Diagnostics and Monitoring Trap Framework Management Object

Approved Version 1.2 – 08 Oct 2013

Open Mobile Alliance

OMA-TS-DiagMonTrapMOFrame-V1_2-20131008-A

Contents

Figures

1. Scope

This document describes the DiagMon Trap Framework and the management object(s) as a part of the framework to be employed in a DiagMon activity that leverages the OMA DM v1.2 protocol. It provides standard DM Management Objects and associated client-side and server-side behaviour necessary to utilize the event monitoring capabilities on the mobile devices.

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_3, URL:http://www.openmobilealliance.org
[ISO8601]	ISO 8601:2004, Data elements and interchange formats -- Information interchange -- Representation of dates and times. URL:http://www.iso.ch/
[OMNA]	“Open Mobile Naming Authority”. Open Mobile Alliance™. URL:http://www.openmobilealliance.org/tech/omna/index.html
[RFC2119]	“Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt
[SCRRULES]	“SCR Rules and Procedures”, Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/
[TrapEvents]	“Diagnostics and Monitoring Trap Events”, Version 1.2. Open Mobile Alliance ™. URL:http://www.openmobilealliance.org/

2.2 Informative References

[OMADICT]	“Dictionary for OMA Specifications”, Version 2.9, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, 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

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

3.3 Abbreviations

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

4. Introduction

The OMA DiagMon Trap Framework v1.2 specified in this document builds on top of the existing OMA DM based management system to enable in an interoperable way specifying and using any kind of events worthwhile for managing and monitoring the networked services or applications that are deployed on the device, or faults on the general software and hardware, etc.

The OMA DiagMon Trap Framework provides the common structure of the DiagMon Trap MOs on which further definitions for the specific events in [TrapEvents], or by vendors and standard bodies will be based, and the DiagMon Trap mechanisms for sending and receiving Notifications about the events.

The OMA DiagMon Trap Framework v1.2 is compatible with [DM].

5. The DiagMon Trap Framework

5.1 Trap Identifier

Any event that is specified as a Trap MUST be assigned an identifier, the Trap identifier. The Trap identifier MUST be an URN and it MAY be registered with OMNA. The assignment of the identifier can be done by the entity who specifies the Trap, following the guidance or the administration from Open Mobile Naming Authority [OMNA] in order for the identifier to be unique and persistent.

Trap identifiers defined by OMA SHALL be an URN consistent with the Management Object identifier as follows: "`urn:oma:mo:oma-diagmontrap:1.1`", for example, `urn:oma:mo:oma-diagmontrap:trapid-abc:1.0`.

5.2 Registrations

If the DiagMon Client supports a Trap, it means the DiagMon Client is capable of monitoring the event and send Notifications whenever it detects the event. If the Management Authority wants to use the capability, it has to register for it. The Trap will have two events being monitored – when the Trap becomes active and when the Trap becomes inactive.

There are two types of Registration depending on the direction in which the Notification is bound, outward and inward. The first type, outward Registration is used when the Management Authority wants to receive the Notification as soon as the Traps are generated; whereas, the second type, inward Registration allows that Traps can be transferred to some other functional components on the same DiagMon Client so as to trigger a certain operation in the destination. Note that the default triggering is when the Trap becomes active. It is possible to set the Trigger node to allow the Notification to take place when the Trap becomes Inactive as well. There are three possibilities for triggering: Active (default), Inactive and Both.

The OMA DiagMon Trap Framework supports multiple Trap recipients. Therefore, it MUST be possible that more than one DiagMon Server can register on one Trap at the same time with the maximum allowed number being limited by the implementations based on the Occurrence framework property of the Trap node [DM]. In this case, the order in which the DiagMon Client sends the Notifications for each DiagMon Server should be decided at the discretion of the implementations.

Inward Registration involves relatively more complex procedure compared to outward one. This is mainly due to the fact that security mechanisms in [DM] are not applicable in addressing the security issues pointed out in section 7.1.

5.3 Notifications

When the Trap occurs, the DiagMon Client MUST send Notifications as practical to all the registered recipients, subject to the Trigger settings (Active, Inactive or Both). Corresponding to the Registration type, there are two types of Notification - outward and inward.

5.3.1 Outward Notifications

The outward Notifications MUST be transmitted to the servers using the Generic Alert mechanism

[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_3, URL:http://www.openmobilealliance.org

. On top of that, additional restrictions on the contents and the usage of the elements in the Generic Alert message are specified for the Notification as follows.

- **Usage of Meta, Type**: When the trap transitions to Active, this element MUST be specified and the content of this element MUST specify the Trap alert type string, "`urn:oma:mo:diagmon:1.0:TrapNotification`", which implies that this Generic Alert message contains the Active Trap Notification. When the trap transitions to Inactive, this element MUST be specified and the content of this element MUST specify the Trap alert type string, "`urn:oma:mo:diagmon:1.0:TrapInactiveNotification`", which implies that this Generic Alert message contains the Inactive Trap Notification.

- **Usage of Meta, Format**: This element MUST be specified and the content of this element MUST specify the format of the `Item/Data` element as "chr".
- **Usage of Source, LocURI**: This element MUST be specified and the content of this element MUST specify the URI of the root node for the Trap MO.
- **Usage of Data (inside Item):** This element MUST be specified and the content of this element MUST specify the value of the MOID of the Trap MO.
- **Usage of Other Fields**: Correlator MUST NOT be used, and the use of the Mark elements is vendor specific.

5.3.2 Inward Notifications

Although the details of the delivery mechanism are implementation specific, the Notification MUST provide the same information to the recipients as for the outward Notification.

The inward Notifications MUST be processed as same as when the DM Server (identified by `<x>/ToRef/TargetURI/<x>/RegisteredServerID`) performed 'Exec' command against the device internal URIs which are stored as `<x>/ToRef/TargetURI/<x>/URI`.

6. DiagMon Trap MO

A trap is a mechanism employed by a Management Authority to enable the DiagMon Client to capture and report events and other relevant information generated from various components of the device, such as a protocol stack, device drivers, or applications.

The DiagMon Trap MO framework defined in this specification will be used for specific Trap MOs to be defined and uniquely identified by a Trap ID.

6.1 The Trap MO

In particular, a TrapMO is used to report the occurrence of an event of interest. A Trap is associated with a trap identifier and a server identifier. It also defines a collection method and a reference node to refer to other management objects or URI, which may be associated with additional activities, e.g. DiagMonMO.

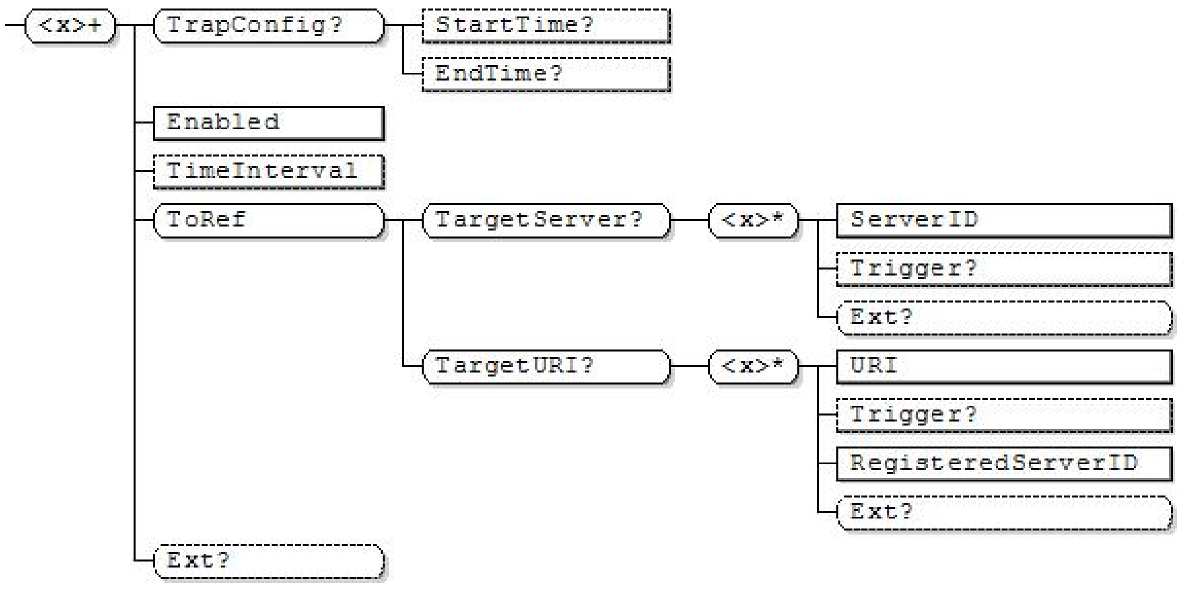

Figure 1: DiagMon Trap Management Object

6.2 Management Object Properties

This section describes the properties for Trap Management Object

<x>

Status	Tree Occurrence	Format	Min. Access Types
Required	OneOrMore	node	Get, Add

This interior node groups together the parameters of the DiagMon Trap 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 Trap Management Object ID "`urn:oma:mo:oma-diagmontrap:1.1`", but this identifier SHALL be replaced by the MOID of actual Trap MO which is defined in [TrapEvents].

<x>/TrapConfig

Status	Tree Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get,

This interior node is a placeholder for the configuration information associated with the DiagMon Trap MO.

<x>/TrapConfig/StartTime

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

This node contains the start time and date, encoded as per the UTC based [ISO8601] basic format. This is the start time of monitoring the Trap MO events..

<x>/TrapConfig/EndTime

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

This node contains the end time and date, encoded as per the UTC based [ISO8601] basic format. This is the end time of monitoring Trap MO events.

<x>/Enabled

Status	Tree Occurrence	Format	Min. Access Types
Required	One	bool	Get, Replace

The Enabled node is used to indicate if the Trap is enabled ('`true`') or disabled ('`false`'). If the Trap is disabled, no action related to the Trap is performed.

<x>/TimeInterval

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

This leaf node is used to define a time interval by second. A Trap is only allowed to send Outward or Inward Notification after the time interval that previous Notification was sent. The Notification is discarded if the Notification is not allowed to be sent. If the node does not exist or does not contain the positive value, no time interval restricts for the DiagMon Client to send the Notification

<x>/ToRef

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

This interior node is a placeholder for all recipient reference.

<x>/ToRef/TargetServer

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

This interior node is a placeholder for specifying targeted server as a trap recipient.

<x>/ToRef/TargetServer/<x>

Status	Tree Occurrence	Format	Min. Access Types
Required	ZeroOrMore	node	Add, Delete, Get

This interior node is a placeholder for each registration for outward notification.

<x>/ToRef/TargetServer/<x>/ServerID

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get, No Replace,

This leaf node specifies the server identifier of the registered DM Server.

<x>/ToRef/TargetServer/<x>/Trigger

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

This leaf node indicates when to send notification to this particular server. If this node is missing, it has the same effect as Active.

Values	Meaning
0	Trap transitions to Active
1	Trap transitions to Inactive
2	Trap transitions between Active and Inactive

<x>/ToRef/TargetServer/<x>/Ext

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

This interior node is a placeholder for extension by vendors or standards organizations.

<x>/ToRef/TargetURI

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

This leaf node stores the device internal URI which will be performed 'Exec' command when the trap event occurred..

<x>/ToRef/TargetURI/<x>

Status	Tree Occurrence	Format	Min. Access Types
Required	ZeroOrMore	node	Add, Get, Replace

This interior node is a placeholder for each registration for inward notification.

<x>/ToRef/TargetURI/<x>/URI

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

This leaf node specifies the device internal target URI reference which will be invoked by trap.

<x>/ToRef/TargetURI/<x>/Trigger

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

This leaf node indicates when to invoke the URI. If this node is missing, it has the same effect as Active.

Values	Meaning
0	Trap transitions to Active
1	Trap transitions to Inactive
2	Trap transitions between Active and Inactive

<x>/ToRef/TargetURI/<x>/RegisteredServerID

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	No Get, No Replace

This leaf node specifies the server identifier of the DM Server that registered the inward trap (e.g. ServerID which added the `TargetURI/<x>` subtree). The DiagMon Client MUST set the value of this node when the trap is registered. This leaf node MUST only be used by the DiagMon Client to check whether the DM Server has the proper authorization for notifying the inward trap (i.e. Exec permission on the executable node pointed by `ToRef/TargetURI/<x>/URI`).

<x>/ToRef/TargetURI/<x>/Ext

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

This interior node is a placeholder for extension by vendors or standards organizations.

<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.

7. Secure Trap Operations

7.1 Secure Inward Trap

For secure trap operations, the DiagMon Client SHOULD verify that the DM Server has proper authorizations for the inward Registration and the Notification as described below.

- **Registration**: The DM Server can register to the Inward Trap by adding sub-tree under the `<x>/ToRef/TargetURI` node. This registration MUST NOT be granted if the DM Server does not have Exec permission on the executable node pointed by the `<x>/ToRef/TargetURI/<x>/URI` node. The Device MUST verify this Exec permission in addition to the basic ACL rules (e.g., Add right for adding a sub-tree). If the DM Server does not have Exec permission, the registration MUST be rejected with TrapMO Result Code 1400 (Registration failed due to insufficient authorizations). After the successful registration, the <x>/`ToRef/TargetURI/<x>/RegisteredServerID` node MUST be set by the Device with the server identifier that registered this inward trap.

- **Notification**: The trap MUST be notified to the executable node pointed by the `<x>/ToRef/TargetURI/<x>/URI` node only if the DM Server, identified by the sibling RegisteredServerID node, has the Exec permission on the executable node pointed by the <x>/`ToRef/TargetURI/<x>/URI` node. For verifying this Exec permission, it MUST be considered that the ACL can be dynamically changed. Hence, the DiagMon Client MUST check Exec permission right before notifying the trap event, or the DiagMon Client MUST trigger a check process upon the ACL change.

If the DM Server does not have the Exec permission on the executable node, the corresponding inward trap MUST be deregistered by the DM Client as soon as practical. After the inward trap is deregistered, the DiagMon Client MUST remove the corresponding sub-tree under the ToRef/TargetURI node. A Generic Alert MAY be sent to the DM Server, identified by the RegisteredServerID node, for notifying the deregistration as described in section 8.1.

7.2 Secure Outward Trap

The outward trap mechanism is intended for sending trap events to DM Servers. The trap events might cover a wide array of device information including sensitive and privacy data such as configuration changes, SMS/MMS usage, or location information. To ensure the confidentiality, trap events SHOULD be sent to an outside DM Server only if that DM Server is trusted. In this section, how to secure outward trap is explained.

The DM Server will add sub-tree under the <x>/`ToRef/TargetServer` node if it is interested to receive traps from the Device. For the successful registration, the Device MUST verify that the <x>/`ToRef/TargetServer/<x>/ServerID` node is set with the DM Server’s own server identifier (i.e., the DM Server MUST NOT register other DM Servers). If the registration fails, the Device MUST send the status code 403 Forbidden. Once it is successfully registered, the value of the <x>/`ToRef/TargetServer/<x>/ServerID` node MUST NOT be changed.

8. Behaviour associated with the Management Object

8.1 Notifying the Deregistration Using the Generic Alert

As explained in section 5.2, the DiagMon Client MAY send a Generic Alert to the relevant DM Server for notifying the deregistration of an inward trap. The Generic Alert MUST include the following data.

- `<Meta>/<Type>`: The value MUST be the alert type identifier 'urn:oma:at:diagmon:1.0:TrapDeregistered'.
- `<Meta>/<Format>`: The value MUST be 'int'.
- `<Source>/<LocURI>`: Contains the target URI reference which was invoked by trap. The value MUST be the address of ToRef/TargetURI/<x>/URI.
- `<Item>/<Data>`: Used to report status of the operation. The value MUST be the appropriate TrapMO Result Code defined in section 8.2.

The following is an example of the Generic Alert, TrapDeregistered.

```
<Alert>
   <CmdID>2</CmdID>
   <Data>1226</Data>             <!-- Generic Alert -->
   <Item>

<Source><LocURI>./trap/BatteryTrap/ToRef/TargetURI/uri1/URI</LocURI></Source>
      <Meta>
         <Type xmlns='syncml:metinf'>
           urn:oma:at:diagmontrap:1.0:TrapDeregistered
         </Type>
         <Format xmlns='syncml:metinf'>int</Format>
      </Meta>
      <Data>1401</Data>
   </Item>
</Alert>
```

8.2 TrapMO Result Code

The Result Code MUST be sent as an integer value in `<Item>/<Data>` element of the Generic Alert

[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_3, URL:http://www.openmobilealliance.org

. The Result Code MUST be one of the values defined below:

Result Code	Result Message	Informative Description of Status Code Usage
1400	Registration failed due to insufficient authorizations	A trap Registration failed because it doesn't have enough rights.
1401	Deregistered due to insufficient authorizations	The Notification fails, and the relevant inward trap is deregistered.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-DiagMonTrapMOFrame-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 Trap MO Tree Structure

Item	Function	Reference	Requirement
TRAP-T-001-M	Use of appropriate Management Object identifier for the TRAP MO node	Section 6.2	
TRAP-T-002-M	Support for REQUIRED nodes under root node	Section 6.2	

B.2 SCR for DiagMon Client

Item	Function	Reference	Requirement
Trap-C-001-M	DM Client allows more than one DiagMon System to register on one Trap	Section 5.2	
Trap-C-002-M	DiagMon Client allows DiagMon System to invoke monitoring of trap events and send notifications via Trap MOs	Section 5.3	
Trap-C-003-M	Alert Type provides information on Trap MO	Section 5.3.1	
Trap-C-004-M	Registration Failure Results are provided to the Trap notification recipient(s) defined by /<x>/ToRef/TargetServer and /<x>/ToRef/TargetURI	Sections 7.1& 7.2	
Trap-C-005-O	Deregistration is provided asynchronously to the Trap notification recipient(s)	Section 8	

B.3 SCR for DiagMon System

Item	Function	Reference	Requirement
TRAP-S-001-M	Support for the Trap Management Object	Section 6.2	
TRAP-S-002-M	DiagMon System can register a Trap via Trap MOs	Section 5.2	
TRAP-S-003-M	DiagMon System can receive notifications via Trap MOs	Section 5.3	

TRAP-S-004-M	Registration Failure Results are received asynchronously by the DiagMon System	Sections 7.1&7.2	
TRAP-S-005-M	Alert Type is received that provides information on the Trap MO	Section 5.3.1	
TRAP-S-006-O	Deregistration is received via Trap MOs	Section 8	