(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 August 2017 (24.08.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/140476 A1

(51) International Patent Classification:
G06F 9/445 (2006.01) *G06F 12/02* (2006.01)

(21) International Application Number:
PCT/EP2017/051855

(22) International Filing Date:
27 January 2017 (27.01.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
16305179.0 17 February 2016 (17.02.2016) EP

(71) Applicant: **GEMALTO SA** [FR/FR]; 6, rue de La Verrerie, 92190 Meudon (FR).

(72) Inventors: **FRANCHI, Christophe**; c/o Gemalto SA, Intellectual Property Department, 525, avenue du Pic de Bertagne - CS12023, 13881 Gemenos Cedex (FR). **MARSEILLE, François-Xavier**; c/o Gemalto SA, Intellectual Property Department, 525, avenue du Pic de Bertagne - CS12023, 13881 Gemenos Cedex (FR).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *with international search report (Art. 21(3))*

WO 2017/140476 A1

(54) Title: METHOD FOR MANAGING OBJECTS IN A SECURE ELEMENT

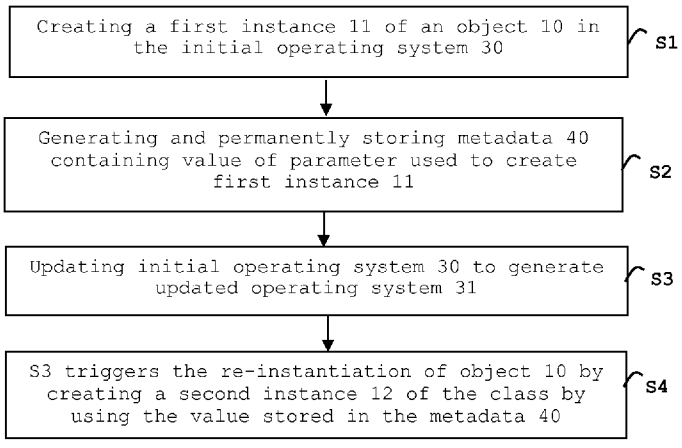

FIG. 1

(57) Abstract: The invention is a method of managing an object which is represented by a first instance of a class. The first instance is stored in a secure element comprising an initial operating system. The method comprises a step of updating the initial operating system to generate an updated operating system, a step of creating a metadata uniquely associated with the object, said metadata being permanently stored in the secure element and comprising a value of a parameter of said class which has been used to create said first instance. The method comprises a step of re-instantiating the object by generating an updated instance of the class in the updated operating system by using said value to set said parameter of the class, said updated instance representing the object. The re-instantiating step is automatically triggered by the step of updating the initial operating system.

METHOD FOR MANAGING OBJECTS IN A SECURE ELEMENT

(Field of the invention)

The present invention relates to methods of managing objects in a secure element. It relates particularly to methods of managing objects which are represented by instance of class in a secure element when the operating system of the secure element is updated.

(Background of the invention)

Secure elements are small devices comprising a memory, a microprocessor and an operating system for computing treatments. Such secure elements may comprise a plurality of memories of different types, like non-volatile memory and volatile memory. They are called "secure" because they are able to control the access to the data they contain and to authorize or not the use of data by other machines. The secure elements may also provide computation services based on cryptographic components. In general, secure elements have limited computing resources and limited memory resources and they are intended to be connected to a host machine which provides them with electric power. Secure elements may be removable or fixed to a host machine. For example, smart cards are a kind of secure elements.

A secure element may contain applications and their associated applicative data which encompass user data, file systems and secret key. Such an application may be developed as a package or a set of packages which is stored into the secure element. One or several instances

of the application are then created as needed. Each instance owns, handles and store its own instance data.

Secure elements may be accessed by a remote server via a wireless channel or through a wired network, like Internet for instance. For example, secure elements which are intended to be used in Telecom domain or Machine-To-Machine (M2M) domain are able to manage an OTA (Over-The-Air) channel. These secure elements may also be accessed through the HyperText Transfer Protocol, usually called HTTP or HTTPS for the secure mode. Thus, a distant server can remotely manage the content of a secure element like an UICC (Universal Integrated Circuit Card) through a dedicated communication session using a specific protocol. For example, the server may use the RAM (Remote Applet Management) mechanism as defined by GlobalPlatform ® v 2.2 standard - Amendment B "RAM over HTTP" or the OMA-DM (Open Mobile Alliance - Device Management) protocol as defined by OMA-TS-DM V1.2.1 standard.

According to JavaCard™ specifications, a package may be loaded in a secure element through a CAP file. The package generally comprises several classes. When an application is installed from the package, an instance of a class is created. This instance corresponds to an object.

JavaCard™ objects are not created each time a JavaCard™ OS or a JavaCard™ applet is started. They are created once (for example when a JavaCard™ applet is installed, personalized, etc.) and then become persistent, except if deleted, for the rest of the life of the JavaCard™ OS or the JavaCard™ applet. A remote server can send a new version or an upgrade of the

operating system of the secure element. In this case, the existing objects (e.g. existing instances of class) must be re-instantiated when the operating system has been updated. In other words, new instances of class corresponding to the old ones must be instantiated in the new operating system context in order to re-create the previously existing objects. This instantiation is carried out from the packages stored in the secure element. It is to be noted that old class instances are no more used or even deleted and their content is lost in the new operating system context.

The re-instantiation of object from a package may require time and a large part of the available computing resources of the secure element. This problem is exacerbated when a large number of objects (i.e. class instances) must be re-created in a secure element. Moreover, the remote server may have no access to the relevant package if this package has been installed by another entity like a third party.

There is a need for allowing to maintain existing objects in a functional state when the operating system of a secure element is updated.

(Summary of the Invention)

An object of the invention is to solve the above mentioned technical problem.

The object of the present invention is a method for managing an object which is represented by a first instance of a class. This first instance is stored in a secure element comprising an initial operating system.

The method comprises a step of updating the initial operating system to generate an updated operating system. The method comprises the step of creating a metadata uniquely associated with the object, said metadata being permanently stored in the secure element and comprising a value of a parameter of said class which has been used to create said first instance. The method comprises the step of re-instantiating the object by generating an updated instance of the class in the updated operating system by using the value to set the parameter of the class, the updated instance representing the object. The re-instantiating step is automatically triggered by the step of updating the initial operating system.

Advantageously, the first instance may comprise a set of data which are dependent on the initial operating system. During the re-instantiating step said set of data may be deleted from the secure element and a new set of data which are dependent on the updated operating system may be created and allocated to the updated instance.

Advantageously, the first instance may comprise a group of data which are independent of the initial operating system and the data of the group may be kept unchanged and allocated to the updated instance.

Another object of the invention is a secure element comprising an initial operating system and an object which is represented by a first instance of a class. The secure element comprises an upgrading agent adapted to generate an updated operating system by updating the initial operating system. The secure element comprises a keeper agent adapted to create a metadata uniquely associated with the object and comprising a value of a

parameter of the class which has been used to create said first instance. The keeper agent is adapted to permanently store the metadata in the secure element. The secure element comprises an instantiating agent adapted to re-instantiate the object by creating a second instance of the class in the updated operating system by using said value to set said parameter of the class. The upgrading agent is adapted to trigger the re-instantiating step at the end of the updating of the initial operating system.

Advantageously, the first instance may comprise a set of data which are dependent on the initial operating system. The instantiating agent may be adapted to delete the set of data from the secure element and to create a new set of data and to generate the updated instance during the re-instantiating step.

Advantageously, the first instance may comprise a group of data which are independent of the initial operating system and the instantiating agent may be adapted to allocate the data of said group to the updated instance while keeping said data of said group unchanged.

Advantageously, the keeper agent may be adapted to create and permanently store the metadata in a format different from that of the first instance.

(Brief description of the drawings)

Other characteristics and advantages of the present invention will emerge more clearly from a reading of the following description of a number of preferred

embodiments of the invention with reference to the corresponding accompanying drawings in which:

- Figure 1 shows an exemplary flow diagram of objects managing in a secure element according to the invention, and
- Figure 2 shows a diagram of a secure element according to an example of the invention,
- Figure 3 shows an example of an application instance with an initial operating system according to the invention and
- Figure 4 shows an example of an application instance with an upgraded operating system according to the invention.

(Detailed description of the preferred embodiments)

The invention may apply to any types of secure element intended to embed class instances and an operating system which may be upgraded when deployed on the field. Such secure elements may be coupled to a host machine like a smartphone, a smart watch, a vehicle, a meter, a slot machine, a TV or a computer.

In the present description, a package is a container as defined by object-oriented programming languages. In particular, the invention applies to objects as defined in Java™ and JavaCard™ languages.

Figure 1 shows an example of a flow diagram for managing an object when the operating system of a secure element is upgraded according to the invention.

A secure element 20 is supposed to have been issued with an original operating system 30. For example, this

original operating system 30 may comprise an object-oriented virtual machine and high level Application programming Interfaces (APIs). For instance, the virtual machine (VM) may be compliant with JavaCard™ specifications version 2.2 or upper.

At step S1, a first instance 11 of a class is created in the secure element 10. This first class instance represents an object 10.

At step S2, a metadata 40 is generated and permanently stored in the secure element 20.

The class instance 11 is assumed to comprise two parts (i.e. two sets of data): a first part whose content is independent from the current operating system and a second part whose content is dependent of the current operating system. Preferably, the metadata 40 is dynamically generated when the very first class instance is created. For example, the class constructor (i.e. the subroutine which is called to create an object) may be designed to generate and store the metadata 40 in addition to the creation of the class instance 11.

The metadata 40 contains a value of at least one parameter used to initialize the first class instance 11. In particular, the metadata 40 will be used to set the second part of the class instance to be initialized after the updating of the operating system of the secure element 20.

Let's consider a Cipher object, metadata associated may be embedded in a Cipher_metadata object as follows:

```
Cipher_metadata {

        byte algorithm;
```

```
        boolean shared_access;
        byte encryption_mode;
        Key_metadata key;
    }

    Key_metadata {

        ...
    }
    Here is an example of structure of a Cipher object:
    Cipher {
        Cipher_metadata metada;
        Key key;
        byte[] data; // volatile data to cipher
        byte[] icv;  // volatile initial vector
    }
    Key {
        Key_metadata metadata;
        ...
    }
```

The content of the metadata 40 is required to create the second part of a new class instance when the operating system has been updated.

The metadata 40 is permanently stored in the secure element 20. For instance, the metadata 40 may be stored in a non-volatile memory of the secure element 20. In the present description, "permanently stored" means that the metadata is kept in the secure element 20 for a long time. For example, these metadata is kept until the object 10 is removed from the secure element 20.

The metadata may be stored in a customized format which is different from the format of the class instance so as to minimize its footprint or to ease its parsing.

Although steps S1 and S2 have been described as two distinct steps they may be considered as a single step.

At step S3, an upgrade is applied to the original operating system 30 to generate a new (i.e. updated) operating system 31.

In a first example, the step S3 is carried out via a bootloader agent which is designed to administrate the current operating system of the secure element 20. In this case, the updated Operating system 31 is fully sent to the secure element 20.

In this case, the bootloader is activated so that the behavior of the secure element 20 does not rely on the original operating system 30. Then the new operating system 31 is downloaded in the secure element 20 via the bootloader. Then an activation command is sent to the secure element to switch from bootloader mode to the new operating system 31 mode and a further reset (i.e. hardware reboot) is applied to the secure element so that the secure element takes the new operating system 31 into account by starting the new operating system 31. In this example, the step S3 of updating the initial operating system 30 consists of the combination of execution of both the activation command and the reset.

In a second example, the step S3 may be carried out by downloading a bulk of data aiming at slightly modify the initial operating system (like a corrective patch or a new feature). In this case, the new/updated feature is active as soon as installed in the secure element (e.g.

without requiring a reset). In this example, the step S3 of updating the initial operating system 30 consists of the installation of the new/updated feature.

It is to be noted that in all cases, the re-instantiation operation is performed when the updated operating system 31 is active on the secure element 20.

The end of the step S3 triggers the step S4 where a new instantiation (also named re-instantiation) operation of the object 10 is started in context of the updated operating system 31. This re-instantiation operation is carried out using the stored metadata 40.

In other words, the re-instantiation step is automatically triggered by the end of the step S3 and is carried out using data permanently stored in the secure element 20. Preferably, the re-instantiation step is executed as soon as the step S3 is finished (i.e. directly executed). Alternatively, other intermediate operations may occur between the end of the step S3 and the re-instantiation step.

Preferably, the metadata 40 which is permanently stored according to the invention, remains unchanged after the re-instantiation step so that a further re-instantiation step can be performed again after another operating system update.

Preferably, the re-instantiation operation of step S4 is carried out as follows.

First, the secure element 20 checks consistency (i.e. compatibility) of the metadata 40 with the updated operating system 31. This checking aims at controlling that the second part (i.e. the part whose content is dependent of the operating system) of the object 10 will

be set with a value which is compliant with the features of the updated operating system 31.

In case of error, the re-instantiation operation may abort. Alternatively the re-instantiation operation may continue and existing objects which are not compatible with the new operating system 31 are removed or lost. In case of successful consistency checking, the secure element 20 create an instance 12 of the object 10 (i.e. a class instance) and initializes the content of the second part of the instance 12 by using the metadata 40.

It is to be noted that the object re-instantiation process is applied to all objects previously installed in the secure element provided that these objects are compatible with the updated operating system.

Figure 2 shows an example a diagram of a secure element according to an example of the invention.

In this example, the secure element 20 is a smart card comprising a communication interface (not shown) configured to exchange data with the outside according to ISO-7816 standards.

Both the original operating system 30 and the updated operating system 31 comprise a JavaCard™ virtual machine VM.

The secure element 20 comprises an initial operating system 30 and an object which is represented by an instance 11 of a class. The secure element 20 comprises an upgrading agent 70 which is configured to generate an updated operating system 31 (shown in dotted line) by updating the initial operating system 30. The upgrading agent 70 is able to get a bulk of data coming from a distant machine and to update the original operating

system 30 using the received data. For instance, the received bulk may comprise a patch for an existing function, an upgrade for a new feature or even a full operating system.

The secure element 20 comprises a keeper agent 90 adapted to create a metadata 40 that is uniquely associated with the object and comprising a value of a parameter of the class which has been used to create the instance 11. The keeper agent 90 is configured to permanently store the generated metadata 40 in the secure element 20.

For example, the following items characterize the Cipher class of JavaCard™ API:

- The used cipher algorithm,
- The multiple applet instances access,
- The used key,
- The mode (encrypt or decrypt).

All of the above-listed items are used for generating the metadata associated with a JavaCard™ object instantiated from the Cipher class.

Thus when an object is instantiated from the Cipher class, the created object contains a first part which is independent of the operating system and a second part which is operating system-dependent.

The second part includes element whose value is set according to the items used for generating the metadata.

The secure element 20 also comprises an instantiating agent 80 which is configured to re-instantiate an object by creating a new instance (like instance 12 shown in dotted line) of the class when the

updated operating system 31 has been activated (or generated). The instantiating agent 80 creates the new class instance by using the metadata associated with the object. In particular, the instantiating agent 80 uses at least one value included in the metadata 40 to set a parameter of the class. The upgrading agent 70 is configured to trigger the re-instantiating step at the end of the updating of the initial operating system 30.

Preferably, the upgrading agent 70 is configured to automatically delete the previous class instances when new class instances are created during the re-instantiation step.

The keeper agent 90 may be configured to store metadata in a customized format different from the format used by class instance so as to minimize the footprint or to ease parsing of the metadata. Alternatively, the keeper agent 90 may be configured to store metadata in a format that is identical to the format used by class instance.

Figure 3 shows an example of an application instance with an initial operating system according to the invention.

At this stage, the initial operating system is still activated and an application instance (Applet instance) has been created in the secure element 20. In the example of Figure 3, the application instance comprises two application objects 10 and 15.

The instance 11 of the object 10 includes a first part 93 (also named Applet data) and a second part 91 (also named OS data). Content of the first part 93 is independent from the current operating system. Content of

the second part 91 is dependent of the current operating system. A metadata 40 has been created and uniquely associated with the second part 91.

The instance of the object 15 includes only a part 95 whose content is dependent of the current operating system. A metadata 45 has been created and uniquely associated with the part 95 (i.e. associated with the object 15 only).

Figure 4 shows an example of an application instance in the context of an upgraded operating system according to the invention.

At this stage, the initial operating system has been updated so that the current operating system is now the upgraded operating system. The application instance (Applet instance) has been kept under an updated form further to the re-instantiating operation in the secure element.

The class instance 11 of the object 10 has been replaced by a new class instance 12 which includes the first part 93 (remaining unchanged) and a new second part 92 whose content has been initialized using the metadata 40. The metadata 40 remains unchanged and is uniquely linked to the new second part 92. In other words, the metadata 40 remains linked to the object 10.

It is to be noted that the content of the first part 93 (also named group of data) is kept unchanged and is allocated to the updated class instance.

The class instance of the object 15 has been replaced by a new class instance (instance of the same class) which includes a single part 96 whose content has been initialized using the metadata 45. The metadata 45

remains unchanged and is still associated with the object 15.

The invention allows to keep active and functional applications based on objects embedded in a secure element when the operating system is updated. The invention avoids having to save data objects outside the secure element and to perform a complete cycle of application installation and application personalization.

Thanks to the invention it is possible to easily and smoothly manage the migration of existing objects when the operating system is updated, by automatically maintaining alive the objects already created in the secure element. Thus the updating operation of the operating system can be performed in a transparent way for the user who keeps a full continuity of service since the application instance (based on created objects) remain functional.

It is to be noted that metadata may be kept unchanged over successive evolutions of the operating system. The metadata may also be updated. In this case, the secure element 20 (e.g. the upgrading agent 70) should be configured to manage and ensure backwards compatibility.

The invention is not limited to the described embodiments or examples. In particular, the secure element may comprise several application instances whose objects are automatically re-instantiated as soon as the operating system is updated.

It is to be noted that the invention applies to any object represented by a class instance.

CLAIMS

1. A method for managing an object (10) which is represented by a first instance (11) of a class, said first instance (11) being stored in a secure element (20) comprising an initial operating system (30), the method comprising a step of updating the initial operating system (30) to generate an updated operating system (31),

characterized in that the method comprises the step of creating a metadata (40) uniquely associated with the object (10), said metadata (40) being permanently stored in the secure element (20) and comprising a value of at least one parameter of said class which has been used to create said first instance (11),

in that the method comprises the step of re-instantiating said object (10) by generating a second instance (12) of the class in the updated operating system (31) by using said value to set said at least one parameter of the class, said second instance (12) representing the object (10) and

in that said re-instantiating step is automatically triggered by the step of updating the initial operating system (30).

2. A method according to claim 1, wherein said first instance (11) comprises a set (91, 95) of data which are dependent on the initial operating system (30), wherein during the re-instantiating step said set (91, 95) of data is deleted from the secure element (20) and a new set (92, 96) of data which are dependent on the updated

operating system (31) is created and allocated to the second instance (12).

3. A method according to claim 2, wherein said first instance (11) comprises a group (93) of data which are independent of the initial operating system (30) and wherein the data of the group (93) are kept unchanged and allocated to the second instance (12).

4. A secure element (20) comprising an initial operating system (30) and an object (10) which is represented by a first instance (11) of a class,

characterized in that said secure element (SE) comprises an upgrading agent (70) adapted to generate an updated operating system (31) by updating the initial operating system (30),

in that said secure element (SE) comprises a keeper agent (90) adapted to create a metadata (40) uniquely associated with the object (10) and comprising a value of at least one parameter of the class which has been used to create said first instance (11), said keeper agent (90) being adapted to permanently store the metadata (40) in the secure element (20),

in that said secure element (20) comprises an instantiating agent (80) adapted to re-instantiate the object (10) by creating a second instance (12) of the class in the updated operating system (31) by using said value to set said at least one parameter of the class,

and in that said upgrading agent (70) is adapted to trigger the re-instantiating step at the end of the updating of the initial operating system (30).

5. A secure element (20) according to claim 4 wherein said first instance (11) comprises a set (91, 95) of data which are dependent on the initial operating system (30) and wherein the instantiating agent (80) is adapted to delete said set (91, 95) of data from the secure element (20), to create a new set (92, 96) of data and to generate the second instance (12) during the re-instantiating step.

6. A secure element (20) according to claim 5 wherein said first instance (11) comprises a group (93) of data which are independent of the initial operating system (30) and wherein the instantiating agent (80) is adapted to allocate the data of said group (93) to the second instance (12) while keeping said data of said group (93) unchanged.

7. A secure element (20) according to claim 4 wherein the keeper agent (90) is adapted to create and permanently store the metadata (40) in a format different from that of the first instance (11).

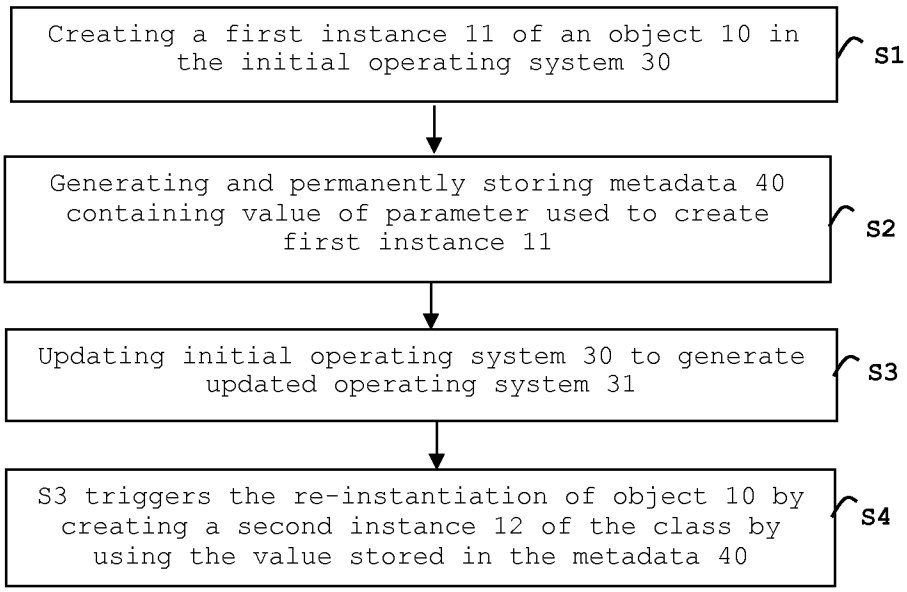
Creating a first instance 11 of an object 10 in the initial operating system 30
S1
Generating and permanently storing metadata 40 containing value of parameter used to create first instance 11
S2
Updating initial operating system 30 to generate updated operating system 31
S3
S3 triggers the re-instantiation of object 10 by creating a second instance 12 of the class by using the value stored in the metadata 40
S4

FIG. 1

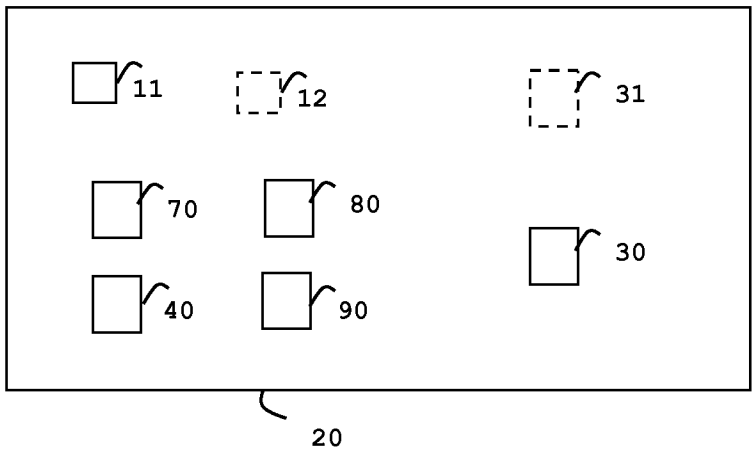
11
12
31
70
80
30
40
90
20

FIG. 2

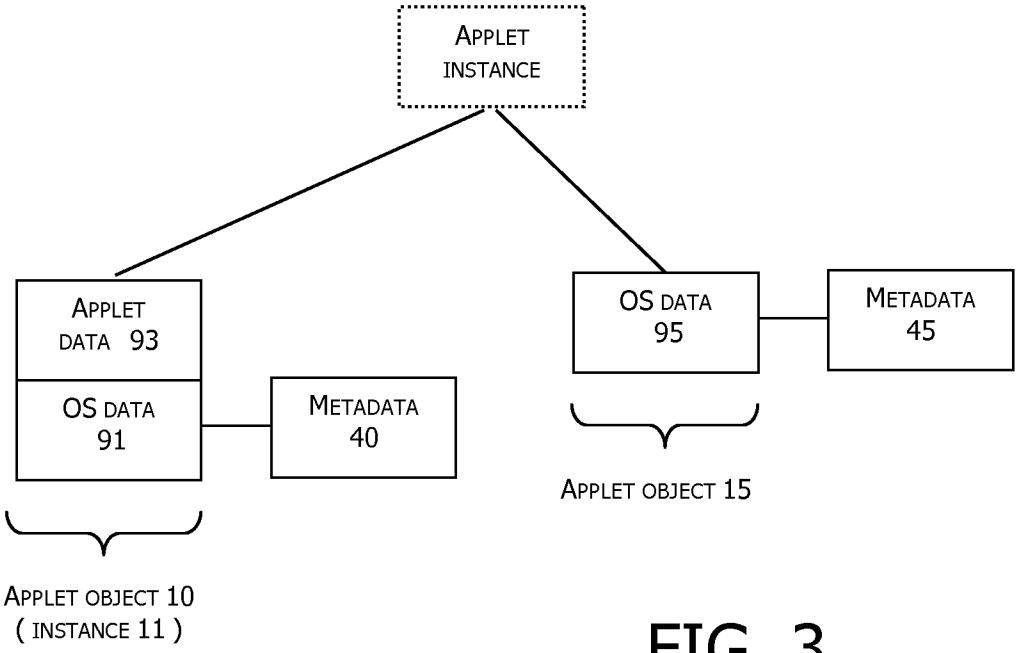
APPLET INSTANCE
APPLET DATA 93
OS DATA 91
METADATA 40
APPLET OBJECT 10 (INSTANCE 11)
OS DATA 95
METADATA 45
APPLET OBJECT 15

FIG. 3

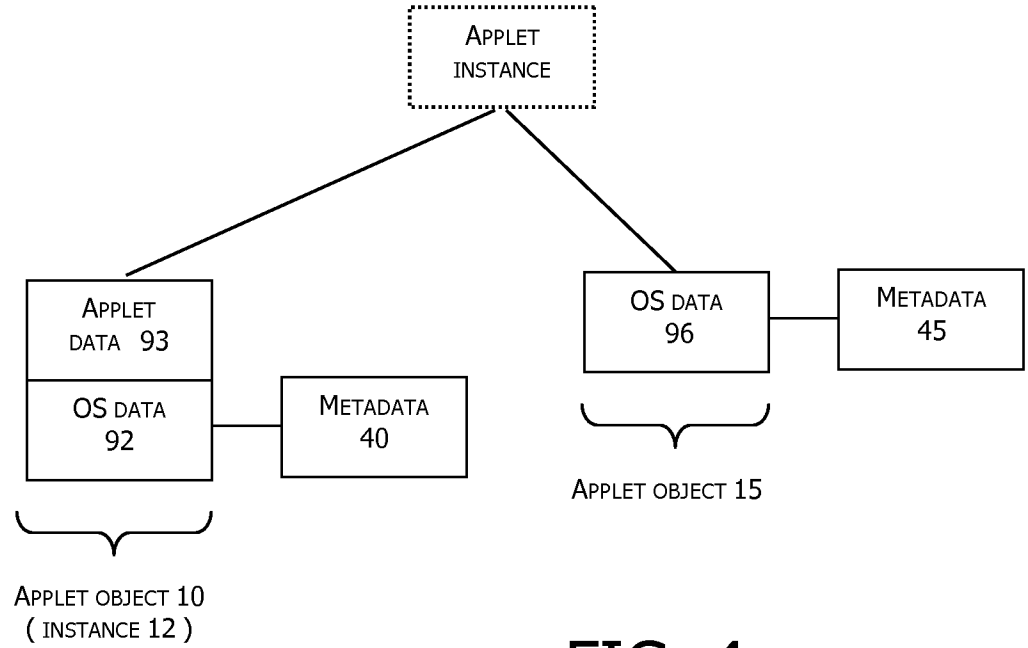
APPLET INSTANCE
APPLET DATA 93
OS DATA 92
METADATA 40
APPLET OBJECT 10 (INSTANCE 12)
OS DATA 96
METADATA 45
APPLET OBJECT 15

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2017/051855

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/445 G06F12/02
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2015/193224 A1 (ZIAT MEHDI [US] ET AL) 9 July 2015 (2015-07-09) figures 2,9 claims 1,4 page 4, paragraph [0036], lines 1-5 page 5, paragraph [0041], lines 1-3, lines 20-27 page 11, paragraph [0100], lines 5-9	1-7

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 February 2017	03/03/2017

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Leineweber, Harald

1

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	A. ORSO ET AL: "A technique for dynamic updating of Java software", PROCEEDINGS IEEE INTENATIONAL CONFERENCE ON SOFTWARE MAINTENANCE. ICSM-2002. MONTREAL, QUEBEC, CANADA, OCT. 3 - 6, 2002., 1 January 2002 (2002-01-01), pages 649-658, XP055288633, US DOI: 10.1109/ICSM.2002.1167829 ISBN: 978-0-7695-1819-0 section 1, page 2, left column, lines 8-11 section 2.1, page 3, left column, last paragraph section 2.1, right column, lines 21-27	1-7
A	US 2007/277168 A1 (VETILLARD ERIC [FR]) 29 November 2007 (2007-11-29) abstract paragraph [0014] claims 1,5,6	1-7
A	EP 2 827 275 A1 (GEMALTO SA [FR]) 21 January 2015 (2015-01-21) abstract figures 2,3	1-7

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2017/051855

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2015193224 A1	09-07-2015	US 2015193224 A1	09-07-2015
		US 2016335078 A1	17-11-2016
US 2007277168 A1	29-11-2007	FR 2864650 A1	01-07-2005
		US 2007277168 A1	29-11-2007
		WO 2005064459 A2	14-07-2005
EP 2827275 A1	21-01-2015	CN 105378748 A	02-03-2016
		EP 2827275 A1	21-01-2015
		EP 3022678 A1	25-05-2016
		US 2016171207 A1	16-06-2016
		WO 2015007491 A1	22-01-2015