 5 chiffres d'un serveur dit *Large Account (LA)* de format: "<numéro de LA>" "<numéro de LA>:<numéro de port>"

2.2.2.2 Cas Siemens:

Remarque: dans tous les cas, le préfixe est "`SMS://`" et non "`sms://`".

Il existe deux fonctions statiques de la classe `com.siemens.mp.gsm.SMS` pour envoyer des SMS texte sans numéro de port:

- `send(String number, byte[] data)`
- `send(String number, String data)`

où `number` doit être une pseudo-constante de format "`SMS://<numéro à 5 chiffres>`".

Pour envoyer un SMS binaire avec un numéro de port particulier, il faut d'abord créer un objet de la classe `com.siemens.mp.io.Connection`. avec une adresse pseudo-constante de format "`SMS://`<numéro à 5 chiffres>:<numéro de port>" puis envoyer le SMS par appel de la méthode `send`.
Exemple:

```
Connection con = new Connection("SMS://12345:54321");
con.send(byte[] data);
```

2.2.2.3 Cas Nokia:

Remarque: dans tous les cas, le préfixe est "`com.nokia.sms://`" et non "`sms://`".
Il existe deux méthodes du package `com.nokia.mid.messaging` permettant de spécifier l'adresse de destination des SMS.

- `MessageConnection.newMessage(int type, String address)`
- `Message.setAddress(String addr)`

Dans les 2 cas, la `String` spécifiant l'adresse doit être une pseudo-constante de format "`com.nokia.sms://`<numéro à 5 chiffres>:<numéro de port>"

2.3 Réception de SMS

Certains numéros de ports utilisés dans les SMS sont réservés pour des applications ou le paramétrage du terminal. Le but de cette règle de développement et de vérifier que l'application ne va pas intercepter de tels SMS. Cette règle est une extension de la norme WMA 1.1 aux terminaux proposant une implémentation propriétaire de la réception de SMS.

REQ-SMSMT-1 A la réception d'un SMS destiné à une midlet, le terminal détermine à quelle midlet le SMS est destiné en analysant le numéro de port présent dans le corps du SMS (TP-UDH du TP-UD).
Les SMS avec numéro de port sont utilisés par d'autres applications du mobile que les midlets. Ces applications utilisent des ports particuliers, qui doivent leur être réservés, et que les midlets ne doivent par conséquent pas utiliser. Les champs interdits sont spécifiés dans la spécification WMA 1.1 (appendix A.6.0 "Restrictions on Port Numbers for SMS messages"). Dans le cas d'implémentations propriétaires Siemens et Nokia d'une API SMS, les éditeurs doivent prendre garde à ne pas utiliser un de ces numéros de ports.

Port number	Description
2805	WAP WTA secure connection-less session service
2923	WAP WTA secure session service
2948	WAP Push connectionless session service (client side)
2949	WAP Push secure connectionless session service (client side)
5502	Service Card reader
5503	Internet access configuration reader
5508	Dynamic Menu Control Protocol
5511	Message Access Protocol
5512	Simple Email Notification
9200	WAP connectionless session service
9201	WAP session service
9202	WAP secure connectionless session service
9203	WAP secure session service
9207	WAP vCal Secure
49996	SyncML OTA configuration
49999	WAP OTA configuration

Numéros de ports interdits (source: spécification WMA 1.1 - appendix A.6.0)

2.4 Fonctionnalités Multimédia (MMAPI : Mobile Multimedia API)

Ces règles de développements s'appliquent aux fonctionnalités multimédia de l'API Mobile Media (MMAPI).
Elles visent à s'assurer que l'usage de ces fonctionnalités n'engendre pas d'accès réseau (téléchargement de fichiers multimédia ou lecture en streaming d'un contenu en ligne déclenché par le contenu exécutable Java).

REQ-MMAPI-1	Seules les fonctionnalités multimédia de l'API MMAPI ne procédant pas à un accès réseau à un serveur sont autorisées.
REQ-MMAPI-2	Lors de l'appel de la méthode `Manager.createPlayer(String locator)` l'argument `locator` doit être une pseudo-constante.
REQ-MMAPI-3	Lors de l'appel de la méthode `Manager.createPlayer(String locator)` L'argument `locator` doit avoir pour préfixe: "`capture://`".

REQ-MMAPI-4	La méthode
	`Manager.createPlayer(InputStream stream, String type)`
	ne doit pas être utilisée avec un objet `InputStream` créé avec une des méthodes suivantes, car celles-ci permettent d'ouvrir des flux à travers le réseau mobile :
	`Connector.openDataInputStream(String name)`
	`Connector.openInputStream(String name)`

REQ-MMAPI-5	La méthode `Manager.createPlayer(DataSource source)` ne doit pas être utilisée car elle sert à ouvrir des flux à travers le réseau mobile.

2.5 Protection des données privées des RMS

REQ-RMS-1 Toute information relative au client et stockée dans un RMS doit être stockée dans un RMS de type « private » (i.e. accessible seulement par les midlets de la midlet suite) et non de type « shareable » (i.e. accessible par toutes les midlets de toutes les midlet suites).

2.6 Propriétés du terminal

Cette règle permet de s'assurer que l'application n'accède pas à des informations confidentielles sur l'utilisateur (via des mécanismes propriétaires ou non) en recueillant des propriétés du terminal (par exemple l'accès à IMEI, localisation, … est interdit)

REQ-PROP-1 La méthode `java.lang.System.getProperty` n'est autorisée qu'avec une pseudo constante de valeur une des valeurs suivantes:

- `microedition.platform`
- `microedition.encoding`
- `microedition.configuration`
- `microedition.profiles`
- `microedition.locale`
- `microedition.commports`
- `microedition.media.version`
- `supports.mixing`
- `supports.audio.capture`
- `supports.video.capture`
- `supports.recording`
- `audio.encodings`
- `video.encodings`
- `video.snapshot.encodings`

2.7 Cas de la méthode `platformRequest()`

REQ-PFRQ-1 : Toute utilisation de la méthode `platformRequest(String URL)` est interdite. En effet, cette méthode permet d'établir un appel voix, de lancer le browser, d'installer une midlet; actions aboutissant à une facturation du client.

2.8 Cas de la méthode forName()

REQ-FNAME-1: Toute utilisation de la méthode `java.lang.Class.forName` est interdite.

2.9 Modification du fichier JAD

REQ-JAD-1 : Seuls les champs suivants du fichier .jad pourront être modifiés sans nouvelle soumission du contenu :

- MIDlet-Jar-URL (URL du fichier JAR),
- MIDlet-Install-Notify (URL où faire un POST HTTP à la fin de l'installation)
- MIDlet-Delete-Notify (URL où faire un POST HTTP à la suppression de la midlet suite)
- MIDlet-Info-URL (URL où le client pourra trouver de l'info sur la midlet suite).

3 Références

JSR 118	MIDP 2.0 Mobile Information Device Profile v2.0 (November 2002) http://jcp.org/en/jsr/detail?id=118
JSR 185	JTWI R1 Java Technology for the Wireless Industry (may 2003) http://jcp.org/en/jsr/detail?id=185
JSR 120	WMA 1.0 Wireless Messaging API v1.0 (2003) http://jcp.org/aboutJava/communityprocess/final/jsr120/index.html WMA 1.1 Wireless Messaging API v1.1 (2003) http://jcp.org/aboutJava/communityprocess/final/jsr120/index2.html
JSR 135	MMAPI 1.1 Mobile Media API v1.1 (2003) http://jcp.org/en/jsr/detail?id=135
JSR 82	Java APIs for Bluetooth http://www.jcp.org/en/jsr/detail?id=82
Siemens SMS API	Spécification : Siemens Mobility Toolkit for Java Development Programmer's Reference Version 3.4, 02/2004 *Available from Siemens developer portal : http://communication-market.siemens.de after registration*
Nokia SMS API	Nokia SMS API v0.9 http://www.forum.nokia.com/ puis lien "Documents" puis taper "SMS" comme mot clé de la recherche puis lien vers le résultat intitulé "Nokia SMS API"
JAR File Specification	http://java.sun.com/j2se/1.4.2/docs/guide/jar/jar.html