SyncML Device Management Bootstrap

Abstract

This document defines the how a SyncML DM device is brought from a ‘clean’ state, to a state where it is capable to initiate a management session with a provisioned management server.

SyncML Initiative

The following companies are Sponsors of the SyncML Initiative:

Ericsson
IBM
Lotus
Matsushita Communication Industrial Co, Ltd.
Motorola
Nokia
Openwave Systems, Inc.
Starfish Software
Symbian

Revision History

Revision	Date	Comments
1.1	2002-02-15	First release.

Copyright Notice

Table of Contents

1 Introduction

Other SyncML DM specifications define how a management session is established and maintained. However, in order for a device to be able to initiate a management session it must be provisioned with SyncML DM settings. The process of moving a device from an un-provisioned, empty, state to a state where it is able to initiate a management session is called SyncML DM bootstrap.

This document defines SyncML DM bootstrap. SyncML DM bootstrap is only meant to bootstrap the functions necessary for the SyncML DM protocol itself. It is not intended as a replacement to other provisioning standards.

2 Formatting Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in [1].

3 Terminology

This section defines terminology used throughout the specification.

ACL

Access Control List. A list of identifiers and access rights associated with each identifier. This is the only mandatory property for management objects.

Device Description Framework, DDF

A specification for how to describe the management syntax and semantics for a particular device type, see [9].

MAC

Message Authentication Code. A value computed based on a message hash and some form of shared secret. The MAC is transported outside the bootstrap package.

Management object

A manageable entity in a managed device. A management object can have other management objects linked to it as children and a collection of management objects can thus form a tree structure. Management objects can be dynamic or permanent, permanent objects cannot be deleted but their value can be changed. Management objects must support the ACL property. All other properties are optional.

Management client

A software component in a managed device that correctly interprets SyncML DM

commands, executes appropriate actions in the device and sends back relevant responses to the issuing management server.

Management server

A network based entity that issues SyncML DM commands to devices and correctly interprets responses sent from the devices.

Management tree

The mechanism by which the management client interacts with the device, e.g. by storing and retrieving values from it and by manipulating the properties of it, for example access control.

Server identifier

The SyncML DM internal name for a management server. A management server is associated with an existing server identifier in a device through SyncML DM authentication.

4 Bootstrap scenarios

SyncML DM devices must be able to function in diverse network environments and using a large set of protocols. This makes it hard to find a 'one size fits all' solution to the bootstrap problem. This section starts with the most basic requirements for bootstrap and continues to define two different processes for bootstrap.

4.1 Requirements

A SyncML DM solution capable of transforming a empty, clean, device into a state where it is able to initiate a management session needs to address these requirements.

- Re-use technology (WAP Push)
- Tightly standardised and simple ⇒ Highly interoperable
- Self sufficient and complete
- Secure
- Data format should be XML based
- Content mappable to the SyncML DM managent object
- Transport encoding should be WBXML

4.2 Solutions

This document defines two different ways to perform the bootstrap process.

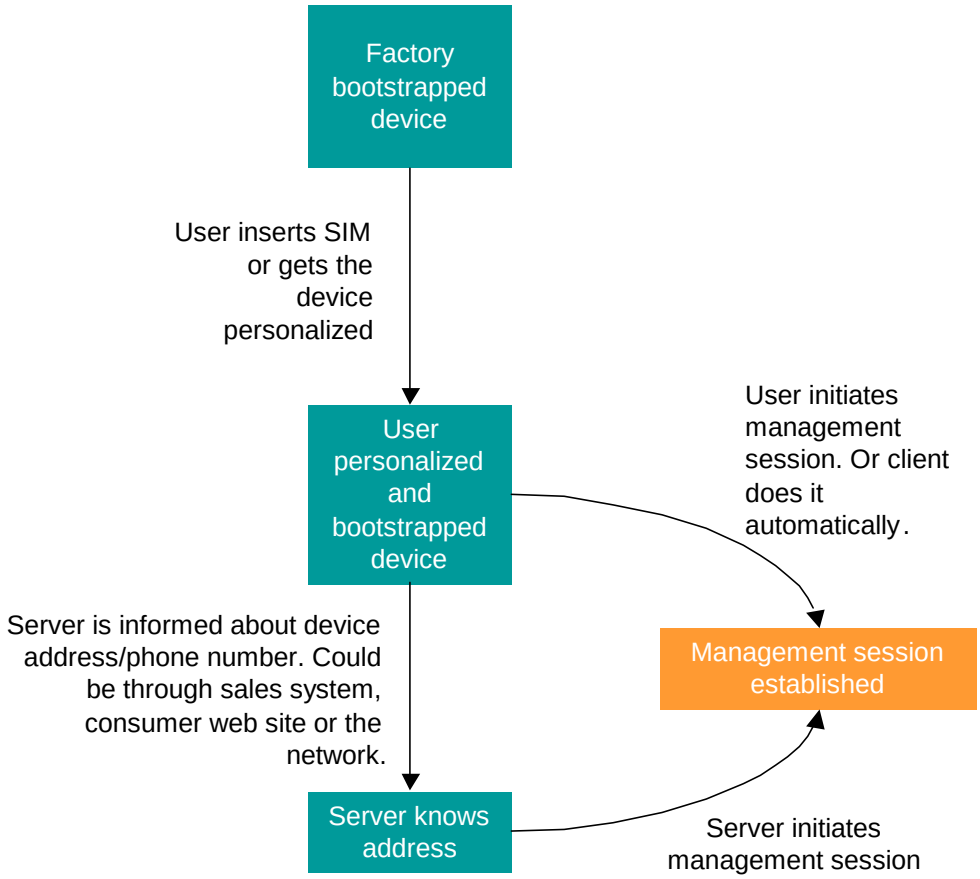
- Customized bootstrap
  Devices are loaded with SyncML DM bootstrap information at manufacture. Also referred to as factory bootstrap.

- Server initiated bootstrap
  Server sends out bootstrap information via some push mechanism, e.g. WAP Push

or OBEX. Server must be told the device address/phone number beforehand.

4.2.1 Customized bootstrap

This is the most convenient way to bootstrap a device from an end user perspective because the user does not have to do anything. In this scenario, an operator orders the devices pre-configured from a device manufacturer. All the information about the operators network and device management infrastructure is already in the devices when they leave the factory. Another advantage of this method is that it is very secure. There is no need to transport sensitive bootstrap information, e.g. shared secrets, over the air. The method is however not very flexible and not all device manufactures may provide this service.



Customized bootstrap

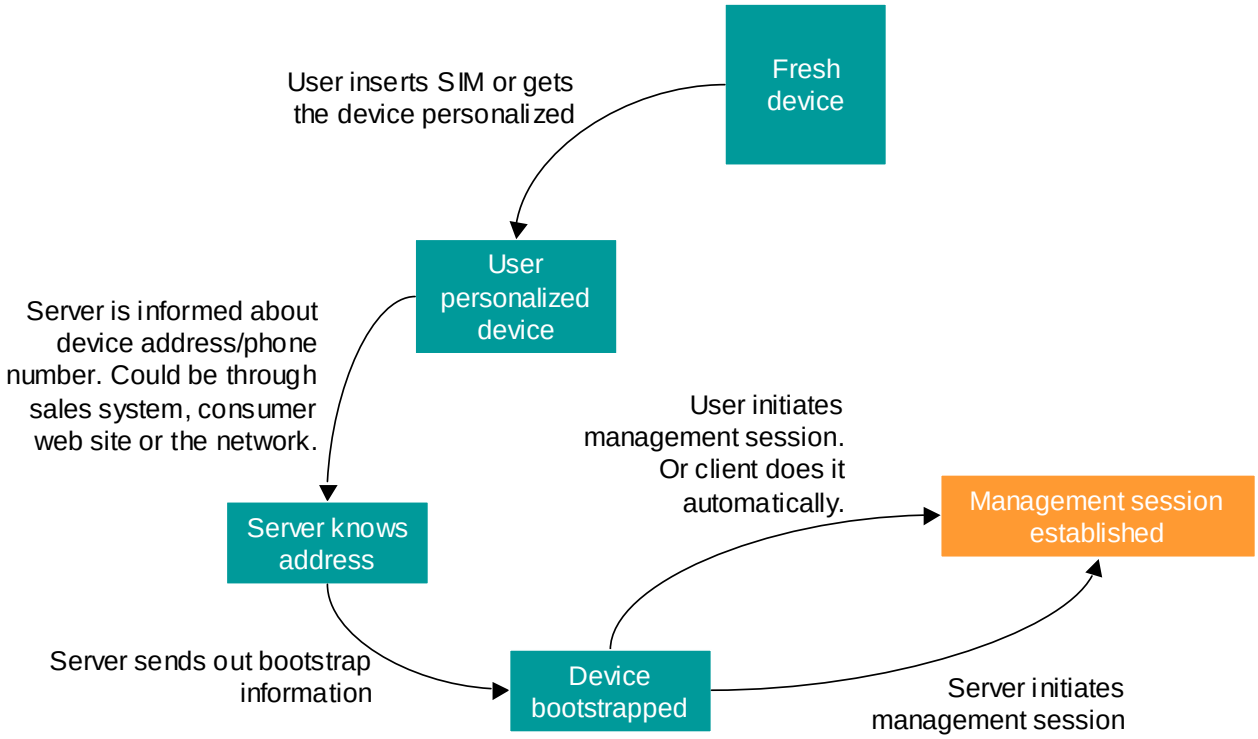
4.2.2 Server initiated bootstrap

In this scenario, the devices leave the assembly line in a clean and empty state. Once a user acquirers a device and personalizes it, e.g. by inserting a SIM, the prerequisites for this process are in place. The problem is now to inform the server of the identity, address or phone number of the device and this can be achieved in many ways.

- It could be done at the point-of-sales where a sales system ties in with the management system and delivers the information.

- It could be done through a self-service web site where the user enters her own phone number.
- It could be done by the network the first time the device is attaching to the network. When this happens a trigger could be sent from the core network to the management server with the number used by the device.
- It could be done with a voice prompt system where the user is prompted to key in her phone number using DMTF.

Regardless of how the phone number or device address reaches the management server, the server is now in a position where it can send out a SyncML DM bootstrap message. This bootstrap message, who's structure and content is defined in this document, contains enough information for the device to be able to initiate a management session with the management server that sent out the bootstrap message.



Server initiated bootstrap

5 Bootstrap profiles

SyncML DM has been designed to meet the management requirements of many different types of devices. For some of these device types there already exists bootstrap or provisioning mechanisms. In these cases SyncML DM leverages the existing mechanisms so that backwards compatibility and simple deployment can be achieved. To define how

different kinds of devices can be bootstrapped and to specify how SyncML DM leverages existing standards this document introduces the concept of bootstrap profiles. Each profile defines it's own security, transport and data format. Support for any particular profile is OPTIONAL.

Currently two profiles are planned, but as interest in SyncML DM grows and usage of it increases more profiles can be added. The two profiles are:

WAP

This profile specifies how the WAP Provisioning architecture [6] can be used to provision SyncML DM. The profile defines a number of parameter extensions to the WAP Provisioning Content format [5] and how these, and other parameters, are mapped in to the SyncML DM management object. The WAP profile is to be specified by the WAP Forum.

Plain

This profile specifies how the SyncML DM management object can be bootstrapped using a SyncML DM format. For security it leverages the methods developed for WAP Provisioning. This type of bootstrap message can be transported using OBEX or WAP Push, but other transport mechanisms can be defined in the future. This profile is specified in this document.

6 The Plain profile

6.1 Transport

Bootstrapping a SyncML DM device should be possible to do through all the transport mechanisms defined for SyncML DM. This includes transports providing both local and remote connectivity. In the local case, the transport used would be OBEX and in the remote case WAP Push, since these are both capable of delivering unprompted messages to the device.

Unprompted messages can arrive to a SyncML DM client for other purposes than bootstrap. In fact, the most common type of unprompted message that a client will receive will be a notification alert, telling the client that the server wishes that the client should initiate a session. A client can tell these different messages apart by their respective MIME types.

6.1.1 WAP Push

SyncML DM bootstrap messages can be sent via WAP Push, [7]. The following values MUST be specified for the listed parameters.

WAP Push application ID: `0x07`

WAP content type code (for the MIME type below): `0x40`

MIME type: `application/vnd.syncml.dm+wbxml`

Devices with out a full WAP implementation can be made to parse the WAP Push headers relatively easy. Hence it is possible to use this delivery method also for non-WAP devices that are capable of receiving concatenated SMS.

6.1.2 OBEX

Local bootstrap over OBEX is done inside the PUT command of the OBEX protocol. This happens in the same way as sending SyncML messages over OBEX to a SyncML client. See the SyncML OBEX Binding specification [2].

6.2 Security

Bootstrapping is a sensitive process that may involve communication between two parties without any previous relationship or knowledge about each other. In this context, security is very important. The receiver of a bootstrap message needs to know that the information originates from the correct source and that it has not been tampered with en-route. WAP Forum has specified different security mechanisms that are suitable for this purpose and SyncML DM reuses these.

6.2.1 Mechanisms

In WAP Provisioning Bootstrap, [4], chapter 6, a number of different security methods are defined for WAP bootstrap.

6.2.1.1 NETWPIN

This method relies on some kind of shared secret that the device and the network both know before the bootstrap process starts. This could be things like IMSI (for GSM) or ESN (for CDMA). What the shared secret actually is depends on the network and this is specified in [4]. One advantage with this method is that is can be used without user intervention.

The NETWPIN method requires a MAC value to be calculated and sent with the message and the protocol used to send the bootstrap message must be capable of transporting both the MAC value and the SyncML DM bootstrap package.

SyncML DM compliant devices and servers MUST support the NETWPIN method.

6.2.1.2 USERPIN

This method relies on a PIN that must be communicated to the user out-of-band, or agreed to before the bootstrap process starts.

The USERPIN method requires a MAC value to be calculated and sent with the message and the protocol used to send the bootstrap message must be capable of transporting both the MAC value and the SyncML DM bootstrap package.

SyncML DM compliant devices and servers MUST support the USERPIN method.

6.2.1.3 USERNETWPIN

This is a combination of the NETWPIN and USERPIN methods. It requires the use of a network shared secret and a user PIN.

The USERNETWPIN method requires a MAC value to be calculated and sent with the message and the protocol used to send the bootstrap message must be capable of transporting both the MAC value and the SyncML DM bootstrap package.

SyncML DM compliant devices and servers MAY support the USERNETWPIN method.

6.2.1.4 USERPINMAC

This method relies on out-of-band transport of the PIN. The PIN is computed based on the bootstrap message and there is no digest or message authentication code sent with the bootstrap message, the PIN itself contains this information. When a device receives the message, the user should be prompted to enter the PIN delivered out-of-band. The computation is then performed again and if the results match the message can be accepted.

One drawback of this method is that the PIN's tend to get rather long; the minimum size allowed by the algorithm is 10 decimal digits.

SyncML DM compliant devices and servers MAY support the USERPINMAC method.

6.2.1.5 No security

If security is provided in some other way, a bootstrap message can be sent without any security information. This is indicated by omitting the SEC and MAC parameters in the content type header.

SyncML DM compliant devices and servers MAY implement no security. SyncML DM compliant devices that implement no security, MAY automatically discard all bootstrap messages received without any security.

6.2.2 Security Parameters

SyncML DM uses the same security parameters as WAP Provisioning. These parameters are SEC and MAC. They are defined in section 5.3 of [5]. The calculation of their values is defined in chapter 6 of [4].

6.2.3 Management tree ACL and bootstrap

During processing of a bootstrap package the normal behaviour of the ACL, as specified in [9], does not apply. Each item in the Replace command in the bootstrap package MUST be processed successfully, provided that no name space collisions occur.

6.3 Bootstrap Message Content

The content of a bootstrap message is a SyncML DM package. However, it is special a package in many ways since it is not part of an ongoing SyncML DM session but rather a one-time transfer of information. Hence, many of the elements needed to manage the

session are superfluous in the context of bootstrapping, but they must still be included so that the package is well formed XML.

Inside the bootstrap package's SyncBody, there is one Replace command with several items. The items all have Target and Data elements. The Target element contains the URI for the object to the bootstrapped and the Data element contains the value to be assigned to the object.

All SyncML DM bootstrap packages MUST be sent WBXML encoded.

SyncML DM servers MUST NOT expect any Status for the command in a bootstrap package. An implicit acknowledgement of successful bootstrap can be concluded when the client connects to the server for the first management session.

6.3.1 Use of protocol elements

6.3.1.1 Elements in SyncHdr

The following table defines how the mandatory elements in the SyncHdr MUST be used.

Element	Usage
VerDTD and VerProto	These elements MUST be used as specified in [3]
SessionID	MUST be "0"
MsgID	MUST be "0"
Target LocURI	SHOULD reflect the identity of the device. This could be bearer dependent
Source LocURI	MUST be used as specified in [3]

The Target and Source elements MUST NOT contain a LocName element.

The SyncHdr MUST NOT contain any other elements than the ones above.

6.3.1.2 Elements in SyncBody

In the SyncBody, there MUST be only one Replace command. The CmdID element of the Replace command MUST have the value "1". The Replace command MUST NOT contain any NoResp, Cred or Meta elements.

This Replace command MUST contain a number of Item elements.

Each Item MUST contain a Target and a Data element. Each Item MAY Contain a Meta element. Each Item MUST NOT contain a Source element.

The Target element MUST NOT contain a LocName element.

The optional Meta in Item can be used to specify if a parameter has a NULL value.

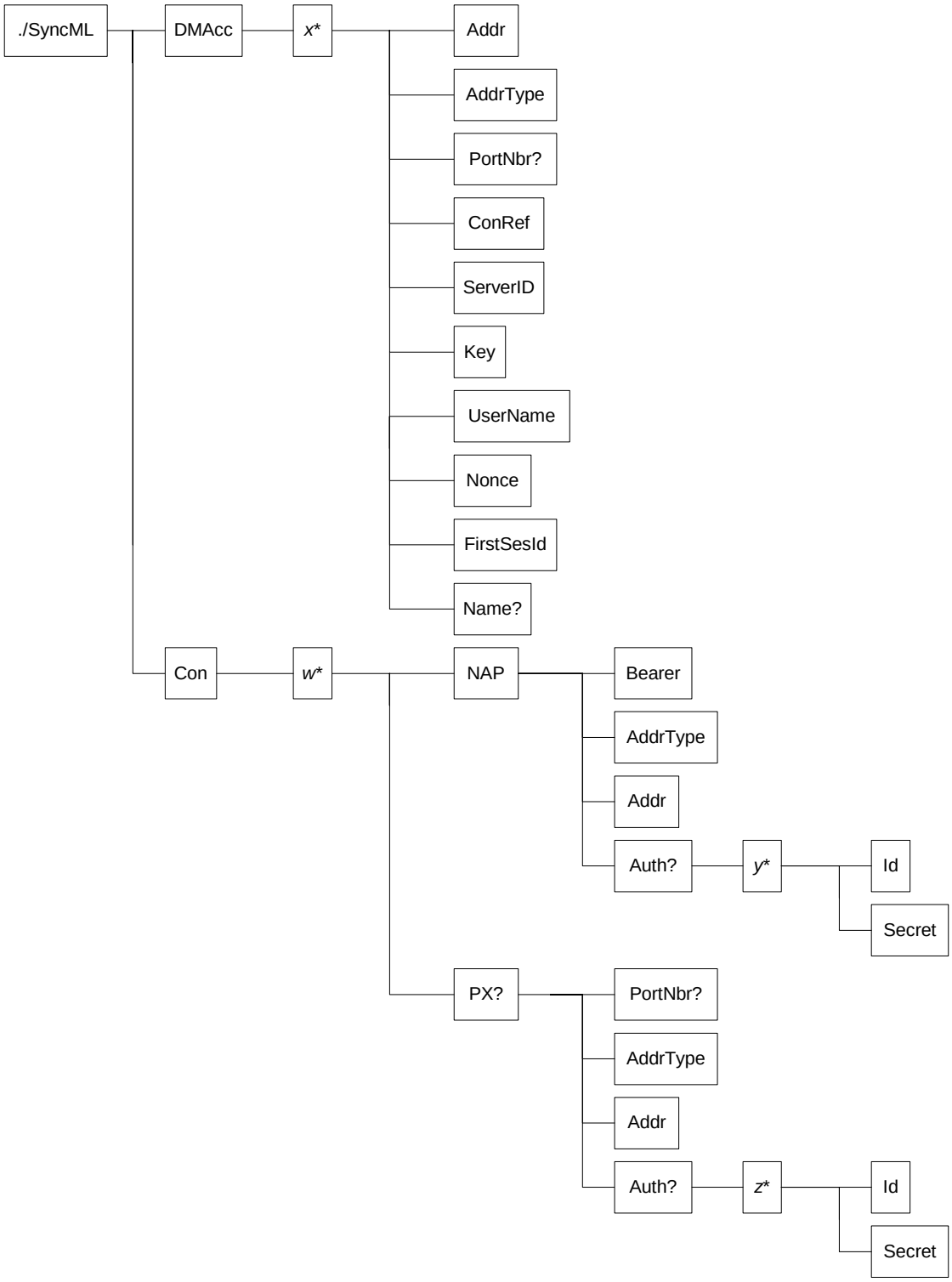
6.3.2 Example bootstrap message

```
<?xml version="1.0" encoding="UTF-8"?>
<SyncML>
   <SyncHdr>
      <VerDTD>1.1</VerDTD>
      <VerProto>SyncMLDM/1.1</VerProto>
      <SessionID>0</SessionID>
      <MsgID>0</MsgID>
      <Target>
         <LocURI>My_SyncML_DM_Device</LocURI>
      </Target>
      <Source>
         <LocURI>http://www.TheUltimateManagementServer.com</LocURI>
      </Source>
   </SyncHdr>
   <SyncBody>
      <Replace>
         <CmdID>1</CmdID>
         <Item>
            <Target>
               <LocURI>
                     ./SyncML/DMAcc/UltimateManagement
               </LocURI>
            </Target>
            <Meta>
               <Format xmlns="syncml:metinf">node</Format>
            </Meta>
            <Data/>
         </Item>
         <Item>
            <Target>
               <LocURI>./SyncML/DMAcc/UltimateManagement/Addr</LocURI>
            </Target>
            <Data>www.TheUltimateManagementServer.com</Data>
         </Item>
         <Item>
            <Target>
               <LocURI>./SyncML/Con/My_ISP</LocURI>
            </Target>
         </Item>
         <Item>
            <Target>
               <LocURI>./SyncML/Con/My_ISP/PX/Addr</LocURI>
            </Target>
            <Data>123.123.123.123</Data>
         </Item>
         <!--...and so on until all bootstrap parameters are transferred.-->
      </Replace>
   </SyncBody>
</SyncML>
```

6.3.3 The Bootstrap Parameters

This section lists the objects that must be set in a bootstrap message in order to assure that a device can initiate a session after receiving the bootstrap message. Note that no upper limit for the number of parameters included in a bootstrap message is specified. This makes it possible to bootstrap more than just the SyncML DM client in a device with one single, large, bootstrap message. However, the recommendation is to bootstrap the bare minimum parameters and then use a SyncML DM management session to provision other services.

The complete SyncML DM object is shown in the following figure.



The SyncML DM management object

A bootstrap message does not need to contain all these objects. Which objects that are actually present in a bootstrap message are dependent on many things and the final decision on what to include lies with the server. Note that clients and server still have to implement the object as it is specified in [8], even though all of the objects are not used in

bootstrap. However, servers should always include sufficient information for the device so that it can initiate a management session with the server. Never the less, there are some objects that must be included in any meaningful bootstrap message. These are indicated bellow.

The objects are listed here by their URI. They are all part of the SyncML DM object defined in [8]; also see this document for a more detailed explanation of each object. Note that some parts of the URI will have to be assigned by the server in the bootstrap message, these parts are indicated with a lowercase italics character, e.g. *x*, in the table below. To increase readability, all URI is specified as relative and the base URI is listed in the table header.

Base URI: ./SyncML		
URI	**Object meaning**	**Usage**
DMAcc/*x*/Addr	DM server address	MUST
DMAcc/*x*/AddrType	Type of address used	MUST
DMAcc/*x*/PortNbr	DM server port number	MAY
DMAcc/*x*/ConRef	Logical reference to connectivity information stored elsewhere in the management tree	MUST
DMAcc/*x*/ServerID	SyncML DM Server ID	MUST
DMAcc/*x*/Key	The HMAC value for this server-device combination.	MUST
DMAcc/*x*/UserName	SyncML DM users user name	MUST
DMAcc/*x*/Nonce	The next nonce to be used by the device	MUST
DMAcc/*x*/FirstSesId	The Session Id to be used by the device during the first session with this server.	MUST
DMAcc/*x*/Name	Displayable name for this management account	MAY
Con/*w*/	Logical name for the following connectivity object	MUST
Con/*w*/NAP/Bearer	Which bearer to use, e.g. GPRS, CSD, OBEX…	MUST
Con/*w*/NAP/Addr	Network access point address	MAY
Con/*w*/NAP/AddrType	Type of address used	MAY
Con/*w*/NAP/Auth/*y*	Interior object specifying authentication scheme	MAY
Con/*w*/NAP/Auth/*y*/Id	NAP authentication id, e.g. user name.	MAY

Con/*w*/NAP/Auth/*y*/Secret	NAP authentication secret, e.g. password.	MAY
Con/*w*/PX/PortNbr	Proxy port number	MAY
Con/*w*/PX/AddrType	Type of address used	MAY
Con/*w*/PX/Addr	Proxy address	MAY
Con/*w*/PX/Auth/*z*	Interior object specifying authentication scheme	MAY
Con/*w*/PX/Auth/*z*/Id	Proxy authentication id, e.g. user name.	MAY
Con/*w*/PX/Auth/*z*/Secret	Proxy authentication secret, e.g. password.	MAY

The detailed definition of the object and valid object values are found in [8].

7 Re-bootstrapping

The bootstrap process is useful in several situations:

1. Bootstrap
   The management relationship between the client and server has not yet been established. This is the typical scenario.

2. Re-Bootstrap
   A previous bootstrap attempt contained incorrect or unusable data.

3. Nonce reset
   The client and server, due to a failure on one or the other, no longer have the same value stored for the next nonce to be used in a Notification message.

4. Client recovery
   The client fails in such a way that it cannot establish a management session with the server.

Most of this document addresses situation 1. Situations 2, 3, and 4 may in fact be addressed in several ways. For example, a "master reset" of the client device, which brings the device back to a pristine factory state, although drastic, does allow for recovery from those situations, because it changes them back into situation 1, in which the typical bootstrap message is accepted and processed by the device. Note that this type of master reset is typically triggered only by some kind of local UI event on the device.

Another form of "reset" can also be used to address situations 2, 3, and 4. The manufacturer could design another reset technique, triggered by a UI event, which applied only to management provisioning data. When the user places the device in this state, the next provisioning message is accepted, even if the server identifier matches an existing one. In other words, when the device is in this special state, new provisioning information may overwrite existing provisioning information for a known server. Obviously, this technique should be used with great care. It would be unwise to make this the default state

for the device, for example, because it provides an opportunity for malicious servers (or devices) to pose as a legitimate server, which already has many access rights in the device. If the malicious server can succeed in creating an acceptable re-bootstrap message, it can take over the device.

Finally, situation 3 (Nonce reset) can be solved in ways other than by bootstrapping. The nonce indicated is used by the server to send an acceptable notification to the device, which causes the device to initiate a management session with the server. The client is able to initiate such a session without receiving a notification, i.e. without needing this nonce. (Note that during a management session, use of an incorrect nonce can be challenged by the other party, which provides a new nonce.) In conclusion, if it can be determined that the client and server are both in good health except that they have difference notification nonce values stored, the problem can be solved by having the client initiate a management session without notification.

A device manufacturer may design a UI event that causes the client to initiate a management action. Alternatively, a timer could expire and trigger a management session, e.g. if there have been no management sessions in some number of days. (Such a technique would have to be implemented very carefully, however.) Server manufacturers can take several lessons from this discussion. First, well-designed device will only allow re-bootstrapping of a known server when the device is in a special state; the user must put the device in this state. So it is very important to perform the initial bootstrap correctly the first time.

Second, if the server should experience a failure in which it forgets which nonce is stored in which client, it will not be able to contact those clients with a notification message until this problem is fixed. Fixing the problem will very likely require action on the part of the user. Therefore, it is very important to safeguard the nonce (and shared secret) values.

Third, if a client initiates a management session on its own, i.e. without responding to a notification, the server knows this because of the alert code used in the first package sent by the client. Because the server may have sent notifications, which were ignored, perhaps due to the problem of the nonce being incorrect, the server might reset the nonce value during this management session to correct the (potential) problem.

8 References

[1] Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, **IETF**.

[2] SyncML OBEX Binding Specification, version 1.1, **SyncML**.

[3] SyncML Representation Protocol: Device Management Usage, version 1.1, **SyncML**.

[4] Provisioning Bootstrap Specification, WAP-184-ProvBoot, **WAP Forum**.

[5] Provisioning Content Specification, WAP-183-ProvCont, **WAP Forum**.

[6] Provisioning Architecture Overview, WAP-182-ProvArch, **WAP Forum**.

[7] Push OTA Protocol Specification, WAP-235-PushOTA, **WAP Forum**.

[8] SyncML Device Management Standardised Objects, version 1.1, **SyncML**

[9] SyncML Device Management Tree and Description, version 1.1, **SyncML**.