



1

2 **Web Services Brokered Notification 1.3**
3 **(WS-BrokeredNotification)**

4 **Public Review Draft 01, 7 July 2005**

5

6 **Document identifier:**

7 wsn-ws-brokered-notification-1.3-spec-pr-01

8 **Location:**

9 http://docs.oasis-open.org/wsn/wsn-ws-brokered_notification-1.3-spec-pr-01.pdf

10 **Editors:**

11 Dave Chappell, Sonic Software <chappell@sonicsoftware.com>

12 Lily Liu, webMethods <lily.liu@webmethods.com>

13 **Abstract:**

14 The Event-driven, or Notification-based, interaction pattern is a commonly used pattern
15 for inter-object communications. Examples exist in many domains, for example in
16 publish/subscribe systems provided by Message Oriented Middleware vendors, or in
17 system and device management domains. This notification pattern is increasingly being
18 used in a Web services context.

19 WS-Notification is a family of related specifications that define a standard Web services
20 approach to notification using a topic-based publish/subscribe pattern. It includes:
21 standard message exchanges to be implemented by service providers that wish to
22 participate in Notifications, standard message exchanges for a notification broker service
23 provider (allowing publication of messages from entities that are not themselves service
24 providers), operational requirements expected of service providers and requestors that
25 participate in notifications, and an XML model that describes topics. The WS-Notification

26 family of documents includes three normative specifications: [\[WS-BaseNotification\]](#), WS-
27 BrokeredNotification, and [\[WS-Topics\]](#).

28 This document defines the Web services interface for the NotificationBroker. A
29 NotificationBroker is an intermediary, which, among other things, allows publication of
30 messages from entities that are not themselves service providers. It includes standard
31 message exchanges to be implemented by NotificationBroker service providers along
32 with operational requirements expected of service providers and requestors that
33 participate in brokered notifications. This work relies upon WS-BaseNotification.

34 **Status:**

35 On July 7th, 2005, the OASIS WS-Notification Technical Committee approved this
36 document for publication as a Public Review Draft. Committee members should send
37 comments on this specification to the wsn@lists.oasis-open.org list. Others may submit
38 comments to the TC via the web form found on the TC's web page at [http://www.oasis-
40 open.org/committees/wsn](http://www.oasis-
39 open.org/committees/wsn). Click the button for "Send A Comment" at the top of the page.
41 Submitted comments (for this work as well as other works of the TC) are publicly
archived and can be viewed at <http://lists.oasis-open.org/archives/wsn-comment/>.

42 For information on whether any patents have been disclosed that may be essential to
43 implementing this specification, and any offers of patent licensing terms, please refer to
44 the Intellectual Property Rights section of the WSN TC web page ([http://www.oasis-
open.org/committees/wsn/](http://www.oasis-
45 open.org/committees/wsn/)).

Table of Contents

47	1	Introduction.....	4
48	1.1	Goals and Requirements.....	4
49	1.1.1	Requirements.....	4
50	1.1.2	Non-Goals.....	5
51	1.2	Notational Conventions.....	5
52	1.3	Namespaces.....	6
53	1.4	Fault Definitions.....	7
54	2	Relationship to Other Specifications.....	7
55	3	Terminology and Concepts.....	8
56	4	Publishing.....	10
57	5	NotificationBroker Interface.....	12
58	5.1	NotificationBroker Resource Properties.....	13
59	5.2	Notify.....	13
60	5.3	Subscribe.....	14
61	5.4	GetCurrentMessage.....	14
62	5.5	RegisterPublisher.....	14
63	5.6	CreatePullPoint.....	14
64	6	RegisterPublisher Interface.....	15
65	6.1	RegisterPublisher.....	15
66	6.1.1	Example SOAP Encoding of the RegisterPublisher Message Exchange.....	18
67	7	PublisherRegistrationManager Interface.....	19
68	7.1	PublisherRegistration Resource Properties.....	20
69	7.2	Destroy.....	21
70	7.2.1	Example SOAP Encoding of the Destroy Message Exchange.....	21
71	8	Security Considerations.....	22
72	8.1	Securing PublisherRegistration.....	22
73	9	References.....	24
74	9.1	Normative.....	24
75	9.2	Non-Normative.....	24
76		Appendix A. Acknowledgments.....	25
77		Appendix B. XML Schema.....	26
78		Appendix C. WSDL 1.1.....	31
79		Appendix D. Notices.....	37

80 1 Introduction

81 The Event-driven, or Notification-based, interaction pattern is a commonly used pattern for inter-
82 object communications. Examples exist in many domains, for example, in publish/subscribe
83 systems or in system and device management domains. Message brokers are involved in many
84 of these systems, such as the ones provided by Message Oriented Middleware vendors.

85 This specification defines the Web services interface for the NotificationBroker. A
86 NotificationBroker is an intermediary between message Publishers and message Subscribers.
87 Common functions of Publishers and Subscribers, such as messaging dissemination and security
88 measurements, can be implemented at the NotificationBroker to produce lightweight Producers
89 and Consumers. A NotificationBroker decouples NotificationProducers and Notification
90 Consumers and can provide advanced messaging features such as demand-based publishing
91 and load-balancing. A NotificationBroker also allows publication of messages from entities that
92 are not themselves service providers. This is very similar to a traditional Message Oriented
93 Middleware model.

94 The NotificationBroker interface includes standard message exchanges to be implemented by
95 NotificationBroker service providers along with operational requirements expected of service
96 providers and requestors that participate in brokered notifications.

97 1.1 Goals and Requirements

98 The goal of WS-BrokeredNotification is to standardize message exchanges involved in Web
99 services publish and subscribe of a message broker. The overall objectives of WS-Notification
100 are presented in [[WS-BaseNotification](#)]. The following section lists the specific subset of those
101 objectives realized by WS-BrokeredNotification.

102 1.1.1 Requirements

103 In meeting this goal, the WS-BrokeredNotification specification must explicitly address the
104 following requirements:

- 105 • **Must allow for a notification broker as an intermediary.** A NotificationBroker is an
106 intermediary Web service that decouples NotificationConsumers from Publishers. A
107 notification broker can relieve a Publisher from having to implement message exchanges
108 associated with NotificationProducer; the NotificationBroker takes on the duties of
109 subscription management and distributing Notifications on behalf of the Publisher. It
110 implements NotificationProducer interface. It may implement SubscriptionManager or may
111 delegate the subscription management work to another component.
- 112 • **Must allow for federation of brokers.** It must be possible to build configurations with
113 multiple intermediary broker services in a dynamic fashion. This specification must allow for
114 a variety of broker topology usage patterns. Among other things, these allow for greater
115 scalability and permit sharing of administrative workload.

- 116 • **Must provide runtime metadata:** There must be a mechanism that lets a potential
117 Subscriber discover what elements available for a subscription are provided by a
118 NotificationBroker, and in what formats the subscription for a notification can be made.
- 119 • **Must conform to WS-BaseNotification:** A NotificationBroker must support required
120 message exchanges defined by the [\[WS-BaseNotification\]](#) specification. It must conform to
121 the NotificationProducer and the NotificationConsumer interfaces defined in WS-
122 BaseNotification.
- 123 • **WS-BrokeredNotification must be independent of binding-level details:** Transport
124 protocol details must be orthogonal to the subscription and the delivery of the notifications, so
125 that the specification can be used over a variety of different transports.
- 126 • **Must not exclude non-service producers and subscribers:** WS-BrokeredNotification
127 design must not exclude a non-service entity to deliver a notification message to a
128 NotificationBroker. It must not exclude a NotificationBroker to send a notification message to
129 a non-service consumer.
- 130 • **Must provide publisher registration:** WS-BrokeredNotification must define standard
131 message exchanges for registering a NotificationPublisher with a NotificationBroker.

132 1.1.2 Non-Goals

133 The following topics are outside the scope of the WS-BrokeredNotification specification:

- 134 • **Defining the format of notification payloads:** The data carried in Notification payloads is
135 application-domain specific, and WS-BrokeredNotification does not prescribe any particular
136 format for this data.
- 137 • **Defining any Events or Notifications:** The WS-BrokeredNotification specification does not
138 define any “standard” or “built-in” notification situations, events, or messages.
- 139 • **Defining the means by which NotificationBrokers are discovered by subscribers:** It is
140 beyond the scope of this specification to define the mechanisms for runtime discovery of
141 NotificationBrokers.

142 1.2 Notational Conventions

143 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
144 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be
145 interpreted as described in [RFC 2119].

146 When describing abstract data models, this specification uses the notational convention used by
147 the [XML Infoset]. Specifically, abstract property names always appear in square brackets (e.g.,
148 [some property]).

149 This specification uses a notational convention, referred to as “Pseudo-schemas” in a fashion
150 similar to the WSDL 2.0 Part 1 specification [WSDL 2.0]. A Pseudo-schema uses a BNF-style
151 convention to describe attributes and elements:

- 152 • '?' denotes optionality (i.e. zero or one occurrences),
- 153 • '*' denotes zero or more occurrences,
- 154 • '+' one or more occurrences,
- 155 • '[' and ']' are used to form groups,
- 156 • '|' represents choice.
- 157 • Attributes are conventionally assigned a value which corresponds to their type, as
- 158 defined in the normative schema.

```

159 <!-- sample pseudo-schema -->
160 <element
161     required_attribute_of_type_QName="xs:QName"
162     optional_attribute_of_type_string="xs:string"?>
163   <required_element />
164   <optional_element /> ?
165   <one_or_more_of_these_elements /> +
166   [ <choice_1 /> | <choice_2 /> ] *
167 </element>

```

168 Where there is disagreement between the separate XML schema and WSDL files describing the
 169 messages defined by this specification and the normative descriptive text (excluding any pseudo-
 170 schema) in this document, the normative descriptive text will take precedence over the separate
 171 files. The separate files take precedence over any pseudo-schema and over any schema and
 172 WSDL included in the appendices.

173 1.3 Namespaces

174 The following namespaces are used in this document:

Prefix	Namespace
s	http://schemas.xmlsoap.org/soap/envelope/ OR http://www.w3.org/2003/05/soap-envelope
xsd	http://www.w3.org/2001/XMLSchema
wsa	http://www.w3.org/2005/03/addressing
wsn-b	http://docs.oasis-open.org/wsn/b-1
wsn-br	http://docs.oasis-open.org/wsn/br-1

wsn-bw	http://docs.oasis-open.org/wsn/bw-1
wsn-brw	http://docs.oasis-open.org/wsn/brw-1
wsrf-bf	http://docs.oasis-open.org/wsrf/bf-1
wsrf-bfw	http://docs.oasis-open.org/wsrf/bfw-1

175 **1.4 Fault Definitions**

176 All faults generated by a NotificationBroker, RegisterPublisher, or PublisherRegistrationManager
177 SHOULD be compliant with the WS-BaseFaults [[WS-BaseFaults](#)] specification.

178 All faults defined by this specification MUST use the following URI for the WS-Addressing [action]
179 Message Addressing Property:

180 <http://docs.oasis-open.org/wsn/fault>.

181 **2 Relationship to Other Specifications**

182 This specification builds on the basic notification mechanism defined in [[WS-BaseNotification](#)], by
183 adding the concept of an intermediary NotificationBroker, and describing additional variants on
184 the publisher role. A NotificationBroker takes on the role of both NotificationProducer and
185 NotificationConsumer (as defined in [[WS-BaseNotification](#)]), and its interactions with other
186 NotificationProducers and NotificationConsumers are largely defined by the WS-BaseNotification
187 specification.

188 This means that a NotificationBroker, implemented to conform to this specification, must also
189 conform to [[WS-BaseNotification](#)]. Such a NotificationBroker can deliver notifications to
190 NotificationConsumers that are implemented to conform to [[WS-BaseNotification](#)], and can
191 subscribe to Notifications distributed by NotificationProducers as defined in [[WS-
192 BaseNotification](#)].

193 A NotificationBroker may support hierarchical topics as defined in [[WS-Topics](#)]. By supporting
194 topics, NotificationBroker can manage enterprise messaging systems more efficiently.

195 WS-BrokeredNotification must be composable with other Web services specifications.

196

3 Terminology and Concepts

197

In addition to the terminology and usage described in the WS-BaseNotification specification, the following are the terms defined in this specification:

198

199

Publisher:

200

- A Publisher is an entity that creates Notifications, based upon Situation(s) that it is capable of detecting and translating into Notification artifacts. It does not need to be a Web service.

201

202

203

- A Publisher can register what topics it wishes to publish with a NotificationBroker.

204

205

206

- A Publisher MAY be a Web service that implements the message exchanges associated with the NotificationProducer interface, in which case it also distributes the Notifications to the relevant NotificationConsumers.

207

208

209

210

- If a Publisher does not implement the message exchanges associated with NotificationProducer, then it is not required to support the Subscribe request message and does not have to maintain knowledge of the NotificationConsumers that are subscribed to it; a NotificationBroker takes care of this on its behalf.

211

NotificationBroker:

212

213

214

215

216

- A NotificationBroker is an intermediary Web service that decouples NotificationConsumers from Publishers. A NotificationBroker is capable of subscribing to notifications, either on behalf of NotificationConsumers, or for the purpose of messaging management. It is capable disseminating notifications on behalf of Publishers to NotificationConsumers.

217

218

- A NotificationBroker aggregates NotificationProducer, NotificationConsumer, RegisterPublisher, and CreatePullPoint interfaces.

219

220

- Acting as an intermediary, a NotificationBroker provides additional capabilities to the base NotificationProducer interface:

221

222

223

224

- It can relieve a Publisher from having to implement message exchanges associated with NotificationProducer; the NotificationBroker takes on the duties of a SubscriptionManager (managing subscriptions) and NotificationProducer (distributing Notifications) on behalf of the Publisher.

225

226

- It can reduce the number of inter-service connections and references, if there are many Publishers and many NotificationConsumers.

227

228

- It can act as a finder service. Potential Publishers and Subscribers can in effect find each other by utilizing a common NotificationBroker.

229

230

- It can provide anonymous Notification, so that the Publishers and the NotificationConsumers need not be aware of each other's identity.

231

232

- An implementation of a NotificationBroker may provide additional added-value function that is beyond the scope of this specification, for example, logging Notifications, or

- 233 transforming Topics and/or Notification content. Additional function provided by a
234 NotificationBroker can apply to all Publishers that utilize it.
- 235 • It may be the factory for Subscription resources or it may delegate the subscription
236 factory to another component.
 - 237 • A NotificationBroker provides publisher registration functions.
 - 238 • A NotificationBroker may subscribe and disseminate messages that are not WS-
239 Notification conforming.
- 240 **PublisherRegistration:**
- 241 • PublisherRegistration is a resource. A PublisherRegistration represents the relationship
242 between a Publisher and a NotificationBroker, in particular, which topic(s) the publisher is
243 permitted to publish to.
 - 244 • A PublisherRegistration resource is created when a Publisher sends the
245 RegisterPublisher request message to a NotificationBroker and the NotificationBroker
246 succeeds in processing the registration.
 - 247 • PublisherRegistration resources can be manipulated by messages sent to a
248 PublisherRegistrationManager Web service.
- 249 **RegisterPublisher:**
- 250 • A RegisterPublisher is a Web service that implements the message exchanges
251 associated with the RegisterPublisher interface. A PublisherRegistration resource is
252 created as a result of a RegisterPublisher request to a NotificationBroker.
- 253 **PublisherRegistrationManager:**
- 254 • A PublisherRegistrationManager is a Web service that implements the message
255 exchanges associated with the PublisherRegistrationManager interface.
 - 256 • A publisher registration resource can be manipulated through
257 PublisherRegistrationManager message exchanges.
 - 258 • A PublisherRegistrationManager provides services that allow a service requestor to query
259 and manipulate PublisherRegistration resources that it manages.
 - 260 • A PublisherRegistrationManager is subordinate to the NotificationBroker, and MAY be
261 implemented by the NotificationBroker service provider. However WS-
262 BrokeredNotification permits it to be implemented by a separate service provider, should
263 an implementer so desire.
- 264 **Demand-Based Publishing:**
- 265 • Some Publishers may be interested in knowing whether they have any Subscribers or
266 not, since producing a Notification may be a costly process. Such Publishers can register
267 with the NotificationBroker as a Demand-Based Publisher.
 - 268 • Demand-Based Publishers implement message exchanges associated with the
269 NotificationProducer interface.

- 270
- 271
- 272
- 273
- The NotificationBroker subscribes to the Demand-Based Publisher. When the NotificationBroker knows that there are no Subscribers for the Notifications from a Demand-Based Publisher it pauses its Subscription with that Publisher; when it knows that there are some Subscribers, it resumes the Subscription.
- 274
- This way the Demand-Based Publisher does not need to produce messages when there are no Subscribers, however a Demand-Based Publisher is only required to support a single Subscriber on any given Topic, and so can delegate the management of multiple Subscribers, the delivery to multiple NotificationConsumers, and other related issues (for example security) to the NotificationBroker.
- 275
- 276
- 277
- 278

279 **4 Publishing**

280 There are three distinct stages in the Notification process

- 281
- Observation of the Situation and its noteworthy characteristics;
- 282
- Creation of the Notification artifact that captures the noteworthy characteristics of the Situation; and
- 283
- Distribution of copies of the Notification to zero or more interested parties.
- 284

285 Stages 1 and 2 happen largely outside of the scope of the WS-Notification architecture; this
286 specification does not restrict the means by which these stages must occur. We refer to an entity
287 that performs stages 1 and 2 as a Publisher,

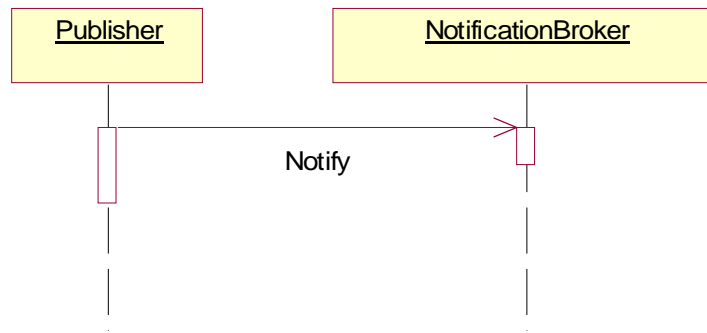
288 However, the WS-Notification family of specifications does specify how dissemination of
289 messages SHOULD occur. There are two dominant patterns by which Notifications are
290 disseminated in WS-Notification: direct and brokered.

291 In the direct case, the publishing Web service implements message exchanges associated with
292 the NotificationProducer interface; it is responsible for accepting Subscribe messages and
293 sending Notifications to interested parties. The implementer of this Web service can choose to
294 program this behavior or delegate to specialized implementations of the Subscribe and
295 Notification delivery behavior. This case is addressed by the WS-BaseNotification specification
296 [[WS-BaseNotification](#)].

297 In the brokered case, an intermediary - a NotificationBroker - is responsible for disseminating
298 messages produced by one or more Publishers to zero or more NotificationConsumers.

299 There are three patterns associated with the relationship between the Publisher and the
300 NotificationBroker: simple publishing, broker initiated publishing, and demand-based publishing.

301 The following figure illustrates simple publishing:



302

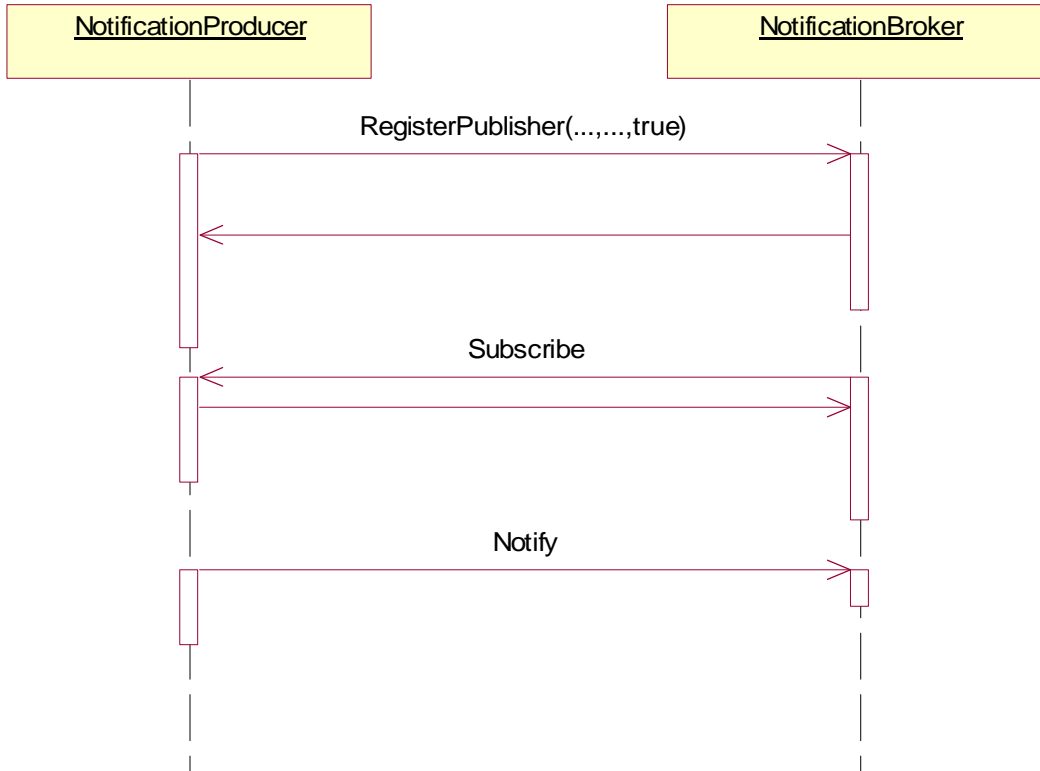
303 In the simple publishing scenario, the Publisher entity is responsible only for the core Publisher
 304 functions - observing the Situation and formatting the Notification artifact that describes the
 305 Situation. The dissemination step occurs when the Publisher sends the Notify message to the
 306 NotificationBroker.

307 In the broker initiated publishing pattern, the role of the Publisher is played by a Web service that
 308 implements NotificationProducer. The act of observing the Situation and formatting the
 309 Notification happens within the implementation logic of the NotificationProducer itself. The
 310 Notification is disseminated by the NotificationProducer sending the Notify message to a
 311 NotificationBroker. The Notification may also be disseminated by sending the Notify message to
 312 any NotificationConsumer that are subscribing to the NotificationProducer.

313 Note: in either of the above two cases, the NotificationBroker MAY require the Publisher to
 314 register with it prior to sending the Notify message. For example, if the broker wishes to control
 315 who can publish to a given Topic, it can perform an access control check during this registration.
 316 However a NotificationBroker MAY choose to allow Publishers to publish without pre-registration,
 317 if it so chooses.

318 The last pattern, the demand-based pattern, requires the Publisher to be a NotificationProducer,
 319 and thereby accept the Subscribe message. Demand-based publication is intended for use in
 320 cases where the act of observing the Situation or the act of formatting the Notification artifact
 321 might be expensive to perform, and therefore should be avoided if there are no interested parties
 322 for that Notification. To use this pattern, the Publisher must register with the NotificationBroker,
 323 using the registration to express the intent to provide demand-based publishing only. Based upon
 324 this style of registration, the NotificationBroker sends the Subscribe message to the Publisher
 325 (recall: in this situation the Publisher must implement the message exchanges associated with
 326 the NotificationProducer interface).

327 Furthermore, the NotificationBroker is expected to pause its Subscription whenever it has no



328 active Subscribers for the information provided by the Publisher. When the NotificationBroker
329 does have active Subscribers, it is obliged to resume its Subscription to the Publisher.

330 5 NotificationBroker Interface

331 The NotificationBroker interface defines a standard set of message exchanges to describe a
332 message broker, providing an intermediary between Publishers and Subscribers on a collection
333 of Topics, similar to a traditional Message Oriented Middleware model.

334 NotificationBroker MAY be a WS-Resource, and if it is, it MUST support the required message
335 exchanges defined by the WS-ResourceProperties specification [[WS-ResourceProperties](#)] and
336 MAY support the optional message exchanges defined by WS-ResourceProperties.

337 A NotificationBroker MUST also support message exchanges and Resource Property elements
338 defined by the following interfaces:

- 339 • NotificationProducer

- 340 • NotificationConsumer
- 341 • RegisterPublisher
- 342 • CreatePullPoint

343 The NotificationBroker portType aggregates the four portTypes and is not the only way to
344 implement a broker. A distributed broker implementation can be achieved by hosting
345 NotificationProducer, NotificationConsumer, CreatePullPoint, or RegisterPublisher portTypes at
346 one or more physical endpoints.

347 The NotificationBroker does not specify any subscription durability or continuity.
348 NotificationBrokers SHOULD advertise their durability or reliability features, either through
349 policies or other means.

350 NotificationBrokers MAY offer flow control and MAY implement Pull-Style notifications. If so,
351 NotificationBrokers SHOULD advertise these features, either through policies or other means.

352

353 5.1 NotificationBroker Resource Properties

354 In addition to the message exchanges described in this specification, a NotificationBroker MAY
355 also support the required message exchanges defined in the WS-ResourceProperties
356 specification and MAY support the optional message exchanges defined in the WS-
357 ResourceProperties specification. If it does so, the Resource Properties document defined by the
358 NotificationBroker MUST include references to resource properties defined in
359 NotificationProducer and NotificationConsumer, and also MUST include a reference to the
360 following resource property element:

```
361 ...  
362     targetNamespace="http://docs.oasis-open.org/wsn/br-1">  
363 ...  
364     <xsd:element name="RequiresRegistration" type="xsd:boolean" />  
365 ...
```

366 Furthermore, this reference MUST reflect the minOccurs and maxOccurs properties as follows:

```
367 <xsd:element ref="wsn-br:RequiresRegistration"  
368             minOccurs="1" maxOccurs="1" />
```

369 This resource property element is further constrained as follows:

370 /wsn-br:RequiresRegistration

371 The value is "true" if the NotificationBroker requires a publisher to register (see 6.1)
372 before sending it a Notify (i.e. publish) message on a Topic. The default is "false".

373 5.2 Notify

374 The NotificationBroker MUST support the Notify message exchange from the
375 NotificationConsumer interface [[WS-BaseNotification](#)], with the following clarifications/restrictions:

376 A Publisher sends a Notify message to a NotificationBroker in order to publish a Notification on a
377 given Topic. As a result of the Publisher sending this message, Notifications are delivered to all
378 NotificationConsumers subscribed on the given Topic. For some Topics (those that require a
379 Publisher to pre-register), the sender must be a registered Publisher in order to successfully
380 publish a Notification on the given Topic (see 6.1).

381 **5.3 Subscribe**

382 The NotificationBroker MUST support the Subscribe message exchange from the
383 NotificationProducer interface [[WS-BaseNotification](#)]. A NotificationBroker MAY support any
384 TopicExpression dialect.

385 A NotificationBroker is capable of routing or producing a sequence of zero or more Notifications.
386 A Subscriber can register the interest of a NotificationConsumer to receive a subset of this
387 sequence. A Subscriber sends a Subscribe message to a NotificationBroker in order to register
388 this interest.

389 If the processing of a Subscribe message is successful, the NotificationBroker MUST produce a
390 response message, as described in WS-BaseNotification, containing an endpoint reference to a
391 Subscription resource representing a Subscription created as a result of processing the
392 Subscribe request. Otherwise, the NotificationBroker must fault. WS-BaseNotification defines a
393 set of these faults.

394 **5.4 GetCurrentMessage**

395 The NotificationBroker MUST support the GetCurrentMessage message exchange from the
396 NotificationProducer interface [[WS-BaseNotification](#)].

397 As defined in WS-BaseNotification, in response to a GetCurrentMessage message, the
398 NotificationBroker MAY return the last Notification published on a given Topic. This is a non-
399 destructive read, allowing a newly-subscribed NotificationConsumer to get the last Notification
400 that other NotificationConsumers have received.

401 **5.5 RegisterPublisher**

402 The NotificationBroker MUST support the RegisterPublisher message exchange from the
403 RegisterPublisher interface.

404 A Publisher can register its interest to publish messages through the NotificationBroker by
405 sending a RegisterPublisherRequest. The NotificationBroker is responsible for managing the
406 registration, and sending a RegisterPublisherResponse to the Publisher if the registration process
407 succeeds. Otherwise, the NotificationBroker MUST fault. These message exchanges are further
408 specified in the following Section 6.

409 **5.6 CreatePullPoint**

410 The NotificationBroker MUST support the CreatePullPoint interface. The CreatePullPoint
411 interface standardizing the means by which a PullPoint resource is created. If a requestor wishes

412 to create a new PullPoint resource, it MUST send a CreatePullPoint request to the
413 NotificationBroker.
414 The NotificationBroker MAY support pull-style notification and attempt to create a PullPoint
415 resource upon receiving a CreatePullPoint request. The NotificationBroker does not define
416 additional constraints to its usage of the CreatePullPoint interface.
417 If the NotificationBroker does not support pull-style notification, it MUST response with the
418 following fault upon receiving a CreatePullPoint request:
419 PullNotificationNotSupportedFault
420

- The NotificationBroker does not support pull-style notification.

421 **6 RegisterPublisher Interface**

422 The RegisterPublisher interface contains message exchanges for publisher registration.
423 NotificationBroker implements the RegisterPublisher interface and is responsible for publisher
424 registration. A NotificationBroker may reject processing certain publisher registrations for reasons
425 such as lacking of authorization.

426 **6.1 RegisterPublisher**

427 The RegisterPublisher message is used by the Publisher to confirm its ability to publish on a
428 given Topic or set of Topics. If an entity wishes to register a publisher, it MUST send a
429 RegisterPublisher request message to the NotificationBroker. The format of the RegisterPublisher
430 request message is:

```
431 ...  
432 <wsn-br:RegisterPublisherRequest>  
433   <wsn-br:PublisherReference>  
434     wsa:EndpointReference  
435   </wsn-br:PublisherReference>?  
436   <wsn-br:Topic dialect = "xsd:anyURI">  
437     {any}  
438   </wsn-br:Topic> *  
439   <wsn-br:Demand>  
440     xsd:Boolean  
441   </wsn-br:Demand>?  
442   <wsn-br:InitialTerminationTime>  
443     xsd:dateTime  
444   </wsn-br:InitialTerminationTime>?  
445   {any} *  
446 </wsn-br:RegisterPublisherRequest>  
447 ...
```

448 The [WS-Addressing](#) [action] Message Addressing Property MUST contain the URI
449 <http://docs.oasis-open.org/wsn/brw-1/RegisterPublisher/RegisterPublisherRequest>.

450

451 The components of the RegisterPublisher request message are further described as follows:

452 /wsn-br:PublisherReference

453 An OPTIONAL endpoint reference element from WS-Addressing [WS-Addressing], used
454 to identify an entity that wishes to become a Publisher. This component MUST appear if
455 the /wsn-br:Demand component has value "true". If this component is missing, the
456 Publisher is either not a Web service, or does not wish to receive messages from the
457 NotificationBroker.

458 /wsn-br:Topic

459 A set of TopicExpressions that identifies one or more Topics. If included, the given
460 Publisher is registered to publish only on the set of Topics identified by this component. If
461 this is missing the Publisher is registered to publish on any Topic supported by the
462 NotificationBroker.

463 /wsn-br:Demand

464 A Boolean element with the default value "false". If its value is "true", then the intent of the
465 Publisher is to use a demand-based model from the NotificationBroker (see Section 4). In
466 this case, the NotificationBroker must observe the rules associated with demand-based
467 publishing, including establishing a Subscription with the Publisher on those Topics and
468 pausing/resuming those Subscriptions as the NotificationBroker receives Subscriptions
469 for those Topics.

470 /wsn-br:InitialTerminationTime

471 This component contains the service requestor's suggestion for the initial termination
472 time of the PublisherRegistration resource being created. This time is relative to the time
473 source used by the NotificationBroker. If the NotificationBroker is unable or unwilling to
474 set the TerminationTime to the given value or greater, then the RegisterPublisher request
475 MUST fault. If the value is not "in the future" relative to the current time as known by the
476 NotificationBroker, the RegisterPublisher request MUST fault. The use of the xsi:nil
477 attribute with value "true" indicates there is no scheduled termination time requested for
478 the RegisterPublisher. If the element does not include the time zone designation, the
479 value of the element MUST be interpreted as universal time (UTC).

480 The publisher should take care when choosing a value for InitialTerminationTime, and
481 any subsequent values that modify the TerminationTime property of the publisher
482 registration. It is RECOMMENDED that the publisher choose termination time values that
483 are significantly (several magnitude) greater than the network latency expected in the
484 interaction between the publisher and the broker. In so doing, the designer avoids
485 undesirable results, such as the termination time having expired prior to the receipt of the
486 published message. The WS-Resource Lifetime specification [WS-ResourceLifetime]
487 (Section 6.1 Regarding time) contains further suggestions on how designers should
488 reason about time values in a WS-Resource Lifetime application.

489 If this component is not included, the initial value of the TerminationTime resource
490 property is dependent on the implementation of the NotificationBroker.

491 / wsn-br:RegisterPublisherRequest/{any}

492 The RegisterPublisherRequest message allows for open content, in order to
493 accommodate elements that may be needed by extensions built on WS-
494 BrokeredNotification.

495 If a /wsn-br:Topic component is included in the message, the NotificationBroker MUST register
496 the Web service specified by the /wsn-br:PublisherReference component as a Publisher on the
497 set of Topics identified by the /wsn-br:Topic component. If for any reason the registration fails, the
498 NotificationBroker MUST fault.

499 As part of the processing of a RegisterPublisher request, the NotificationBroker creates a
500 PublisherRegistration resource representing the registration. A new resource is created
501 regardless of whether the same Publisher has previously registered with the NotificationBroker.
502 The NotificationBroker returns a PublisherRegistrationReference in the response to the
503 RegisterPublisher request. This PublisherRegistrationReference is a WS-Addressing endpoint
504 reference and includes the address of a PublisherRegistrationManager service and a reference
505 property identifying the PublisherRegistration resource.

506 If the NotificationBroker accepts the RegisterPublisher request message, it must respond with a
507 message of the following form:

```
508 ...  
509 <wsn-br:RegisterPublisherResponse>  
510   <wsn-br:PublisherRegistrationReference>  
511     <wsa:Address>  
512       Address of PublisherRegistration Manager  
513     </wsa:Address>  
514     <wsa:ReferenceParameters>  
515       PublisherRegistration Identifier  
516     </wsa:ReferenceParameters>  
517     ...  
518   </wsn-br:PublisherRegistrationReference>  
519 </wsn-br:RegisterPublisherResponse>  
520 ...
```

521 The WS-Addressing [action] Message Addressing Property MUST contain the URI

522 <http://docs.oasis-open.org/wsn/brw-1/RegisterPublisher/RegisterPublisherResponse>

523 The components of the RegisterPublisher response message are further described as follows:

524 /wsn-br:PublisherRegistrationReference

525 A WS-Addressing endpoint reference to the PublisherRegistration resource created by
526 the RegisterPublisher request message.

527 If the NotificationBroker does not succeed in responding to the RegisterPublisher request
528 message with the RegisterPublisherResponse message, then it MUST send a fault. The
529 NotificationBroker MUST fault if it rejects the publisher registration. This specification defines the
530 following faults associated with failure to process the RegisterPublisher request message:

531

532

- 533 ResourceUnknownFault
- 534 • The NotificationBroker is acting as a WS-Resource, and the resource identified in the
535 message (which follows the WS-Resource Access Pattern) is not known to the Web
536 service. This fault is specified by the WS-Resource [WS-Resource] specification.
- 537 InvalidTopicExpressionFault
- 538 • The TopicExpression presented in the request message is invalid.
- 539 TopicNotSupportedFault
- 540 • The TopicExpression does not match any Topic supported by the NotificationBroker.
- 541 PublisherRegistrationRejectedFault
- 542 • The publisher registration is rejected by the NotificationBroker.
- 543 PublisherRegistrationFailedFault
- 544 • The publisher registration process has failed.

545 **6.1.1 Example SOAP Encoding of the RegisterPublisher Message** 546 **Exchange**

547 The following is a non-normative example of a RegisterPublisher request message using SOAP:

```
548 <s:Envelope ... >
549   <s:Header>
550     <wsa:Action>
551       http://docs.oasis-open.org/wsn/brw-1/RegisterPublisher/
552 RegisterPublisherRequest
553     </wsa:Action>
554     ...
555   </s:Header>
556   <s:Body>
557     <wsn-br:RegisterPublisher>
558       <wsn-br:PublisherReference>
559         <wsa:Address>
560           http://www.producer.org/PublisherEndpoint
561         </wsa:Address>
562         <wsa:ReferenceParameters>
563           <npex: NPResourceDisambiguator>
564             uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
565           </npex: NPResourceDisambiguator>
566         </wsa:ReferenceParameters>
567       </wsn-br:PublisherReference>
568       <wsn-br:Topic Dialect="http://docs.oasis-open.org/wsn/t-
569 1/SimpleTopicExpression">
570         npex:SomeTopic
571       </wsn-br:Topic>
572       <wsn-br:Demand>
573         true
574       </wsn-br:Demand>
```

```
575     <wsn-br:InitialTerminationTime>
576         2003-12-25T00:00:00.000000Z
577     </wsn-br:InitialTerminationTime>
578 </wsn-br:RegisterPublisher>
579 </s:Body>
580 </s:Envelope>
```

581 The following is a non-normative example of a RegisterPublisher response message using
582 SOAP:

```
583 <s:Envelope ... >
584   <s:Header>
585     <wsa:Action>
586       http://docs.oasis-open.org/wsn/brw-
587 1/RegisterPublisher/RegisterPublisherResponse
588     </wsa:Action>
589     ...
590   </s:Header>
591   <s:Body>
592     <wsn-br:RegisterPublisherResponse>
593       <wsn-br:PublisherRegistrationReference>
594         <wsa:Address>
595           http://www.producer.org/PublisherRegisterEndpoint
596         </wsa:Address>
597         <wsa:ReferenceParameters>
598           <npex:NPubResourceId>
599             uuid:95fefeb3-f37d-5dfe-44fe-221d9fceed99
600           </npex:NPubResourceId>
601         </wsa:ReferenceParameters>
602       </wsn-br:PublisherRegistrationReference>
603     </wsn-br:RegisterPublisherResponse>
604   </s:Body>
605 </s:Envelope>
```

606 7 PublisherRegistrationManager Interface

607 The PublisherRegistrationManager interface defines message exchanges to manipulate
608 PublisherRegistration resources. The PublisherRegistrationManager MAY be a WS-Resource,
609 and if it is, request messages defined in this specification MUST follow the WS-Resource Access
610 Pattern defined by [WS-Resource] and the PublisherRegistrationManager WS-Resource MUST
611 support the immediate termination interface defined by WS-RF Resource Lifetime and it MAY
612 support the scheduled termination interface defined by WS-RF Resource Lifetime.

613 If the PublisherRegistrationManager does not respond to a request message with a respond
614 message defined in this specification, then it MUST send a fault. The WS-ResourceProperties
615 and WS-ResourceLifetime define some of these fault messages.

616 7.1 PublisherRegistration Resource Properties

617 In addition to the message exchanges described in this specification, a
618 PublisherRegistrationManager MAY also support the required message exchanges defined in the
619 WS-ResourceProperties specification and MAY support the optional message exchanges defined
620 in the WS-ResourceProperties specification. If it does so, the Resource Properties document
621 defined by the PublisherRegistrationManager MUST also include references to the following
622 resource property elements:

```
623 .....  
624     targetNamespace="http://docs.oasis-open.org/wsn/br-1"  
625     ...  
626     <xsd:element name="PublisherReference"  
627         type="wsa:EndpointReference" />  
628     <xsd:element name="Topic" type="wsn-b:TopicExpressionType" />  
629     <xsd:element name="Demand" type="xsd:boolean" />  
630     <xsd:element name="CreationTime" type="xsd:dateTime" />  
631     ...
```

632 Furthermore, these references MUST reflect the minOccurs and maxOccurs properties as
633 follows:

```
634     <xsd:element ref="wsn-br:PublisherReference"  
635         minOccurs="0" maxOccurs="1" />  
636     <xsd:element ref="wsn-br:Topic"  
637         minOccurs="0" maxOccurs="unbounded" />  
638     <xsd:element ref="wsn-br:Demand"  
639         minOccurs="1" maxOccurs="1" />  
640     <xsd:element ref="wsn-br:CreationTime"  
641         minOccurs="0" maxOccurs="1" />
```

642 These resource property elements are further constrained as follows:

643 /wsn-br:PublisherReference, /wsn-br:Topic, and /wsn-br:Demand

644 These elements are defined in the description of the RegisterPublisher request message
645 (see 6.1).

646 /wsn-br:CreationTime

647 Indicates the date and time at which the PublisherRegistration was created. This is an
648 optional component, supporting resource constrained devices which cannot associate a
649 creation time with PublisherRegistration resources they create.

650 If PublisherRegistrationManager is a WS-Resource, the following resource properties MAY be
651 modified by the requestor, by sending a SetResourceProperties request message as defined in
652 the WS-ResourceProperties specification:

- 653 • /wsn-br:TopicExpression and /wsn-br:Demand
 - 654 ○ Note: /wsn-br:Demand may not take the value "true" if there is no /wsn-
655 br:PublisherReference resource property element in the resource property
656 document.

657 **7.2 Destroy**

658 The PublisherRegistrationManager interface provides a destroy operation, providing a means by
659 which a requestor can terminate the publisher registration manager resource. To terminate
660 PublisherRegistrationManager resource, a requestor MUST send a Destroy request message to
661 the PublisherRegistrationManager. The Destroy request message has the following form:

```
662 <wsn-br:DestroyRequest>  
663   {any} *  
664 </wsn-br:DestroyRequest>  
665  
666
```

667 The WS-Addressing [action] Message Addressing Property MUST contain the URI
668 <http://docs.oasis-open.org/wsn/brw-1/PublisherRegistrationManager/DestroyRequest>.

669 The Destroy request message allows for open content and contains an extension component
670 / wsn-br:DestroyRequest/{any}.

671 Upon receipt of the Destroy request, the PublisherRegistrationManager MUST attempt to destroy
672 itself. If the Destroy request message is successfully processed, the
673 PublisherRegistrationManager MUST respond with the following message:

```
674 <wsn-br:DestroyResponse/>  
675  
676
```

677 The WS-Addressing [action] Message Addressing Property MUST contain the URI
678 <http://docs.oasis-open.org/wsn/brw-1/PublisherRegistrationManager/DestroyResponse>.

679 If the PublisherRegistrationManager does not respond to the Destroy request message with the
680 DestroyResponse message, then it MUST send a fault. This specification defines the following
681 faults associated with failure to process the Destroy request message:

682 ResourceUnknownFault

- 683 • The PublisherRegistrationManager is a WS-Resource, and the resource identified in the
684 message is not known to the Web service. This fault is specified by the WS-Resource
685 [WS-Resource] specification.

686 ResourceNotDestroyedFault

- 687 • The PublisherRegistrationManager was unable to destroy the
688 PublisherRegistrationManager resource for some reason.

689 **7.2.1 Example SOAP Encoding of the Destroy Message Exchange**

690 The following is a non-normative example of a Destroy request message using SOAP:

```
691 <s:Envelope ... >  
692   <s:Header>
```

```
693     <wsa:Action>
694         http://docs.oasis-open.org/wsn/brw-
695 1/PublisherRegistrationManager/DestroyRequest
696     </wsa:Action>
697     ...
698 </s:Header>
699 <s:Body>
700     <wsn-br:DestroyRequest/>
701 </s:Body>
702 </s:Envelope>
```

703 The following is a non-normative example of a Destroy response message using SOAP:

```
704 <s:Envelope ... >
705 <s:Header>
706     <wsa:Action>
707         http://docs.oasis-open.org/wsn/brw-
708 1/PublisherRegistrationManager/DestroyResponse
709     </wsa:Action>
710     ...
711 </s:Header>
712 <s:Body>
713     <wsn-br:DestroyResponse/>
714 </s:Body>
715 </s:Envelope>
```

716

717 **8 Security Considerations**

718 Baseline security considerations for WS-Notification are discussed in WS-BaseNotification
719 specification. This section only covers additional broker specific security measurements.

720 **8.1 Securing PublisherRegistration**

721

722 In addition to the security policies for Notification process and Subscription process, WS-
723 BrokeredNotification should provide policies such that:

- 724 1. only authorized Publishers can register with a NotificationBroker
- 725 2. only messages of the authorized Publishers and of registered topics, can be accepted by
726 a NotificationBroker
- 727 3. only authorized principals can modify or delete PublisherRegistration resource

728 Given that WS-BrokeredNotification may implement WS-ResourceProperties and WS-
729 ResourceLifetime, the security considerations outlined in these specifications need to be taken
730 into account where appropriate. Authorization policies for those Resource Properties should be

731 put in place so that the implications of providing the state information (through
732 GetResourceProperty request messages) or through notification of state change and modification
733 of the resource properties (through SetResourceProperty request messages), are taken into
734 account.

735 A NotificationBroker can support the security measurements of NotificationProducers and
736 NotificationConsumers mentioned in WS-BaseNotification. Acting as an intermediary,
737 NotificationBroker MAY also provide convenience to security management, including but not
738 limited to:

- 739 • Controlling who can publish on a topic at publisher registration time
- 740 • Refusing to relay messages from unauthorized publishers
- 741 • Imposing security measurements on all messaging routing through the broker
- 742 • Providing convenience in messaging security management based on topics.

743 NotificationBrokers SHOULD advertise, whether through policy assertions or other means, what
744 security measures they take.

745

746 9 References

747 9.1 Normative

748	[RFC2119]	S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt , IETF RFC 2119, March 1997.
749		
750		
751	[XML]	http://www.w3.org/TR/REC-xml
752	[XML-Infoset]	http://www.w3.org/TR/xml-infoset/
753	[XPath]	http://www.w3.org/TR/xpath
754	[WS-Addressing]	http://www.w3.org/TR/ws-addr-core
755	[WS-BaseNotification]	http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-cd-01.pdf
756		
757	[WS-Topics]	http://docs.oasis-open.org/wsn/wsn-WS-Topics-1.3-cd-01.pdf
758	[WS-Resource]	http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-cd-01.pdf
759		
760	[WS-ResourceLifetime]	http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-cd-01.pdf
761		
762	[WS-ResourceProperties]	http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-cd-01.pdf
763		
764	[WS-BaseFaults]	http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-cd-01.pdf
765		
766		

767 9.2 Non-Normative

768	[SOAP 1.2]	http://www.w3.org/TR/soap12-part1/
769	[WS-Security]	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
770		
771		

772 **Appendix A. Acknowledgments**

773 The following individuals were members of the committee during the development of this
774 specification:

775 Sid Askary, Individual, Fred Carter, AmberPoint, Martin Chapman, Oracle, Dave Chappell, Sonic
776 Software, Rick Cobb, KnowNow, Ugo Corda, SeeBeyond Technology Corporation, John Fuller,
777 Individual, Stephen Graham, IBM, David Hull, Tibco, Hideharu Kato, Hitachi, Lily Liu,
778 webMethods, Tom Maguire, IBM, Susan Malaika, IBM, Samuel Meder, Argonne National
779 Laboratory, Bryan Murray, Hewlett-Packard, Peter Niblett, IBM, Sanjay Patil, SAP, Mark Peel,
780 Novell, Matt Roberts, IBM, Igor Sedukhin, Computer Associates, David Snelling, Fujitsu, Latha
781 Srinivasan, Hewlett-Packard, William Vambenepe, Hewlett-Packard, Kirk Wilson, Computer
782 Associates.

783 Special thanks to the Global Grid Forum's Open Grid Services Infrastructure working group,
784 which defined the OGSi v1.0 specification which was a large inspiration for the ideas expressed
785 in this specification.

786 In addition, the following people who are not members of the committee made contributions to
787 this specification:

788 Tim Banks (IBM), Nick Butler (IBM), Doug Davis (IBM), John Dinger (IBM), Don Ferguson (IBM),
789 Jeff Frey (IBM), Andreas Koepfel (SAP), Heather Kreger (IBM), Amy Lewis (TIBCO Software),
790 Kevin Liu (SAP), Nataraj Nagaratnam (IBM), Martin Nally (IBM), Jeff Nick (IBM), Jay Parikh
791 (Akamai Technologies), Claus von Riegen (SAP), Rick Rineholt (IBM), John Rofrano (IBM),
792 Shivajee Samdarshi (TIBCO Software), Igor Sedukhin (Computer Associates), Eugène
793 Sindambiwe (SAP), Jay Unger (IBM), Bill Weihl (Akamai Technologies), Mark Weitzel (IBM), Dan
794 Wolfson (IBM).

795 Appendix B. XML Schema

796 The XML types and elements used in WS-BrokeredNotification are defined in the following XML
797 Schema

```
798 <?xml version="1.0" encoding="UTF-8"?>
799 <!--
800 OASIS takes no position regarding the validity or scope of any
801 intellectual property or other rights that might be claimed to pertain
802 to the implementation or use of the technology described in this
803 document or the extent to which any license under such rights might or
804 might not be available; neither does it represent that it has made any
805 effort to identify any such rights. Information on OASIS's procedures
806 with respect to rights in OASIS specifications can be found at the
807 OASIS website. Copies of claims of rights made available for
808 publication and any assurances of licenses to be made available, or the
809 result of an attempt made to obtain a general license or permission for
810 the use of such proprietary rights by implementors or users of this
811 specification, can be obtained from the OASIS Executive Director.
812
813 OASIS invites any interested party to bring to its attention any
814 copyrights, patents or patent applications, or other proprietary rights
815 which may cover technology that may be required to implement this
816 specification. Please address the information to the OASIS Executive
817 Director.
818
819 Copyright (C) OASIS Open (2005). All Rights Reserved.
820
821 This document and translations of it may be copied and furnished to
822 others, and derivative works that comment on or otherwise explain it or
823 assist in its implementation may be prepared, copied, published and
824 distributed, in whole or in part, without restriction of any kind,
825 provided that the above copyright notice and this paragraph are
826 included on all such copies and derivative works. However, this
827 document itself may not be modified in any way, such as by removing the
828 copyright notice or references to OASIS, except as needed for the
829 purpose of developing OASIS specifications, in which case the
830 procedures for copyrights defined in the OASIS Intellectual Property
831 Rights document must be followed, or as required to translate it into
832 languages other than English.
833
834 The limited permissions granted above are perpetual and will not be
835 revoked by OASIS or its successors or assigns.
836
837 This document and the information contained herein is provided on an
838 "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
839 INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
840 INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
841 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
```

```

842 -->
843
844 <xsd:schema
845   xmlns="http://www.w3.org/2001/XMLSchema"
846   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
847   xmlns:wsa="http://www.w3.org/2005/03/addressing"
848   xmlns:wsn-br="http://docs.oasis-open.org/wsn/br-1"
849   xmlns:wsn-b="http://docs.oasis-open.org/wsn/b-1"
850   xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-1"
851   targetNamespace="http://docs.oasis-open.org/wsn/br-1"
852   elementFormDefault="qualified"
853   attributeFormDefault="unqualified">
854
855 <!-- ===== Imports ===== -->
856
857   <xsd:import namespace="http://www.w3.org/2005/03/addressing"
858             schemaLocation="http://www.w3.org/2005/03/addressing"/>
859
860   <xsd:import namespace="http://docs.oasis-open.org/wsrf/bf-1"
861             schemaLocation="http://docs.oasis-open.org/wsrf/bf-1"/>
862
863   <xsd:import namespace="http://docs.oasis-open.org/wsn/b-1"
864             schemaLocation="http://docs.oasis-open.org/wsn/b-1"/>
865
866 <!-- ===== Resource Properties for NotificationBroker ===== -->
867   <xsd:element name="RequiresRegistration" type="xsd:boolean"/>
868
869 <!-- ===== Resource Properties for PublisherRegistration ===== -->
870   <xsd:element name="PublisherReference"
871               type="wsa:EndpointReferenceType"/>
872   <xsd:element name="Topic"
873               type="wsn-b:TopicExpressionType"/>
874   <xsd:element name="Demand"
875               type="xsd:boolean"/>
876   <xsd:element name="CreationTime"
877               type="xsd:dateTime"/>
878   <xsd:element name="NotificationBrokerRP">
879     <xsd:complexType>
880       <xsd:sequence>
881         <!-- From NotificationProducer -->
882         <xsd:element ref="wsn-b:TopicExpression"
883                       minOccurs="0" maxOccurs="unbounded" />
884         <xsd:element ref="wsn-b:FixedTopicSet"
885                       minOccurs="0" maxOccurs="1" />
886         <xsd:element ref="wsn-b:TopicExpressionDialect"
887                       minOccurs="0" maxOccurs="unbounded" />
888
889         <!-- NotificationBroker specific -->
890         <xsd:element ref="wsn-br:RequiresRegistration"
891                       minOccurs="1" maxOccurs="1" />
892       </xsd:sequence>

```

```

893         </xsd:complexType>
894     </xsd:element>
895
896     <!-- ===== Resource Properties for PublisherRegistration ===== -->
897     <xsd:element name="PublisherRegistrationRP">
898         <xsd:complexType>
899             <xsd:sequence>
900                 <!-- From WS-ResourceLifetime ScheduledResourceTermination -->
901                 <xsd:element ref="wsn-b:CurrentTime"
902                     minOccurs="0" maxOccurs="1" />
903                 <xsd:element ref="wsn-b:TerminationTime"
904                     minOccurs="1" maxOccurs="1" />
905
906                 <!-- PublisherRegistration specific -->
907                 <xsd:element ref="wsn-br:PublisherReference"
908                     minOccurs="0" maxOccurs="1" />
909                 <xsd:element ref="wsn-br:Topic"
910                     minOccurs="0" maxOccurs="unbounded" />
911                 <xsd:element ref="wsn-br:Demand"
912                     minOccurs="1" maxOccurs="1" />
913                 <xsd:element ref="wsn-br:CreationTime"
914                     minOccurs="0" maxOccurs="1" />
915             </xsd:sequence>
916         </xsd:complexType>
917     </xsd:element>
918
919     <!-- ===== Message Types for NotificationBroker ===== -->
920     <xsd:element name="RegisterPublisher">
921         <xsd:complexType>
922             <xsd:sequence>
923                 <xsd:element name="PublisherReference"
924                     type="wsa:EndpointReferenceType"
925                     minOccurs="0" maxOccurs="1" />
926                 <xsd:element name="Topic"
927                     type="wsn-b:TopicExpressionType"
928                     minOccurs="0" maxOccurs="unbounded" />
929                 <xsd:element name="Demand"
930                     type="xsd:boolean" default="false"
931                     minOccurs="0" maxOccurs="1" />
932                 <xsd:element name="InitialTerminationTime"
933                     type="xsd:dateTime"
934                     minOccurs="0" maxOccurs="1" />
935                 <xsd:any namespace="##other" processContents="lax"
936                     minOccurs="0" maxOccurs="unbounded" />
937             </xsd:sequence>
938         </xsd:complexType>
939     </xsd:element>
940
941     <xsd:element name="RegisterPublisherResponse">
942         <xsd:complexType>
943             <xsd:sequence>

```

```

944         <xsd:element name="PublisherRegistrationReference"
945                     type="wsa:EndpointReferenceType"
946                     minOccurs="0" maxOccurs="1" />
947     </xsd:sequence>
948 </xsd:complexType>
949 </xsd:element>
950
951
952 <xsd:complexType name="InvalidTopicExpressionFaultType">
953     <xsd:complexContent>
954         <xsd:extension base="wsrf-bf:BaseFaultType" />
955     </xsd:complexContent>
956 </xsd:complexType>
957 <xsd:element name="InvalidTopicExpressionFault"
958             type="wsn-br:InvalidTopicExpressionFaultType" />
959
960 <xsd:complexType name="TopicNotSupportedFaultType">
961     <xsd:complexContent>
962         <xsd:extension base="wsrf-bf:BaseFaultType" />
963     </xsd:complexContent>
964 </xsd:complexType>
965 <xsd:element name="TopicNotSupportedFault"
966             type="wsn-br:TopicNotSupportedFaultType" />
967
968 <xsd:complexType name="PublisherRegistrationRejectedFaultType">
969     <xsd:complexContent>
970         <xsd:extension base="wsrf-bf:BaseFaultType" />
971     </xsd:complexContent>
972 </xsd:complexType>
973 <xsd:element name="PublisherRegistrationRejectedFault"
974             type="wsn-br:PublisherRegistrationRejectedFaultType" />
975
976 <xsd:complexType name="PublisherRegistrationFailedFaultType">
977     <xsd:complexContent>
978         <xsd:extension base="wsrf-bf:BaseFaultType" />
979     </xsd:complexContent>
980 </xsd:complexType>
981 <xsd:element name="PublisherRegistrationFailedFault"
982             type="wsn-br:PublisherRegistrationFailedFaultType" />
983
984 <xsd:complexType name="PullNotificationNotSupportedType">
985     <xsd:complexContent>
986         <xsd:extension base="wsrf-bf:BaseFaultType" />
987     </xsd:complexContent>
988 </xsd:complexType>
989 <xsd:element name="PullNotificationNotSupportedFault"
990             type="wsn-br:PullNotificationNotSupportedType" />
991
992 <xsd:element name="Destroy">
993     <xsd:complexType>
994         <xsd:sequence>

```

```

995         <xsd:any namespace="##other" processContents="lax"
996             minOccurs="0" maxOccurs="unbounded" />
997     </xsd:sequence>
998     <xsd:anyAttribute />
999 </xsd:complexType>
1000 </xsd:element>
1001
1002 <xsd:element name="DestroyResponse">
1003     <xsd:complexType>
1004         <xsd:sequence>
1005             <xsd:any namespace="##other" processContents="lax"
1006                 minOccurs="0" maxOccurs="unbounded" />
1007         </xsd:sequence>
1008         <xsd:anyAttribute />
1009     </xsd:complexType>
1010 </xsd:element>
1011
1012 <xsd:complexType name="ResourceNotDestroyedFaultType">
1013     <xsd:complexContent>
1014         <xsd:extension base="wsrf-bf:BaseFaultType" />
1015     </xsd:complexContent>
1016 </xsd:complexType>
1017 <xsd:element name="ResourceNotDestroyedFault"
1018     type="wsn-br:ResourceNotDestroyedFaultType" />
1019
1020 </xsd:schema>

```

1021

Appendix C. WSDL 1.1

1022
1023

The following illustrates the WSDL 1.1 for the Web service methods described in this specification:

1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062

```
<?xml version="1.0" encoding="utf-8"?>
<!--
OASIS takes no position regarding the validity or scope of any
intellectual property or other rights that might be claimed to pertain
to the implementation or use of the technology described in this
document or the extent to which any license under such rights might or
might not be available; neither does it represent that it has made any
effort to identify any such rights. Information on OASIS's procedures
with respect to rights in OASIS specifications can be found at the
OASIS website. Copies of claims of rights made available for
publication and any assurances of licenses to be made available, or the
result of an attempt made to obtain a general license or permission for
the use of such proprietary rights by implementors or users of this
specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any
copyrights, patents or patent applications, or other proprietary rights
which may cover technology that may be required to implement this
specification. Please address the information to the OASIS Executive
Director.

Copyright (C) OASIS Open (2005). All Rights Reserved.

This document and translations of it may be copied and furnished to
others, and derivative works that comment on or otherwise explain it or
assist in its implementation may be prepared, copied, published and
distributed, in whole or in part, without restriction of any kind,
provided that the above copyright notice and this paragraph are
included on all such copies and derivative works. However, this
document itself may not be modified in any way, such as by removing the
copyright notice or references to OASIS, except as needed for the
purpose of developing OASIS specifications, in which case the
procedures for copyrights defined in the OASIS Intellectual Property
Rights document must be followed, or as required to translate it into
languages other than English.

The limited permissions granted above are perpetual and will not be
revoked by OASIS or its successors or assigns.
```

1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113

```
This document and the information contained herein is provided on an
"AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
-->

<wsdl:definitions name="WS-BrokeredNotification"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wsa="http://www.w3.org/2005/03/addressing"
  xmlns:wsn-br="http://docs.oasis-open.org/wsn/br-1"
  xmlns:wsn-brw="http://docs.oasis-open.org/wsn/brw-1"
  xmlns:wsn-b="http://docs.oasis-open.org/wsn/b-1"
  xmlns:wsn-bw="http://docs.oasis-open.org/wsn/bw-1"
  xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-1"
  xmlns:wsrf-rw="http://docs.oasis-open.org/wsrf/rw-1"
  targetNamespace="http://docs.oasis-open.org/wsn/brw-1">

  <!-- ===== Imports ===== -->
  <wsdl:import namespace="http://docs.oasis-open.org/wsrf/rw-1"
    location="http://docs.oasis-open.org/wsrf/rw-1"/>

  <wsdl:import namespace="http://docs.oasis-open.org/wsn/bw-1"
    location="http://docs.oasis-open.org/wsn/bw-1"/>

  <!-- ===== Types Definitions ===== -->
  <wsdl:types>
    <xsd:schema>
      <xsd:import
        namespace="http://docs.oasis-open.org/wsn/br-1"
        schemaLocation="http://docs.oasis-open.org/wsn/br-1"/>
      </xsd:schema>
    </wsdl:types>

  <!-- ===== NotificationBroker::RegisterPublisher =====
  RegisterPublisher(PublisherReference, TopicExpression* ,
    [Demand], [InitialTerminationTime])
  returns: WS-Resource qualified EPR to a PublisherRegistration -->
  <wsdl:message name="RegisterPublisherRequest">
    <wsdl:part name="RegisterPublisherRequest"
      element="wsn-br:RegisterPublisher"/>
  </wsdl:message>

  <wsdl:message name="RegisterPublisherResponse">
    <wsdl:part name="RegisterPublisherResponse"
      element="wsn-br:RegisterPublisherResponse"/>
  </wsdl:message>
```



```

1114 <wsdl:message name="InvalidTopicExpressionFault">
1115   <wsdl:part name="InvalidTopicExpressionFault"
1116     element="wsn-br:InvalidTopicExpressionFault" />
1117 </wsdl:message>
1118
1119 <wsdl:message name="TopicNotSupportedFault">
1120   <wsdl:part name="TopicNotSupportedFault"
1121     element="wsn-br:TopicNotSupportedFault" />
1122 </wsdl:message>
1123
1124 <wsdl:message name="PublisherRegistrationRejectedFault">
1125   <wsdl:part name="PublisherRegistrationRejectedFault"
1126     element="wsn-br:PublisherRegistrationRejectedFault" />
1127 </wsdl:message>
1128
1129 <wsdl:message name="PublisherRegistrationFailedFault">
1130   <wsdl:part name="PublisherRegistrationFailedFault"
1131     element="wsn-br:PublisherRegistrationFailedFault" />
1132 </wsdl:message>
1133
1134 <wsdl:message name="PullNotificationNotSupportedFault">
1135   <wsdl:part name="PullNotificationNotSupportedFault"
1136     element="wsn-br:PullNotificationNotSupportedFault" />
1137 </wsdl:message>
1138
1139 <wsdl:message name="DestroyRequest">
1140   <wsdl:part name="DestroyRequest"
1141     element="wsn-br:Destroy" />
1142 </wsdl:message>
1143
1144 <wsdl:message name="DestroyResponse">
1145   <wsdl:part name="DestroyResponse"
1146     element="wsn-br:DestroyResponse" />
1147 </wsdl:message>
1148
1149 <wsdl:message name="ResourceNotDestroyedFault">
1150   <wsdl:part name="ResourceNotDestroyedFault"
1151     element="wsn-br:ResourceNotDestroyedFault" />
1152 </wsdl:message>
1153
1154 <!-- ===== PortType Definitions ===== -->
1155
1156 <!-- ===== RegisterPublisher ===== -->
1157 <wsdl:portType name="RegisterPublisher">
1158   <wsdl:operation name="RegisterPublisher">
1159     <wsdl:input message="wsn-brw:RegisterPublisherRequest" />
1160     <wsdl:output message="wsn-brw:RegisterPublisherResponse" />
1161     <wsdl:fault name="ResourceUnknownFault"
1162       message="wsrf-rw:ResourceUnknownFault" />
1163     <wsdl:fault name="InvalidTopicExpressionFault"
1164       message="wsn-brw:InvalidTopicExpressionFault" />

```

```

1165     <wsdl:fault name="TopicNotSupportedFault"
1166             message="wsn-brw:TopicNotSupportedFault" />
1167     <wsdl:fault name="PublisherRegistrationRejectedFault"
1168             message="wsn-brw:PublisherRegistrationRejectedFault" />
1169     <wsdl:fault name="PublisherRegistrationFailedFault"
1170             message="wsn-brw:PublisherRegistrationFailedFault" />
1171     </wsdl:operation>
1172 </wsdl:portType>
1173
1174
1175 <!-- ===== NotificationBroker PortType Definition ===== -->
1176 <wsdl:portType name="NotificationBroker">
1177     <!-- ===== extends NotificationConsumer ===== -->
1178     <wsdl:operation name="Notify">
1179         <wsdl:input message="wsn-bw:Notify" />
1180     </wsdl:operation>
1181
1182     <!-- ===== extends NotificationProducer ===== -->
1183     <wsdl:operation name="Subscribe">
1184         <wsdl:input message="wsn-bw:SubscribeRequest" />
1185         <wsdl:output message="wsn-bw:SubscribeResponse" />
1186         <wsdl:fault name="ResourceUnknownFault"
1187             message="wsrf-rw:ResourceUnknownFault" />
1188         <wsdl:fault name="InvalidFilterFault"
1189             message="wsn-bw:InvalidFilterFault" />
1190         <wsdl:fault name="TopicExpressionDialectUnknownFault"
1191             message="wsn-
1192 bw:TopicExpressionDialectUnknownFault" />
1193         <wsdl:fault name="InvalidTopicExpressionFault"
1194             message="wsn-bw:InvalidTopicExpressionFault" />
1195         <wsdl:fault name="TopicNotSupportedFault"
1196             message="wsn-bw:TopicNotSupportedFault" />
1197         <wsdl:fault name="InvalidProducerPropertiesExpressionFault"
1198             message="wsn-
1199 bw:InvalidProducerPropertiesExpressionFault" />
1200         <wsdl:fault name="InvalidMessageContentExpressionFault"
1201             message="wsn-bw:InvalidMessageContentExpressionFault" />
1202         <wsdl:fault name="InvalidUseRawValueFault"
1203             message="wsn-bw:InvalidUseRawValueFault" />
1204         <wsdl:fault name="UnacceptableInitialTerminationTimeFault"
1205             message="wsn-bw:UnacceptableInitialTerminationTimeFault" />
1206         <wsdl:fault name="SubscribeCreationFailedFault"
1207             message="wsn-bw:SubscribeCreationFailedFault" />
1208     </wsdl:operation>
1209     <wsdl:operation name="GetCurrentMessage">
1210         <wsdl:input message="wsn-bw:GetCurrentMessageRequest" />
1211         <wsdl:output message="wsn-bw:GetCurrentMessageResponse" />
1212         <wsdl:fault name="ResourceUnknownFault"
1213             message="wsrf-rw:ResourceUnknownFault" />
1214         <wsdl:fault name="TopicExpressionDialectUnknownFault"
1215             message="wsn-bw:TopicExpressionDialectUnknownFault" />

```

```

1216     <wsdl:fault name="InvalidTopicExpressionFault "
1217             message="wsn-bw:InvalidTopicExpressionFault" />
1218     <wsdl:fault name="TopicNotSupportedFault "
1219             message="wsn-bw:TopicNotSupportedFault" />
1220     <wsdl:fault name="NoCurrentMessageOnTopicFault "
1221             message="wsn-bw:NoCurrentMessageOnTopicFault" />
1222     <wsdl:fault name="MultipleTopicsSpecifiedFault "
1223             message="wsn-bw:MultipleTopicsSpecifiedFault" />
1224 </wsdl:operation>
1225
1226 <!-- ===== extends RegisterPublisher ===== -->
1227 <wsdl:operation name="RegisterPublisher">
1228     <wsdl:input message="wsn-brw:RegisterPublisherRequest" />
1229     <wsdl:output message="wsn-brw:RegisterPublisherResponse" />
1230     <wsdl:fault name="ResourceUnknownFault "
1231             message="wsrf-rw:ResourceUnknownFault" />
1232     <wsdl:fault name="InvalidTopicExpressionFault "
1233             message="wsn-brw:InvalidTopicExpressionFault" />
1234     <wsdl:fault name="TopicNotSupportedFault "
1235             message="wsn-brw:TopicNotSupportedFault" />
1236     <wsdl:fault name="PublisherRegistrationRejectedFault "
1237             message="wsn-brw:PublisherRegistrationRejectedFault" />
1238     <wsdl:fault name="PublisherRegistrationFailedFault "
1239             message="wsn-brw:PublisherRegistrationFailedFault" />
1240 </wsdl:operation>
1241
1242 <!-- ===== extends CreatePullPoint ===== -->
1243 <wsdl:operation name="CreatePullPoint">
1244     <wsdl:input name="CreatePullPointRequest "
1245             message="wsn-bw:CreatePullPointRequest" />
1246     <wsdl:output name="CreatePullPointResponse"
1247             message="wsn-bw:CreatePullPointResponse" />
1248     <wsdl:fault name="UnableToCreatePullPoint "
1249             message="wsn-bw:UnableToCreatePullPoint" />
1250     <wsdl:fault name="PullNotificationNotSupportedFault "
1251             message="wsn-brw:PullNotificationNotSupportedFault" />
1252 </wsdl:operation>
1253 </wsdl:portType>
1254
1255 <!-- ===== PublisherRegistrationManager PortType Definition ===== -->
1256 <wsdl:portType name="PublisherRegistrationManager">
1257
1258     <!-- === Destroy:ImmediateResourceTermination===== -->
1259     <wsdl:operation name="Destroy">
1260         <wsdl:input name="DestroyRequest "
1261                 message="wsn-brw:DestroyRequest" />
1262         <wsdl:output name="DestroyResponse"
1263                 message="wsn-brw:DestroyResponse" />
1264         <wsdl:fault name="ResourceUnknownFault "
1265                 message="wsrf-rw:ResourceUnknownFault" />
1266         <wsdl:fault name="ResourceNotDestroyedFault "

```

1267
1268
1269
1270

```
                message="wsn-brw:ResourceNotDestroyedFault" />  
        </wsdl:operation>  
    </wsdl:portType>  
</wsdl:definitions>
```

1271

Appendix D. Notices

1272 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1273 that might be claimed to pertain to the implementation or use of the technology described in this
1274 document or the extent to which any license under such rights might or might not be available;
1275 neither does it represent that it has made any effort to identify any such rights. Information on
1276 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1277 website. Copies of claims of rights made available for publication and any assurances of licenses
1278 to be made available, or the result of an attempt made to obtain a general license or permission
1279 for the use of such proprietary rights by implementers or users of this specification, can be
1280 obtained from the OASIS Executive Director.

1281 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1282 applications, or other proprietary rights which may cover technology that may be required to
1283 implement this specification. Please address the information to the OASIS Executive Director.

1284 Copyright © OASIS Open 2004. All Rights Reserved.

1285 This document and translations of it may be copied and furnished to others, and derivative works
1286 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1287 published and distributed, in whole or in part, without restriction of any kind, provided that the
1288 above copyright notice and this paragraph are included on all such copies and derivative works.
1289 However, this document itself does not be modified in any way, such as by removing the
1290 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
1291 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1292 Property Rights document must be followed, or as required to translate it into languages other
1293 than English.

1294 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1295 successors or assigns.

1296 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1297 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1298 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1299 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1300 PARTICULAR PURPOSE.