MTOSI XML Implementation User Guide

Abstract

This document provides an overview of the MTOSI data and service interface models and their associated XML definitions, which are described and captured in the various MTOSI Delivery Document Packages (DDPs) .

This document provides behavior descriptions and usage guidelines that:

1. require a rather long description
2. require supporting diagrams

appear repeatedly throughout the XML (and are easier to document in just one place, i.e., SD2-2).

Table of Contents

MTOSI XML Implementation User Guide 1
Abstract 1
Table of Contents 1
1 MTOSI XML Definitions 3
2 MTOSI XML Message Exchange Patterns (MEPs) 3
3 MTOSI XML Message Structure 4
4 Anatomy of an MTOSI XML Message 5
4.1 SOAP Header in MTOSI 6
4.1.1 Mandatory Header fields applicable to all MEPs 8
4.1.2 Correlation of asynchronous messages in the MTOSI message style 8
4.1.3 Header fields setting for Asynchronous Request/Reply (ARR) 10
4.1.4 Header fields setting for Synchronous Request/Reply (SRR) 10
4.1.5 Header fields setting for Asynchronous MultipleResponse (ABR) 10
4.1.6 Header fields setting for Synchronous Iterator (SIT) 11
4.1.6.1 Header fields setting for Asynchronous (File) Bulk (AFB) 11
4.1.6.2 Header fields setting for Synchronous (File) Bulk (SFB) 12
4.1.6.3 Subscription for Notifications 12
4.2 SOAP Body in MTOSI 12
4.2.1 MTOSI Name syntax 13
4.2.2 MTOSI names are defined according to the network object containment hierarchy. See [SD2-7] for details about MTOSI Object Naming. General (default) behaviors on missing (empty) message parameters 13
4.2.3 Missing or empty XML elements in a request message (Query) 13

4.2.3.1 Missing or empty XML elements in a response 14
5 MTOSI Extensibility 15
6 MTOSI WS Discovery/Registration 15
7 References 16
8 Administrative Appendix .... 17~~18~~
8.1 Document History .... 17~~18~~
8.2 Acknowledgments .... 17~~18~~
8.3 How to comment on this document .... 17~~18~~

1 MTOSI XML Definitions

A significant aspect of the MTOSI information model is the definition of the network objects along with their containment relationships. MTOSI has a formal definition of each network object in terms of an XML schema (XSD). These XSD definitions are the foundation of the formal arguments and the result sets of the MTOSI messages. The top level messages (both request, reply, and notification) contain a set of Network Objects as formal argument and the result set is arranged in an XML document style to accomplish a business activity. See the following references:

- [SD0-5] for a detailed description of the MTOSI service interface model structure and XML definitions
- [SD2-12]Inventory Layout Description, for specific details on how the Network Objects are organized into a comprehensive XML document for the getInventory() operation.

Since the Network Objects XSDs (defined in the Network Resource Definitions) are used in a variety of contexts, (e.g. query filter template, response set, event notification) the definition of the objects are necessarily loose (all elements are optional) and has to be intended as a skeleton to support optional information.

Furthermore, MTOSI XSDs are occasionally under constrained to limit the complexity of XSD validation and facilitate extensibility. (e.g MTOSI name sequences, TransmissionParameters, etc)

Specific business rules may apply in a specific context and have to be enforced beyond the XSD mechanisms.

The following sections list a set of rules, which in conjunction of the XSD constraints guarantees interoperability and cooperation in an MTOSI deployment.

2 MTOSI XML Message Exchange Patterns (MEPs)

MTMN and MTOSI share the same information model TMF 608. This model identifies the managed objects as well as the logical operations to manipulate the objects. In MTOSI, we call the operations at this level of abstraction “Business Activities”. In MTOSI, we identified four “communication patterns” [SD2-5] Communication Styles to accomplish all the business activities:

- Simple Response (to perform simple request-reply interactions)
- Multiple response (to perform request with multiple reply interactions due to long running processing)
- Bulk Response (to transfer the response set through a side channel)
- Notifications (to disseminate events)

Orthogonal to the communication patterns are the communication styles: MSG (i.e., message) and RPC (i.e., remote procedure call) describing the interaction mode. Due to resource constraints in this MTOSI release, we only address the MSG communication Style.

The combination of a communication pattern with a communication style leads to a Message Exchange Pattern (MEP). The MEP fully identifies the messages and associated choreography (sequencing and cardinality) of messages involved in a business activity. Table 1 summarizes the MEP supported in this release of MTOSI.

Table 1 - MEPs supported in MTOSI R2.0

	Communication Pattern			
Communication Style	Simple Response	Multiple Response	File Bulk Response	Notification
MSG (Asynch)	**(ARR)** Asynchronous Request/Reply	**(ABR)** Asynchronous Response	**(AFB)** Asynchronous (File) Bulk	**(AN)** Notification with async subscription
RPC (Synch)	**(SRR)** Synchronous Request/Reply	**(SIT)** Synchronous Iterator	**(SFB)** Synchronous (File) Bulk	**(SN)** Notification with sync subscription

We structured the XSD set to clearly separate the abstract interface supporting the business activities from the additional information related to the particular communication pattern used to accomplish the business activity.

[SD2-1] MTOSI Implementation Statement, provides a detailed description of the MTOSI functional data (static) and operational (dynamic/behavioral) models for each DDP .

For example: the getInventory operation of the ResourceInventoryRetrieval service interface can be traced as followed in the Managed Resource Inventory (MRI):

- Business Agreement (BA) with requirements and use cases
- Information Agreement (IA) with UML class diagrams
- Interface Implementation Specifications (IIS)with XML Schema and WSDL modules

Note that the references in the artifacts (IA and IIS) will lead to definitions from DM DDPs and the Framework DDP.

3 MTOSI XML Message Structure

The MTOSI abstract interface specifies the syntax and semantics of a service. At this technology neutral level a business activity is accomplished by the exchanges of MTOSI SOAP compliant messages. MTOSI adopts the SOAP envelope as external wrapper and structure for the MTOSI messages. The following figure shows a typical MTOSI message inside a SOAP envelope.

```
<soap:Envelope xmlns:soap=" http://schemas.xmlsoap.org/soap/envelope">
            <soap:Header>
                        ------ MTOSI header goes here -------
            </soap:Header>
            <soap:Body>
                        ------ MTOSI message body goes here ------
            </soap:Body>

</soap:Envelope>
```

Refer to http://schemas.xmlsoap.org/soap/envelope for a complete definition of the SOAP XSD.

While SOAP addresses the envelope of a message, MTOSI specifies the namespace and root elements of the MTOSI specific Header and Body.

- MTOSI header root element is <header>, which is defined in the Framework DDP in the http://www.tmforum.org/mtop/fmw/xsd/hdr/v1 namespace.
- MTOSI body root element is the name of the message (SOAP Document/literal wrapped style) which namespace is specifically bound to the service interface definition found in an MTOSI OM DDP (i.e. <getInventoryRequest> element defined in the ResourceInventoryRetrieval (RIR) found in the MRI OM DDP).

MTOSI messages, while compliant to the SOAP standard in terms of structure, do not necessarily require a SOAP stack to be processed. Beside the SOAP envelope, MTOSI does not require any other SOAP specific constraints or functionalities to be implemented by the service provider nor service consumer.

4 Anatomy of an MTOSI XML Message

The purpose of MTOSI is to identify and standardize Business Activities in terms of syntax (XSD) and behavior (WSDL + MEP) in other to achieve interoperability. Any Business Activity is fulfilled by the exchange of XML messages. An interface groups together similar business activity messages.

E.g., getManagedElement business activity defined in the ManagedElementRetrieval service interface of the MRI DDP has:

- getManagedElementRequest, the request message,getManagedElementResponse, the response message, and
- getManagedElementException, the exception or fault message

All service interface messages are XML instance documents with associated XSDs (XML Schema Definition) for validation.

Each MTOSI message is composed of a header and a body combined together with a SOAP [SOA] envelope. See following example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:hdr="http://www.tmforum.org/mtop/fmw/xsd/hdr/v1" xmlns:mer="http://www.tmforum.org/mtop/mri/xsd/mer/v1"
xmlns:nam="http://www.tmforum.org/mtop/fmw/xsd/nam/v1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.tmforum.org/mtop/mri/xsd/mer/v1 ../xsd/ManagedElementRetrievalMessages.xsd
http://schemas.xmlsoap.org/soap/envelope/ http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <hdr:header>
      <hdr:activityName>getManagedElement</hdr:activityName>
      <hdr:msgName>getManagedElementRequest</hdr:msgName>
      <hdr:msgType>REQUEST</hdr:msgType>
      <hdr:senderURI>/MTOSI/InventoryOS</hdr:senderURI>
      <hdr:destinationURI>/MTOSI/EmsOS</hdr:destinationURI>
      <hdr:correlationId>1111</hdr:correlationId>
      <hdr:communicationPattern>SimpleResponse</hdr:communicationPattern>
      <hdr:communicationStyle>MSG</hdr:communicationStyle>
      <hdr:timestamp>2007-07-30T09:26:00</hdr:timestamp>
    </hdr:header>
  </soap:Header>
  <soap:Body>
    <mer:getManagedElementRequest>
      <mer:meName>
        <nam:rdn>
          <nam:type>MD</nam:type>
          <nam:value>string</nam:value>
        </nam:rdn>
        <nam:rdn>
        <nam:type>ME</nam:type>
        <nam:value>string</nam:value>
       </nam:rdn>
     </mer:meName>
    </mer:getManagedElementRequest>
  </soap:Body>
</soap:Envelope>
```

Refer to the xml folder in each DDP as it captures sample XML files.

While the body carries all the information related to the business activity (e.g., the name of the associated operation and the filter, in the case of the getManagedElement operation), the header is responsible for the additional information related to the particular MEP used in the interaction (e.g., correlation ID, requestedBatchSize, etc). Refer to the SD2-1, MTOSI Implementation Statement, for the exact position in the envelope and definition of the formal argument of each operation.

The next section will summarize the Header element and their use in the message.

4.1 SOAP Header in MTOSI

The following Table 2 summarizes the field included in the MTOSI header.

Header field is the tag name in the XML, O=optional, M=required (mandatory), NA=not applicable and not used in the header.

Table 2 - MTOSI Header fields

Header field	Req	Resp	Notif	Description
activityName	**M**	**M**	**M**	**Identifies the name of the business transaction activity for the message being exchanged. The value is the name of the TMF 608 operation, e.g. getInventory. This value corresponds to the “operation” in the WSDL.**
msgName	**M**	**M**	**M**	**Identifies the name of the message (or contract) that is being exchanged as part of an operation. E.g. getInventoryResponse. This field corresponds to the message name in the WSDL.**
msgType	**M**	**M**	**M**	**Identifies the type of the message. One of: REQUEST, RESPONSE, NOTIFICATION, ERROR.**
senderURI	**M**	**M**	**M**	**Identifies the application sending the message.**
destinationURI	**M**	**M**	**M**	**Used to identify the final destination of the message (the OS that will process the content of the message). This field may point to a logical end point or abstract topic name to be resolved by the communication transport middleware in order to deliver the message.**
originatorURI	**O**	**O**	**O**	**Identifies the originator of the Business Activity.**
replyToURI	**O**	**NA**	**NA**	**Used by request messages to specify destination for response message, if not specified the SenderURI endpoint is used to send back the response.**
failureToURI	**O**	**NA**	**NA**	**Identifies the application receiving the potential error notification message. Required for all response messages and notifications which are triggered by request/response. If**

Header field	Req	Resp	Notif	Description
				not specified the replyToURI is used. If the replyToURI is also not specified, senderURI endpoint is used to send back the failure response.
activityStatus	NA	M	NA	Specifies the high-level response status for an activity. Required for response messages, including error responses. One of: SUCCESS. FAILURE, WARNING
correlationId	O*	O*	NA	This field may be set by the originator of an asynchronous request that will allow it to correlate the response to the request. If this field is set, it's value must be reflected in the header of the response message. *=mandatory for MSG style communication.
security	O	O	O	Contains credential information used to secure message processing.
securityType	O	O	O	Identifies the type of credential in the security element.
priority	O	O	O	Indicates message-handling priority for messages. It must be in the range 0-9 (lowest-highest). Default: 4
msgSpecificProperties	O	O	O	Conditionally required for request, response, and notification messages as identified by the documentation for a specific interface message. The communications infrastructure or the receiving application can use this value for routing or filtering messages.
communicationPattern	M	M	M	Communication Pattern - SimpleResponse, MultipleResponse, BulkResponse, Notification
communicationStyle	M	M	M	Communication Style: RPC, MSG
requestedBatchSize	M*	O*	NA	logical size of the batch for a mutli-response communication pattern. (can be set to 0 to imply a single response). *NA for SimpleResponse and Notification comm. pattern
batchSequenceNumber	NA	M*	NA	Used in a multiple response Comm. Pattern to identify the batch seq number in a sequence. *NA for SimpleResponse and Notification comm.. pattern
batchSequenceEndOfReply	NA	M*	NA	Used in a multiple response Comm. Pattern, true if it is the last result batch in a sequence. *NA for SimpleResponse and Notification comm. pattern
iteratorReferenceURI	NA	M*	NA	Mandatory in the Synchronous Iteration MEP (SIT)
fileLocationURI	M*	O*	NA	Used for file retrieval. Specify the base name of the file(s) to be generated and the remote destination. *Mandatory in the BulkResponse comm. Pattern, NA otherwise.

Header field	Req	Resp	Notif	Description
compressionType	**M***	**O***	**NA**	**Used for file retrieval. Specify if compression is to be performed.** ***Mandatory in the BulkResponse comm. Pattern, NA otherwise.**
packingType	**M***	**O***	**NA**	**Used for file retrieval. Specify if the output file(s) are to be packed.** ***Mandatory in the BulkResponse comm. Pattern, NA otherwise.**
timeStamp	**O**	**O**	**O**	**Used to reflect the time when the message was created.**
vendorExtensions	**O**	**O**	**O**	**Additional vendor specific information.**

The MTOSI Header contains generic meta information related to the MTOSI body (business payload) as well as the data related to the specific communication pattern and communication style on the implemented MEP. We will refer to these latter elements (shaded as green in Table 2) as MEP elements in the rest of this document.

4.1.1 Mandatory Header fields applicable to all MEPs

Independently from the MEP here is the list of Header element mandatory in the MTOSI messages:

Mandatory in Both request response and notification:

- **activityName** - Identifies the name of the ***business transaction activity*** [SD2-5] (a.k.a. operation in the WSDL) for the message being exchanged. (e.g. getInventory)
- **msgName** Identifies the name of the message that is being exchanged as part of an operation. E.g. getInventoryResponse.
- **msgType** - Identifies the type of the message. One of: REQUEST, RESPONSE, NOTIFICATION
- **senderURI** - Identifies the application sending the message. (actual value depends on the specific transport adopted in the MTOSI deployment: e.g JNDI name for JMS transport)
- **destinationURI** - Identifies the final destination of the message (actual value depends on the specific transport adopted in the MTOSI deployment: e.g JNDI name for JMS transport)

Mandatory only in the response:

Each response message will have to specify the response status of the operation in the **activityStatus** header element. The possible values are: SUCCESS. FAILURE, WARNING

4.1.2 Correlation of asynchronous messages in the MTOSI message style

MTOSI uses the header field **correlationID** to correlate messages belonging to the same business activity in the context of a communication pattern instance (e.g. all the responses belonging to a getInventory request instance) See Figure 1~~Figure 2~~.

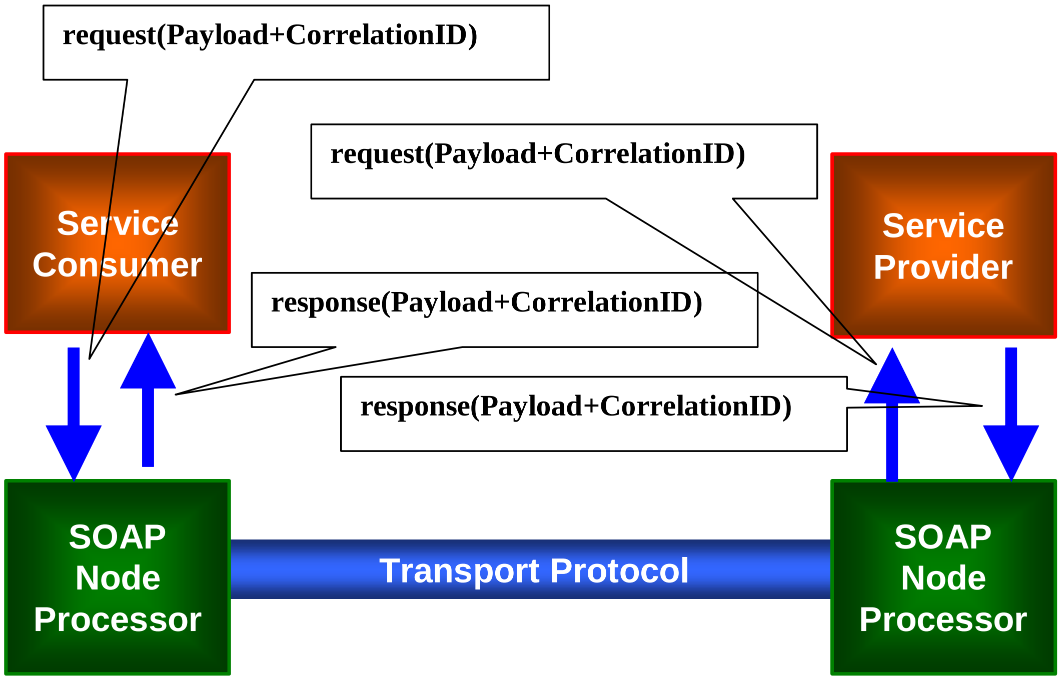
request(Payload+CorrelationID)
request(Payload+CorrelationID)
Service Consumer
Service Provider
response(Payload+CorrelationID)
response(Payload+CorrelationID)
SOAP Node Processor
Transport Protocol
SOAP Node Processor

Figure 1~~2~~ - use of CorrelationID in MTOSI

MTOSI mandates that the response messages related to a request MUST have the same correlationID value of the initial request. Figure 2~~Figure 3~~ and illustrate the detailed behavior of the correlation mechanism.

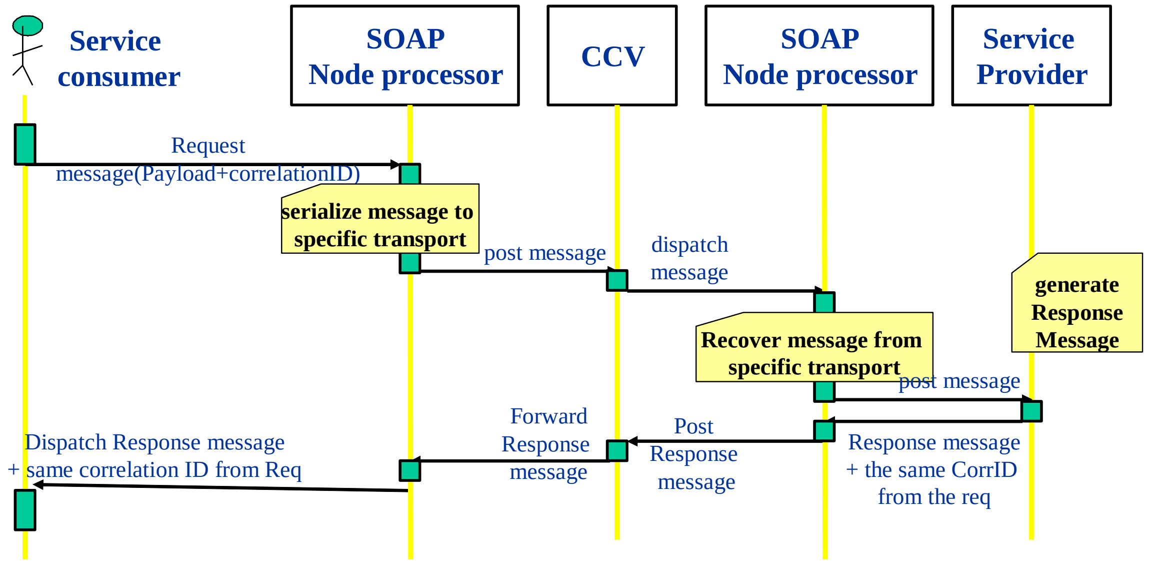
Service consumer
SOAP Node processor
CCV
SOAP Node processor
Service Provider
Request message(Payload+correlationID)
serialize message to specific transport
post message
dispatch message
generate Response Message
Recover message from specific transport
post message
Dispatch Response message + same correlation ID from Req
Forward Response message
Post Response message
Response message + the same CorrID from the req

Figure 2~~3~~ – MTOSI CorrelationID mechanism

While the MTOSI correlationID can be used at the application level to correlate and dispatch a message, other orthogonal correlation mechanisms can be adopted at different abstraction levels. For instance, in JMS MTOSI suggests the adoption of the fields JMS message ID and JMS Correlation ID to perform the correlation at the transport level. See SD2-9, Using JMS as an MTOSI Transport, for the details on how the MTOSI correlationID is mapped into the JMS transport stack.

4.1.3 Header fields setting for Asynchronous Request/Reply (ARR)

This is the simplest pattern and can be seen as a degenerate case of the ABR when there is only a single response.

Mandatory MEP elements:

- **communicationPattern = SimpleResponse**
- **communicationStyle = MSG**
- **correlationID** = the field has to exist, but the actual value may be provided by the underlined transport in the implicit modality.
- **replyToURI =** this field is necessary to indicate the address of the service end-point which is to receive the response

4.1.4 Header fields setting for Synchronous Request/Reply (SRR)

The synchronous variant of a Request response is identical in syntax to the ARR. The only difference is the coordination and correlation of the request and response. In the SRR the response is implicitly correlated to the request as the interaction is blocking and executed in the same communication logical channel.

Mandatory MEP elements:

- **communicationPattern = SimpleResponse**
- **communicationStyle = RPC**

4.1.5 Header fields setting for Asynchronous MultipleResponse (ABR)

This MEP allows fragmentation of the response into several logical units.

Mandatory MEP elements in both request and responses:

- **communicationPattern = MultipleResponse**
- **communicationStyle = MSG**
- **correlationID** = the field has to exist, but the actual value may be provided by the underlined transport in the implicit modality
- **replyToURI =** this field is necessary to indicate the address of the service end-point which is to receive the responses

Mandatory MEP elements in request:

- **requestedBatchSize** logical size of the batch for the result fragments. For example in the getInventory it is the maximum number of Inventory top level objects to be included in an inventory XML file (batch).

Mandatory MEP elements in responses:

- **batchSequenceNumber** - Used in a multiple response Comm. Pattern to identify the batch seq number in a sequence. (we start from 0)
- **batchSequenceEndOfReply** - Used in a multiple response Comm. Pattern, true if it is the last result batch in a sequence.

4.1.6 Header fields setting for Synchronous Iterator (SIT)

This MEP is the synchronous variation of the Batch Response and allows fragmentation of the response into several logical units. According to the SD2-5 the SIT is implemented with the Iterator Design pattern.

This pattern is composed of two primitives request/reply operations and four message types:

<OP> initial request

<OP>Response – initial response + Iterator reference

<OP>IteratorRequest – request for the subsequent data set responses

<OP>IteratorResponse – responses from the Iterator

Mandatory MEP elements in both request and responses:

- **communicationPattern = MultipleResponse**
- **communicationStyle = RPC**

Mandatory MEP elements in request (both initial request and Iterator request):

- **requestedBatchSize** logical size of the batch for the result fragments. For example in the getInventory it is the maximum number of Inventory top level objects to be included in an inventory XML file (batch).

Mandatory MEP elements in responses(both initial response and Iterator responses):

- **iteratorReferenceURI** – contains the to the Iterator endpoint reference
- **batchSequenceNumber** - Used in a multiple response Comm. Pattern to identify the batch seq number in a sequence. (we start from 0)
- **batchSequenceEndOfReply** - Used in a multiple response Comm. Pattern, true if it is the last result batch in a sequence.

4.1.6.1 Header fields setting for Asynchronous (File) Bulk (AFB)

This MEP allows transferring the entire response set through a side channel (secondary to the main CCV transport). E.g. this MEP allows sending a result set using an FTP stack.

Mandatory MEP elements in both request and responses:

- **communicationPattern = BulkResponse**
- **communicationStyle=MSG**
- **correlationID** = this is used only in the request and responses (acknowledge and notifications) controlling the FTP transfer.
- **replyToURI =** this field is necessary to indicate the address of the service end-point which is to receive the responses

Mandatory MEP elements in request:

- **requestedBatchSize** logical size of the batch for the transfer fragments. For example in the getInventory it is the maximum number of Inventory top level objects to be included in an inventory XML file (batch).
- **fileLocationURI** - URI provided by the requesting OS indicating the rootname of the file(s) to be produced and location of where to place the retrieved XML inventory file(s). See RFC 3986 for details on URI syntax
- **compressionType**- The type of compression to apply to the generated file(s). MTOSI v1.0 will define the following two options; NO_COMPRESSION, GZIP. Default behavior (when request parameter is omitted) is NO_COMPRESSION. Implementation of this file processing instruction by the Target OS is optional, and any incompatible request shall be handled with the appropriate exception. Vendor extension of this attribute shall be permitted
- **packingType** - The type of packing to apply to all the inventory file(s) generated from the same request. MTOSI v1.0 will define the following three options; NO_PACKING, ZIP, TAR. Default behavior (when request parameter is omitted) is NO_ PACKING. Implementation of this file processing instruction by the Target OS is optional, and any incompatible request shall be handled with the appropriate exception. Vendor extension of this attribute shall be permitted

Mandatory MEP elements in response:

The XML response to an AFB request is an acknowledge of the FTP transfer and optionally a set of intermediate notification regarding the state of the FTP transfer.

The header element **activityStatus** should be used (values: SUCCESS. FAILURE, WARNING) in the acknowledge to notify the service consumer of the result of the activity.

No additional elements are necessary in the header.

4.1.6.2 Header fields setting for Synchronous (File) Bulk (SFB)

This MEP is almost identical to the AFB as the style affect only the control flow of the file transfer.

The only difference is that the correlationID is no longer mandatory as the correlation is implicit and realized by the transport.

4.1.6.3 Subscription for Notifications

MTOSI Notification service interfaces are based on a minimal subset of functionality found in the OASIS WS-Notification specification. See the following references for specification details:

- SD2-6 for a description of the service interfaces
- SD2-9 Using JMS as an MTOSI Transport has a complete description of the JMS bindings and mechanisms for the JMS stack
- SD2-16 Using HTTP as MTOSI Transport promotes this style
-

Beside the common mandatory header element described in 4.1.1~~4.1.1~~ no other MEP element are necessary in the notification MEP.

4.2 SOAP Body in MTOSI

In MTOSI, we adopted the SOAP Document/literal wrapped style. In detail this means that the first element in the MTOSI body has to be the operation name (message name in MTOSI) and the rest of the body is a plain XML structure as defined in the XSD set. Note that the header contains the name of the message carried in the body. A SOAP-compatible envelope is provided in this MTOSI release to allow a designer/developer to assemble and validate MTOSI messages in isolation. Furthermore, while it is expected that an MTOSI deployment will take full advantage of the WS facilities such as SOAP parser, dispatchers and validators, nevertheless, MTOSI does not mandate nor require supporting a complete SOAP stack. In this later scenario, the WS framework will directly assemble the MTOSI header and body into a SOAP envelope.

We refer to the MTOSI body message as top level message and top level message type for the related XSD defining the message.

4.2.1 MTOSI Name syntax

4.2.2 MTOSI names are defined according to the network object containment hierarchy. See [SD2-7] for details about MTOSI Object Naming. General (default) behaviors on missing (empty) message parameters

This section describes general implementation behaviors of MTOSI service interface operations with respect to omitted input and output parameters (request and response message elements). These implementation behaviors are normative unless specified otherwise for a specific operation description.

4.2.3 Missing or empty XML elements in a request message (Query)

The implementation behavior by a Target OS of an MTOSI query-based operation must support a default behavior when message request elements/parameters are omitted by the Requesting OS.

In the case of a request with a filter, all the XML components are to be joined by a logical AND. The Target OS must compute the intersection of the various filter predicates. Not specifying optional elements in the filter is to be intended as relaxing a constraint on that element (allowing all the values to be matched).

The same mechanism can be generalized and extended to the other elements of a request message that concur to the definition of a scope of a result set.

Example#1: getActiveAlarms operation of the AlarmRetrieval service interface in ResourceTroubleManagement DDP

```
<getActiveAlarms>
  <filter>
        <acknowledgeIndication>AI_EVENT_ACKNOWLEDGED</acknowledgeIndication>
        <perceivedSeverityList>
                <perceivedSeverity>PS_CLEARED</perceivedSeverity>
        </perceivedSeverityList>
  </filter>
</getActiveAlarms>
```

This filter will retrieve all activeAlarms matching all alarms with acknowledgeIndication= AI_EVENT_ACKNOWLEDGED AND perceivedSeverity= PS_CLEARED

Removing the perceivedSeverityList from the filter will cause the query to return alarm matching any kind of perceivedSeverity.

Example#2: getInventory operation of the RersourceInventoryRetrieval service interface in the ManageResourceInventory DDP.

The diffDateAndTime element if not specified will be interpreted by the service provider as “retrieve all the known resources regardless of the change date”.

4.2.3.1 Missing or empty XML elements in a response

When an attribute is requested via the MTOSI, there are several possibilities concerning the availability of the value for the attribute. In the following table, we ~~should~~ show the various attribute value availability possibilities and the associated behavior by the Target OS (i.e., the OS to which the attribute retrieval request is made. It is critical to keep in mind that MTOSI Release ~~1~~ 2 does not allow for the retrieval of object attributes in isolation, i.e., the entire object needs to be retrieved and the Requesting OS can not specify a subset of an object’s attributes for retrieval.

Attribute Availability	**Expected Behavior by the Target OS**	**XML representation**	**Comment**
1. The value of the requested attribute is known to the Target OS. And is not empty	The Target OS returns the attribute value.	<TAG>value here</TAG> <TAG>23</TAG>	If the attribute is a string and it is known that the string is intended to be empty, the Target OS needs to indicate this to the Requesting OS.
2. The value of the requested attribute is known to the Target OS. And the value is empty.	This is only applicable to Strings. The Target OS returns the attribute value. Example name alias <nameAlias/>	If Type is String: <TAG></TAG> Or <TAG/> We cannot use this for any types other than string	If the attribute is a string and it is known that the string is intended to be empty, the Target OS needs to indicate this to the Requesting OS.
3. This attribute is supported by the Target OS, but the current value is not known to the Target OS.	The Target OS needs to inform the Requesting OS that the attribute is supported but its value is currently unknown. In enumerations we use the value UNKNOWN	Valid for String, Numeral and Booleans: <TAG nill=true></TAG> Or <TAG nill=true />	In the case of strings, it is important to distinguish this case from the case where the string is intentionally set to be empty.
4. The value of the requested attribute is known to the Target OS, but the value is suspect.	The Target OS returns the attribute value with an indication that the value is suspect.	We use the quality indicator in the inventory Layout	
5. The attribute is not supported by the Target OS.	The Target OS needs to inform the Requesting OS that the attribute is **not** supported by not including it in the response.	Tag is omitted from the XML	
6. The attribute is not understood by the Target OS.	-	-	This case cannot happen (not even between major MTOSI releases) since the Requesting OS cannot ask for specific attributes with regard to a give object type. The Target OS will always be working from its understand of the object

			type defintion and associated attributes.

In a request message a missing tag in a request can be in general interpreted as a relaxation of the request constraint. The next chapter addresses the partial specification of a filter in the context of a request message.

5 MTOSI Extensibility

The MTOSI service interfaces specifications are designed to support the following two extension mechanisms:

- Vendor extensions, and
- Minor version changes

Refer to the following supporting documents for details on these extensions:

- SD0-5 includes the MTOSI design guidelines for the definition of the service interfaces (WSDL and XML Schema)
- SD2-6 includes the MTOSI methodology regarding versioning and extensions

6 MTOSI WS Discovery/Registration

The MTOSI service interface specifications do not provide any normative requirements nor guidelines regarding the development, deployment, and runtime management of the implemented Web services. However, it is recommended that implementers and providers follow guidelines from W3C and OASIS regarding the discovery and registration of Web services.

Note that these topics will have to be accounted for as part of the future development of the service interface specifications to meet all the new requirements associated with a Service ~~Interface~~ Oriented Architecture (SOA).

7 References

[SOA] SOAP Version 1.1 (used with WSDL 1.1), W3C Note 08 May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[SD0-5]	SD0-5_mTOPGuidelines_WebServices
[SD2-1]	MTOSI Implementation Statement
[SD2-5]	Communication Styles
[SD2-6]	Versioning and Extensibility
[SD2-7]	Object Naming
[SD2-9]	Using JMS as an MTOSI Transport
[SD2-12]	Inventory Layout Description

8 Administrative Appendix

8.1 Document History

Version	Date	Description of Change
1.0	May 2005	This is the first version and as such, there are no changes to report.
1.1	June 2006	Added support for RPC style (SIT MEP)
1.2	August 2006	Minor editorial changes
2.0	May 2008	Updated for MTOSI R2

8.2 Acknowledgments

First Name	Last Name	Company
Michel	Besson	Cramer > Amdocs OSS Division
Francesco	Caruso	Telcordia Technologies Inc.
Shlomo	Cwang	Cramer > Amdocs OSS Division
Steve	Fratini	Telcordia Technologies Inc.
Elisabetta	Gardelli	Siemens
Jérôme	Magnet	Nortel Networks

8.3 How to comment on this document

Comments and requests for information must be in written form and addressed to the contact identified below:

Francesco	Caruso	Telcordia Technologies Inc.
Phone:	+1 732 699 3072	

Fax:	
e-mail:	caruso@research.telcordia.com

Please be specific, since your comments will be dealt with by the team evaluating numerous inputs and trying to produce a single text. Thus we appreciate significant specific input. We are looking for more input than wordsmith" items, however editing and structural help are greatly appreciated where better clarity is the result.