OMA Web Services Enabler (OWSER) Best Practices:

WSDL Style Guide

Candidate Version 1.0 – 16 Mar 2004

Open Mobile Alliance

OMA-OWSER-Best_Practice-WSDL_Style_Guide-V1_0-20040316-A

Contents

1. Scope

This document provides informative information for the use of the WSDL that should be followed for OMA defined Web Services.

The scope of content is consistent with the capabilities described in the MWS Overview and MWS Core Specification documents, and does not provide information beyond that required to support these capabilities.

This is the first in a series of best practices that are expected to be produced as part of the OWSER. Through the use of these best practices, it is expected that there will be a coherent and consistent approach to the use of Web Services in an OMA environment.

2. References

2.1 Normative References

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

2.2 Informative References

[XML1.0] "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, 6 October 2000, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, URL:http://www.w3.org/TR/REC-xml

[WSI1.0] "Web Services Basic Profile Version 1.0", Web Services Interoperability Organization, Board Approval Draft, 2003/03/28, URL:http://www.ws-i.org/Profiles/Basic/2003-03/BasicProfile-1.0-BdAD.html

[Schema-0] "XML Schema Primer", URL:www.w3.org/TR/xmlschema-0

[Schema-1] "XML Schema Part 1", Structures, URL:www.w3.org/TR/xmlschema-0

[Schema-2] "XML Schema Part 2", Datatypes, URL:www.w3.org/TR/xmlschema-0

[SOAP1.1] "Simple Object Access Protocol (SOAP) 1.1", Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Nielsen, Satish Thatte, Dave Winer, W3C Note, May 8, 2000, URL:http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[WSDL1.1] "Web Services Description Language (WSDL) 1.1", Erik Christensen, Francisco Cubrera, Greg Meredith, Sanjiva Weeravarana, W3C NOTE, March 15, 2001, URL:http://www.w3.org/TR/wsdl.html

[WSDL1.2] "Web Services Description Language (WSDL) Version 1.2", R. Chinnici, M. Gudgin, J-J. Moreau, S. Weerawarana , W3C Working Group Draft, July 9, 2001, URL:http://www.w3.org/TR/wsdl12/

[OWSERSpec] "OMA Web Services Enabler (OWSER): Core Specifications, Version 1.0", Open Mobile Alliance™, OMA-OWSER-Core-Specification-V1_0-20040216-D, URL:http://www.openmobilealliance.org

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", IETF RFC 2396, T. Berners-Lee, R. Fielding, L. Masinter, August 1998, URL:http://www.ietf.org/rfc/rfc2396.txt

[HTTP1.0] "Hypertext Transfer Protocol -- HTTP/1.0", IETF RFC 1945, Fielding, R., Frystyk, H. and T. Berners-Lee, May 1996, URL:http://www.ietf.org/rfc/rfc1945.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

Style Guide	A set of guidelines for use of a technology, providing specific directions for use and best practices.
Web service	A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. (Source: [WSGloss])

3.3 Abbreviations

HTTP	Hypertext Transfer Protocol
RPC	Remote Procedure Call
SOAP	Not an abbreviation, now a protocol name
UDDI	Universal Description, Discovery and Integration
WSDL	Web Services Description Language
XML	Extensible Markup Language

4. Introduction

Web Services Definition Language (WSDL) is the manner in which Web Services interfaces are described. It should be readable and understandable to a wide audience, following consistent styles for common type definitions and message patterns. The purpose of the WSDL Style Guide is to provide the information to allow for consistency, and to impart a set of best practices for WSDL creators to follow.

The target audience for this document is writers of OMA specifications that include WSDL documents as part of their deliverables.

With any style guide, there will be always be areas where multiple options are available and each option has its own merits. This guide will endeavor to first address the common WSDL constructs and message patterns in a best practices manner, while considering the readability and usability of the WSDL documents.

WSDL is not dependent on a particular underlying protocol, though many current implementations are based on SOAP over HTTP bindings. The [OWSERSpec] mandates the use of the SOAP over HTTP binding.

4.1 Document Notes

Throughout the document ‘www.openmobilealliance.org’ is used as a namespace. It is used to show correct usage, it does not represent any specification output of OMA.

Best practices guidelines are provided for each section, and a reference summarizing the guidelines is attached as Appendix A.

5. Decisions Influencing the Content of the WSDL Style Guide

When creating the WSDL Style Guide, a number of decisions on what to include and exclude must be made. Some of these decisions are influenced by technical direction, some by common sense, and some based on experience with a variety of WSDL documents. However, not all the decisions taken are easily recognized in the resulting guide, and these are explained here.

5.1 SOAP Message Style

In WSDL 1.1, two message styles were defined;

- Document, commonly used for exchange of documents fully defined with XML Schema.
- RPC, commonly used for providing a list of datum to an operation and receiving discrete datum from the operation.

These message styles affect the SOAP binding definition, and the definition of WSDL messages to be used with the SOAP binding.

However, there are a number of conventions that have been created to allow the RPC style message to be assimilated within the document style message definition. While a standard convention has not yet been adopted, there is significant effort underway to have one message style.

This WSDL Style Guide will provide information on how to write document style WSDL definitions.

5.2 SOAP Encoding

SOAP Encoding, as defined in part 5 of the SOAP 1.1 specification, provided a mechanism for representing data transported using SOAP in a compact form. However, this form can not be represented by an XML Schema.

The WS-I Basic Profile disallows use of SOAP encoding for messages, and the WSDL Style Guide does not include coverage of SOAP encoding.

5.3 Guidelines

- Evaluate which style will best fit the requirements of the service. Use of document style is recommended.
- Do not use SOAP encoding when developing OMA defined Web services.

6. Service Definition and Documents

Service definitions are expressed using the facilities of WSDL. While it is possible to produce a single document that represents an entire service definition, this is not a desirable approach for any non-trivial Web Service.

Decomposition provides the following benefits:

- XML Schema is used for data type definitions
- Faults that are shared across interfaces are defined independently
- Service interface definitions are defined independent of bindings
- Bindings are defined independently and consistent with UDDI best practices

Following these conventions improves the overall definition and maintenance process and improves deployment by supporting separation of interface and binding.

6.1 Single PortTypes and PortType Sets

A Web Service definition may contain a single portType or a set of portTypes (or interfaces in WSDL 1.2). The Web Service being considered will determine whether one portType or multiple porTypes are appropriate – using some of the same criteria as is used for other technologies.

The term *Single PortType* will be used to describe a Web Service with a single portType. A portType is not limited in the number of operations it may contain.

The term *PortType Set* will be used to describe a Web Service that contains multiple portTypes. When a *PortType Set* is defined, is provides a mechanism to group a set of related portTypes using well defined conventions for document and namespace naming.

For reference, other technologies group related interfaces using ‘module’ or ‘package’ conventions, achieving a similar result for organizing related interfaces.

6.2 Preparing for Document Definition

To provide a consistent use of naming within document sets, and across document sets, a number of conventions are defined that rely on a small amount of preparation to be done before creating the documents.

An initial decision is whether the Web Service will have a single porType or multiple portTypes. If a single portType is to be defined, follow the directions for creating a *Single PortType*. If multiple portTypes are to be defined, follow the directions for creating a *PortType Set*.

For each *PortType Set*, a *Base Name* is selected. For each portType (either a *Single PortType* or each portType within a *PortType Set*), a *Short Name* is selected. These names will be used as part of a common naming convention for the related set of documents defined and for definition naming within the documents.

A couple of examples will provide clarity for these preparation steps. Consider first a simple Web Service definition containing one portType to retrieve the stock price of a specified stock.

- This is a *Single PortType*, so there is no *Base Name* defined
- The single portType, GetQuote, is assigned the *Short Name* ‘quote’

In another example, a group of portTypes for a short messaging service (SMS) is defined. This Web Service contains multiple portTypes.

- Since there are multiple portTypes, a *PortType Set* is defined (SMS PortType Set)
- The *Base Name* for the *PortType Set* is assigned the name ‘sms’

- Each portType within the *PortType Set* is assigned a *Short Name*
    - The SendSms portType is assigned the *Short Name* ‘send’
    - The ReceiveSms portType is assigned the *Short Name* ‘receive’

Base Names and *Short Names* are always defined using only lower case letters, numbers or underscore characters. They must not start with a non-alphabetic character. An underscore should be used to separate words when the name consists of multiple words. These restrictions apply since these names are used in the construction of file names and URI content.

With these preparations complete, the document set may be created.

6.3 Documents

There are four document types that can be utilized in a Web Service definition. Each has a specific role, and contributes to the goal of supporting a well organized and useful decomposition of the individual elements of a Web Service definition.

6.3.1 Types Definition Document

The *Type Definitions Document* contains data type definitions within a schema namespace.

When the document is related to a specific *Single PortType* or *PortType Set*, it will use the *Base Name* (or *Short Name* for a Single PortType) with the suffix ‘_types’ and the extension ‘.xsd’. When the *Type Definitions Document* is used across multiple *Single PortTypes* or *PortType Sets*, it will use an independent name with the suffix ‘_types’ and the extension ‘.xsd’.

This document is optional, since not all services will define new datatypes.

6.3.2 Shared Faults Document

The *Shared Faults Document* contains fault definitions that are shared across *Single PortTypes*, across multiple portTypes in a *PortType Set*, or across *PortType Sets*.

The faults are defined within their own namespace within the WSDL definition namespace. The document name for this document will use the suffix ‘_faults’ and the extension ‘wsdl’. The first part of the name of the document is based on its usage, with the following guidance;

- If it is used by multiple *Single PortTypes* or multiple *PortType Sets*, an independent name reflective of the faults defined will be chosen by the author
- If it used only by multiple portTypes within a *PortType Set*, then the *Base Name* will be used for the first part of the name.

This document is optional, since not all WSDL definitions will define faults that are shared with other WSDL definitions.

6.3.3 Service Interface Document

The *Service Interface Document* contains the message and portType definitions. One portType definition is included in each document. This document may import *Type Definition Documents* and *Shared Faults Documents*. This document may be used for a variety of *Service Bindings Documents* without change.

The document name for this document will use the suffix ‘_interface’ and the extension ‘wsdl’. The name of the document is determined as follows;

- For a *Single PortType* the *Short Name* is used
- For a *PortType Set*, the name is a combination of the *Base Name* followed by an underscore followed by the *Short Name* for the portType defined in this document. Thus multiple documents will have the same *Base Name* as the first portion of the name and the individual portType *Short Name* as the second portion

6.3.4 Service Bindings Document

The *Service Bindings Document* contains both the binding to be used and the service definition associated with the binding. One service definition is defined in each document. This document imports one *Service Interface Document*.

The document name for this document will use the suffix '_service' with the extension 'wsdl'. Optionally, text representing the specific binding may be added immediately before the '_service' suffix. The name of the document is determined as follows;

- For a *Single PortType* the *Short Name* is used
- For a *PortType Set*, the name is a combination of the *Base Name* followed by an underscore followed by the *Short Name* for the portType defined in this document. Thus multiple documents will have the same *Base Name* as the first portion of the name and the individual portType *Short Name* as the second portion

6.4 Document Separation Rationale

The four document types approach satisfies a number of desirable goals for WSDL creation, use and maintenance.

- Types and shared faults are defined in common documents, eliminating redundant definitions
- PortTypes are defined in individual documents, providing easier reading (only relevant message definitions in same document), while using a naming convention that group related portTypes together
- Services are defined in individual documents, providing easy consumption by service registries and easy creation of alternate binding documents. Like the portType documents, the naming conventions for these documents group related services together.

By following this approach, the document decomposition supports modularization for reuse goals, in a manner that reflects a useful level of granularity, and useful document form for use with tools and for deployment use.

6.5 Document Naming Example

Using the SMS PortType Set described previously, the following document set would be produced. Additional assumptions for this example are that there are some data type definitions and that multiple portTypes in the *PortType Set* use a common set of faults.

One *Type Definitions Document* – sms_types.xsd

One *Shared Faults Document* – sms_faults.wsdl

Two *Service Interface Documents* – sms_send_interface.wsdl and sms_receive_interface.wsdl

Two *Service Bindings Documents* – sms_send_service.wsdl and sms_receive_service.wsdl

The two *Service Interface Documents* import the *Type Definition Document* and *Shared Faults Document*. The two *Service Bindings Documents* import their respective *Service Interface Document*.

6.6 Guidelines

- Define XML Schema types in Type Definition Documents.
- Define XML Schema for message parts, message definitions, and portType definitions in the Service Interface Document.
- Do not define other XML Schema types in the Service Interface Document.
- Do not define bindings or service definitions in the Service Interface Document.
- Define Faults that are shared across documents in a Shared Faults document.

- Define only binding and service definition content in a Service Binding Document.

7. Namespaces

The definitions tag has a number of attributes for namespace definitions. These definitions will include a set of common definitions and WSDL specific definitions. The common definitions will be provided in all WSDL documents.

7.1 Use of Namespaces

Correct use of namespaces is essential for both creating WSDL that will be usable by a variety of tools, and creating references that allow use of reusable content across the set of documents for a Web Service.

The following are the key namespaces defined,

- XML Schema namespaces for data type definitions
- Shared fault namespaces, for easy sharing of common fault definitions
- WSDL interface namespace for Web Service interface definitions
- WSDL schema local interface namespace for XML Schema definitions contained in the WSDL interface definition
- WSDL binding namespace for service bindings definitions

Each namespace has a distinct role. Managing them in a consistent way provides highly flexible definitions, while ensuring easy use by WSDL creators and readers.

7.2 Namespace Components

The namespace definition includes three defined components – the hierarchical name component, the version component and the namespace type component.

The hierarchical name component provides a fully qualified name in a hierarchical form for the namespace. This component is the Web Service specific information.

If a namespace contains a version number it will be a separate namespace component, immediately following the hierarchical name component, and preceding the namespace type. Version numbers are recommended, and are used in the examples.

There are two version conventions that may be used, one based on release numbering and another using date.

A version number based on release numbering consists of the lowercase letter ‘v’, followed by a number indicating major version number, followed by an underscore ‘_’, followed by a minor version number. Any numbering beyond the minor version number follows the same convention with an underscore separator. Numbers are not limited to single digits.

A version number based on date will follow these conventions;

- May use (year), (year/month), or (year/month/day), with forward slash separating each element of the date
- Year is 4 digits, month is two digits (with leading zero if necessary), day is two digits (with leading zero if necessary)

It is recommended that the first convention (e.g. v1_1) be used for documents that will be maintained and overlap with new revisions. This, for example, allows the use of a v1_2 and v2_0 documents when v1_2 is created after the release of v2_0. The date approach is recommended for documents that are expected to be replaced by later versions.

Following the version number is the namespace type, which is always the last element in the namespace. The namespace type is one of ‘faults’ for *Shared Faults Documents*, ‘interface’ or ‘local’ for *Service Interface Documents*, or ‘service’ for *Service Bindings Documents*. The *Type Definitions Document* does not have a namespace type; since it does not share its namespace ‘schema’ (XML Schema definitions in the *Service Interface Document* use the ‘local’ namespace type).

7.3 Common Namespaces

Each document type has some common namespaces will be used in every instance of that document type.

Type Definition Documents

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

Shared Faults Documents and *Service Interface Documents*

```
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

Service Bindings Documents for SOAP over HTTP

```
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

Other bindings will have other namespace definitions that will be common to all *Service Bindings Documents* using that binding.

7.4 Target Namespace

The target namespace defines the namespace that is the default namespace for the elements within a document. A special case is the target namespace defined for the schema section within the wsdl:types section of a *Service Interface Document*, where the target namespace applies specifically to the XML Schema definitions within this section.

The base namespace for WSDL related namespaces is http://www.openmobilealliance.org/wsdl. For sub-namespaces, they will extend this namespace. All components defined within the root namespace are defined within this base target namespace.

For example target namespaces may include,

Root namespace, http://www.openmobilealliance.org/wsdl/v1_0/service

Sub-namespace, http://www.openmobilealliance.org/wsdl/accounts/v1_0/service

Multi-level sub-namespace, http://www.openmobilealliance.org/wsdl/accounts/payables/v1_0/service

The target namespace is the same namespace that will be defined later as the XML Schema or WSDL namespace.

7.5 WSDL and Schema Namespaces

Namespaces are defined for the WSDL and Schema elements that are defined within this document, and for those that are referenced by elements in this document. For each instance, a pair of namespaces may be defined (if applicable). The WSDL namespace is defined with its *Short Name*. The Schema namespace is defined with the *Short Name* plus the suffix '_xsd'.

For the WSDL reference used for this document, the name space definition is the same as the targetNamespace. For the Schema reference, the base namespace is http://www.openmobilealliance.org/schema, with the same hierarchy reference following the base namespace, but with no ending qualifier since the schema namespace is not shared across documents.

Examples,

Base namespace

```
xmlns:example="http://www.openmobilealliance.org/wsdl/v1_0"
```

```
xmlns:example_xsd="http://www.openmobilealliance.org/schema/v1_0"
```

Sub-namespace

```
xmlns:accounts="http://www.openmobilealliance.org/wsdl/accounts/v1_0/service"
xmlns:accounts_xsd="http://www.openmobilealliance.org/schema/accounts/v1_0"
```

7.6 Local Namespace Use

Within the WSDL service definition, XML Schema is used to define messages. These are defined within the wsdl:types section of the *Service Interface Document*. Since namespaces must be unique across documents, and within different sections of the same document, a local namespace is used for the XML Schema types defined within the wsdl:types section.

The local namespace definition within a *Service Interface Document* uses the 'schema' namespace, with the *Base Name* and *Short Name* components followed by the version element and '/local'. The namespace is defined as the *Short Name* plus '_local_xsd'. This approach guarantees unique and predictable name use.

7.7 Examples

Base definitions,

```
<definitions
   name="example"
   targetNamespace="http://www.openmobilealliance.org/wsdl/v1_0/service"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:example="http://www.openmobilealliance.org/wsdl/v1_0/service"
   xmlns:example_xsd="http://www.openmobilealliance.org/schema/v1_0">
```

Sub-namespace definitions,

```
<definitions
   name="accounts"
   targetNamespace="http://www.openmobilealliance.org/wsdl/accounts/v1_0/service"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:example="http://www.openmobilealliance.org/wsdl/v1_0"
   xmlns:example_xsd="http://www.openmobilealliance.org/schema/v1_0"
   xmlns:accounts="http://www.openmobilealliance.org/wsdl/accounts/v1_0/service"
   xmlns:accounts_xsd="http://www.openmobilealliance.org/schema/accounts/v1_0">
```

7.8 Namespaces for OMA Defined Web Services

7.8.1 Base Namespace

For OMA interfaces, the base namespace is http://www.openmobilealliance.org/wsdl.

For each set of WSDL documents defined within a group, a namespace will be defined under the appropriate base namespace.

An example of a namespace for location services is http://www.openmobilealliance.org/wsdl/location

7.8.2 WSDL and Schema Namespaces

For the WSDL reference used for this document, the name space definition is the same as the targetNamespace. For the Schema reference, the base namespace is http://www.openmobilealliance.org/schema, with the same hierarchy reference following the base namespace.

For example,

OMA defined Web Services base namespaces

```
xmlns:oma=”http://www.openmobilealliance.org/wsdl”
xmlns:oma_xsd=”http://www.openmobilealliance.org/schema”
```

Location service

```
xmlns:location="http://www.openmobilealliance.org/wsdl/location"
xmlns:location_xsd="http://www.openmobilealliance.org/schema/location"
```

7.8.3 Common Type Definitions for OMA Defined Web Services

Across the set of OMA Web Services definitions, there will be a number of types that will be used by a number of services. These types will be defined in the namespace http://www.openmobilealliance.org/schema/common.

7.9 Guidelines

- Assign each document a unique target namespace.
- Include a version element in OMA namespaces. If a version element is used in a namespace, it should follow one of the defined formats – release numbering or date.
- If a version number is used in a namespace for WSDL documents, it should be the second last element of the namespace. The last element should contain the namespace type appropriate for the document type.
- If a version number is used in a namespace for XML Schema documents, it should be the last element.
- Include the specified list of common namespaces.
- Use ‘wsdl’ as the top level namespace element for WSDL documents and ‘schema’ as the top level element for XML Schema documents.
- Use ‘www.openmobilealliance.org’ as the host name in namespace definitions.

8. Authoring Style – Document Content and Names

8.1 General WSDL Document Information

The following are general guidelines for WSDL document information

- WSDL documents will use UTF-8 as their encoding. UTF-16 may also be used if required.
- A date in a comment at the top of the WSDL document will indicate the last revision date of the definition.

8.2 Names

Names will be normal language names, without prefixes (e.g. type or interface markers). The names will be meaningful, and not abbreviated in a way that makes the name hard to understand for users of the WSDL that are not literate in computer programming.

As a guideline, a person using the WSDL will be able to load the WSDL file into an XML viewer and see the names displayed and have reasonable understanding of the content.

This does not preclude the use of commonly understood acronyms within names (e.g. ATM) or commonly used abbreviations (e.g. max). However, the resulting name must still be meaningful.

8.3 Case Usage for Names

Two general cases are provided for, both using mixed case names; one with a leading capital letter, the other with a leading lowercase letter.

Names for all elements (all cases where the text name='Name' is used) will start with a letter and be mixed case, with the leading letter of each word capitalized. Words will not be separated by white space, underscore, hyphen or other non-letter character.

The following elements will have a leading uppercase letter – simpleType name, complexType name, portType name, binding name, service name.

The following elements will have a leading lowercase letter – sub-element names (those names used for elements within other elements), message name (message name portion, service prefix will have uppercase letter if used), message part name, portType operation name, binding operation name.

For example, valid names include 'Name', 'FirstName', 'Name1', 'mixedCaseName'. Invalid names include '1Name', 'NAME', 'nAME'.

8.4 Naming Conventions for Special Names

Some names have special meaning, and are often recognized by a naming convention. For example, in some conventions constants are identified by using all upper case letters and underscores between words.

In WSDL, the case usage for names will be followed as described in the previous section. No other conventions for case usage will be used.

For faults, the fault name will be suffixed with the word 'Exception'.

In many technologies, the return value of an operation is not named. However, in WSDL the response message contains a named part. The part representing the response message content will use the name 'result'.

8.5 Document Layout

To provide easy and consistent reading of WSDL files, the following layout patterns are recommended.

Each tag level is indented one level relative to the previous tag indent level. The xml tag, date comment and root tag are not indented, they are on the left margin.

Indents of 3 spaces are used, and tabs are not used for storage (store files with spaces).

Namespaces are defined one per line, single spaced, indented one indent level.

Import statements are defined one per line, single spaced, with attributes on the same line.

Each primary element within the schema is separated by one blank line.

Each element within a primary element is single spaced.

Restrictions, extensions and elements are defined on a single line with their attributes.

XML Schema types are laid out according to their respective sections in this document.

Messages are defined single spaced, with one blank line separating each message definition. Messages with no parts are defined with one tag.

PortTypes are defined with its attributes on a single line, with one blank line separating each portType definition. The first operation starts on the line following the portType definition, with each operation defined single spaced and with a blank line separating each operation definition. Each element defined within an operation (input, output and fault) is defined on a single line with its attributes within one tag.

Bindings will be laid out consistently with portTypes.

Each service is defined single spaced.

8.6 Guidelines

- Use UTF-8 or UTF-16 encoding
- Follow the case usage guidelines for names
- Follow the recommended naming conventions for special names
- Follow the recommended document layout form

9. Data Type Definitions

All data type definition examples are shown as their XML Schema type definitions.

9.1 Types Section Declaration

All data types are defined using XML Schema in the Type Definitions Document.

Base document definition,

```
<xsd:schema>
   targetNamespace="http://www.openmobilealliance.org/schema/v1_0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <!-- types -->
</xsd:schema>
```

Sub-namespace document definition,

```
<xsd:schema>
   targetNamespace="http://www.openmobilealliance.org/schema/accounts/v1_0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <!-- types -->
</xsd:schema>
```

9.1.1 Optional Elements

XML Schema allows elements to be defined as optional, meaning that the element may or may not be present within an XML document. This is convenient for applications that process XML documents as a whole, however it is problematic for use with applications receiving the message content as discrete elements, where optional parameters are either not supported, or have strict rules on their use. Therefore, optional elements should only be used in cases where the XML content is to be processed as a whole document.

Elements may be declared as optional by use of the minOccurs attribute with a value of zero.

```
   <xsd:element name="example" type="xsd:int" minOccurs="0" maxOccurs="1"/>
```

An alternative to using this XML convention is to specify an optional element is to instead designate a particular value that is to be treated as a value to ignore. For example, an integer may be assigned of -1 to indicate no valid value is provided. In each case, this value (or values) must be documented.

Note that this is more restrictive than [WSI 1.0], since this is a practice that improves programmability but does not affect interoperability.

9.1.2 Nillable Elements

XML Schema allows elements to be declared as nillable, but like optional elements there can be problems with how this is translated into programming languages. Therefore, types that translate to primitive types in programming languages (like boolean and int) should not be declared as nillable.

An alternative is to specify a particular value that is to be treated as a value to be treated as a nil value. For example, an integer may be assigned of -1 to indicate no valid value is provided. In each case, this value (or values) must be documented.

Note that this is more restrictive than [WSI 1.0], since this is a practice that improves programmability but does not affect interoperability.

9.1.3 Elements with default values

Elements may be declared with a default value by use of the default attribute with an associated value.

```
<xsd:element name="example" type="xsd:int" default="10"/>
```

9.1.4 User Defined Simple Data Types

User defined simple data types are defined as XML Schema simpleType elements with a name attribute and a restriction to the base XML Schema type. Simple data types do not extend other simple data types.

```
<xsd:simpleType name="Example">
    <xsd:restriction base="xsd:int"/>
</xsd:simpleType>
```

9.1.5 User Defined Constants

User defined constants are always defined within XML Schema complexType elements using the 'fixed' attribute. If a constant is a 'standalone' constant, that is that it does not belong to a particular type, then a generic type must be defined around it – e.g. a complexType with the name 'Constants'.

It is expected that constants that are related to each other will be defined within independent complexTypes.

```
<xsd:complexType name="TitleProperties">
    <xsd:sequence>
       <xsd:element name="defaultSize" type="xsd:int" fixed="24"/>
       <xsd:element name="defaultFont" type="xsd:string" fixed="Serif"/>
    </xsd:sequence>
 </xsd:complexType>
```

9.1.6 Data Structures

Data structures are defined as complex types with a sequence of elements within the complex type.

```
<xsd:complexType name="Account">
   <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="address" type="example_xsd:Address"/>
   </xsd:sequence>
</xsd:complexType>
```

9.1.7 Enumerations

Enumerations are defined using XML Schema enumerations.

```
<xsd:simpleType name="Days">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Sunday"/>
      <xsd:enumeration value="Monday"/>
```

```
      <xsd:enumeration value="Tuesday"/>
   </xsd:restriction>
</xsd:simpleType>
```

Enumerations are assigned the literal values from the list provided, not a generated integer representing each enumeration values.

9.1.8 Unions

Unions are defined using a data structure composed of the selection element, and an XML Schema choice element containing the list of union elements. The selection element is an enumeration type indicating the choice element that is valid.

```
<xsd:simpleType name="PhoneType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="home"/>
      <xsd:enumeration value="work"/>
      <xsd:enumeration value="mobile"/>
   </xsd:restriction>
</xsd:simpleType>


<xsd:complexType name="PhoneNumber">
   <xsd:sequence>
      <xsd:element name="switchName" type="example_xsd:PhoneType">
      <xsd:choice>
         <xsd:element name="home" type="xsd:string"/>
         <xsd:element name="work" type="xsd:string"/>
         <xsd:element name="mobile" type="example_xsd:MobileNumber"/>
      </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
```

9.2 Guidelines

- Do not use optional elements for OMA Web services in cases where the XML content will not be processed as a whole document..
- Do not use nillable elements for XML Schema primitive types.
- Follow the appropriate data definition form for element definitions.

10. Other WSDL Document Content

Other portions of the WSDL documents include the definitions for messages, port types, bindings and services. The content for these elements is covered in [WSDL1.1] and [WSI 1.0].

10.1 Messages

Messages are used in the operation elements of portTypes, providing the definition of the content that is exchanged on input, output and faults.

10.2 PortTypes

PortTypes make a set of operations available, and define the messages that will be used for each.

For the request / response message pattern, a portType will have an operation definition that contains a single input message, a single output message and zero or more fault messages, in that order.

For the one way message pattern, a portType will have an operation definition that contains a single input message only. No output message or fault messages are permitted.

No other message pattern is to be used for OMA Web Services.

10.3 Bindings

The binding defines how the WSDL definitions will be utilized in interacting with the network. The binding defines the protocols and operational style of the binding. Only the SOAP / HTTP binding is supported for OMA Web Services.

This document does not address bindings in detail, as the binding is independent of the WSDL interface definitions. Specific information on use of the SOAP/HTTP binding is covered in [WSI 1,0].

10.4 Service Definitions

Services define an endpoint (port), though the address of the end point specified in this definition is often replaced at runtime when the discovery step determines that actual location that the service is hosted at.

Specific information on the use of the service definition and relationship to UDDI is covered in [WSI 1.0]

10.5 Guidelines

- For the request / response message pattern, define the portType operation to contain a single input message, a single output message and zero or more fault messages, in that order.
- For the one-way message pattern, define the portType operation to contain a single input message, no output message, and no faults.
- Do not use message patterns other than request / response or one-way.
- Use the SOAP/HTTP binding guidelines in [WSI 1.0].

11. Example

An example is provided for a Web Service using the document / literal style.

11.1 Example Type Definition Documents

This example includes two *Type Definition Documents*, one for common type definitions and one for service specific type definitions.

common_types.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- September 30, 2003 -->
<xsd:schema
   targetNamespace="http://www.openmobilealliance.org/schema/common/v1_0"
   xmlns:common_xsd="http://www.openmobilealliance.org/schema/common/v1_0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <xsd:complexType name="ArrayOfBankAccount">
      <xsd:sequence>
         <xsd:element name="ArrayOfBankAccount" type="common_xsd:BankAccount" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="BankAccount">
      <xsd:sequence>
         <xsd:element name="type" type="common_xsd:BankAccountType"/>
         <xsd:element name="number" type="xsd:int"/>
         <xsd:element name="balance" type="xsd:float"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Address">
      <xsd:sequence>
         <xsd:element name="number" type="xsd:string"/>
         <xsd:element name="street" type="xsd:string"/>
         <xsd:element name="city" type="xsd:string"/>
         <xsd:element name="state" type="xsd:string"/>
         <xsd:element name="country" type="xsd:string"/>
         <xsd:element name="zipCode" type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:simpleType name="BankAccountType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="savings"/>
         <xsd:enumeration value="checking"/>
         <xsd:enumeration value="certificate"/>
      </xsd:restriction>
   </xsd:simpleType>

   <xsd:element name="ServiceException">
      <xsd:complexType>
         <xsd:sequence>
```

```
            <xsd:element name="ServiceException" type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
```

account_types.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- September 30, 2003 -->
<xsd:schema
   targetNamespace="http://www.openmobilealliance.org/schema/account/v1_0"
   xmlns:account_xsd="http://www.openmobilealliance.org/schema/account/v1_0"
   xmlns:common_xsd="http://www.openmobilealliance.org/schema/common/v1_0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <xsd:import namespace="http://www.openmobilealliance.org/schema/common/v1_0"
schemaLocation="common_types.xsd"/>

   <xsd:complexType name="CustomerAccount">
      <xsd:sequence>
         <xsd:element name="number" type="xsd:int"/>
         <xsd:element name="name" type="xsd:string"/>
         <xsd:element name="address" type="common_xsd:Address"/>
         <xsd:element name="accounts" type="common_xsd:ArrayOfBankAccount"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:element name="CreateAccountException">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="CreateAccountException" type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="InvalidAccountException">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="InvalidAccountException" type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

11.2 Example Shared Faults Documents

Faults may be provided in multiple documents, since their usage may be applicable over multiple WSDL interfaces. It is common for some generic faults to be captured in a common faults document, and faults in a particular area to be grouped in a shared document. When a fault is specific to a document, it should be defined within that document. Two *Shared Faults Documents* are shown; one for generic faults and one for faults than span the account interfaces.

common_faults.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- September 30, 2003 -->
<wsdl:definitions
   name="common_faults"
```

```
   targetNamespace="http://www.openmobilealliance.org/wsdl/common/v1_0/faults"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:common_xsd="http://www.openmobilealliance.org/schema/common/v1_0">

   <wsdl:types>
      <xsd:schema elementFormDefault="qualified">
         <xsd:import namespace="http://www.openmobilealliance.org/schema/common/v1_0"
schemaLocation="common_types.xsd"/>
      </xsd:schema>
   </wsdl:types>

   <wsdl:message name="ServiceException">
      <wsdl:part name="ServiceException" element="common_xsd:ServiceException"/>
   </wsdl:message>
</wsdl:definitions>
```

account_faults.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- September 30, 2003 -->
<wsdl:definitions
   name="account_faults"
   targetNamespace="http://www.openmobilealliance.org/wsdl/account/v1_0/faults"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:account_xsd="http://www.openmobilealliance.org/schema/account/v1_0">

   <wsdl:types>
      <xsd:schema elementFormDefault="qualified">
         <xsd:import namespace="http://www.openmobilealliance.org/schema/account/v1_0"
schemaLocation="account_types.xsd"/>
      </xsd:schema>
   </wsdl:types>

   <wsdl:message name="CreateAccountException">
      <wsdl:part name="CreateAccountException" element="account_xsd:CreateAccountException"/>
   </wsdl:message>

   <wsdl:message name="InvalidAccountException">
      <wsdl:part name="InvalidAccountException" element="account_xsd:InvalidAccountException"/>
   </wsdl:message>
</wsdl:definitions>
```

11.3 Example Service Interface Documents

Two *Service Interface Documents* are shown; one for each service portType.

account_management_interface.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- September 30, 2003 -->
<wsdl:definitions
   name="account_management_interface"
   targetNamespace="http://www.openmobilealliance.org/wsdl/account/management/v1_0/interface"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
```

```
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:account_management="http://www.openmobilealliance.org/wsdl/account/management/v1_0/interface"
   xmlns:account_faults="http://www.openmobilealliance.org/wsdl/account/v1_0/faults"
   xmlns:common_faults="http://www.openmobilealliance.org/wsdl/common/v1_0/faults"
   xmlns:common_xsd="http://www.openmobilealliance.org/schema/common/v1_0"
   xmlns:account_xsd="http://www.openmobilealliance.org/schema/account/v1_0"

xmlns:account_management_local_xsd="http://www.openmobilealliance.org/schema/account/management/v1_0/
local">

   <wsdl:import namespace="http://www.openmobilealliance.org/wsdl/account/v1_0/faults"
location="account_faults.wsdl"/>
   <wsdl:import namespace="http://www.openmobilealliance.org/wsdl/common/v1_0/faults"
location="common_faults.wsdl"/>

   <wsdl:types>
      <xsd:schema elementFormDefault="qualified"
targetNamespace="http://www.openmobilealliance.org/schema/account/management/v1_0/local">
         <xsd:import namespace="http://www.openmobilealliance.org/schema/common/v1_0"
schemaLocation="common_types.xsd"/>
         <xsd:import namespace="http://www.openmobilealliance.org/schema/account/v1_0"
schemaLocation="account_types.xsd"/>

         <xsd:element name="createAccount">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="account" type="account_xsd:CustomerAccount"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

         <xsd:element name="createAccountResponse">
            <xsd:complexType>
               <xsd:sequence/>
            </xsd:complexType>
         </xsd:element>
      </xsd:schema>
   </wsdl:types>

   <wsdl:message name="BankAccount_createAccountRequest">
      <wsdl:part name="parameters" element="account_management_local_xsd:createAccount"/>
   </wsdl:message>

   <wsdl:message name="BankAccount_createAccountResponse">
      <wsdl:part name="result" element="account_management_local_xsd:createAccountResponse"/>
   </wsdl:message>

   <wsdl:portType name="BankAccount">
      <wsdl:operation name="createAccount">
         <wsdl:input message="account_management:BankAccount_createAccountRequest"/>
         <wsdl:output message="account_management:BankAccount_createAccountResponse"/>
         <wsdl:fault name="ServiceException" message="common_faults:ServiceException"/>
         <wsdl:fault name="CreateAccountException" message="account_faults:CreateAccountException"/>
      </wsdl:operation>
   </wsdl:portType>
```

```
</wsdl:definitions>
```

account_access_interface.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- September 30, 2003 -->
<wsdl:definitions
   name="account_access_interface"
   targetNamespace="http://www.openmobilealliance.org/wsdl/account/access/v1_0/interface"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:account_access="http://www.openmobilealliance.org/wsdl/account/access/v1_0/interface"
   xmlns:account_faults="http://www.openmobilealliance.org/wsdl/account/v1_0/faults"
   xmlns:common_faults="http://www.openmobilealliance.org/wsdl/common/v1_0/faults"
   xmlns:common_xsd="http://www.openmobilealliance.org/schema/common/v1_0"
   xmlns:account_xsd="http://www.openmobilealliance.org/schema/account/v1_0"

xmlns:account_access_local_xsd="http://www.openmobilealliance.org/schema/account/access/v1_0/local">

   <wsdl:import namespace="http://www.openmobilealliance.org/wsdl/account/v1_0/faults"
location="account_faults.wsdl"/>
   <wsdl:import namespace="http://www.openmobilealliance.org/wsdl/common/v1_0/faults"
location="common_faults.wsdl"/>

   <wsdl:types>
      <xsd:schema elementFormDefault="qualified"
targetNamespace="http://www.openmobilealliance.org/schema/account/access/v1_0/local">
         <xsd:import namespace="http://www.openmobilealliance.org/schema/common/v1_0"
schemaLocation="common_types.xsd"/>
         <xsd:import namespace="http://www.openmobilealliance.org/schema/account/v1_0"
schemaLocation="account_types.xsd"/>

         <xsd:element name="getAccount">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="accountNumber" type="xsd:int"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>

         <xsd:element name="getAccountResponse">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="result" type="account_xsd:CustomerAccount"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
      </xsd:schema>
   </wsdl:types>

   <wsdl:message name="BankAccountAccess_getAccountRequest">
      <wsdl:part name="parameters" element="account_access_local_xsd:getAccount"/>
   </wsdl:message>

   <wsdl:message name="BankAccountAccess_getAccountResponse">
      <wsdl:part name="result" element="account_access_local_xsd:getAccountResponse"/>
```

```
   </wsdl:message>

   <wsdl:portType name="BankAccountAccess">
      <wsdl:operation name="getAccount">
         <wsdl:input message="account_access:BankAccountAccess_getAccountRequest"/>
         <wsdl:output message="account_access:BankAccountAccess_getAccountResponse"/>
         <wsdl:fault name="ServiceException" message="common_faults:ServiceException"/>
         <wsdl:fault name="InvalidAccountException"
message="account_faults:InvalidAccountException"/>
      </wsdl:operation>
   </wsdl:portType>
</wsdl:definitions>
```

11.4 Example Service Bindings Documents

Two example documents are shown; using SOAP over HTTP as the transport.

account_management_service.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- September 30, 2003 -->
<wsdl:definitions
   name="account_management_service"
   targetNamespace="http://www.openmobilealliance.org/wsdl/account/management/v1_0/service"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:tns="http://www.openmobilealliance.org/wsdl/account/management/v1_0/service"
   xmlns:interface="http://www.openmobilealliance.org/wsdl/account/management/v1_0/interface">

   <wsdl:import namespace="http://www.openmobilealliance.org/wsdl/account/management/v1_0/interface"
location="account_management_interface.wsdl"/>

   <wsdl:binding name="BankAccountBinding" type="interface:BankAccount">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

      <wsdl:operation name="createAccount">
         <soap:operation soapAction="" style="document"/>
            <wsdl:input>
               <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body use="literal"/>
            </wsdl:output>
            <wsdl:fault name="ServiceException">
               <soap:fault name="ServiceException" use="literal"/>
            </wsdl:fault>
            <wsdl:fault name="CreateAccountException">
               <soap:fault name="CreateAccountException" use="literal"/>
            </wsdl:fault>
      </wsdl:operation>
   </wsdl:binding>

   <wsdl:service name="BankAccountService">
      <wsdl:port name="BankAccount" binding="tns:BankAccountBinding">
```

```
            <soap:address location="http://localhost:6080/BankAccountService/services/BankAccount"/>
         </wsdl:port>
      </wsdl:service>
   </wsdl:definitions>
```

account_access_service.wsdl

```
   <?xml version="1.0" encoding="UTF-8"?>
   <!-- September 30, 2003 -->
   <wsdl:definitions
      name="account_access_service"
      targetNamespace="http://www.openmobilealliance.org/wsdl/account/access/v1_0/service"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:tns="http://www.openmobilealliance.org/wsdl/account/access/v1_0/service"
      xmlns:interface="http://www.openmobilealliance.org/wsdl/account/access/v1_0/interface">

      <wsdl:import namespace="http://www.openmobilealliance.org/wsdl/account/access/v1_0/interface"
   location="account_access_interface.wsdl"/>

      <wsdl:binding name="BankAccountAccessBinding" type="interface:BankAccountAccess">
         <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

         <wsdl:operation name="getAccount">
            <soap:operation soapAction="" style="document"/>
               <wsdl:input>
                  <soap:body use="literal"/>
               </wsdl:input>
               <wsdl:output>
                  <soap:body use="literal"/>
               </wsdl:output>
               <wsdl:fault name="ServiceException">
                  <soap:fault name="ServiceException" use="literal"/>
               </wsdl:fault>
               <wsdl:fault name="InvalidAccountException">
                  <soap:fault name="InvalidAccountException" use="literal"/>
               </wsdl:fault>
         </wsdl:operation>
      </wsdl:binding>

      <wsdl:service name="BankAccountAccessService">
         <wsdl:port name="BankAccountAccess" binding="tns:BankAccountAccessBinding">
            <soap:address
   location="http://localhost:6080/BankAccountAccessService/services/BankAccountAccess"/>
         </wsdl:port>
      </wsdl:service>
   </wsdl:definitions>
```

Appendix A. Guideline Summary

This summary is a compilation of the guidelines. It does not carry any implications of OMA Web services conformance, but is supplied to facilitate WSDL design based on the best practices outlined in this document.

Section	Guideline
5	Evaluate which style will best fit the requirements of the service. Use of document style is recommended. Do not use SOAP encoding when developing OMA defined Web services.
6	Decompose XML Schema types in Type Definition Documents. Define XML Schema for message parts, message definitions, and portType definitions in the Service Interface Document. Do not define other XML Schema types in the Service Interface Document. Do not define bindings or service definitions in the Service Interface Document. Define Faults that are shared across documents in a Shared Faults Document. Define only binding and service definition content in a Service Binding Document.
7	Assign each document a unique target namespace. Include a version element in OMA namespaces. If a version element is used in a namespace, it should follow one of the defined formats – release numbering or date. If a version number is used in a namespace for WSDL documents, it should be the second last element of the namespace. The last element should contain the namespace type appropriate for the document type. If a version number is used in a namespace for XML Schema documents, it should be the last element. Include the specified list of common namespaces. Use ‘wsdl’ as the top level namespace element for WSDL documents and ‘schema’ as the top level element for XML Schema documents. Use ‘www.openmobilealliance.org’ as the host name in namespace definitions.
8	Use UTF-8 or UTF-16 encoding Follow the case usage guidelines for names Follow the recommended naming conventions for special names Follow the recommended document layout form
9	Do not use optional elements for OMA Web services in cases where the XML content will not be processed as a whole document. Do not use nillable elements for XML Schema primitive types.

	Follow the appropriate data definition form for element definitions.
10	For the request / response message pattern, define the portType operation to contain a single input message, a single output message and zero or more fault messages, in that order. For the one-way message pattern, define the portType operation to contain a single input message, no output message, and no faults. Do not use message patterns other than request / response or one-way. Use the SOAP/HTTP binding guidelines in [WSI 1.0].

Appendix B. Change History

B.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

B.2 Draft/Candidate Version 1_0 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-OWSER-Best_Practice-WSDL_Style_Guide-V1_0	11 Dec 2003		Initial draft
	16 Feb 2004	2.2	Consistency review comments
Candidate Version OMA-OWSER-Best_Practice-WSDL_Style_Guide-V1_0	16 Mar 2004	n/a	Status changed to Candidate by TP TP ref # OMA-TP-2004-0089-OWSER-V1_0-for-Candidate-Approval