SyncML Device Management Bootstrap, version 1.1.2

Approved version 09-December-2003

Open Mobile Alliance
OMA-SyncML-DM-Bootstrap-V1_1_2-20031209-A

Continues the Technical Activities
Originated in the SyncML Initiative

Contents

1. Scope

This document defines 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.

The SyncML Initiative, Ltd. was a not-for-profit corporation formed by a group of companies who co-operated to produce an open specification for data synchronization and device management. Prior to SyncML, data synchronization and device management had been based on a set of different, proprietary protocols, each functioning only with a very limited number of devices, systems and data types. These non-interoperable technologies have complicated the tasks of users, manufacturers, service providers, and developers. Further, a proliferation of different, proprietary data synchronization and device management protocols has placed barriers to the extended use of mobile devices, has restricted data access and delivery and limited the mobility of the users.

SyncML Components

SyncML is a specification that contains the following main components:

- An XML-based representation protocol
- A synchronization protocol and a device management protocol
- Transport bindings for the protocol
- A device description framework for device management

2. References

2.1 Normative References

[DMCONF] "Device Management Conformance Requirements, Version 1.1.2". Open Mobile Alliance™. OMA-SyncML-DMConReqs-V1_1_2. URL:http:www.openmobilealliance.org/tech/docs

[DMREPU] "SyncML Representation Protocol, Device Management Usage, Version 1.1.2". Open Mobile Alliance™. OMA-SyncML-DMRepPro-V_1_1_2. URL:http:www.openmobilealliance.org/tech/docs

[DMSTDOBJ] "SyncML Device Management Standardized Objects, Version 1.1.2". Open Mobile Alliance™. OMA-SyncML-DMStdObj-V1_1_2. URL:http:www.openmobilealliance.org/tech/docs

[DMTND] "SyncML Device Management Tree and Description, Version 1.1.2". Open Mobile Alliance™. OMA-SyncML-DMTND-V1_1_2. URL:http:www.openmobilealliance.org/tech/docs

[OBEXBD] "SyncML OBEX Binding, version 1.1.2". Open Mobile Alliance™. OMA-SyncML-OBEXBinding-V1_1_2. URL:http:www.openmobilealliance.org/tech/docs

[PROVARCH] "Provisioning Architecture Overview 1.1". Open Mobile Alliance™. OMA-WAP-ProvArch-v1_1. URL:http:www.openmobilealliance.org/tech/docs

[PROVBOOT] "Provisioning Bootstrap 1.1". Open Mobile Alliance™. OMA-WAP-ProvBoot-v1_1. URL:http:www.openmobilealliance.org/tech/docs

[PROVCONT] "Provisioning Content 1.1". Open Mobile Alliance™. OMA-WAP-ProvCont-v1_1. URL:http:www.openmobilealliance.org/tech/docs

[PUSH] "WAP Push OTA Specification". Open Mobile Alliance™. OMA-WAP-235-PushOTA. URL:http:www.openmobilealliance.org/tech/docs

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997. URL:http://www.ietf.org/rfc/rfc2119.txt

2.2 Informative References

None.

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

Any reference to components of the DTD's or XML snippets are specified in this `typeface`.

3.2 Definitions

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

See the DM Tree and Description [DMTND] document for definitions of terms related to the management tree.

3.3 Abbreviations

WAP	Wireless Application Protocol
MAC	Message Authentication Code

4. 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.

5. Bootstrapping

5.1 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

5.1.1 Requirements

A SyncML DM solution capable of transforming an 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 standardized and simple ⇒ Highly interoperable
- Self sufficient and complete
- Secure
- Data format should be XML based
- Content mappable to the SyncML DM management object
- Transport encoding should be WBXML

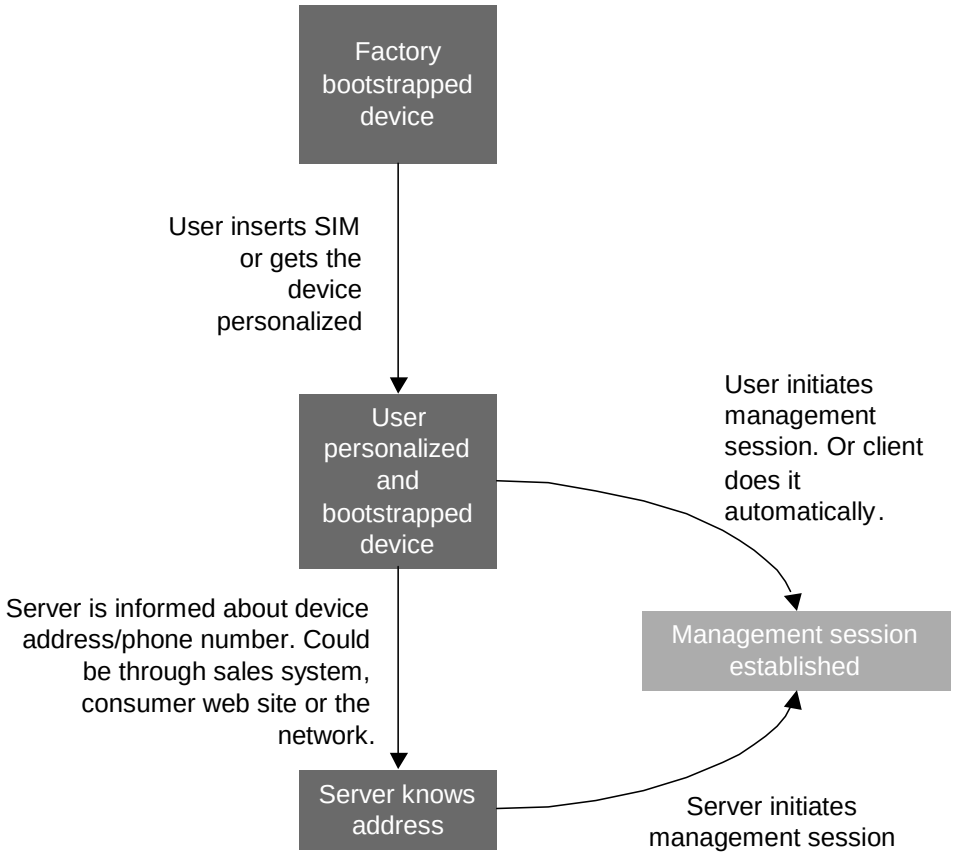
5.1.2 Solutions

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.

5.1.2.1 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 operator's 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 manufacturers may provide this service.



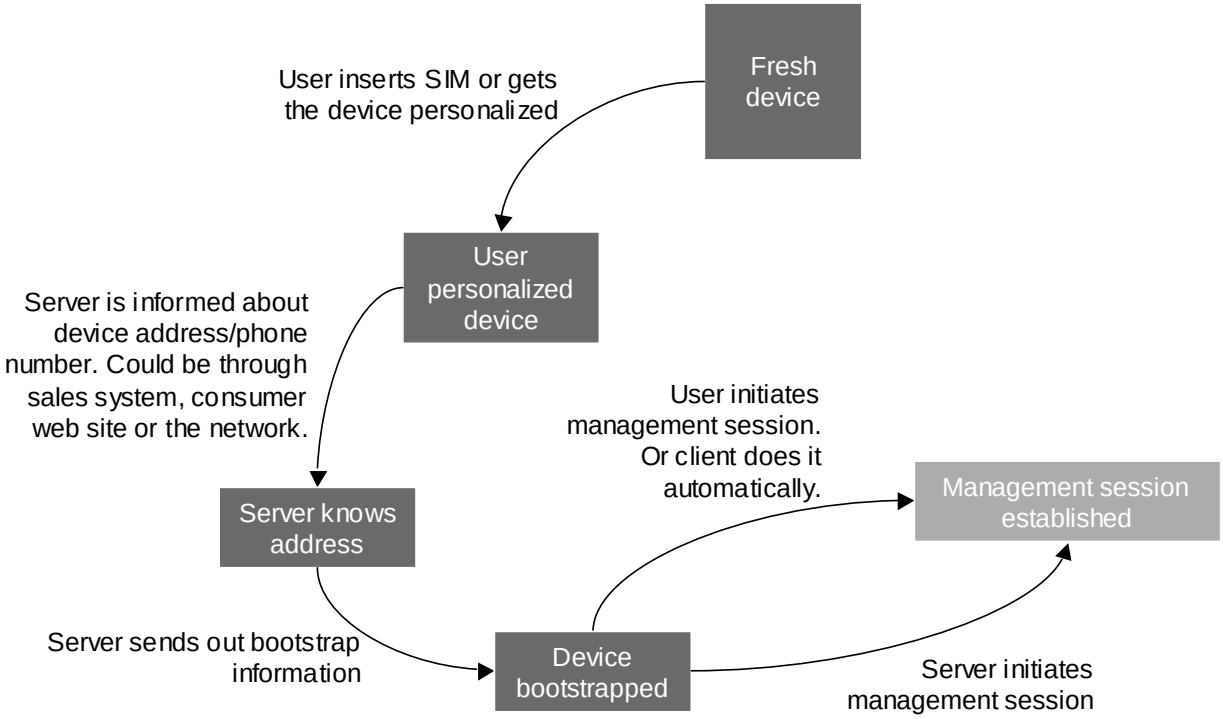
Customized bootstrap

5.1.2.2 Server initiated bootstrap

In this scenario, the devices leave the assembly line in a clean and empty state. Once a user acquires 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 attaches 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 DTMF.

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, whose 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

5.2 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 a bootstrap or provisioning mechanism. 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 its 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 OMA Provisioning architecture [PROVARCH] can be used to provision SyncML DM. The profile defines a number of parameter extensions to the WAP Provisioning Content format [PROVCONT] and how these, and other parameters, are mapped in to the DM management object. It is RECOMMENDED to use this profile

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. This profile will be removed from future versions of this specification.

5.3 WAP Profile

The Device Management Bootstrap is an application of OMA Provisioning, and based on the OMA Client Provisioning specifications version 1.1 (http://www.openmobilealliance.org/documents.asp). This mechanism has native support for a large number of bearers, both IP as well as more specialized bearers (such as various flavors of SMS).

The content of the Bootstrap message is based on the OMA Provisioning Content Specification [PROVCONT]. In order to enable the usage of the OMA Provisioning Content Specification within the Device Management Framework, the DM application registration document w7 is released by DM group to provide information how the APPLICATION characteristic in OMA Provisioning content [PROVCONT] is used.

5.3.1 Transports

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

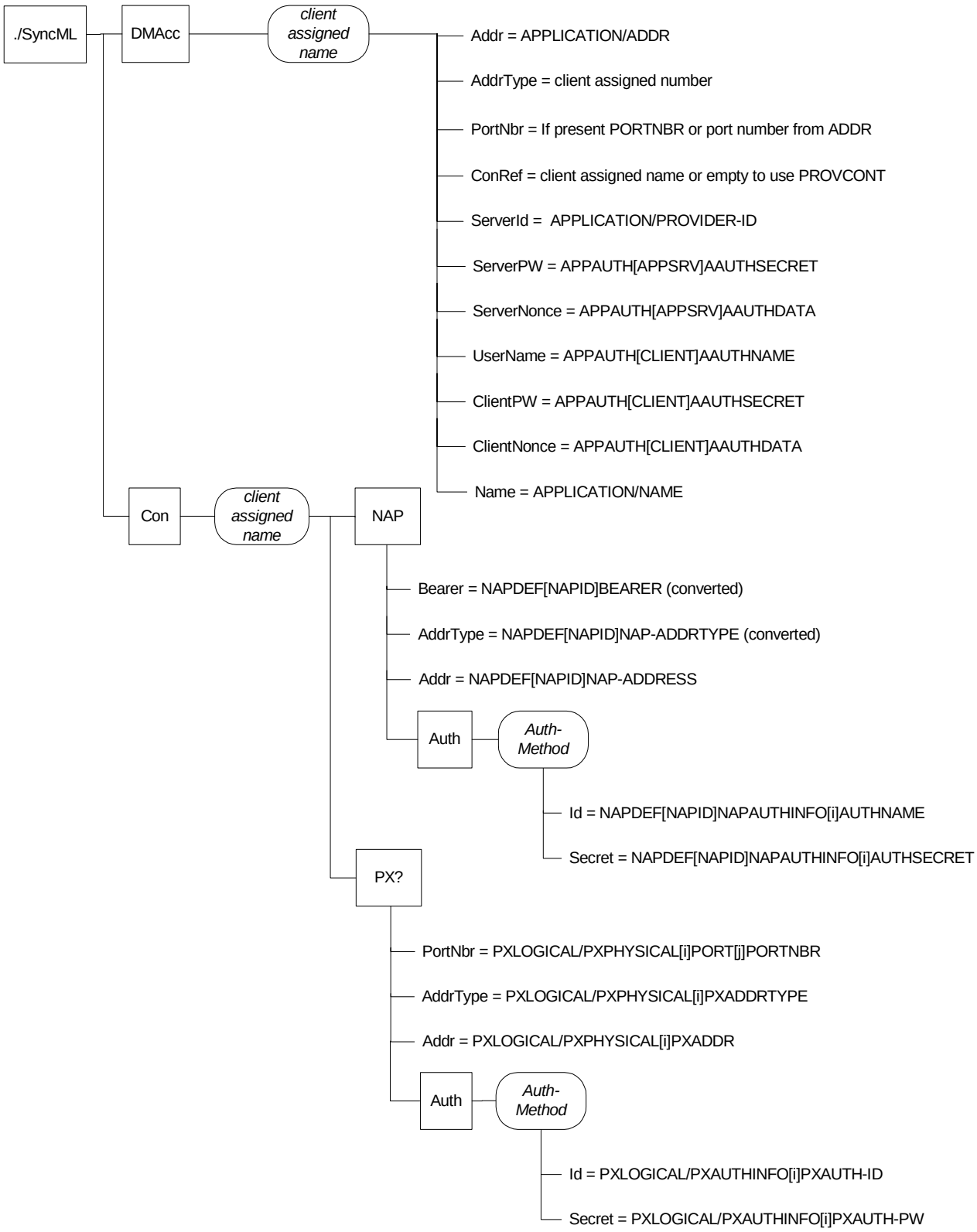
5.3.1.1 WAP Push

OTA bootstrapping using WAP profile is done as defined in the OMA Provisioning Bootstrap specification [PROVBOOT].

5.3.2 Mapping Characteristic Data to the Management Tree

The Device Management DMAcc sub-tree is initialized using values from the w7 APPLICATION characteristic, and MUST be created by the DM client. The Con sub-tree MAY be omitted by leaving the ConRef value in the DMAcc sub-tree empty. If the ConRef value is omitted, the client may use connection data that is external to the management tree, such as established in the Provisioning Content (e.g., PXLOGICAL and NAPDEF). In either case the initial management tree is intentionally sparse, as it is expected that the provisioning server will add any additional nodes and properties once a Device Management session is established.

The following diagram shows how information from the provisioning content and the w7 characteristic are mapped to the management tree. Note that the discussion concerning the creation of the DMAcc sub-tree is normative, while the example of building a Con sub-tree from bootstrapped PXLOGICAL and NAPDEF values is informative.



Requirements for DM client when it converts the w7 APPLICATION characteristic to the management tree:

- DM Client MUST assign a unique name for the child node under DMAcc. Should the server need to modify the resulting management object, it must first retrieve the node name as assigned by the DM client.
- The DM Client MUST grant Get, Replace and Delete ACL rights to the specified ServerId for the child node created under DMAcc. The provisioning server may modify this ACL to provide broader access in a subsequent DM session.

The values of each leaf in the DMAcc object is derived from the w7 APPLICATION characteristic as follows:

- **Addr** – takes on the literal value of **APPADDR/ADDR** or **APPLICATION/ADDR** depending on which is sent.
- **AddrType** – DM Client chooses the value based on the transports it supports. The DM Server may modify this in a subsequent DM session.
- **PortNbr –** takes the port value from **PORT/PORTNBR** or from **APPLICATION/ADDR** if present.
- **ConRef** – may be left empty by the DM client to use connection information maintained outside of the management tree, for example as specified within PXLOGICAL and NAPDEF. If a value is specified, it MUST be a URL that points to connectivity information stored within the management tree. Commonly, this value will identify an object under the .SyncML/Con node as discussed in the example below.
- **ServerId** – specifies the ID of the DM management server and takes on the value specified by **APPLICATION/PROVIDER-ID**. Server identification and access control is based on this value. (Note that APPLICATION/PROVIDERID is used, even if a different value is provided in the application characteristic APPAUTH[APPSVR]AAUTHNAME.)
- **ServerPW** – takes the value of **APPAUTH/AAUTHSECRET** as specified for the server by the keyword **APPSRV**.
- **ServerNonce** – takes the value of **APPAUTH/AAUTHDATA** as specified for the server by the keyword **APPSRV**.
- **UserName** – specifies the ID of the device user and takes on the value of **APPAUTH/AAUTHNAME** as specified by the keyword **CLIENT**.
- **ClientPW** – takes the value of **APPAUTH/AAUTHSECRET** as specified by the keyword **CLIENT**.
- **ClientNonce** – takes the value of **APPAUTH/AAUTHDATA** as specified by the keyword **CLIENT**.
- **Name** – takes the value from **APPLICATION/NAME**.

A DM client that has just been bootstrapped with provisioning content with values for PXLOGICAL and NAPDEF sufficient to connect to the DM server, may simply build the DMAcc sub-tree, and leave the ConRef value empty. The client could then use the connection information in PXLOGICAL and NAPDEF to establish a DM session with the server. If the w7 APPLICATION characteristic TO-PROXY is specified, the connection is made using the specified PROXY-ID. If TO-NAPID is specified, the connection is made using the specified NAPID.

Alternatively, the client could use the very same connection data to build a DM management sub-tree from the information in PXLOGICAL and NAPDEF. The following informative discussion shows how a DM client might build the Con sub-tree shown in the preceding diagram from PXLOGICAL and NAPDEF:

- **Con/NAP/…** – data is taken from the NAPDEF information specified by a NAPID, provided by the TO-NAPID value in the w7 APPLICATION characteristic or a TO-NAPID value found within a PXLOGICAL specified by the TO-PROXY in the w7 APPLICATION characteristic. The NAPDEF so identified is used to build the Con/NAP sub-tree as follows:

- **Con/NAP/Bearer** – is converted from NAPDEF/BEARER to conform to the numeric Bearer codes assigned in [DMSTDOBJ] (e.g., "GSM-GPRS" → "11").
- **Con/NAP/AddrType** – is converted from NAPDEF/NAP-ADDRTYPE to conform to the numeric Bearer codes assigned in [DMSTDOBJ] (e.g., "IPV4" → "1").
- **Con/NAP/Addr** – is taken directly from NAPDEF/NAP-ADDRESS.
- **Con/NAP/Auth/<*Auth-Method*>** – defines a node for either PAP or CHAP authentication, indicated respectively by using "PAP" or "CHAP" as the node name.
  - **Con/NAP/Auth/<*Auth-Method*>/ID** – defines the NAP authentication ID, e.g., a user name.
  - **Con/NAP/Auth/<*Auth-Method*>/Secret** – defines the NAP authentication secret, e.g., a password.

- **Con/PX/…** – data is taken from the PXLOGICAL information specified by a PROXY-ID, provided by the TO-PROXY value in the w7 APPLICATION characteristic. The PXLOGICAL so identified is used to build the Con/PX sub-tree as follows:
  - **Con/PX/PortNbr** – is taken directly from PXLOGICAL/PXPHYSICAL/PORT/PORTNBR. In the case of multiple PXPHYSICAL and PXPHYSICAL/PORT values, the DM client must choose a single value.
  - **Con/PX/AddrType** – is converted from PXLOGICAL/PXPHYSICAL/PXADDRTYPE to conform to the numeric Bearer codes assigned in [DMSTDOBJ] (e.g., "IPV4" → "1"). In the case of multiple PXPHYSICAL values, the DM client must choose a single value.
  - **Con/PX/Addr** – is taken directly from PXLOGICAL/PXPHYSICAL/PXADDRESS. In the case of multiple PXPHYSICAL values, the DM client must choose a single value.
  - **Con/PX/Auth/<*Auth-Method*>** – defines a node for HTTP-BASIC, HTTP-DIGEST, or WTLS-SS authentication, indicated respectively by using "HTTP-BASIC", "HTTP-DIGEST", or "WTLS-SS" as the node name.
    - **Con/NAP/Auth/<*Auth-Method*>/ID** – defines the proxy authentication ID, e.g., a user name.
    - **Con/NAP/Auth/<*Auth-Method*>/Secret** – defines the proxy authentication secret, e.g., a password

Note that when a DM client chooses to build a connection object within the management tree, the ConRef of the associated DMAcc object must be set to point to the corresponding connection object.

Transport Level Authentication

At the current time, there is no specific node identified in the management tree in which to store authentication credentials for HTTP "transport level" authentication. This issue is to be addressed in a subsequent release of [DMSTDOBJ]. Although this transport level authentication does not have a standard location in the management tree, its value can be conveyed to the device client by specifying an APPAUTH characteristic that does not contain an AUTHLEVEL value, as shown in the w7 example.

5.4 The Plain profile (to be removed in future DM releases)

5.4.1 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.

5.4.1.1 WAP Push

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

WAP Push application ID: `0x07`

WAP content type code (for the MIME type below): `0x42`

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

Devices without a full WAP implementation can be made to parse the WAP Push headers relatively easily. Hence it is possible to use this delivery method also for non-WAP devices that are capable of receiving concatenated SMS.

5.4.1.2 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 [OBEXBD].

5.4.2 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.

5.4.2.1 Mechanisms

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

5.4.2.1.1 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 [PROVBOOT]. 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.

5.4.2.1.2 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.

5.4.2.1.3 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.

5.4.2.1.4 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, since 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.

5.4.2.1.5 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

5.4.2.2 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 [PROVCONT]. The calculation of their values is defined in chapter 6 of [PROVBOOT].

5.4.2.3 Management tree ACL and bootstrap

During processing of a bootstrap package the normal behavior of the ACL, as specified in [DMTND], does not apply. Each item in the Add command in the bootstrap package MUST be processed successfully, according to the conditions described in section 5.4.4, in order for the bootstrap to be successful.

5.4.3 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 valid XML.

Inside the bootstrap package's SyncBody, there is one Add 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.

5.4.3.1 Use of protocol elements

5.4.3.1.1 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 [DMREPU]
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 [DMREPU]

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

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

5.4.3.1.2 Elements in SyncBody

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

This Add 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.

5.4.3.2 Example bootstrap message

```
<?xml version="1.0" encoding="UTF-8"?>
<SyncML>
   <SyncHdr>
      <VerDTD>1.1</VerDTD>
      <VerProto>DM/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>
      <Add>
         <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.-->
      </Add>
      <Final/>
   </SyncBody>
</SyncML>
```

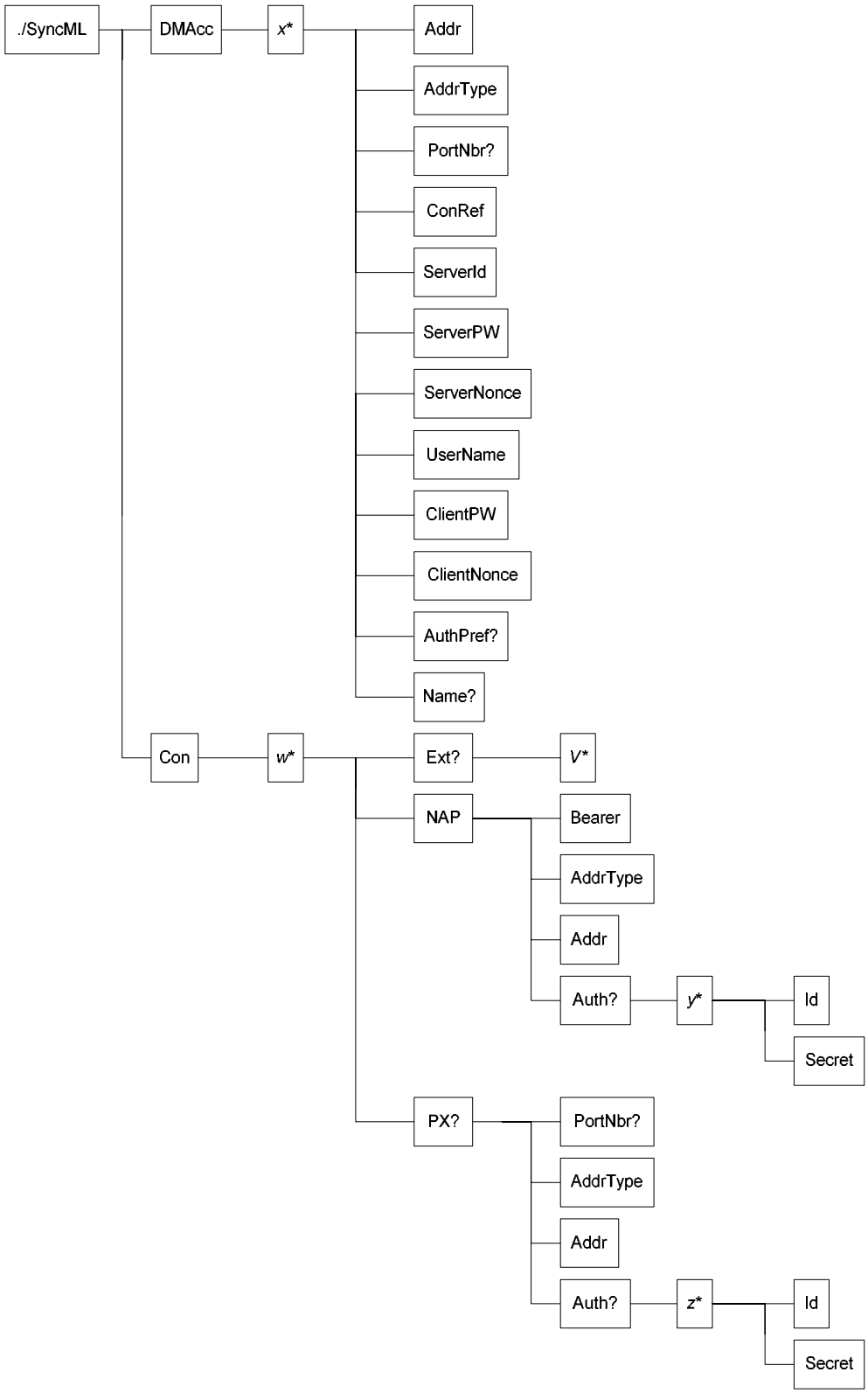
5.4.3.3 The Bootstrap Parameters

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.

Furthermore, each bootstrap message MUST only contain one DMAcc object, i.e. only one management server can be provisioned per bootstrap message.

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 depends 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 [DMSTDOBJ], 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. Nevertheless, there are some objects that must be included in any meaningful bootstrap message. These are indicated below.

The objects are listed here by their URI. They are all part of the SyncML DM object defined in [DMSTDOBJ]; 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, each URI is specified as relative and the base URI is listed in the table header.

The Con node has an interior node called Ext, located underneath *w*, at the same level as NAP and PX. This node is for vendor-specific extensions to the connectivity object. See section 5.4.4 (below) for restrictions on processing Ext items.

No additional objects in DMAcc or Con may be included by a server in a bootstrap message unless they are located under the Ext node in the Con node

Base URI: ./SyncML		
URI	**Node 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	MAY
DMAcc/*x*/ServerId	SyncML DM Server Id	MUST
DMAcc/*x*/ServerPW	The password or secret that the server will use to authenticate itself to the client. The password should be stored using Basic format (recommended locally encrypted) if Basic authentication will be used. If only MD5 or HMAC authentication will be used, the password may be stored hashed.	MUST
DMAcc/*x*/ServerNonce	This node stores the next nonce that the server will use to authenticate itself to the client.	MUST
DMAcc/*x*/UserName	SyncML DM user name. Identifies the device and/or user to the server. Is part of MD5 and HMAC calculations.	MUST
DMAcc/x/ClientPW	The password or secret that the client will use to authenticate itself to the server. The password should be stored using Basic format (recommended locally encrypted) if Basic authentication will be used. If only MD5 or HMAC authentication will be used, the password may be stored hashed.	MAY
DMAcc/x/ClientNonce	This node stores the next nonce that the client will use to authenticate itself to the server.	MAY
DMAcc/*x*/AuthPref	The server-preferred authentication type.	MAY

DMAcc/*x*/Name	Displayable name for this management account	MAY
Con/*w*/	Logical name for the following connectivity node	MAY
Con/w/Ext	Vendor-specific extensions to connectivity node.	MAY
Con/*w*/NAP/Bearer	Which bearer to use, e.g. GPRS, CSD, OBEX…	MAY
Con/*w*/NAP/Addr	Network access point address	MAY
Con/*w*/NAP/AddrType	Type of address used	MAY
Con/*w*/NAP/Auth/*y*	Interior node 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 node 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 node and valid node values are found in [DMSTDOBJ].

5.4.4 Processing of the Bootstrap

There are several situations in which the entire bootstrap message must be discarded by the client, i.e. the client MUST NOT store any information from the bootstrap message in the management tree. These situations include:

- The security of the bootstrap message fails.
- The DMAcc or Con node being created by the bootstrap already exists in the tree. (Note that this is normal behavior for the Add command.)
- The device detects that some information in the bootstrap contents is incorrect or missing.
- The ServerID of the DMAcc being created already exists in another DMAcc node.

The last requirement above means that the client processing the bootstrap message MUST also verify that the ServerID of the server being bootstrapped is unique within the client's DMAcc area of the management tree. The client must check all existing ./SyncML/DMAcc/x/ServerID values against the one that is being bootstrapped. If the new ServerID is not unique, the complete bootstrap message MUST be discarded, and MUST NOT store any information from the bootstrap message in the management tree.

Regarding the Ext area of the Con node, a device which receives a bootstrap message with items in the Ext area MUST ignore those items which it is not equipped to process, in order that the bootstrap will succeed.

After successfully processing the bootstrap, it is RECOMMENDED that the SyncML DM client automatically initiate a client-initiated session to the DM server configured in the bootstrap at the next practical opportunity (i.e., when network connectivity and other factors would allow such a connection.)

Appendix A. Static Conformance Requirements (Normative)

The static conformance requirements can be found in [DMCONF].

Appendix B. Change History (Informative)

B.1 Approved Version History

Reference	Date	Description
n/a	n/a	No previous version within OMA

B.2 Draft/Candidate 1.1.2 History

Dccument Identifier	Date	Section	Description
Class 0	10-Apr-2003	All	The initial version of this document, based on SyncML DM 1.1.1.
Class 2	10-Apr-2003	All	Change Requests listed in OMA-DM-2003-0047R3
Class 3	08-May-2003	All	Editorial corrections
Draft Version OMA-SyncML-DM-Bootstrap-V1_1_2-20030508-D	8 may 2003		Draft for TP approval
Candidate Version OMA-SyncML-DM-Bootstrap-V1_1_2-20030612-C	12 June 2003		Status Changed to Candidate by TP TP ref# OMA-TP-2003-0266R1
Class 3	09 December 2003	5.3	Change request OMA-DM-2003-0181