Diagnostics and Monitoring Management Object Framework

Approved Version 1.2 – 08 Oct 2013

Open Mobile Alliance

OMA-TS-DiagMonMOFrame-V1_2-20131008-A

Contents

1. SCOPE 4
2. REFERENCES 5
2.1 NORMATIVE REFERENCES 5
2.2 INFORMATIVE REFERENCES 5
3. TERMINOLOGY AND CONVENTIONS 6
3.1 CONVENTIONS 6
3.2 DEFINITIONS 6
3.3 ABBREVIATIONS 6
4. INTRODUCTION 7
5. THE DIAGMON FRAMEWORK 8
5.1 DIAGMON MO 8
5.2 FIGURE OF THE MANAGEMENT OBJECT (INFORMATIVE) 9
5.3 DIAGMON MO PARAMETERS 9
6. INVOCATION AND REPORTING BEHAVIOR 14
6.1 INVOKING AND STOPPING DIAGNOSTIC FUNCTIONS 14
6.1.1 Non-continuously Available Functions 14
6.1.2 Continuously Available Functions 14
6.2 RESULT REPORTING 14
6.2.1 Explicitly Invoked Functions 14
6.2.2 Continuously Available Functions 16
6.2.3 Posting data to DiagDataURL 16
6.2.4 Posting Data Over Report Channel 16
6.3 RESULT CODES 17
APPENDIX A. CHANGE HISTORY (INFORMATIVE) 18
A.1 APPROVED VERSION HISTORY 18
APPENDIX B. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) 19
B.1 SCR FOR DIAGMON TREE STRUCTURE 19
B.2 SCR FOR DIAGMON CLIENT 19
B.3 SCR FOR DIAGMON SERVER 19
APPENDIX C. BEST PRACTICES FOR DEFINING DIAGMON FUNCTIONS (INFORMATIVE) 21

Figures

Figure 1: DiagMon Management Object 9

Tables

Table 1: DiagMon Result Codes 17

1. Scope

This document describes DiagMonMO employed in DiagMon management that builds on and leverages the OMA DM protocol [DMPRO]. It provides standard DM Management Objects and associated client-side and server-side behaviour necessary to conduct DiagMon activities on mobile devices.

The DiagMonMO defined in this specification will provide the DiagMon System with a generic framework and interface to the device functions that perform different DiagMon activities. These functions are not specified in this enabler release.

2. References

2.1 Normative References

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

[DMPRO] "OMA Device Management Protocol, Version 1.3". Open Mobile Alliance™.
OMA-TS-DM-Protocol-V1_2,
URL: http://www.openmobilealliance.org/

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

[RFC2388] "Returning Values from Forms: multipart/form-data", L. Masinter, August 1998,
URL: http:/www.ietf.org/rfc/rfc2388.txt

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures,
URL:http://www.openmobilealliance.org/

2.2 Informative References

[OMADICT] "Dictionary for OMA Specifications", Version 2.9, Open Mobile Alliance™,
OMA-ORG-Dictionary-Vx_y, URL:http://www.openmobilealliance.org/

[RFC3986] "Uniform Resource Identifier (URI): Generic Syntax". Network Working Group. January 2005.
URL: http://www.ietf.org/rfc/rfc3986.txt

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.

3.2 Definitions

Interior Node	A node that may have child nodes, but cannot store any value. The Format property of an interior node is node.
Leaf Node	A node that can store a value, but cannot have child nodes. The Format property of a leaf node is not node.
Management Object	A MO is the data model for information which is a logical part of the interfaces exposed by DM components
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 the access control lists.
Node	A node is a single element in a management tree. There can be two kinds of nodes in a management tree: interior nodes and leaf nodes. The Format property of a node provides information about whether a node is a leaf or an interior node.

Kindly consult [OMADICT] for all other definitions used in this document.

3.3 Abbreviations

Kindly consult [OMADICTOMADICT] for all other abbreviations used in this document.

4. Introduction

The Diagnostics and Monitoring (DiagMon) Functions perform various DiagMon activities in order to diagnose problems, report data, or collect measurements data associated with a device, with applications on the device, with the network in which the device operates or with access to services provided by the network.

The purpose of the DiagMonMO is to provide a generic framework and interface for different DiagMon Functions so that they can be managed in a consistent manner by using the same description framework. The specific DiagMon Functions will not be defined in this specification.

5. The DiagMon Framework

The generic management interface for DiagMon Functions can be broadly divided into two parts: the DiagMon Management Objects and the Alert message. Through this interface a DiagMon Server can configure or invoke the functions behind it, and retrieve results of Diagnostics or monitored information. Note that this interface framework does not present any specific function. Instead, it is intended to be used to define a wide range of DiagMon Functions. For this purpose, every DiagMon Function will have its own MO Identifiers which are used to identify the function represented by this framework. With this identifier, the server is able to get further information of the function.

In addition, this generic management interface is designed to cover different processing types of various functions – synchronous, asynchronous or always running - and also to allow multiple instantiation of the same function. In case of multiple instantiations, separate management object instances need to be created for each instantiation: each instance can be distinguished by the URIs of the management object.

As explained above, this generic management interface provides a common template that can be reused among a wide range of functions, whilst allowing flexible extensions (or customization) for different functions.

5.1 DiagMon MO

A DiagMon Function resides on the device. The DiagMonMO makes this function remotely callable by theDiagMon Server. The DiagMonMO is used as the means to expose the DiagMon Function to the DiagMon Server. The DiagMonMO will provide the following management capabilities and diagnostics and monitoring operations

- Invoke the DiagMon Functions
- Stop the DiagMon Functions
- Configure the DiagMon Funtions for reporting, execution, etc.
- Retrieves the DiagMon Data or the results
- Notifications using the Generic Alert

The management objects associated with DiagMon management are assembled under an internal node *x* (dynamically or statically created).

Management Object identifier: `urn:oma:mo:oma-diagmon:1.1`. This identifier SHALL be replaced by the identifier specified in the different DiagMon functions and registered with OMNA.

This MO is the design pattern for various DiagMon Functions. DiagMon Functions are recommended to conform to the structure specified in this specification and are allowed to include or exclude certain nodes if necessary. DiagMon Functions can also add function specific nodes. Additionally the DiagMon Functions can override the status and occurrence of the DiagMonMO nodes but SHOULD NOT change the format of the DiagMonMO nodes.

Protocol Compatibility: This object is compatible with OMA Device Management protocol specifications [DM].

5.2 Figure of the Management Object (Informative)

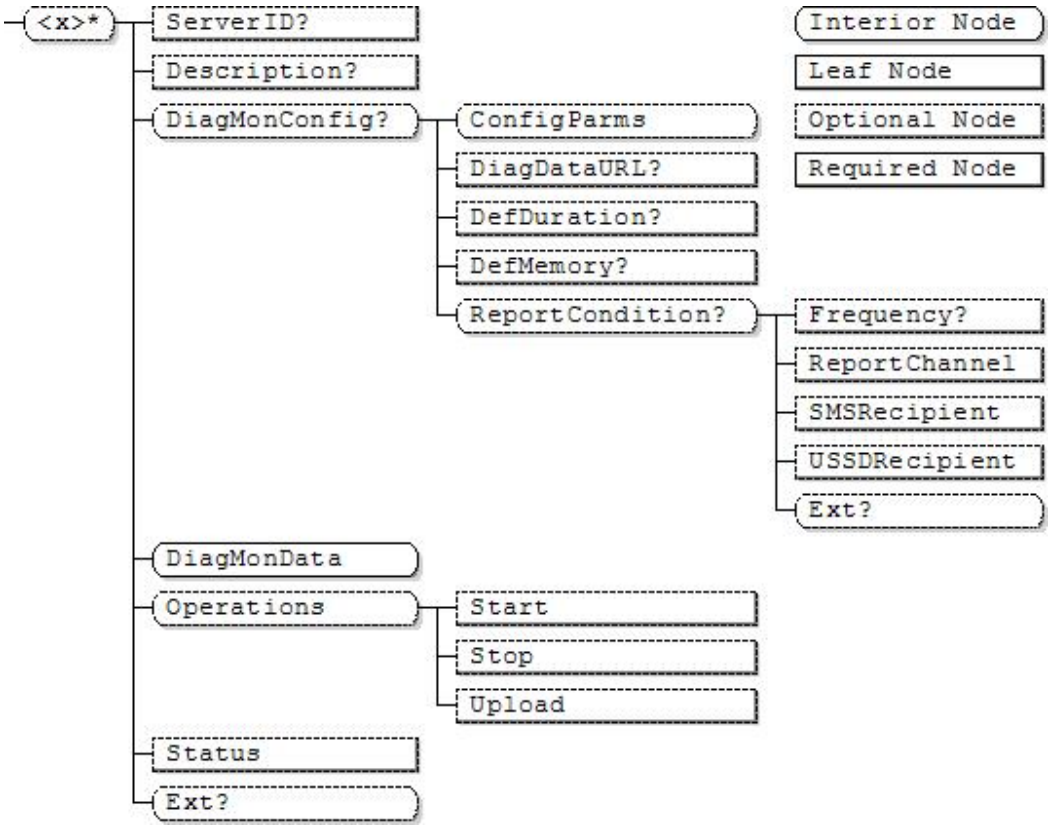

Figure 1: DiagMon Management Object

5.3 DiagMon MO Parameters

This section describes the properties for DiagMon MO

<x>

Status	Tree Occurrence	Format	Min. Access Types
Optional	ZeroOrMore	node	Get

This interior node groups together the parameters of the DiagMon MO. The ancestor elements of this node define the position in the DM tree of this MO.

The type of this node MUST be the Function Management Object ID "`urn:oma:mo:oma-diagmon:1.1`".

<x>/ServerID

Status	Tree Occurrence	Format	Min. Access Types
Optional	ZeroOrOne	chr	Get

The ServerID leaf node is used to identify the DiagMon Server where the alert and collected DiagMon Data SHALL be sent if the node is present. The value of this node MUST be compliant with URI format as in [RFC3986].

<x>/Description

Status	Tree Occurrence	Format	Min. Access Types
Optional	ZeroOrOne	chr	Get

This optional leaf node provides a description of the DiagMon Function.

<x>/DiagMonConfig

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

This interior node is a placeholder for the configuration information associated with behaviour of the DiagMon Function, including the invocation and reporting.

<x>/DiagMonConfig/ConfigParms

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	node	Get

This interior node is a placeholder for any extensions to the configuration information associated with the of the DiagMon Function.

<x>/DiagMonConfig/DiagDataURL

Status	Tree Occurrence	Format	Min. Access Types
Optional	ZeroOrOne	chr	Get

This leaf node contains the URL [RFC3986] of an alternate data server that the client MAY use to deliver the DiagMon Data.This node MUST exist when the node '`<x>/Operations/Upload`' exists.

<x>/DiagMonConfig/DefDuration

Status	Tree Occurrence	Format	Min. Access Types
Optional	ZeroOrOne	int	Get

This node specifies the time-limit threshold for the running of the DiagMon Functions. When the time-limit threshold is exceeded the DiagMon Client MUST stop the DiagMon Function. This time is expressed in seconds, presented as a 32bit unsigned non-negative integer.

<x>/DiagMonConfig/DefMemory

Status	Tree Occurrence	Format	Min. Access Types
Optional	ZeroOrOne	int	Get

This optional node specifies the DiagMon memory-size-limit threshold for running the DiagMon Function. When the memory-size-limit is exceeded, the DiagMon Client MUST stop the DiagMon Function. The amount of memory is expressed in bytes, presented as a 32bit unsigned non-negative integer.

<x>/DiagMonConfig/ReportCondition

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

This interior node acts as a placeholder to contain different reporting conditions which indicate when DiagMon data are reported back to the DiagMon System.

<x>/DiagMonConfig/ReportCondition/Frequency

Status	Tree Occurrence	Format	Min. Access Types
Optional	ZeroOrOne	int	Get

This leaf node contains an integer value indicating the frequency at which alert about DiagMon Data is reported back to the DiagMon System. The DiagMon Server MAY use Get command on the '`DiagMonData`' node to retrieve diagnostics data.If the node does not exist, then the DiagMon Client MUST send generic alert as soon as practical to the DIagMon Server when all of diagnostic data has been completely collected, and DiagMon Client SHOULD NOT send any more generic alters to the DiagMon Server.

Note: Each diagnostic function will define what the set of diagnostic data is to be collected.

Values	Description
0	The DiagMon Client SHOULD NOT send generic alert to the DiagMon Server when the set of diagnostics data has been completely collected.
positive integer value n	The DiagMon Client MUST send generic alert as soon as practical to the DiagMon Server after the n^{th} set of diagnostics data has been completely collected.

<x>/DiagMonConfig/ReportCondition/ReportChannel

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	int	Get, Replace

The value of this node indicates the channel(s) which SHOULD be used by the DiagMon Client to report DiagMon Data to the DiagMon Server. If multiple channels are to be indicated, the sum of their numeric value MUST be indicated: for instance, if the DiagMon Server would like to indicate to send DiagMon data using Generic Alert Package and USSD channels, the node value is 10 (2+8).

The following table contains the numeric value associated with each report channel.

Values	Channel
0	Report Channel is disabled
1	Generic Alert Package without DiagMon data
2	Generic Alert Package containing DiagMon data
4	SMS
8	USSD

Note: in case the value of the node is 0 (disabled) or the node does not exist, no report from the DiagMon Client is required but the DiagMon data can be still retrieved by the DiagMon Server using suitable DM Commands during a normal DM Session.

<x>/DiagMonConfig/ReportCondition/SMSRecipient

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	chr	Get, Replace

If the value of `<x>/DiagMonConfig/ReportCondition/ReportChannel` node indicates DM Client to use SMS channel. This node MUST contain the SMS recipient (i.e. phone number).

<x>/DiagMonConfig/ReportCondition/USSDRecipient

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	chr	Get, Replace

If the value of `<x>/DiagMonConfig/ReportCondition/ReportChannel` node indicates DM Client to use USSD channel, this node MUST contain the USSD recipient.

<x>/DiagMonConfig/ReportCondition/Ext

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

This interior node allows for the extensions on report configurations.

<x>/DiagMonData

Status	Tree Occurrence	Format	Min. Access Types
Required	One	node	Get

This node is a placeholder for the data associated with the DiagMon Function. Sub-nodes of this node identify the collected DiagMon Data and are DiagMon Function specific. Therefore the tree structure under this node is outside the scope of this specification. Depending upon the nature of the DiagMon Function, sub-nodes MAY either be always present, or created dynamically. For DiagMon Functions that generate data at run-time, Devices MUST support adding of leaf nodes under this node.

<x>/Operations

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	node	Get

This node is a placeholder for operations that can be executed on DiagMon Function.

<x>/Operations/Start

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	null	Exec

This leaf node is the target of an ‘Exec’ command to start the DiagMon Function resident on the device.

<x>/Operations/Stop

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	null	Exec

This leaf node is the target of an 'Exec' command to stop the DiagMon Function resident on the device.

<x>/Operations/Upload

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	null	Get, Exec

This leaf node is the target of an 'Exec' command to upload the data associated with the DiagMon Function. If this node exists, '`<x>/DiagMonConfig/DiagDataURL`' MUST exist too. The Server SHOULD set valid URL on '`<x>/DiagMonConfig/DiagDataURL`' before performing Exec on '`<x>/Operations/Upload`' node. If the Upload operation failed, the result code '1400 (Operation Failed)' MUST be returned.

<x>/Status

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	chr	Get

This leaf node specifies the operational state of the DiagMon Function. The value of this node MUST be one of the following, although all values might not be applicable to a specific DiagMon Function:

State	Meaning
Stopped	The DiagMon Function is stopped.
Running	The DiagMon Function is running, following explicit invocation.
Continuous	The DiagMon Function is a continuously available function that is always running. Running and Stopped states are not applicable.

<x>/Ext

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

This interior node is a placeholder for platform or vendor specific extensions.

6. Invocation and Reporting Behavior

DiagMon Functions fall into two broad categories: those that are continuously available and those that have to be explicitly invoked. The invocation and reporting behavior of a DiagMon MO depends on which of the two broad categories it falls into.

6.1 Invoking and Stopping Diagnostic Functions

6.1.1 Non-continuously Available Functions

A DiagMon Function that is not continuously available SHALL be invoked by sending an Exec command to the <x/>Operations/Start node.

If the Operations/Start node is present but the Operations/Stop node is not present, the DiagMon Function, once started by the DiagMon Server, will run indefinitely, unless either (or both) nodes DiagMonConfig/[DefMemory,DefDuration] are present and have non-zero values. Note that DiagMonConfig/[DefMemory,DefDuration] SHOULD be set to values that allow for meaningful results, while not having a negative impact on DiagMon Client behavior.

6.1.2 Continuously Available Functions

For functions that are continuously available an explicit invocation using the `<x>/Operations` node is not required. The MO definitions for such functions MUST NOT support the `<x>/Operations/Start` and `<x>/Operations/Stop` nodes.

6.2 Result Reporting

6.2.1 Explicitly Invoked Functions

Explicitly invoked DiagMon Functions can report their results in two modes: synchronously or asynchronously, as described in the following sub-sections.

The DiagMon Server SHALL support retrieval of result in both modes. The DiagMon Client SHALL support either synchronous or asynchronous reporting. The Client MAY support both modes.

It is up to the DiagMon Client to decide in which mode to operate. The DiagMon Client MAY change its mode between subsequent calls and DiagMon Server MUST NOT assume that DiagMon Client operates always in same mode.

6.2.1.1 Asynchronous Execution (Use of the Generic Alert)

The DiagMon Client MAY perform Asynchronous Exection, and report back to the DM Server later. Explicitly invoked DiagMon Functions that execute asynchronously MUST report their results when the execution was completed by issuing an alert with type "`urn:oma:at:diagmon:1.0:OperationComplete`" in the same or new DM session, according to the Generic Alert mechanism defined in [DMPRO]. The Generic Alert message includes the following data:

- An alert type "`urn:oma:at:diagmon:1.0:OperationComplete`" – Used to identify the result type of the operation; MUST be included in Item/Meta/Type element.
- The URI of the operation node on which the Exec command was invoked– Used to identify the source. This information MAY be omitted if the node is `./Operations/Start`, otherwise it MUST be included in the `Source/LocURI` element within the Generic Alert.
- The URI of the DiagMonData node or its child node provides the location at which the output data associated with the DiagMon Function is present. This information MUST be included in the `Target/LocURI` element within the Generic Alert if the DiagMon Function generates DiagMonData. If no DiagMonData is generated then the Target element MUST NOT be present within the Generic Alert.

- An integer result code – Used to report result of the invocation of the DiagMon Function, as defined in section 6.3. This information MUST be included in an `Item/Data` element within the Generic Alert.
- A correlator – Used to identify the DM command which invoked DiagMon Function.

The DiagMon Server MAY retrieve the collected DiagMon Data by sending a Get command to DiagMonData or its child nodes.

If node `DiagMonConfig/ReportCondition/ReportChannel` is present and its numeric value mandates the usage of the "Generic Alert Package containing DiagMon data" (2) , the Generic Alert message MAY include additional Item elements to convey the `DiagMonData`, with the following data in each Item element:

- If multiple DiagMon Data are reported to the DiagMon Server, each of them is included in an Item element
- The URI of the DiagMon Data node or one of its child nodes in `Target/LocURI`, to identify the reported data.
- If the URI points to a leaf node of the DiagMonData subtree, the Data element will include the value of the node and the Meta/Type element MUST contain alert type "`urn:oma:at:diagmon:1.0:CollectedSimpleDataReport`".

The following is an example message:

```
      <Alert>
         <CmdID>2</CmdID>
         <Data>1226</Data>    <!-- Alert Code for Generic Alert -->
         <Item>
          <Source>
            <LocURI>./DiagMon/abc/Operations/Start</LocURI>
          </Source>
          <Target>
               <LocURI>./DiagMon/abc/DiagMonData</LocURI>
          </Target>
               <Meta>
           <Type
xmlns="syncml:metinf">urn:oma:at:diagmon:1.0:OperationComplete</Type>
                 <Format xmlns="syncml:metinf">int</Format>
                 <Mark xmlns="syncml:metinf">warning</Mark>
               </Meta>
               <Data>1200</Data>     <!-- DiagMon Result Code -->
         </Item>
      </Alert>
```

6.2.1.2 Synchronous Reporting

For explicitly invoked DiagMon Functions that execute synchronously, the DiagMon Server MAY retrieve the collected DiagMon Data by sending a Get command to DiagMonData or its child nodes.

If the DiagMon Function is executed synchronously by Exec command, the DiagMon Client MUST send result for Exec command as described below:

- The `<Data>` element MUST contain a valid DiagMon Result Code defined in section 6.3.
- The URI of the operation node on which the Exec command was invoked – Used to identify the source; MUST be included in the `<Source>/<LocURI>` of `<Status>/<Item>` element.
- The URI of the DiagMonData node or its child node provides the location at which the output data associated with the DiagMon Function is present. This information MUST be included in the `<Target>/<LocURI>` of `<Status>/<Item>` element if the DiagMon Function generates DiagMonData. If no DiagMonData is generated then the Target element MUST NOT be present.

The following is an example message:

```
<Status>
  <MsgRef>1</MsgRef>
  <CmdRef>2</CmdRef>
  <Cmd>Exec</Cmd>
  <Data>1200</Data>        <!--DiagMon Result status Code -->
  <Item>
    <Source>
      <LocURI>./DiagMon/abc/Operations/Start</LocURI>
    </Source>
    <Target>
         <LocURI>./DiagMon/abc/DiagMonData</LocURI>
    </Target>
  </Item>
</Status>
```

These functions SHALL NOT report any results via Generic Alerts.

6.2.2 Continuously Available Functions

For DiagMon Functions that are continuously available, the DiagMon Server MAY retrieve the DiagMon Data by sending a Get command to DiagMonData or its child nodes. These functions SHALL NOT report any results via Generic Alerts.

6.2.3 Posting data to DiagDataURL

The Client MAY support posting the DiagMon Data to the URL specified by '`DiagMonConfig/DiagDataURL`'. If the Client supports this feature, the node '`Operations/Upload`' MUST be present.

If the DiagMon Server wishes to request the DiagMon Client to post the data, the DiagMon Server MUST issue 'Exec' command on '`Operations/Upload`' explicitly. When the '`Operations/Upload`' is executed, the DiagMon Client MUST send the DiagMon Data over HTTP in form of '`multipart/form-data`' as defined in [RFC2388].

This multipart '`multipart/form-data`' MUST contain the content of the leaf nodes under the '`<x>/DiagMonData`' node, and MUST have both '`Content-Type`' and '`Content-Disposition`' headers. The `Content-Type` header MUST contain appropriate MIME type of the node contents, and the `Content-Disposition` header MUST contain the parameter '`name`' with the relative path name from the '`<x>/DiagMonData`'. The '`name`' parameter values MUST be URL escaped as defined in [RFC3986]. For example, if the sending leaf node was '`DiagMonData/xxx/yyy`', the name parameter MUST be formatted as '`xxx%2Fyyy`'.

The `multipart/form-data` also MUST contain the part named '`devid`' which contains the URI string to identify the sender device.

The `multipart/form-data` MAY contain other data, which is not part of DiagMonData sub-tree, but its '`name`' parameter string in Content-Disposition header MUST not conflict with any other name of the '`DiagMonData`' sub-tree nodes.

6.2.4 Posting Data Over Report Channel

If node `DiagMonConfig/ReportCondition/ReportChannel` is present and its numeric value mandates the usage of the "SMS" or "USSD" channel (4 or 8), the DiagMon data MAY be conveyed over these transport using the following format:

`[<Correlator>] ";" <DiagMonData>`

Note: in case DiagMon data contains binary data, it MUST be encoded as b64.

6.3 Result Codes

The Result Code of the DiagMon operation MUST be sent as an integer value in Item/Data element of the Generic Alert [DMPRO] message or in response to an Exec command. The Result Code MUST be one of the values defined below:

Code	Meaning	Usage
1200	Successful	Successful - The Request has Succeeded
1201	Successful – Data Available	Successful – DiagMon Data is available
1240-1249	Successful – Vendor Specified	Successful Operation with Vendor Specified Result Code. Note: this extension applies to DiagMon framework.
1250-1259	Successful - DiagMon Framework Future Releases Specified	Successful Operation. Note: this extension is for future DiagMon framework releases.
1260-1299	Successful - DiagMon Function Specified	Successful Operation. Note: this extension is for DiagMon Function Specified Result Code.
1400	Operation Failed	Operation failed to start or stop DiagMon Function
1401	Not authorized	The DiagMon Server is not authorized to conduct this operation.
1402	Not Implemented	The DiagMon Client does not support the requested operation.
1403	Out of memory	The operation failed due to insufficient memory in the device to execute the DiagMon Function
1404	Storage is Full	The operation failed due to insufficient storage in the recipient device to store events or data for DiagMon Funtion.
1440-1449	Operation Failed – Vendor Specified	Operation Failure with Vendor Specified Result Code. Note: this extension applies to DiagMon framework.
1450-1459	Operation Failed – DiagMon Framework Future Releases Specified	Operation Failure. Note: this extension is for future DiagMon framework releases.
1460-1499	Operation Failed – DiagMon Function Specified	Operation Failure. Note: this extension is for DiagMon Function Specified Result Code.

Table 1: DiagMon Result Codes

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-DiagMonMOFrame-V1_2-20131008-A	08 Oct 2013	Status changed to Approved by TP TP Ref # OMA-TP-2013-0311-INP_DiagMon_V1_2_ERP_for_Final_Approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

B.1 SCR for DiagMon Tree Structure

Item	Function	Reference	Requirement
DIAG-T-001-M	Use of appropriate Management Object identifier for the DiagMon node	Section 5.1	
DIAG-T-002-M	Support for Required nodes under root node	Section 5.2	
DIAG-T-003-O	Support for Optional nodes	Section 5.3	DIAG-T-004-O
DIAG-T-004-O	Support for Required nodes under an Optional node if the Optional node is supported	Section 5.3	

B.2 SCR for DiagMon Client

Item	Function	Reference	Requirement
DIAG-C-001-M	DM Client allows DiagMon Server to invoke diagnostic functions via DiagMon MO	Section 5	
DIAG-C-002-M	DiagMon Functions result reporting	Section 6.2	DIAG-C-003-O OR DIAG-C-004-O
DIAG-C-003-O	DiagMon Functions are executed asynchronously and results are reported to DiagMon Server using Generic Alert	Section 6.2.1	
DIAG-C-004-O	DiagMon Functions are executed synchronously and results are reported using a Status message	Section 6.2.1.2	
DIAG-C-005-M	Alert Type provides information on MO	Section 6.2.4	

B.3 SCR for DiagMon Server

Item	Function	Reference	Requirement
DIAG-S-001-M	Support for the DiagMon Management Object	Section 5.3	
DIAG-S-002-M	DiagMon Server can invoke DiagMon Functions via DiagMon MO	Section 6.2.1	
DIAG-S-003-M	DiagMon Functions are executed asynchronously by DiagMon Server and	Section 6.2.16.2.1	

Item	Function	Reference	Requirement
	report the result using Generic Alert		
DIAG-S-004-M	DiagMon Functions are executed synchronously by DiagMon Server and report the result using Status message	Section 6.2.1	
DIAG-S-005-M	DiagMon Server sends a Get command to retrieve collected DiagMon Data	Section 6.2.1.2	
DIAG-S-006-M	Alert Type is received that provides information on MO	Section 6.2.4	

Appendix C. Best Practices for defining DiagMon Functions (Informative)

DiagMon Functions are essentially MOs, and therefore all the best practices for defining MOs, as documented in [ACMO], apply to them as well. There are also some additional best practices that apply to DiagMon Functions.

The following is a list of best practices to be followed for documenting DiagMon Functions. These best practices are also applicable to external Standards Developing Organizations (SDOs) that may define their own DiagMon Functions.

1. Identify the broad category and the subcategory, under the broad category, into which the DiagMon Function falls. If the DiagMon Function does not fall into one of the existing categories, create a new one.
2. Create a new subsection for the DiagMon Function under the section pertaining to its subcategory. Let us assume this section is Section x.
3. Under Section x create the following subsections:

1 Introduction

This subsection provides a brief description of the DiagMon Function and highlights its salient features.

2 MOID

This subsection provides the MOID of the DiagMon Function.

3 Non-applicable nodes from DiagMon MO definition

This subsection lists the nodes from the DiagMon MO definition that do not apply to the DiagMon function, if any.

For DiagMon Functions that are continuously available, the following nodes generally do not apply:

- ServerID
- DiagMonConfig
- Operations

For explicitly invoked DiagMon Functions that do not report their results asynchronously, the following node generally does not apply:

- ServerID

4 Function Description

This subsection includes a figure that gives the complete structure of the DiagMon Function, preferably in the enhanced graphical notation, as described in [ACMO]. A description of the various nodes in the MO definition is also included, as per [ACMO] guidelines. However, nodes that are not different from the DiagMon framework description are not described in this section.

5 Additional Information

This optional subsection provides any additional information that is specific to the DiagMon Function.

Note: If the DiagMon Function is defined by an external SDO, only subsection x.1 should be present. In that case, subsection x.1 should only provide a reference to the pertinent document that describes the DiagMon Function.