White Paper on Management Object Design Guidelines

Approved Version 1.0 – 29 Jan 2013

Open Mobile Alliance

OMA-WP-Management_Object_Design_Guidelines-20130129-A

Contents

Figures

1. Scope

This informative document enumerates guidelines for design of Management Objects (MOs) which comply with the OMA DM v1.2 and v1.3 enablers.

2. References

[ACMO] "OMA White Paper on Provisioning Objects, Version 1.0.1". Open Mobile Alliance™. URL:http://www.openmobilealliance.org/

[ConnMO] "OMA Device Management Connectivity Management Object, Version 1.0". Open Mobile Alliance™. URL:http://www.openmobilealliance.org/

[DM13] "OMA Device Management", Version 1.3, Open Mobile Alliance™, OMA-ERELD-DM-V1_3, URL:http://www.openmobilealliance.org/

[DMBOOT] "OMA Device Management Bootstrap", Version 1.3, Open Mobile Alliance™, OMA-TS-DM_Bootstrap-V1_3, URL:http://www.openmobilealliance.org/

[DMDICT] "OMA Device Management Dictionary", Version 1.0, Open Mobile Alliance™, OMA-SUP-DM_Dictionary-V1_0, URL:http://www.openmobilealliance.org/

[DMPROTO] "OMA Device Management Protocol", Version 1.3, Open Mobile Alliance™, OMA-TS-DM_Protocol-V1_3, URL:http://www.openmobilealliance.org/

[DMSTDOBJ] "OMA Device Management Stardardized Management Objects", Version 1.3, Open Mobile Alliance™, OMA-TS-DM_StdObj-V1_3, URL:http://www.openmobilealliance.org/

[DMTND] "OMA Device Management Tree and Description", Version 1.3, Open Mobile Alliance™, OMA-TS-DM_TND-V1_3, URL:http://www.openmobilealliance.org/

[ISO8601] "Data elements and interchange formats -- Information interchange -- Representation of dates and times", ISO 8601:2000,, URL:http://www.iso.ch

[RFC2045] "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", N. Freed & N. Borenstein, November 1996, URL:http://www.ietf.org/rfc/rfc2045.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax". T. Berners-Lee, et al. August 1998. URL:http://www.ietf.org/rfc/rfc2396.txt

3. Terminology and Conventions

3.1 Conventions

This is an informative document, which is not intended to provide testable requirements to implementations.

3.2 Definitions

Kindly consult **Error! Reference source not found.** for all definitions used in this document.

3.3 Abbreviations

Kindly consult **Error! Reference source not found.** for all abbreviations used in this document.

4. Introduction

The OMA Device Management [DM13] protocol [DMPROTO] supports the notion of Management Objects (MOs). These are abstract representations of remote management capabilities supported by the device. All available MOs pertaining to a device are organized in a hierarchical tree structure known as Management Tree. The DM Server performs remote management actions by executing operations on the nodes of the various MOs in the Management Tree. Managing resources on a device entails a two-way communication between the DM Client and the DM Server.

MOs are described using the Device Description Framework (DDF) syntax, which is described in [DMTND]. The OMA DM Working Group has laid down many guidelines for OMA and other SDOs, as well as device vendors, to define MOs. These guidelines are captured in this document.

4.1 Relationship with the ACMO White Paper

In 2005, the OMA DM Working Group (WG) started developing the "OMA Provisioning Objects - Device Management Application Characteristics Management Object" (AC_MO) to ease the transition from the Client Provisioning enabler to the Device Management enabler. Initially the White Paper discussed design guidelines for designing Application Characteristics (ACs) to be used by the Client Provisioning enabler and for designing Management Objects (MOs) to be used by the Device Management enabler. Guidelines for converting ACs to MOs were also specified. In course of time the scope of the White Paper was expanded to include a short overview of OMA DM and other enablers developed by the OMA DM WG e.g. FUMO, SCOMO, DCMO, DiagMon, etc.

The AC_MO White Paper [ACMO] was released as an Approved Reference Release in 2009. The goal was only to help out on how to define the AC and/or the corresponding MO. This transition is described in an Appendix C – "General Mapping" in the [DMBOOT]. Since the release of the AC_MO White Paper, it has become clear that some design guidelines for designing MOs were inadvertently left out, while the design guidelines for ACs seem to be covered quite extensively in the AC_MO WP. The DM Enabler defines the rules on what is a valid DDF but the AC_MO WP describes best practice and describes good combinations.

The OMA DM Working Group has laid down many guidelines for OMA and other SDOs, as well as device vendors, to define MOs. This White Paper focuses exclusively on these guidelines and captures them. While this White Paper covers only a subset of topics discussed in the AC_MO Whitepaper, it does so in much greater depth than the AC_MO White Paper.

5. MO Design Considerations

5.1 MO Description Syntax

The OMA DM MOs are defined using the OMA DM Device Description Framework (DDF) [DMTND]. The use of this description framework produces detailed information about the Management Tree of the device.

5.2 MO Description in Graphical Notation

Due to the high level of detail in the DDF, it is sometimes hard for humans to get an overview of a particular object's structure by examining the DDF. In order to make it easier to quickly get an overview of how a Management Object is organized and intended to be used, the OMA DM Working Group has developed a graphical notation in the shape of a tree diagram. The graphical notation borrows from the syntax of DTDs for XML. The characters and their meaning are defined in the following table.

Character	Meaning
+	one or many occurrences
*	zero or more occurrences
?	zero or one occurrences

If none of these characters is used, then the default occurrence is exactly once.

Another feature of the DDF that needs to have a corresponding graphical notation is the un-named block. Un-named are nodes which act as placeholders in the description and are instantiated with information when the nodes are used at run-time. Un-named blocks in the description are represented by less than ("<") and a greater than (">") character containing a lower case character, (e.g. "<x>").

Each block in the graphical notation corresponds to a described node and the text is the name of the node. If a block contains an <x>, it means that the name is not known in the description and that it will be assigned at run-time creation.

The names of all ancestral nodes are used to construct the URI for each node in the MO. It is not possible to see the actual parameters, or data, stored in the nodes by looking at the graphical notation of a MO.

Some MOs specify explicit names of nodes but the name is still assigned at run-time. These nodes may not be described as <x> in the DDF file and it is possible to use the syntax [NodeName] where "NodeName" is a logical name for the node. In this case the graphical representation and the DDF file will contain the logical name of the node to improve the readability.

The nodes which the DM Client is required to support are drawn in the graphical notation with solid line, while nodes whose support is not mandatory for the DM Client are drawn with a dotted line.

Leaf nodes are drawn as rectangle while interior nodes are drawn as rectangle with rounded corners.

The following is an example of what a MO can look like when it is expressed using the graphical notation:

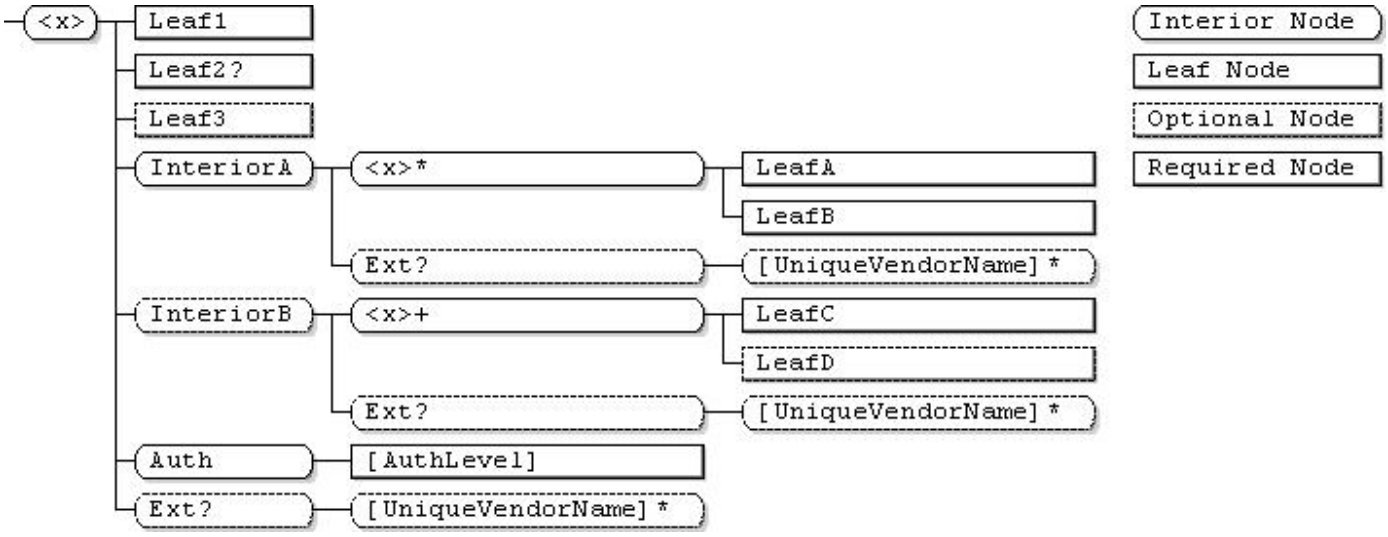

Figure 1: Example of a MO pictured using the graphical notation

Naturally, this graphical overview does not show all details of the full description, but it provides a good map of the description so that it is easier to find the individual node. Although the figure only provides an overall view of the description, there are still some things worth noticing.

In example shown in Figure 1 all blocks with names in place occur exactly once, except Leaf2, InteriorA/<x>, InteriorB/<x>, all Ext nodes and their children.

Leaf3, InteriorB, all Ext and their children nodes are optional to be supported by the DM Client.

All nodes whose name starts with “Leaf” and the node “[AAuthLevel]” are *leaf* nodes. They may contain data but cannot contain child nodes; all other nodes are *interior* nodes, they cannot contain data but can contain child nodes.

The un-named leaf nodes are marked with * or +. This means that although the description only contains one node description at this position in the tree, there can be any number of instantiated nodes at run-time, including none in the first case, at least one in the second. The only limit is that the node names must be unique and the DM Client must have sufficient memory to store the nodes.

The Figure 2 shows an example of what the previous MO could look like at run-time.

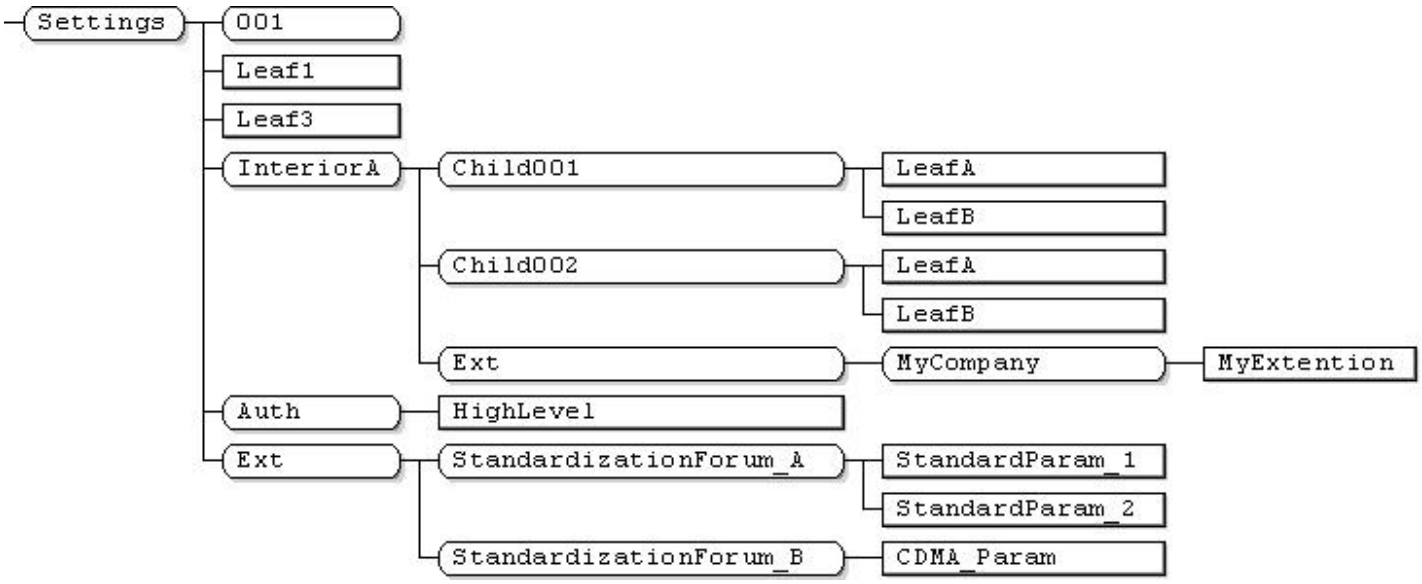

Figure 2: Example of an instance of MO

The difference between this figure and the previous one is that now the un-named blocks have been instantiated and some optional nodes are not shown.

Note that none of the stored data in the leaf nodes is shown in the figure: only the node names are visible.

5.2.1 DDF Compliance

The MO descriptions are normative. However, they also contain a number of informative aspects that could be included to enhance readability or to serve as examples. Other informative aspects are, for instance, the `ZeroOrMore` and `OneOrMore` elements, where implementations may introduce restrictions. All these exceptions are listed here:

- All XML comments, e.g. "`<!-- some text -->`", are informative.
- The descriptions do not contain an `RTProperties` element, or any of its child elements, but a description of an actual implementation of this object may include these.
- If a default value for a leaf node is specified in a description, by the `DefaultValue` element, an implementation must supply its own appropriate value for this element. If the `DefaultValue` element is present in the description of a node, it must be present in the implementation, but may have a different value.
- The value of all `Man, Mod, Description` and `DFTitle` elements are informative and included only as examples.
- Below the interior `Ext` node, an implementation may add further nodes at will.
- The contents of the `AccessType` element may be extended by an implementation.
- If any of the following `AccessType` values are specified, they must not be removed in an implementation: `Copy, Delete, Exec, Get,` and `Replace`.
- If the `AccessType` value of `Add` is specified, the node may be removed in an implementation if the implementation only supports a fixed number of child nodes.

- An implementation may replace the `ZeroOrMore` or `OneOreMore` elements with `ZeroOrN` or `OneOrN` respectively. An appropriate value for `N` must also be given with the `…OrN` elements.
- Path element is informative.
- All nodes are included but the specification may allow an implementation to not require that all nodes are to be supported.

5.3 Definition and Description of Nodes

Each node in the MO definition is characterized by five node description characteristics: *Name*, *Status*, *Occurrence*, *Format* and *Minimum Set of Access Types*. These node description characteristics are described further on in this White Paper.

OMA DM recommends describing each MO node in the tabular structure shown below.

Note: it is recommended to report the MOID of the MO in the description of the root node of the MO.

This is an example:

<ID>

Status	Occurrence	Format	Min. Access Types
Required	One	chr	Get

This leaf node defines the identity of the one specific network access point which an instance of this management object represents

5.3.1 NodeName

The node Name is a string specifing the name of the described Node, and it is the name by which the node is addressed in the Management Tree. Since it is used as last segment of Target URI, see [RFC2396] for general restrictions.

The maximum length of node Name is defined in ./DevDetail/URI/MaxSegLen as described in [DMSTDOBJ].

The node Name element may be empty (“un-named”, see Section **Error! Reference source not found.**), in which case the name of the node must be assigned when the node is created with the last segment of the URI specified as Target for the command that results in the node being created.

5.3.2 Node Status

The node Status indicates whether or not the DM Client needs to support the node. It has two values: *Required* and *Optional*. If the Status is “Required” then the Client must support that node, provided the parent node of this node is supported. If the Status is “Optional”, the Client is not required to support the node.

The node Status makes sense only for DDFs issued by SDOs. A vendor DDF should not provide this information, since it is implied: if a vendor’s DM implementation supports a certain *Optional* node, it will be present in the vendor’s DDF file; if the vendor’s DM implementation does not support a certain *Optional* node, it will not be present in the vendor’s DDF file.

It needs to be noted that the node Status is not supported by the DDF DTD. The node Status information needs to be conveyed via an XML comment in the DDF file. Here is an example:

```
  <Node>     <!-- node definition begin -->
    <NodeName>SomeName</NodeName>
    <!--Status: Required-->
    <DFProperties>
      <DFFormat>
        <chr />
      </DFFormat>
      <Occurrence>
        <ZeroOrOne />
      </Occurrence>
    </DFProperties>
</Node>     <!-- node definition end -->
```

5.3.3 Node Occurrence

The node Occurrence specifies the allowed number of instances for a node within the subtree that is rooted at its parent node. The possible values for node Occurrence are as follows:

- ZeroOrOne
- ZeroOrMore
- ZeroOrN (N is a character string representing a positive integer)
- One
- OneOrN (N is a character string representing a positive integer)
- OneOrMore

It is important to note that Status and Occurrence are orthogonal concepts. The following table summarizes the semantics of node Status and Occurrence:

Node Status Value	Node Occurrence Value	Significance
Required	ZeroOrOne	The Device is required to support this node, provided the parent node of this node is supported. At any time, the node may or may not be present in the subtree rooted at the parent node of this node
	ZeroOrMore	The Device is required to support this node, provided the parent node of this node is supported. At any time there can be any number of instances of this node in the subtree rooted at the parent node of this node

Node Status Value	Node Occurrence Value	Significance
	ZeroOrN	The Device is required to support this node, provided the parent node of this node is supported. At any time there can be up to N instances of this node in the subtree rooted at the parent node of this node
	One	The Device is required to support this node, provided the parent node of this node is supported. At any time there will be precisely one instance of this node in the subtree rooted at the parent node of this node. In other words, this node will come into existence the moment its parent node is instantiated
	OneOrN	The Device is required to support this node, provided the parent node of this node is supported. At any time there will be at least 1 and at most N instances of this node in the subtree rooted at the parent node of this node
	OneOrMore	The Device is required to support this node, provided the parent node of this node is supported. At any time there will be at least 1 instance of this node in the subtree rooted at the parent node of this node
Optional	ZeroOrOne	If the Device supports this node, at any time the node may or may not be present in the subtree rooted at the parent node of this node
	ZeroOrMore	If the Device supports this node, at any time there can be any number of instances of this node in the subtree rooted at the parent node of this node
	ZeroOrN	If the Device supports this node, at any time there can be up to N instances of this node in the subtree rooted at the parent node of this node
	One	If the Device supports this node, at any time there will be precisely one instance of this node in the subtree rooted at the parent node of this node.
	OneOrN	If the Device supports this node, at any time there will be at least 1 and at most N instances of this node in the subtree rooted at the parent node of this node
	OneOrMore	If the Device supports this node, at any time there will be at least 1 instance of

Node Status Value	Node Occurrence Value	Significance
		this node in the subtree rooted at the parent node of this node

5.3.4 Minimum Set of Access Types

When deciding upon minimum set of access type guidance, the recommendation from OMA DM is as a general rule to grant only *Get* access type for *Min. Access Types* on the various nodes within the standardized MO definition, except in some special cases, where other access types may be required.

The allowed values for access type are any combination of the following:

- *Add*
- *Copy*
- *Delete*
- *Exec*
- *Get*
- *Replace*

In some instances there may be a need to explicitly disallow a certain type of access for a given node. This is indicated by pre-pending the access type with the keyword "*No*". For example, it is strongly recommended that any node that deals with security related information, such as user-id and password, should not be granted the *Get* or *Copy* access type. The *Min. Access Types* for such a node should include *No Get* and *No Copy*.

It needs to be noted that specifying only Get access type for *Min. Access Types* of the node within the MO does not restrict an implementation from supporting additional access types for that node since the standardized MO only specifies the minimum set of access types that the node is required to support.

5.3.5 Node Format

The node Format provides information about the data format of the current node value. The possible Format values are listed in the following table:

`b64`	The format of the node content information is binary data that has been character encoded using the Base64 transfer encoding defined by [RFC2045].
`bin`	The format of the node content information is raw binary data. The value may be encoded with base64 transport encoding.
`bool`	The encoding of the node content information is either case sensitive "true" or case sensitive "false".
`chr`	The format of the node content information is clear-text in the character set specified on the transport protocol, the MIME content type header or the XML prolog.
`int`	The format of the node content information is numeric text representing a 32-bit signed integer.
`node`	The node content represents an interior object in the management tree.
`null`	The node has no content. For example, when a leaf node serves only as the target of an Exec command, the Format value can be set as "null". That is, it is never intended to have a value.

`xml`	The format of the node content information is XML structured mark-up data.
`date`	The format of the node content information is compliant to [ISO8601] with the century being included in the year.
`time`	The format of the node content information is compliant to [ISO8601].
`float`	The format of the node content information is a single precision 32 bit floating point real number.

5.4 Instantiating MOs on the Device

Management Objects are created on the device in one of three ways:

- Fully static MOs — the DM Server cannot create or delete these MOs. The MOs are fully specified in static form by the DM Client vendor. Node values within the MOs can be replaced, but nodes within the MOs cannot be added or removed.
- Fully dynamic MOs are created node-by-node typically by the DM Server (but could sometimes be created by the DM Client, with notification to the DM Server) at run-time.
- Hybrid MOs— after the dynamic creation of a parent node by the DM Server, the device automatically creates a factory-provisioned set of sub-nodes, and may fill in some default data as well. The static parts of these MOs are the automatically created sub-nodes and the default data (which is up to the implementation). This model may also be used within an MO to provide for repeating groups of related nodes.

Devices will likely employ all three models.

Device manufacturers will decide if they implement nodes as static or dynamic. They will publish which method they have used in the corresponding DDF file for all nodes. It is common to specify the minimum access type to only GET since the commands REPLACE, ADD and DELETE are dependent of the device manufacturers' implementation choice.

5.5 MO Content Guidance

When creating a MO, the creators should take into account that the aim is to have a standardized way to manage the information present in the device.

It should also be noted that not all the parameters of a standardized MO have to be mandatory. Some information may be relevant for a substantial number of actors, and irrelevant for others. For this case, allowing optional elements lets both groups use the MO without undue burden.

It should also be noted that, in order to decrease interoperability problems, it is better to make sure all relevant parameters are in the lowest versions of the MO (if there is more than one of this), even if they are optional, since incrementing the number of parameters in a later version could break backward compatibility. When in doubt about a node, add it in.

When creating a MO, there are some fields that are recommended.

- Parameters to be used by an application on the device, but that may be changed by the Management Authority.
- Date or Version of the data
    - Information of when the data was last set (this will allow the Management Authority and the Device to know which ones are the most updated parameters).
- Connectivity parameters.
    - Connectivity parameters should reuse, if possible, other already existing ConnMOs that may be on the device. This means that in order to allow more than one application the use of the same connectivity parameters (for example in case of using the same Server), the MO should point, as an example, to existing instances of ConnMO.

    - QoS Parameters

      These parameters are typically bearer specific. The MO should reference already existing bearer specific QoS parameters that are specified by ConnMO. If insufficient, the additional information specific QoS parameters may have to be specified as well.

- Application information.
    - Information of the application that may later be interesting for trouble shooting (by diagnosis and monitoring, customer care, etc).
    - Information of the application that may be interesting to be accessed remotely (by customer care).
- Behaviour: it is also possible to indicate part of the behaviour associated to the MO in the MO description.
- Nodes to be targeted with commands.
    - It is possible to relate a node that would be used to start an action in the device: e.g. start a firmware download, install a software component on the device.
    - When possible, it is recommended to group such related nodes under (one or more) Operations parent node(s).
- Proprietary Extensions:
    - All MO should have a sub-node allowing a vendor to include any proprietary sub-nodes in order to include any vendor specific extensions. The name of this node should be “Ext”.
- Exec command:
    - For related operations, realized via the Exec command, should be defined on nodes under the same parent node, which is preferably named “Operations”.

6. MO Design Guidelines

Over time, the OMA-DM Working Group has developed a number of useful guidelines for designing MOs. Some of these guidelines are sound design practices while others are imposed by the DMTND requirements.

These guidelines are listed below:

1. Any node that deals with security related information, such as user-id and password, should not be granted the *Get* or *Copy* access type.
2. Interior nodes are required to have "node" as the Format value.
3. The name "Ext" is reserved for nodes that are vendor extensions to the MO definitions.

   This optional interior node designates the single top-level branch of the management object tree into which vendor's extensions may be supported, permanently or dynamically.

   For example:

   IP/IPv4/Ext

Status	*Occurrence*	*Format*	*Min. Access Types*
Optional	*ZeroOrOne*	*node*	*Get*

   Ext sub trees MAY be included in various places within a Management Object. to provide flexible points of extension for implementation-specific parameters. Vendor extensions must not be defined outside of one of these Ext sub-trees.
4. An unnamed node is not allowed to have any sibling node.
5. If a collection of related nodes within an MO definition is expected to be repeated a certain number of times, it is recommended to introduce an unnamed node in the MO definition and have all the nodes subtend from this unnamed node.
6. Named nodes can only have Occurrence values of *ZeroOrOne* or *One*.
7. Any node with Occurrence of *ZeroOrOne* or *ZeroOrN* or *ZeroOrMore* should be granted the *Get* access type, in addition to other access types that it may support. This is to enable a DM Server to query whether the node is present prior to executing an operation on the node. For example, an executable node with Occurrence of *ZeroOrOne* should have its Min. Access Types set to "*Get, Exec*" instead of "*Get*" only.
8. When a leaf node serves only as the target of an Exec command, its Format value should be set as "null".
9. Nodes with Scope *Permanent* cannot have the *Add* or *Delete* access types.
10. When the MIME Type of the node content value needs to be specified (i.e. in case of binary content), a specific node should be defined with this purpose, avoiding the usage of DFType properties.
11. The MOID is a string which identifies uniquely (via OMNA registry) the content of a MO and should not be related to the version of specifications which in case act as a container of the MO. For instance, if different versions of an OMA Enabler use the same MO (in terms of content and structure), the MOID should be the same despite the different enabler version.

6.1 DDF Files Considerations

6.1.1 Standardized DDF Files

Since the standardized DDF files contain machine readable information, the DM Client vendor may use standardized DDF files to verify the implementation compliance to the standard specifications. The standardized DDF file may contain

additional information compared to what is valid according to the DTD of the DDF files: for instance, a standardized DDF file contains both mandatory and optional nodes and nodes which occurrence can be `ZeroOrMore`, while an implementation DDF file reports strictly the node which are implemented.

The rules defined by the [DMDDFDTD] define how DDF must be handled by the DM Client and the DM Server; in addition to these rules, other information can be helpful for the DM Client vendors in order to create implementation DDF files that define the DM Tree supported by a DM Client. These DDF files may also be used as input for creating the diagram of the MO Tree structure. For this reason, the following information may be included in standardized DDF files:

- The node status is defined by a XML Comment as next sibling to the node "NodeName"; if the value is *"Required"* then the DM Client must support the node (if the parent node is supported); if the value is *"Optional"*, then the node is not unconditional mandatory to support in the implementation.

  The syntax for this is: "<!-- Status: Required -->" or "<!-- Status: Optional -->".

  If an interior node <x> is *Optional* and its child ChildA is *Required*, then DM Client must support ChildA only if <x> is supported.

  If the support of a node depends from conditions external to the MO (for instance, if the device support a physical feature, then MO must include the specific node), then the node occurrence should be *Optional*.

- The Path element is used to define the location of the MO in the DM Tree. If the standard specification does not define a fixed location into the DM Tree, then value of the Path element is "…"; if the specifications defines an unconditional fixed location into the DM Tree, then the Path element contains this exact location.
- In addition to specifying the minimum set of DM Commands which the node must support, standard specification can specify explicitly the DM Commands which are NOT allowed for that node. This is achieved by adding a XML Comment for each allowed DM Command as child of the "AccessType" Element.

  The valid syntax is to use the word "No" plus the space character plus the unallowed DM Command, for example "<!-- No Get -->".

6.1.2 Vendor DDF Files

Standardized MO definitions, issued by SDOs, generally do not specify the value for certain DDF elements like Path, Scope and CaseSense. These may be provided in the vendor DDFs. Also, as mentioned previously, a vendor DDF should not provide the node Status information, since it is implied. For example, if a vendor's DM implementation supports a certain *Optional* node, it will be present in the vendor's DDF file and if the vendor's DM implementation does not support a certain *Optional* node, it will not be present in the vendor's DDF file.

Use of the *RTProperties* element in the DDF is discouraged unless it is used to specify which run-time properties a node will support and/or their default values.

7. MO Registration with OMNA

OMNA (Open Mobile Naming Authority) maintains a registry of MOs at the following location:

http://www.openmobilealliance.org/Tech/omna/omna-dm_mo-registry.aspx

Registering a MO is an important task. It allows a party to associate a Management Object identifier (MOI) to a specific MO and to let the rest of the world know about that MO. The MOI registrations may be made by OMA Working Groups or external organizations.

Registering consists in gathering all information relevant to a MO in a document that will later be referenced in the OMNA website.

In order to register a MO there is a process that needs to be followed. Please note that vendors registering MOs need only to do step 6:

1. OMA Working Groups must use the latest support document template as the baseline for the MO DDF (OMA-Template-SUPgeneric-YYYYMMDD-I.txt). Non OMA parties are encouraged to use this template as well.
2. Create MO specification. OMA Working Groups must use a standalone document for the MO using the latest TS template or DDS template. Typically the Data Model MO will use the DDS template. However, one enabler may contain two functional MOs and one data model MO, then all documents could be based on the TS template. Non OMA parties are encouraged to use these templates as well.
3. Other Standard bodies are highly recommended to send the MO for review to the OMA DM WG. Please note that the MO will not be agreed by the OMA WG, just reviewed for compliance with DM protocol. This review will not have any official status, but will help determine potential problems with the MO.
4. Wait for the OMA review of the document.
5. Modify the MO and the associated DDF according to the comments from the OMA DM WG (if any comments have been received).
6. Complete the electronic form on the OMNA MO Request page for submission. The linkage for this page is http://www.openmobilealliance.org/tech/omna. The required information is for any party registering an MO is:
    a. Name of the submitter.
    b. Email address of the submitter.
    c. URL of MO DDF file (optional).
    d. URL of MO specification that defines the Management Object.
    e. Requested MO registration identifier.
    f. Short description of the MO.

A copy of the MO DDF file needs to be sent to the OMNA Secretary. After some review, the OMNA Secretary will revise the documentation and most likely assign the requested MOI to the MO.

At this moment, the registration will be considered complete and all relevant information (all parameter listed on top plus the assigned MOI) will be posted at the OMA-OMNA website.

OMNA maintains a registry of values used for Managed Object (MO) descriptions. In all cases, the registry provides for allocation of the needed MO URN value and serves as a repository for the MO descriptions. The linkage for the registry is http://www.openmobilealliance.org/tech/omna/omna-dm_mo-registry.htm

The MO registry is divided into three segments: two named collections which are managed by OMNA (one for OMA WG defined objects and one for external) and one undefined set for testing or private use. These labels are described in the following table:

Range	Description
oma-*label*	Assignments of values to MOs defined by OMA Work Groups
ext-*label*	Assignments of values to MOs defined by external organizations
x-*label*	Values that are used for testing or private use - will not be recorded

Here are some examples for URN based MOI for information:

MO Value	MO Identifier
oma-dm-devinfo	urn:oma:mo:oma-dm-devinfo:1.0
ext-3gpp-vcc	urn:oma:mo:ext-3gpp-vcc:1.0
x-private-test	urn:oma:mo:x-private-test:1.0

Appendix A. Change History (Informative)

Document Identifier	Date	Sections	Description
OMA-WP-Management_Object_Design_Guidelines-20130129-A	29 Jan 2013	n/a	Status changed to Approved by TP TP Ref # OMA-TP-2013-0006-INP_Management_Object_Design_Guidelines_V1_0_RRP_for_Final_Approval