



1

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

4 **OASIS Standard, 1 October 2006**

5

6 **Document identifier:**

7 wsn-ws_brokered_notification-1.3-spec-os

8 **Location:**

9 http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.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 that, 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 This document is an OASIS standard.

36 Committee members should send comments on this specification to the [wsn@lists.oasis-](mailto:wsn@lists.oasis-open.org)
37 [open.org](http://lists.oasis-open.org) list. Others may submit comments to the TC via the web form found on the
38 TC's web page at <http://www.oasis-open.org/committees/wsn>. Click the button for "Send
39 A Comment" at the top of the page. Submitted comments (for this work as well as other
40 works of the TC) are publicly archived and can be viewed at [http://lists.oasis-](http://lists.oasis-open.org/archives/wsn-comment/)
41 [open.org/archives/wsn-comment/](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-](http://www.oasis-open.org/committees/wsn/)
45 [open.org/committees/wsn/](http://www.oasis-open.org/committees/wsn/)).

46 The errata document for this specification is maintained at:
47 http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-errata.pdf

48

Table of Contents

50	1	Introduction	4
51	1.1	Goals and Requirements	4
52	1.1.1	Requirements.....	4
53	1.1.2	Non-Goals	5
54	1.2	Notational Conventions	5
55	1.3	Namespaces	6
56	1.4	Fault Definitions.....	7
57	2	Relationship to Other Specifications.....	8
58	3	Terminology and Concepts.....	9
59	4	Publishing.....	12
60	5	NotificationBroker Interface.....	15
61	5.1	NotificationBroker Resource Properties	15
62	5.2	Notify	16
63	5.3	Subscribe	16
64	5.4	GetCurrentMessage	16
65	5.5	RegisterPublisher	16
66	5.6	CreatePullPoint	17
67	6	RegisterPublisher Interface.....	18
68	6.1	RegisterPublisher	18
69	6.1.1	Example SOAP Encoding of the RegisterPublisher Message Exchange	21
70	7	PublisherRegistrationManager Interface	24
71	7.1	PublisherRegistration Resource Properties	24
72	7.2	DestroyRegistration.....	25
73	7.2.1	Example SOAP Encoding of the DestroyRegistration Message Exchange	26
74	8	Security Considerations	27
75	8.1	Securing PublisherRegistration.....	27
76	9	References.....	28
77	9.1	Normative	28
78	9.2	Non-Normative	28
79		Appendix A. Acknowledgments	30
80		Appendix B. XML Schema.....	31
81		Appendix C. WSDL 1.1.....	36
82		Appendix D. Revision History	41
83		Appendix E. Notices	43

84 1 Introduction

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

89 This specification defines the Web services interface for the NotificationBroker. A
90 NotificationBroker is an intermediary between message Publishers and message Subscribers. A
91 NotificationBroker decouples NotificationProducers and Notification Consumers and can provide
92 advanced messaging features such as demand-based publishing and load-balancing. A
93 NotificationBroker also allows publication of messages from entities that are not themselves
94 service providers. This is very similar to a traditional Message Oriented Middleware model.

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

98 1.1 Goals and Requirements

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

103 1.1.1 Requirements

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

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

- 120 • **Must conform to WS-BaseNotification:** A NotificationBroker must support required
121 message exchanges defined by the [WS-BaseNotification] specification. It must conform to
122 the NotificationProducer and the NotificationConsumer interfaces defined in WS-
123 BaseNotification.
- 124 • **WS-BrokeredNotification must be independent of binding-level details:** Transport
125 protocol details must be orthogonal to the subscription and the delivery of the notifications, so
126 that the specification can be used over a variety of different transports.
- 127 • **Must not exclude non-service producers and subscribers:** WS-BrokeredNotification
128 design must not exclude a non-service entity to deliver a notification message to a
129 NotificationBroker. It must not exclude a NotificationBroker to send a notification message to
130 a non-service consumer.
- 131 • **Must provide publisher registration:** WS-BrokeredNotification must define standard
132 message exchanges for registering a NotificationPublisher with a NotificationBroker.

133 1.1.2 Non-Goals

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

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

143 1.2 Notational Conventions

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

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

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

- 153 • '?' denotes optionality (i.e. zero or one occurrences),
154 • '*' denotes zero or more occurrences,
155 • '+' one or more occurrences,

- 156 • '[' and `]' are used to form groups,
- 157 • `|' represents choice.
- 158 • Attributes are conventionally assigned a value which corresponds to their type, as
- 159 defined in the normative schema.
- 160 • Elements with simple content are conventionally assigned a value which corresponds to
- 161 the type of their content, as defined in the normative schema.
- 162 • The use of {any} indicates the presence of an element wildcard (<xs:any/>).
- 163 • The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute/>).

164

```

165 <!-- sample pseudo-schema -->
166 <element
167     required_attribute_of_type_QName="xs:QName"
168     optional_attribute_of_type_string="xs:string"?>
169   <required_element />
170   <optional_element /> ?
171   <one_or_more_of_these_elements /> +
172   [ <choice_1 /> | <choice_2 /> ] *
173 </element>

```

174 Where there is disagreement between the separate XML schema and WSDL files describing the
 175 messages defined by this specification and the normative descriptive text (excluding any pseudo-
 176 schema) in this document, the normative descriptive text will take precedence over the separate
 177 files. The separate files take precedence over any pseudo-schema and over any schema and
 178 WSDL included in the appendices.

179

1.3 Namespaces

180 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/2005/08/addressing
wsn-b	http://docs.oasis-open.org/wsn/b-2
wsn-br	http://docs.oasis-open.org/wsn/br-2

wsn-bw	http://docs.oasis-open.org/wsn/bw-2
wsn-brw	http://docs.oasis-open.org/wsn/brw-2
wsrf-bf	http://docs.oasis-open.org/wsrf/bf-2
wsrf-bfw	http://docs.oasis-open.org/wsrf/bfw-2

181

1.4 Fault Definitions

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

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

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

187

2 Relationship to Other Specifications

188 This specification builds on the basic notification mechanism defined in [\[WS-BaseNotification\]](#) by
189 adding the concept of an intermediary NotificationBroker, and describing additional variants on
190 the publisher role. A NotificationBroker takes on the role of both NotificationProducer and
191 NotificationConsumer (as defined in [\[WS-BaseNotification\]](#)), and its interactions with other
192 NotificationProducers and NotificationConsumers are largely defined by the WS-BaseNotification
193 specification.

194 This means that a NotificationBroker, implemented to conform to this specification, must also
195 conform to [\[WS-BaseNotification\]](#). Such a NotificationBroker can deliver notifications to
196 NotificationConsumers that are implemented to conform to [\[WS-BaseNotification\]](#), and can
197 subscribe to Notifications distributed by NotificationProducers as defined in [\[WS-
198 BaseNotification\]](#).

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

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

202 3 Terminology and Concepts

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

205 Publisher:

- 206 • A Publisher is an entity that creates Notifications, based upon Situation(s) that it is
207 capable of detecting and translating into Notification artifacts. It does not need to be a
208 Web service.
- 209 • A Publisher can register what topics it wishes to publish with a NotificationBroker.
- 210 • A Publisher MAY be a Web service that implements the message exchanges associated
211 with the NotificationProducer interface, in which case it also distributes the Notifications
212 to the relevant NotificationConsumers.
- 213 • If a Publisher does not implement the message exchanges associated with
214 NotificationProducer, then it is not required to support the Subscribe request message
215 and does not have to maintain knowledge of the NotificationConsumers that are
216 subscribed to it; a NotificationBroker takes care of this on its behalf.

217 NotificationBroker:

- 218 • A NotificationBroker is an intermediary Web service that decouples
219 NotificationConsumers from Publishers. A NotificationBroker is capable of subscribing to
220 notifications, either on behalf of NotificationConsumers, or for the purpose of messaging
221 management. It is capable disseminating notifications on behalf of Publishers to
222 NotificationConsumers.
- 223 • A NotificationBroker aggregates NotificationProducer, NotificationConsumer, and
224 RegisterPublisher interfaces.
- 225 • Acting as an intermediary, a NotificationBroker provides additional capabilities to the
226 base NotificationProducer interface:
 - 227 ○ It can relieve a Publisher from having to implement message exchanges
228 associated with NotificationProducer; the NotificationBroker takes on the duties of
229 a SubscriptionManager (managing subscriptions) and NotificationProducer
230 (distributing Notifications) on behalf of the Publisher.
 - 231 ○ It can reduce the number of inter-service connections and references, if there are
232 many Publishers and many NotificationConsumers.
 - 233 ○ It can act as a finder service. Potential Publishers and Subscribers can in effect
234 find each other by utilizing a common NotificationBroker.
 - 235 ○ It can provide anonymous Notification, so that the Publishers and the
236 NotificationConsumers need not be aware of each other's identity.

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

- 274
- 275
- 276
- 277
- 278
- 279
- 280
- 281
- 282
- 283
- 284
- Demand-Based Publishers implement message exchanges associated with the NotificationProducer interface.
 - 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.
 - 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.

285

4 Publishing

286

There are three distinct stages in the Notification process

287

- Observation of the Situation and its noteworthy characteristics;

288

- Creation of the Notification artifact that captures the noteworthy characteristics of the Situation; and

289

290

- Distribution of copies of the Notification to zero or more interested parties.

291

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

292

293

294

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

295

296

297

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

298

299

300

301

302

303

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

304

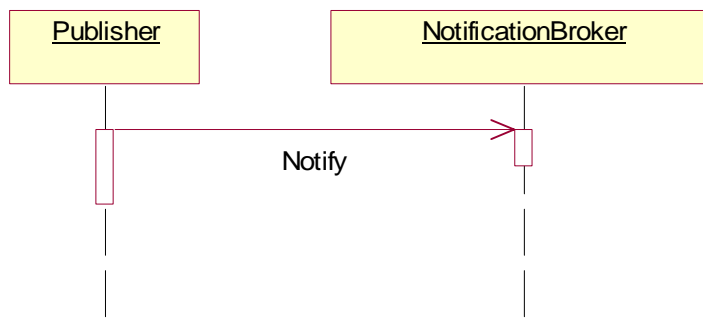
305

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

306

307

The following figure illustrates simple publishing:



308

309

In the simple publishing scenario, the Publisher entity is responsible only for the core Publisher functions - observing the Situation and formatting the Notification artifact that describes the

310

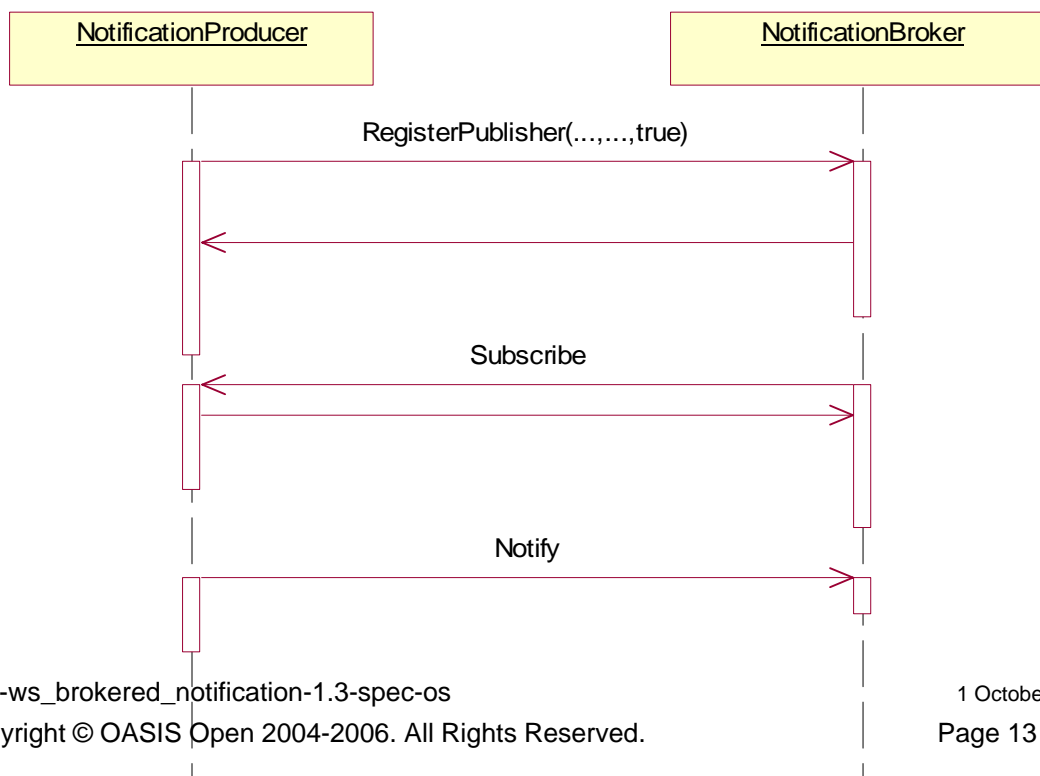
311 Situation. The dissemination step occurs when the Publisher sends the Notify message to the
312 NotificationBroker.

313 In the broker-initiated publishing pattern, the role of the Publisher is played by a Web service that
314 implements NotificationProducer. The act of observing the Situation and formatting the
315 Notification happens within the implementation logic of the NotificationProducer itself. The
316 Notification is disseminated by the NotificationProducer sending the Notify message to a
317 NotificationBroker. The Notification may also be disseminated by sending the Notify message to
318 any NotificationConsumers that are subscribing to the NotificationProducer.

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

324 The last pattern, the demand-based pattern, requires the Publisher to be a NotificationProducer,
325 and thereby accept the Subscribe message. Demand-based publication is intended for use in
326 cases where the act of observing the Situation or the act of formatting the Notification artifact
327 might be expensive to perform, and therefore should be avoided if there are no interested parties
328 for that Notification. A Publisher indicates its intention to use this pattern by registering with the
329 NotificationProducer and setting the Demand component of the RegisterPublisher request
330 message to "true". Based upon this style of registration, the NotificationBroker sends the
331 Subscribe message to the Publisher (recall: in this situation the Publisher must implement the
332 message exchanges associated with the NotificationProducer interface).

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



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

5 NotificationBroker Interface

336

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

340 A NotificationBroker MAY be a WS-Resource, and if it is, it MUST support the required message
341 exchanges defined by the [WS-ResourceProperties] specification and MAY support the optional
342 message exchanges defined by WS-ResourceProperties.

343 A NotificationBroker MUST also support message exchanges and MAY support Resource
344 Property elements defined by the following interfaces:

- 345 • NotificationProducer
- 346 • NotificationConsumer
- 347 • RegisterPublisher

348 The NotificationBroker portType aggregates the three portTypes and is not the only way to
349 implement a broker. A distributed broker implementation can be achieved by hosting
350 NotificationProducer, NotificationConsumer, or RegisterPublisher portTypes at one or more
351 physical endpoints.

352 The NotificationBroker is not required to provide any specific subscription durability or continuity.
353 NotificationBrokers SHOULD advertise their durability or reliability features, either through
354 policies or other means.

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

357

5.1 NotificationBroker Resource Properties

358

359 In addition to the message exchanges described in this specification, a NotificationBroker MAY
360 also support the required message exchanges defined in the WS-ResourceProperties
361 specification and MAY support the optional message exchanges defined in the WS-
362 ResourceProperties specification. In such cases, the Resource Properties document defined by
363 the NotificationBroker MUST include references to resource properties defined in
364 NotificationProducer and NotificationConsumer, and also MUST include a reference to the
365 following resource property element:

366

```
...  
367   targetNamespace="http://docs.oasis-open.org/wsn/br-2">  
368   ...  
369   <xsd:element name="RequiresRegistration" type="xsd:boolean"/>  
370   ...
```

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

372

```
<xsd:element ref="wsn-br:RequiresRegistration"
```

373

```
minOccurs="1" maxOccurs="1" />
```

374 This resource property element is further constrained as follows:

375 /wsn-br:RequiresRegistration

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

378

5.2 Notify

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

381 A Publisher sends a Notify message to a NotificationBroker in order to publish a Notification on a
382 given Topic. As a result of the Publisher sending this message, Notifications are delivered to all
383 NotificationConsumers subscribed on the given Topic. The NotificationBroker may require that a
384 Publisher be registered before the Publisher sends it a Notification (see 6.1).

385

5.3 Subscribe

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

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

393 If the processing of a Subscribe message is successful, the NotificationBroker MUST produce a
394 response message, as described in [WS-BaseNotification](#), containing an endpoint reference to a
395 Subscription resource representing a Subscription created as a result of processing the
396 Subscribe request. Otherwise, the NotificationBroker must fault. [WS-BaseNotification](#) defines a
397 set of these faults.

398

5.4 GetCurrentMessage

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

401 As defined in [WS-BaseNotification](#), in response to a GetCurrentMessage message, the
402 NotificationBroker MAY return the last Notification published on a given Topic. This is a non-
403 destructive read, allowing a newly-subscribed NotificationConsumer to get the last Notification
404 that other NotificationConsumers have received.

405

5.5 RegisterPublisher

406 The NotificationBroker MUST support the RegisterPublisher message exchange from the
407 RegisterPublisher interface.

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

413 **5.6 CreatePullPoint**

414 The NotificationBroker MAY support pull-style notification as defined in WS-BaseNotification and
415 attempt to create a PullPoint resource upon receiving a CreatePullPoint request. The
416 NotificationBroker does not define additional constraints to its usage of the CreatePullPoint
417 operation.

418

6 RegisterPublisher Interface

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

423

6.1 RegisterPublisher

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

428

```
429 <wsn-br:RegisterPublisher>  
430   <wsn-br:PublisherReference>  
431     wsa:EndpointReferenceType  
432   </wsn-br:PublisherReference?>  
433   <wsn-br:Topic Dialect = "xsd:anyURI" >  
434     {any} ?  
435   </wsn-br:Topic> *  
436   <wsn-br:Demand>  
437     xsd:boolean  
438   </wsn-br:Demand?>  
439   <wsn-br:InitialTerminationTime>  
440     xsd:dateTime  
441   </wsn-br:InitialTerminationTime?>  
442   {any} *  
443 </wsn-br:RegisterPublisher>  
444 ...
```

445

446 The [WS-Addressing](#) [action] Message Addressing Property MUST contain the URI
447 <http://docs.oasis-open.org/wsn/brw-2/RegisterPublisher/RegisterPublisherRequest>.

448

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

450 /wsn-br:RegisterPublisher/PublisherReference

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

456 /wsn-br:RegisterPublisher/Topic

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

461 /wsn-br:RegisterPublisher/Demand

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

468 /wsn-br:RegisterPublisher/InitialTerminationTime

469 This component contains the service requestor's suggestion for the initial termination
470 time of the PublisherRegistration resource being created. This time is relative to the time
471 source used by the NotificationBroker. If the NotificationBroker is unable or unwilling to
472 set the TerminationTime to the given value or greater, the RegisterPublisher request
473 MUST return an UnacceptableInitialTerminationTimeFault message. If the value is not "in
474 the future" relative to the current time as known by the NotificationBroker, the
475 RegisterPublisher request MUST also return an UnacceptableInitialTerminationTimeFault
476 message.

477 The use of the xsi:nil attribute with value "true" indicates there is no scheduled
478 termination time requested for the RegisterPublisher. If the element does not include the
479 time zone designation, the value of the element MUST be interpreted as universal time
480 (UTC).

481 The publisher should take care when choosing a value for InitialTerminationTime, and
482 any subsequent values that modify the TerminationTime property of the publisher
483 registration. It is RECOMMENDED that the publisher choose termination time values that
484 are significantly (several orders of magnitude) greater than the network latency expected
485 in the interaction between the publisher and the broker. In so doing, the designer avoids
486 undesirable results, such as the termination time having expired prior to the receipt of the
487 published message. The [\[WS-ResourceLifetime\]](#) specification (Section 5.1 Regarding
488 time) contains further suggestions on how designers should reason about time values in
489 a WS-Resource Lifetime application.

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

492 /wsn-br:RegisterPublisher/{any}

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

496 If a /wsn-br:RegisterPublisher/Topic component is included in the message, the
497 NotificationBroker MUST register the Web service specified by the /wsn-
498 br:RegisterPublisher/PublisherReference component as a Publisher on the set of Topics

499 identified by the /wsn-br:RegisterPublisher/Topic component. If for any reason the registration
500 fails, the NotificationBroker MUST fault.

501 As part of the processing of a RegisterPublisher request, the NotificationBroker creates a
502 PublisherRegistration resource representing the registration. A new resource is created
503 regardless of whether the same Publisher has previously registered with the NotificationBroker.
504 The NotificationBroker MUST return a PublisherRegistrationReference in the response to the
505 RegisterPublisher request.

506 PublisherRegistrationReference is a WS-Addressing endpoint reference and includes the address
507 of a PublisherRegistrationManager service.

508 ConsumerReference is a WS-Addressing endpoint reference to a NotificationConsumer that
509 accepts notifications from this registered Publisher. If Demand value is false in the
510 RegisterPublisher request, the NotificationBroker MUST include a ConsumerReference in the
511 response.

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

```
514 ...  
515 <wsn-br:RegisterPublisherResponse>  
516   <wsn-br:PublisherRegistrationReference>  
517     <wsa:Address>  
518       Address of PublisherRegistration Manager  
519     </wsa:Address>  
520     ...  
521   </wsn-br:PublisherRegistrationReference>  
522   <wsn-br:ConsumerReference>  
523     <wsa:Address>  
524       Address of a NotificationConsumer with which the  
525       Publisher is registered  
526     </wsa:Address>  
527     ...  
528   </wsn-br:ConsumerReference>  
529 </wsn-br:RegisterPublisherResponse>  
530 ...
```

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

532 <http://docs.oasis-open.org/wsn/brw-2/RegisterPublisher/RegisterPublisherResponse>

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

534 /wsn-br:RegisterPublisherResponse/PublisherRegistrationReference

535 A WS-Addressing endpoint reference to the PublisherRegistration resource created by
536 the RegisterPublisher request message. This element MUST be present in the
537 RegisterPublisher response message. The NotificationBroker may choose to include
538 extra information such as reference parameters in this reference.

539 /wsn-br:RegisterPublisherResponse/ConsumerReference

540 A WS-Addressing endpoint reference to a NotificationConsumer resource that accepts
541 notifications for this publisher registration.

542 Any Notification Messages sent by the Publisher (and considered to take place under this
543 registration) MUST be sent to this endpoint reference.

544 The NotificationBroker MAY use this as a mechanism for identifying the Publisher as
545 having registered.

546 If the NotificationBroker does not respond to the RegisterPublisher request message with the
547 RegisterPublisherResponse message, then it MUST send a fault. The NotificationBroker MUST
548 fault if it rejects the publisher registration. This specification defines the following faults associated
549 with failure to process the RegisterPublisher request message:

550

551 ResourceUnknownFault

552 • The NotificationBroker is acting as a WS-Resource, and the resource identified in the
553 message is not known to the Web service. This fault is specified by the WS-Resource
554 [WS-Resource] specification.

555 InvalidTopicExpressionFault

556 • The TopicExpression presented in the request message is invalid. This fault is specified
557 in WS-BaseNotification.

558 TopicNotSupportedFault

559 • The TopicExpression does not match any Topic supported by the NotificationBroker. This
560 fault is specified in WS-BaseNotification.

561 PublisherRegistrationRejectedFault

562 • The publisher registration is rejected by the NotificationBroker. The NotificationBroker
563 MAY provide a hint in the fault message indicating why the registration is rejected.

564 PublisherRegistrationFailedFault

565 • The publisher registration process has failed. The NotificationBroker MAY include a hint
566 in the fault message indicating why the registration is failed.

567 UnacceptableInitialTerminationTimeFault

568 • The value of InitialTerminationTime specified in the RegisterPublisher request message
569 is not acceptable to the NotificationBroker. The NotificationBroker MAY include a hint in
570 the fault message indicating why the value is unacceptable.

571 **6.1.1 Example SOAP Encoding of the RegisterPublisher Message** 572 **Exchange**

573 The following is a non-normative example of a RegisterPublisher request message using SOAP
574 1.1 [SOAP 1.1] or SOAP 1.2 [SOAP 1.2]:

```
575 <s:Envelope ... >  
576 <s:Header>  
577 <wsa:Action>
```

```

578     http://docs.oasis-open.org/wsn/brw-
579 2/RegisterPublisher/RegisterPublisherRequest
580     </wsa:Action>
581     ...
582 </s:Header>
583 <s:Body>
584   <wsn-br:RegisterPublisher>
585     <wsn-br:PublisherReference>
586       <wsa:Address>
587         http://www.example.org/PublisherEndpoint
588       </wsa:Address>
589       <wsa:ReferenceParameters>
590         <npex: NPResourceDisambiguator>
591           uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
592         </npex: NPResourceDisambiguator>
593       </wsa:ReferenceParameters>
594     </wsn-br:PublisherReference>
595     <wsn-br:Topic Dialect="http://docs.oasis-open.org/wsn/t-
596 1/TopicExpression/Simple">
597       npex:SomeTopic
598     </wsn-br:Topic>
599     <wsn-br:Demand>
600       true
601     </wsn-br:Demand>
602     <wsn-br:InitialTerminationTime>
603       2003-12-25T00:00:00.00000Z
604     </wsn-br:InitialTerminationTime>
605   </wsn-br:RegisterPublisher>
606 </s:Body>
607 </s:Envelope>

```

608

609 The following is a non-normative example of a RegisterPublisher response message using
610 SOAP:

```

611 <s:Envelope ... >
612   <s:Header>
613     <wsa:Action>
614       http://docs.oasis-open.org/wsn/brw-
615 2/RegisterPublisher/RegisterPublisherResponse
616     </wsa:Action>
617     ...
618   </s:Header>
619   <s:Body>
620     <wsn-br:RegisterPublisherResponse>
621       <wsn-br:PublisherRegistrationReference>
622         <wsa:Address>
623           http://www.example.org/PublisherRegistrationManager
624         </wsa:Address>
625         <wsa:ReferenceParameters>
626           <npex: NPubResourceId>
627             uuid:95fefeb3-f37d-5dfe-44fe-221d9fceed99

```

```
628         </npex:NPubResourceId>
629     </wsa:ReferenceParameters>
630 </wsn-br:PublisherRegistrationReference>
631 <wsn-br:ConsumerReference>
632     <wsa:Address>
633         http://www.example.org/NotificationConsumer
634     </wsa:Address>
635     ...
636 </wsn-br:ConsumerReference>
637 </wsn-br:RegisterPublisherResponse>
638 </s:Body>
639 </s:Envelope>
```

640

7 PublisherRegistrationManager Interface

641 The PublisherRegistrationManager interface defines message exchanges to manipulate
642 PublisherRegistration resources. The PublisherRegistrationManager MAY expose
643 PublisherRegistrations as WS-Resources, and if it does then the PublisherRegistrationManager
644 MUST support the immediate termination interface defined by WS-ResourceLifetime and it MAY
645 support the scheduled termination interface defined by WS-ResourceLifetime.

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

649

7.1 PublisherRegistration Resource Properties

650 In addition to the message exchanges described in this specification, a
651 PublisherRegistrationManager MAY also support the required message exchanges defined in the
652 WS-ResourceProperties specification and MAY support the optional message exchanges defined
653 in the WS-ResourceProperties specification. In such cases, the Resource Properties document
654 defined by the PublisherRegistrationManager MUST also include references to the following
655 resource property elements:

656

```
.....  
targetNamespace="http://docs.oasis-open.org/wsn/br-2"  
...  
<xsd:element name="PublisherReference"  
            type="wsa:EndpointReferenceType" />  
<xsd:element name="Topic" type="wsn-b:TopicExpressionType" />  
<xsd:element name="Demand" type="xsd:boolean" />  
<xsd:element name="CreationTime" type="xsd:dateTime" />  
...
```

657

658

659

660

661

662

663

664

665

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

666

667

```
<xsd:element ref="wsn-br:PublisherReference"  
            minOccurs="0" maxOccurs="1" />  
<xsd:element ref="wsn-br:Topic"  
            minOccurs="0" maxOccurs="unbounded" />  
<xsd:element ref="wsn-br:Demand"  
            minOccurs="1" maxOccurs="1" />  
<xsd:element ref="wsn-br:CreationTime"  
            minOccurs="0" maxOccurs="1" />
```

668

669

670

671

672

673

674

675

These resource property elements are further constrained as follows:

676

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

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

679 /wsn-br:CreationTime

680 Indicates the date and time at which the PublisherRegistration was created. This
681 component MAY be omitted, for example by resource-constrained devices that cannot
682 associate a creation time with PublisherRegistration resources they create.

683 If PublisherRegistration is exposed as a WS-Resource, the PublisherRegistrationManager MAY
684 permit the following resource properties to be modified by a requestor, by sending a
685 SetResourceProperties request message as defined in the WS-ResourceProperties specification:

- 686 • /wsn-br:Topic and /wsn-br:Demand

687

688 Note: /wsn-br:Demand may not take the value “true” if there is no /wsn-
689 br:PublisherReference resource property element in the resource property document.

690 **7.2 DestroyRegistration**

691 The PublisherRegistrationManager interface provides a destroy operation, providing a means by
692 which a requestor can terminate the PublisherRegistration resource. The DestroyRegistration
693 request message has the following form:

```
694 <wsn-br:DestroyRegistration>  
695 {any} *  
696 </wsn-br:DestroyRegistration>  
697  
698
```

699 The WS-Addressing [action] Message Addressing Property MUST contain the URI
700 <http://docs.oasis-open.org/wsn/brw-2/PublisherRegistrationManager/DestroyRegistrationRequest>

701 The DestroyRegistration request message allows for open content and contains an extension
702 component

703 /wsn-br:DestroyRegistration/{any}.

704

705 Upon receipt of the DestroyRegistration request, the PublisherRegistrationManager MUST
706 attempt to destroy the PublisherRegistration resource. If the DestroyRegistration request
707 message is successfully processed, the PublisherRegistrationManager MUST respond with the
708 following message:

```
709 <wsn-br:DestroyRegistrationResponse>  
710 {any} *  
711 </wsn-br:DestroyRegistrationResponse>  
712  
713
```

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

715 [http://docs.oasis-open.org/wsn/brw-
716 2/PublisherRegistrationManager/DestroyRegistrationResponse](http://docs.oasis-open.org/wsn/brw-2/PublisherRegistrationManager/DestroyRegistrationResponse).

717 If the PublisherRegistrationManager does not respond to the DestroyRegistration request
718 message with the DestroyRegistrationResponse message, then it MUST send a fault. This
719 specification defines the following faults associated with failure to process the
720 DestroyRegistration request message:

721 ResourceUnknownFault

- 722 • The PublisherRegistration is a WS-Resource, and the resource identified in the message
723 is not known to the Web service. This fault is specified by the WS-Resource [WS-
724 Resource] specification.

725 ResourceNotDestroyedFault

- 726 • The PublisherRegistrationManager was unable to destroy the PublisherRegistration
727 resource for some reason.

728 **7.2.1 Example SOAP Encoding of the DestroyRegistration Message** 729 **Exchange**

730 The following is a non-normative example of a DestroyRegistration request message using
731 SOAP:

```
732 <s:Envelope ... >  
733   <s:Header>  
734     <wsa:Action>  
735       http://docs.oasis-open.org/wsn/brw-  
736 2/PublisherRegistrationManager/DestroyRegistrationRequest  
737     </wsa:Action>  
738     ...  
739   </s:Header>  
740   <s:Body>  
741     <wsn-br:DestroyRegistration/>  
742   </s:Body>  
743 </s:Envelope>
```

744 The following is a non-normative example of a DestroyRegistration response message using
745 SOAP:

```
746 <s:Envelope ... >  
747   <s:Header>  
748     <wsa:Action>  
749       http://docs.oasis-open.org/wsn/brw-  
750 2/PublisherRegistrationManager/DestroyRegistrationResponse  
751     </wsa:Action>  
752     ...  
753   </s:Header>  
754   <s:Body>  
755     <wsn-br:DestroyRegistrationResponse/>  
756   </s:Body>  
757 </s:Envelope>
```

758

8 Security Considerations

759

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

760

761

8.1 Securing PublisherRegistration

762

763

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

764

765

1. only authorized Publishers can register with a NotificationBroker

766

2. only messages of authorized Publishers and of registered topics, can be accepted by a NotificationBroker

767

768

3. only authorized principals can modify or delete PublisherRegistration resource

769

770

771

772

773

774

775

Given that WS-BrokeredNotification may implement WS-ResourceProperties and WS-ResourceLifetime, the security considerations outlined in these specifications need to be taken into account where appropriate. Authorization policies for those Resource Properties should be put in place so that the implications of providing the state information (through GetResourceProperty request messages) or through notification of state change and modification of the resource properties (through SetResourceProperty request messages), are taken into account.

776

777

778

779

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

780

- Controlling who can publish on a topic at publisher registration time

781

- Refusing to relay messages from unauthorized publishers

782

- Imposing security measurements on all messaging routing through the broker

783

- Providing convenience in messaging security management based on topics.

784

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

785

786

9 References

787

9.1 Normative

788

[RFC2119]

789

S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", IETF RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>

790

791

[XML]

792

"Extensible Markup Language (XML) 1.0", W3C Recommendation.

793

<http://www.w3.org/TR/REC-xml>

794

[XML-InfoSet]

795

"XML Information Set", W3C Recommendation. <http://www.w3.org/TR/xml-infoset/>

796

[WS-Addressing]

797

"Web Services Addressing 1.0 - Core", W3C Recommendation.

798

<http://www.w3.org/TR/ws-addr-core>

799

[WS-BaseNotification]

800

"Web Services Base Notification 1.3", OASIS Standard.

801

http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

802

[WS-Topics]

803

"Web Services Topics 1.3", OASIS Standard.

804

http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf

805

[WS-Resource]

806

"Web Services Resource 1.2", OASIS Standard.

807

http://docs.oasis-open.org/wsr/wsr-ws_resource-1.2-spec-os.pdf

808

[WS-ResourceLifetime]

809

"Web Services Resource Lifetime 1.2", OASIS Standard.

810

http://docs.oasis-open.org/wsr/wsr-ws_resource_lifetime-1.2-spec-os.pdf

811

[WS-ResourceProperties]

812

"Web Services Resource Properties 1.2", OASIS Standard.

813

http://docs.oasis-open.org/wsr/wsr-ws_resource_properties-1.2-spec-os.pdf

814

[WS-BaseFaults]

815

"Web Services Base Faults 1.2", OASIS Standard.

816

http://docs.oasis-open.org/wsr/wsr-ws_base_faults-1.2-spec-os.pdf

817

818

9.2 Non-Normative

819

[SOAP 1.1]

820

"Simple Object Access Protocol (SOAP) 1.1"

821

<http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

822 **[SOAP 1.2]**
823 “SOAP Version 1.2 Part 1: Messaging Framework”, W3C Recommendation.
824 <http://www.w3.org/TR/soap12-part1/>
825 **[WS-Security]**
826 “Web Services Security: SOAP Message Security 1.0”, OASIS Standard.
827 [http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)
828 [1.0.pdf](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf)

Appendix A. Acknowledgments

830 The following individuals were members of the committee during the development of this
831 specification:

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

839 Special thanks to the Global Grid Forum's Open Grid Services Infrastructure working group,
840 which defined the OGSI v1.0 specification which was a large inspiration for the ideas expressed
841 in this specification.

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

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

851

Appendix B. XML Schema

852 The XML types and elements used in WS-BrokeredNotification are defined in the following XML
853 Schema

```
854 <?xml version="1.0" encoding="UTF-8"?>
855 <!--
856 OASIS takes no position regarding the validity or scope of any
857 intellectual property or other rights that might be claimed to pertain
858 to the implementation or use of the technology described in this
859 document or the extent to which any license under such rights might or
860 might not be available; neither does it represent that it has made any
861 effort to identify any such rights. Information on OASIS's procedures
862 with respect to rights in OASIS specifications can be found at the OASIS
863 website. Copies of claims of rights made available for publication and
864 any assurances of licenses to be made available, or the result of an
865 attempt made to obtain a general license or permission for the use of
866 such proprietary rights by implementors or users of this specification,
867 can be obtained from the OASIS Executive Director.
868
869 OASIS invites any interested party to bring to its attention any
870 copyrights, patents or patent applications, or other proprietary rights
871 which may cover technology that may be required to implement this
872 specification. Please address the information to the OASIS Executive
873 Director.
874
875 Copyright (C) OASIS Open (2004-2006). All Rights Reserved.
876
877 This document and translations of it may be copied and furnished to
878 others, and derivative works that comment on or otherwise explain it or
879 assist in its implementation may be prepared, copied, published and
880 distributed, in whole or in part, without restriction of any kind,
881 provided that the above copyright notice and this paragraph are included
882 on all such copies and derivative works. However, this document itself
883 may not be modified in any way, such as by removing the copyright notice
884 or references to OASIS, except as needed for the purpose of developing
885 OASIS specifications, in which case the procedures for copyrights
886 defined in the OASIS Intellectual Property Rights document must be
887 followed, or as required to translate it into languages other than
888 English.
889
890 The limited permissions granted above are perpetual and will not be
891 revoked by OASIS or its successors or assigns.
892
893 This document and the information contained herein is provided on an "AS
894 IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
895 INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
896 INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
897 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
898 -->
```

899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950

```
<xsd:schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns:wsn-br="http://docs.oasis-open.org/wsn/br-2"
  xmlns:wsn-b="http://docs.oasis-open.org/wsn/b-2"
  xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-2"
  xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
  targetNamespace="http://docs.oasis-open.org/wsn/br-2"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
<!-- ===== Imports ===== -->
  <xsd:import namespace="http://www.w3.org/2005/08/addressing"
    schemaLocation="http://www.w3.org/2005/08/addressing/ws-
addr.xsd"/>
  <xsd:import namespace="http://docs.oasis-open.org/wsrf/bf-2"
    schemaLocation="http://docs.oasis-open.org/wsrf/bf-
2.xsd"/>
  <xsd:import namespace="http://docs.oasis-open.org/wsn/b-2"
    schemaLocation="http://docs.oasis-open.org/wsn/b-2.xsd"/>
  <xsd:import namespace="http://docs.oasis-open.org/wsn/t-1"
    schemaLocation="http://docs.oasis-open.org/wsn/t-1.xsd"/>
<!-- ===== Resource Properties for NotificationBroker ===== -->
  <xsd:element name="RequiresRegistration" type="xsd:boolean"/>
<!-- ===== Resource Properties for PublisherRegistration ===== -->
  <xsd:element name="PublisherReference"
    type="wsa:EndpointReferenceType"/>
  <xsd:element name="ConsumerReference"
    type="wsa:EndpointReferenceType"/>
  <xsd:element name="Topic"
    type="wsn-b:TopicExpressionType"/>
  <xsd:element name="Demand"
    type="xsd:boolean"/>
  <xsd:element name="CreationTime"
    type="xsd:dateTime"/>
  <xsd:element name="NotificationBrokerRP">
    <xsd:complexType>
      <xsd:sequence>
        <!-- From NotificationProducer -->
          <xsd:element ref="wsn-b:TopicExpression"
            minOccurs="0" maxOccurs="unbounded" />
          <xsd:element ref="wsn-b:FixedTopicSet"
            minOccurs="0" maxOccurs="1" />
          <xsd:element ref="wsn-b:TopicExpressionDialect"
```



```

951         minOccurs="0" maxOccurs="unbounded" />
952         <xsd:element ref="wstop:TopicSet"
953             minOccurs="0" maxOccurs="1" />
954         <!-- NotificationBroker specific -->
955         <xsd:element ref="wsn-br:RequiresRegistration"
956             minOccurs="1" maxOccurs="1" />
957     </xsd:sequence>
958 </xsd:complexType>
959 </xsd:element>
960
961 <!-- ===== Resource Properties for PublisherRegistration ===== -->
962 <xsd:element name="PublisherRegistrationRP">
963     <xsd:complexType>
964         <xsd:sequence>
965             <xsd:element ref="wsn-br:PublisherReference"
966                 minOccurs="0" maxOccurs="1" />
967             <xsd:element ref="wsn-br:Topic"
968                 minOccurs="0" maxOccurs="unbounded" />
969             <xsd:element ref="wsn-br:Demand"
970                 minOccurs="1" maxOccurs="1" />
971             <xsd:element ref="wsn-br:CreationTime"
972                 minOccurs="0" maxOccurs="1" />
973         </xsd:sequence>
974     </xsd:complexType>
975 </xsd:element>
976
977 <!-- ===== Message Types for NotificationBroker ===== -->
978 <xsd:element name="RegisterPublisher">
979     <xsd:complexType>
980         <xsd:sequence>
981             <xsd:element name="PublisherReference"
982                 type="wsa:EndpointReferenceType"
983                 minOccurs="0" maxOccurs="1" />
984             <xsd:element name="Topic"
985                 type="wsn-b:TopicExpressionType"
986                 minOccurs="0" maxOccurs="unbounded" />
987             <xsd:element name="Demand"
988                 type="xsd:boolean" default="false"
989                 minOccurs="0" maxOccurs="1" />
990             <xsd:element name="InitialTerminationTime"
991                 type="xsd:dateTime"
992                 minOccurs="0" maxOccurs="1" />
993             <xsd:any namespace="##other" processContents="lax"
994                 minOccurs="0" maxOccurs="unbounded" />
995         </xsd:sequence>
996     </xsd:complexType>
997 </xsd:element>
998
999 <xsd:element name="RegisterPublisherResponse">
1000     <xsd:complexType>
1001         <xsd:sequence>
1002             <xsd:element name="PublisherRegistrationReference"

```

```

1003         type="wsa:EndpointReferenceType"
1004         minOccurs="1" maxOccurs="1" />
1005     <xsd:element name="ConsumerReference"
1006         type="wsa:EndpointReferenceType"
1007         minOccurs="0" maxOccurs="1" />
1008
1009         </xsd:sequence>
1010     </xsd:complexType>
1011 </xsd:element>
1012
1013 <xsd:complexType name="PublisherRegistrationRejectedFaultType">
1014     <xsd:complexContent>
1015         <xsd:extension base="wsrf-bf:BaseFaultType" />
1016     </xsd:complexContent>
1017 </xsd:complexType>
1018 <xsd:element name="PublisherRegistrationRejectedFault"
1019     type="wsn-br:PublisherRegistrationRejectedFaultType" />
1020
1021 <xsd:complexType name="PublisherRegistrationFailedFaultType">
1022     <xsd:complexContent>
1023         <xsd:extension base="wsrf-bf:BaseFaultType" />
1024     </xsd:complexContent>
1025 </xsd:complexType>
1026 <xsd:element name="PublisherRegistrationFailedFault"
1027     type="wsn-br:PublisherRegistrationFailedFaultType" />
1028
1029
1030
1031 <xsd:element name="DestroyRegistration">
1032     <xsd:complexType>
1033         <xsd:sequence>
1034             <xsd:any namespace="##other" processContents="lax"
1035                 minOccurs="0" maxOccurs="unbounded" />
1036         </xsd:sequence>
1037         <xsd:anyAttribute />
1038     </xsd:complexType>
1039 </xsd:element>
1040
1041 <xsd:element name="DestroyRegistrationResponse">
1042     <xsd:complexType>
1043         <xsd:sequence>
1044             <xsd:any namespace="##other" processContents="lax"
1045                 minOccurs="0" maxOccurs="unbounded" />
1046         </xsd:sequence>
1047         <xsd:anyAttribute />
1048     </xsd:complexType>
1049 </xsd:element>
1050
1051 <xsd:complexType name="ResourceNotDestroyedFaultType">
1052     <xsd:complexContent>
1053         <xsd:extension base="wsrf-bf:BaseFaultType" />
1054     </xsd:complexContent>

```

1055
1056
1057
1058
1059

```
</xsd:complexType>  
<xsd:element name="ResourceNotDestroyedFault"  
             type="wsn-br:ResourceNotDestroyedFaultType" />  
  
</xsd:schema>
```

1060

Appendix C. WSDL 1.1

1061 The following illustrates the WSDL 1.1 for the Web service methods described in this
1062 specification:

```
1063 <?xml version="1.0" encoding="utf-8"?>
1064 <!--
1065 OASIS takes no position regarding the validity or scope of any
1066 intellectual property or other rights that might be claimed to pertain
1067 to the implementation or use of the technology described in this
1068 document or the extent to which any license under such rights might or
1069 might not be available; neither does it represent that it has made any
1070 effort to identify any such rights. Information on OASIS's procedures
1071 with respect to rights in OASIS specifications can be found at the OASIS
1072 website. Copies of claims of rights made available for publication and
1073 any assurances of licenses to be made available, or the result of an
1074 attempt made to obtain a general license or permission for the use of
1075 such proprietary rights by implementors or users of this specification,
1076 can be obtained from the OASIS Executive Director.
1077
1078 OASIS invites any interested party to bring to its attention any
1079 copyrights, patents or patent applications, or other proprietary rights
1080 which may cover technology that may be required to implement this
1081 specification. Please address the information to the OASIS Executive
1082 Director.
1083
1084 Copyright (C) OASIS Open (2004-2006). All Rights Reserved.
1085
1086 This document and translations of it may be copied and furnished to
1087 others, and derivative works that comment on or otherwise explain it or
1088 assist in its implementation may be prepared, copied, published and
1089 distributed, in whole or in part, without restriction of any kind,
1090 provided that the above copyright notice and this paragraph are included
1091 on all such copies and derivative works. However, this document itself
1092 may not be modified in any way, such as by removing the copyright notice
1093 or references to OASIS, except as needed for the purpose of developing
1094 OASIS specifications, in which case the procedures for copyrights
1095 defined in the OASIS Intellectual Property Rights document must be
1096 followed, or as required to translate it into languages other than
1097 English.
1098
1099 The limited permissions granted above are perpetual and will not be
1100 revoked by OASIS or its successors or assigns.
1101
1102 This document and the information contained herein is provided on an "AS
1103 IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
1104 INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
1105 INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
1106 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
```

```

1107  -->
1108
1109  <wsdl:definitions name="WS-BrokeredNotification"
1110    xmlns="http://schemas.xmlsoap.org/wsdl/"
1111    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1112    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1113    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1114    xmlns:wsa="http://www.w3.org/2005/08/addressing"
1115    xmlns:wsn-br="http://docs.oasis-open.org/wsn/br-2"
1116    xmlns:wsn-brw="http://docs.oasis-open.org/wsn/brw-2"
1117    xmlns:wsn-b="http://docs.oasis-open.org/wsn/b-2"
1118    xmlns:wsn-bw="http://docs.oasis-open.org/wsn/bw-2"
1119    xmlns:wsrf-bf="http://docs.oasis-open.org/wsr/bf-2"
1120    xmlns:wsrf-rw="http://docs.oasis-open.org/wsr/rw-2"
1121    targetNamespace="http://docs.oasis-open.org/wsn/brw-2">
1122
1123  <!-- ===== Imports ===== -->
1124  <wsdl:import namespace="http://docs.oasis-open.org/wsr/rw-2"
1125    location="http://docs.oasis-open.org/wsr/rw-2.wsdl"/>
1126
1127  <wsdl:import namespace="http://docs.oasis-open.org/wsn/bw-2"
1128    location="http://docs.oasis-open.org/wsn/bw-2.wsdl"/>
1129
1130  <!-- ===== Types Definitions ===== -->
1131  <wsdl:types>
1132    <xsd:schema>
1133      <xsd:import
1134        namespace="http://docs.oasis-open.org/wsn/br-2"
1135        schemaLocation="http://docs.oasis-open.org/wsn/br-2.xsd"/>
1136    </xsd:schema>
1137  </wsdl:types>
1138
1139  <!-- ===== NotificationBroker::RegisterPublisher ===== -->
1140  RegisterPublisher(PublisherReference, TopicExpression* ,
1141    [Demand], [InitialTerminationTime])
1142  returns: WS-Resource qualified EPR to a PublisherRegistration -->
1143  <wsdl:message name="RegisterPublisherRequest">
1144    <wsdl:part name="RegisterPublisherRequest"
1145      element="wsn-br:RegisterPublisher"/>
1146  </wsdl:message>
1147
1148  <wsdl:message name="RegisterPublisherResponse">
1149    <wsdl:part name="RegisterPublisherResponse"
1150      element="wsn-br:RegisterPublisherResponse"/>
1151  </wsdl:message>
1152
1153  <wsdl:message name="PublisherRegistrationRejectedFault">
1154    <wsdl:part name="PublisherRegistrationRejectedFault"
1155      element="wsn-br:PublisherRegistrationRejectedFault"/>
1156  </wsdl:message>
1157
1158  <wsdl:message name="PublisherRegistrationFailedFault">

```

```

1159     <wsdl:part name="PublisherRegistrationFailedFault"
1160         element="wsn-br:PublisherRegistrationFailedFault" />
1161 </wsdl:message>
1162
1163 <wsdl:message name="DestroyRegistrationRequest">
1164     <wsdl:part name="DestroyRegistrationRequest"
1165         element="wsn-br:DestroyRegistration" />
1166 </wsdl:message>
1167
1168 <wsdl:message name="DestroyRegistrationResponse">
1169     <wsdl:part name="DestroyRegistrationResponse"
1170         element="wsn-br:DestroyRegistrationResponse" />
1171 </wsdl:message>
1172
1173 <wsdl:message name="ResourceNotDestroyedFault">
1174     <wsdl:part name="ResourceNotDestroyedFault"
1175         element="wsn-br:ResourceNotDestroyedFault" />
1176 </wsdl:message>
1177
1178 <!-- ===== PortType Definitions ===== -->
1179
1180 <!-- ===== RegisterPublisher ===== -->
1181 <wsdl:portType name="RegisterPublisher">
1182     <wsdl:operation name="RegisterPublisher">
1183         <wsdl:input message="wsn-brw:RegisterPublisherRequest" />
1184         <wsdl:output message="wsn-brw:RegisterPublisherResponse" />
1185         <wsdl:fault name="ResourceUnknownFault"
1186             message="wsrf-rw:ResourceUnknownFault" />
1187         <wsdl:fault name="InvalidTopicExpressionFault"
1188             message="wsn-bw:InvalidTopicExpressionFault" />
1189         <wsdl:fault name="TopicNotSupportedFault"
1190             message="wsn-bw:TopicNotSupportedFault" />
1191         <wsdl:fault name="PublisherRegistrationRejectedFault"
1192             message="wsn-brw:PublisherRegistrationRejectedFault" />
1193         <wsdl:fault name="PublisherRegistrationFailedFault"
1194             message="wsn-brw:PublisherRegistrationFailedFault" />
1195         <wsdl:fault name="UnacceptableInitialTerminationTimeFault"
1196             message="wsn-bw:UnacceptableInitialTerminationTimeFault" />
1197     </wsdl:operation>
1198 </wsdl:portType>
1199
1200 <!-- ===== NotificationBroker PortType Definition ===== -->
1201 <wsdl:portType name="NotificationBroker">
1202     <!-- ===== extends NotificationConsumer ===== -->
1203     <wsdl:operation name="Notify">
1204         <wsdl:input message="wsn-bw:Notify" />
1205     </wsdl:operation>
1206
1207     <!-- ===== extends NotificationProducer ===== -->
1208     <wsdl:operation name="Subscribe">
1209         <wsdl:input message="wsn-bw:SubscribeRequest" />
1210         <wsdl:output message="wsn-bw:SubscribeResponse" />

```

```

1211 <wsdl:fault name="ResourceUnknownFault "
1212         message="wsrf-rw:ResourceUnknownFault " />
1213 <wsdl:fault name="InvalidFilterFault "
1214         message="wsn-bw:InvalidFilterFault " />
1215 <wsdl:fault name="TopicExpressionDialectUnknownFault "
1216         message="wsn-bw:TopicExpressionDialectUnknownFault " />
1217 <wsdl:fault name="InvalidTopicExpressionFault "
1218         message="wsn-bw:InvalidTopicExpressionFault " />
1219 <wsdl:fault name="TopicNotSupportedFault "
1220         message="wsn-bw:TopicNotSupportedFault " />
1221 <wsdl:fault name="InvalidProducerPropertiesExpressionFault "
1222         message="wsn-bw:InvalidProducerPropertiesExpressionFault " />
1223 <wsdl:fault name="InvalidMessageContentExpressionFault "
1224         message="wsn-bw:InvalidMessageContentExpressionFault " />
1225 <wsdl:fault name="UnacceptableInitialTerminationTimeFault "
1226         message="wsn-bw:UnacceptableInitialTerminationTimeFault " />
1227 <wsdl:fault name="UnrecognizedPolicyRequestFault "
1228         message="wsn-bw:UnrecognizedPolicyRequestFault " />
1229 <wsdl:fault name="UnsupportedPolicyRequestFault "
1230         message="wsn-bw:UnsupportedPolicyRequestFault " />
1231 <wsdl:fault name="NotifyMessageNotSupportedFault "
1232         message="wsn-bw:NotifyMessageNotSupportedFault " />
1233 <wsdl:fault name="SubscribeCreationFailedFault "
1234         message="wsn-bw:SubscribeCreationFailedFault " />
1235 </wsdl:operation>
1236 <wsdl:operation name="GetCurrentMessage">
1237     <wsdl:input message="wsn-bw:GetCurrentMessageRequest " />
1238     <wsdl:output message="wsn-bw:GetCurrentMessageResponse " />
1239     <wsdl:fault name="ResourceUnknownFault "
1240             message="wsrf-rw:ResourceUnknownFault " />
1241     <wsdl:fault name="TopicExpressionDialectUnknownFault "
1242             message="wsn-bw:TopicExpressionDialectUnknownFault " />
1243     <wsdl:fault name="InvalidTopicExpressionFault "
1244             message="wsn-bw:InvalidTopicExpressionFault " />
1245     <wsdl:fault name="TopicNotSupportedFault "
1246             message="wsn-bw:TopicNotSupportedFault " />
1247     <wsdl:fault name="NoCurrentMessageOnTopicFault "
1248             message="wsn-bw:NoCurrentMessageOnTopicFault " />
1249     <wsdl:fault name="MultipleTopicsSpecifiedFault "
1250             message="wsn-bw:MultipleTopicsSpecifiedFault " />
1251 </wsdl:operation>
1252
1253 <!-- ===== extends RegisterPublisher ===== -->
1254 <wsdl:operation name="RegisterPublisher">
1255     <wsdl:input message="wsn-brw:RegisterPublisherRequest " />
1256     <wsdl:output message="wsn-brw:RegisterPublisherResponse " />
1257     <wsdl:fault name="ResourceUnknownFault "
1258             message="wsrf-rw:ResourceUnknownFault " />
1259     <wsdl:fault name="InvalidTopicExpressionFault "
1260             message="wsn-bw:InvalidTopicExpressionFault " />
1261     <wsdl:fault name="TopicNotSupportedFault "
1262             message="wsn-bw:TopicNotSupportedFault " />
1263     <wsdl:fault name="PublisherRegistrationRejectedFault "

```

```

1264         message="wsn-brw:PublisherRegistrationRejectedFault" />
1265     <wsdl:fault name="PublisherRegistrationFailedFault"
1266         message="wsn-brw:PublisherRegistrationFailedFault" />
1267     <wsdl:fault name="UnacceptableInitialTerminationTimeFault"
1268         message="wsn-brw:UnacceptableInitialTerminationTimeFault" />
1269 </wsdl:operation>
1270
1271 </wsdl:portType>
1272
1273 <!-- ===== PublisherRegistrationManager PortType Definition ===== -->
1274 <wsdl:portType name="PublisherRegistrationManager">
1275
1276     <!--====DestroyRegistration:ImmediateResourceTermination====-->
1277     <wsdl:operation name="DestroyRegistration">
1278         <wsdl:input name="DestroyRegistrationRequest"
1279             message="wsn-brw:DestroyRegistrationRequest" />
1280         <wsdl:output name="DestroyRegistrationResponse"
1281             message="wsn-brw:DestroyRegistrationResponse" />
1282         <wsdl:fault name="ResourceUnknownFault"
1283             message="wsrf-rw:ResourceUnknownFault" />
1284         <wsdl:fault name="ResourceNotDestroyedFault"
1285             message="wsn-brw:ResourceNotDestroyedFault" />
1286     </wsdl:operation>
1287 </wsdl:portType>
1288 </wsdl:definitions>

```


Appendix D. Revision History

Rev	Date	By Whom	What
1.2 01	2004-05-12	Lily Liu	Initial version
1.2 02	2004-06-07	Dave Chappell	Updates and consistency check w/ other WS-N specs
1.2 03	2004-06-24	Lily Liu, Dave Chappell	Addition of a Goals and Requirements section and minor format changes
1.2 03	2004-07-12	Lily Liu	Addition of a status paragraph
1.3 01a – 1.3 01e	2005-02-01	Dave Chappell, Lily Liu	Series of issue resolution and consistency reviews with WS-BaseNotification
1.3 01f	2005-06-10	Lily Liu	Issues: 3.1, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.17, 3.18, 3.19, 3.20 Updated the Terminology, Introduction, and Security sections. Updated sections about NotificationBroker and PublisherRegistrationManager resource properties.
1.3 01g	2005-07-01	Lily Liu	Updated the status section. Changed term NotificationMessage to Notification. Added CreatePullPoint portType to NotificationBroker. Completed issue resolutions. Replaced the Abstract section.
1.3 02d	2005-11-04	Lily Liu	Included changes to address: WSN 2.62, WSN 3.23, WSN 3.24, WSN 3.25, WSN 3.26, WSN 3.28, and WSN 3.29 Resolved AI 137, AI 138, AI 141, AI 142 AI 144, and AI 145. Updated references.
1.3 pr02	2006-02-17	Lily Liu	Updated the document with changes suggested in the errata document (http://www.oasis-open.org/apps/org/workgroup/wsn/download.php/16679/wsn-

Rev	Date	By Whom	What
			ws_brokered_notification-1.3-errata.doc).
1.3 cd03	2006- 03-23	Lily Liu	Updated the document again with more changes suggested in the errata document (http://www.oasis-open.org/apps/org/workgroup/wsn/download.php/16679/wsn-ws_brokered_notification-1.3-errata.doc). Updated copyrights and status of the document.
1.3 cd03	2006- 04-20	Peter Niblett	Updated the References section to point at OASIS standard versions of WSRF specifications, and moved the WS-BaseNotification and WS-Topics references on to point to the latest committee drafts

1290

Appendix E. Notices

1291 OASIS takes no position regarding the validity or scope of an intellectual property or other rights
1292 that might be claimed to pertain to the implementation or use of the technology described in this
1293 document or the extent to which any license under such rights might or might not be available;
1294 neither does it represent that it has made any effort to identify any such rights. Information on
1295 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1296 website. Copies of claims of rights made available for publication and any assurances of licenses
1297 to be made available, or the result of an attempt made to obtain a general license or permission
1298 for the use of such proprietary rights by implementers or users of this specification, can be
1299 obtained from the OASIS Executive Director.

1300 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1301 applications, or other proprietary rights which may cover technology that may be required to
1302 implement this specification. Please address the information to the OASIS Executive Director.

1303 Copyright © OASIS Open 2004-2006. All Rights Reserved.

1304 This document and translations of it may be copied and furnished to others, and derivative works
1305 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1306 published and distributed, in whole or in part, without restriction of any kind, provided that the
1307 above copyright notice and this paragraph are included on all such copies and derivative works.
1308 However, this document itself does not be modified in any way, such as by removing the
1309 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
1310 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1311 Property Rights document must be followed, or as required to translate it into languages other
1312 than English.

1313 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1314 successors or assigns.

1315 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1316 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1317 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY
1318 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1319 PARTICULAR PURPOSE.