



Web Services Brokered Notification 1.3 (WS-BrokeredNotification)

Committee Specification, 31 July 2006

Document identifier:

wsn-ws_brokered_notification-1.3-spec-cs-01

Location:

http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-cs-01.pdf

Editors:

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

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

Abstract:

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

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

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

35 **Status:**

36 This document is published by the OASIS WS-Notification Technical Committee as a
37 "Committee Specification".

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

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

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

50

51 **Table of Contents**

52 1 Introduction 4
53 1.1 Goals and Requirements 4
54 1.1.1 Requirements..... 4
55 1.1.2 Non-Goals 5
56 1.2 Notational Conventions 5
57 1.3 Namespaces 7
58 1.4 Fault Definitions..... 7
59 2 Relationship to Other Specifications 8
60 3 Terminology and Concepts 9
61 4 Publishing..... 12
62 5 NotificationBroker Interface..... 15
63 5.1 NotificationBroker Resource Properties 16
64 5.2 Notify 16
65 5.3 Subscribe 16
66 5.4 GetCurrentMessage 17
67 5.5 RegisterPublisher 17
68 5.6 CreatePullPoint 17
69 6 RegisterPublisher Interface..... 18
70 6.1 RegisterPublisher 18
71 6.1.1 Example SOAP Encoding of the RegisterPublisher Message Exchange 22
72 7 PublisherRegistrationManager Interface 24
73 7.1 PublisherRegistration Resource Properties 24
74 7.2 DestroyRegistration..... 25
75 7.2.1 Example SOAP Encoding of the DestroyRegistration Message Exchange 26
76 8 Security Considerations 27
77 8.1 Securing PublisherRegistration..... 27
78 9 References..... 28
79 9.1 Normative 28
80 9.2 Non-Normative 28
81 Appendix A. Acknowledgments 30
82 Appendix B. XML Schema..... 31
83 Appendix C. WSDL 1.1..... 36
84 Appendix D. Revision History 41
85 Appendix E. Notices 43

86

1 Introduction

87

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

91

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

97

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

99

100

1.1 Goals and Requirements

101

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

102

103

104

105

1.1.1 Requirements

106

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

107

108

- **Must allow for a notification broker as an intermediary.** A NotificationBroker is an intermediary Web service that decouples NotificationConsumers from Publishers. A notification broker can relieve a Publisher from having to implement message exchanges associated with NotificationProducer; the NotificationBroker takes on the duties of subscription management and distributing Notifications on behalf of the Publisher. It implements NotificationProducer interface. It may implement SubscriptionManager or may delegate the subscription management work to another component.

109

110

111

112

113

114

115

- **Must allow for federation of brokers.** It must be possible to build configurations with multiple intermediary broker services in a dynamic fashion. This specification must allow for a variety of broker topology usage patterns. Among other things, these allow for greater scalability and permit sharing of administrative workload.

116

117

118

119

- **Must provide runtime metadata:** There must be a mechanism that lets a potential Subscriber discover what elements available for a subscription are provided by a NotificationBroker, and in what formats the subscription for a notification can be made.

120

121

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

135 1.1.2 Non-Goals

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

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

145 1.2 Notational Conventions

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

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

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

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

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

166

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

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

181 **1.3 Namespaces**

182 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

183 **1.4 Fault Definitions**

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

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

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

189

2 Relationship to Other Specifications

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

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

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

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

204 3 Terminology and Concepts

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

207 Publisher:

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

219 NotificationBroker:

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

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

- 276
- 277
- 278
- 279
- 280
- 281
- 282
- 283
- 284
- 285
- 286
- 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.

287

4 Publishing

288 There are three distinct stages in the Notification process

- 289 • Observation of the Situation and its noteworthy characteristics;
- 290 • Creation of the Notification artifact that captures the noteworthy characteristics of the
291 Situation; and
- 292 • Distribution of copies of the Notification to zero or more interested parties.

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

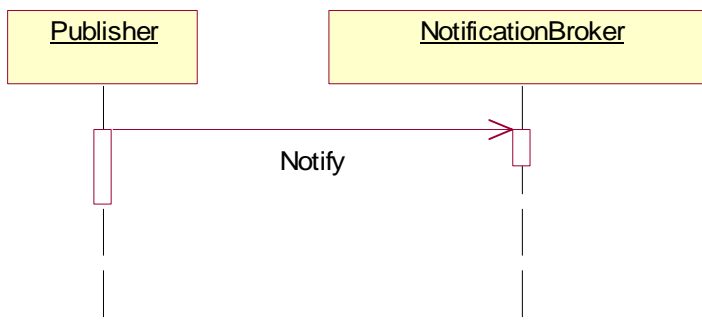
296 However, the WS-Notification family of specifications does specify how dissemination of
297 messages SHOULD occur. There are two dominant patterns by which Notifications are
298 disseminated in WS-Notification: direct and brokered.

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

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

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

309 The following figure illustrates simple publishing:



310

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

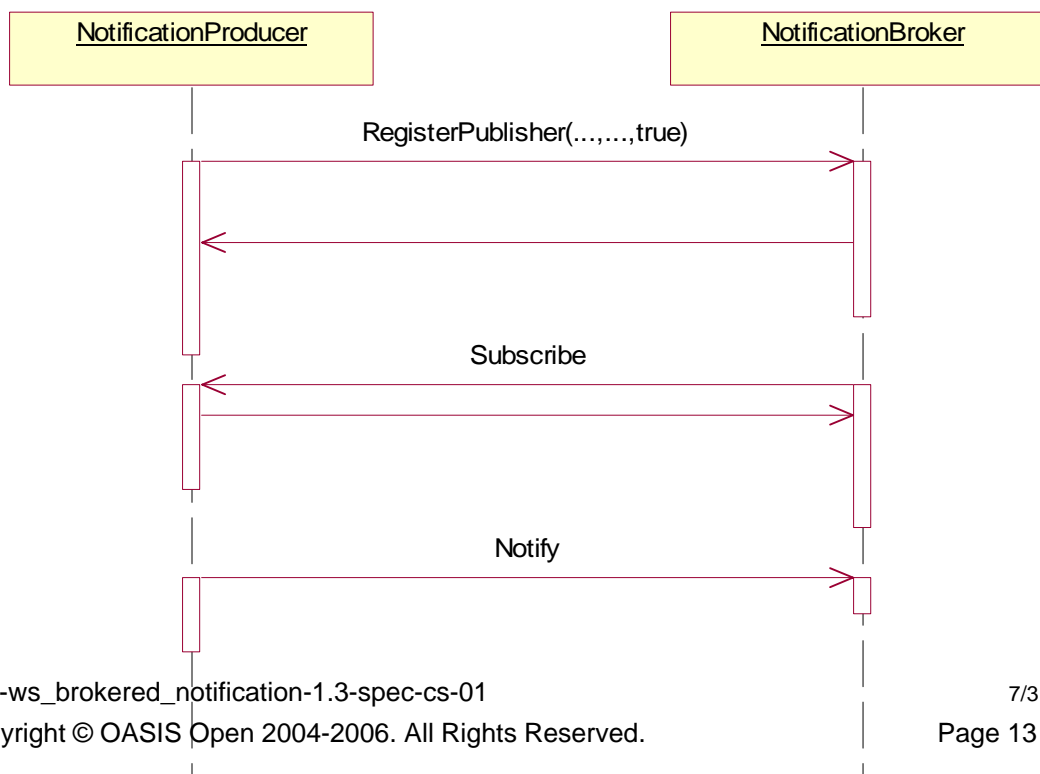
313 Situation. The dissemination step occurs when the Publisher sends the Notify message to the
314 NotificationBroker.

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

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

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

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



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

338 5 NotificationBroker Interface

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

342 A NotificationBroker MAY be a WS-Resource, and if it is, it MUST support the required message
343 exchanges defined by the [\[WS-ResourceProperties\]](#) specification and MAY support the optional
344 message exchanges defined by WS-ResourceProperties.

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

- 347 • NotificationProducer
- 348 • NotificationConsumer
- 349 • RegisterPublisher

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

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

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

359

360 5.1 NotificationBroker Resource Properties

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

```
368 ...  
369     targetNamespace="http://docs.oasis-open.org/wsn/br-2">  
370 ...  
371     <xsd:element name="RequiresRegistration" type="xsd:boolean"/>  
372 ...
```

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

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

376 This resource property element is further constrained as follows:

377 /wsn-br:RequiresRegistration

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

380 5.2 Notify

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

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

387 5.3 Subscribe

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

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

395 If the processing of a Subscribe message is successful, the NotificationBroker MUST produce a
396 response message, as described in [WS-BaseNotification](#), containing an endpoint reference to a
397 Subscription resource representing a Subscription created as a result of processing the

398 Subscribe request. Otherwise, the NotificationBroker must fault. WS-BaseNotification defines a
399 set of these faults.

400 **5.4 GetCurrentMessage**

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

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

407 **5.5 RegisterPublisher**

408 The NotificationBroker MUST support the RegisterPublisher message exchange from the
409 RegisterPublisher interface.

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

415 **5.6 CreatePullPoint**

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

420 6 RegisterPublisher Interface

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

425 6.1 RegisterPublisher

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

```
430 ...  
431 <wsn-br:RegisterPublisher>  
432   <wsn-br:PublisherReference>  
433     wsa:EndpointReferenceType  
434   </wsn-br:PublisherReference?>  
435   <wsn-br:Topic Dialect = "xsd:anyURI" >  
436     {any} ?  
437   </wsn-br:Topic> *  
438   <wsn-br:Demand>  
439     xsd:boolean  
440   </wsn-br:Demand?>  
441   <wsn-br:InitialTerminationTime>  
442     xsd:dateTime  
443   </wsn-br:InitialTerminationTime?>  
444   {any} *  
445 </wsn-br:RegisterPublisher>  
446 ...
```

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

450
451 The components of the RegisterPublisher request message are further described as follows:
452 /wsn-br:RegisterPublisher/PublisherReference

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

458 /wsn-br:RegisterPublisher/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:RegisterPublisher/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:RegisterPublisher/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, the RegisterPublisher request
475 MUST return an UnacceptableInitialTerminationTimeFault message. If the value is not "in
476 the future" relative to the current time as known by the NotificationBroker, the
477 RegisterPublisher request MUST also return an UnacceptableInitialTerminationTimeFault
478 message.

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

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

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

494 /wsn-br:RegisterPublisher/{any}

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

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

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

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

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

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

```
516 ...  
517 <wsn-br:RegisterPublisherResponse>  
518   <wsn-br:PublisherRegistrationReference>  
519     <wsa:Address>  
520       Address of PublisherRegistration Manager  
521     </wsa:Address>  
522     ...  
523   </wsn-br:PublisherRegistrationReference>  
524   <wsn-br:ConsumerReference>  
525     <wsa:Address>  
526       Address of a NotificationConsumer with which the  
527       Publisher is registered  
528     </wsa:Address>  
529     ...  
530   </wsn-br:ConsumerReference>  
531 </wsn-br:RegisterPublisherResponse>  
532 ...
```

533 The WS-Addressing [action] Message Addressing Property MUST contain the URI
534 <http://docs.oasis-open.org/wsn/brw-2/RegisterPublisher/RegisterPublisherResponse>
535 The components of the RegisterPublisher response message are further described as follows:
536 /wsn-br:RegisterPublisherResponse/PublisherRegistrationReference
537 A WS-Addressing endpoint reference to the PublisherRegistration resource created by
538 the RegisterPublisher request message. This element MUST be present in the

539 RegisterPublisher response message. The NotificationBroker may choose to include
540 extra information such as reference parameters in this reference.

541 /wsn-br:RegisterPublisherResponse/ConsumerReference

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

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

546 The NotificationBroker MAY use this as a mechanism for identifying the Publisher as
547 having registered.

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

552

553 ResourceUnknownFault

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

557 InvalidTopicExpressionFault

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

560 TopicNotSupportedFault

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

563 PublisherRegistrationRejectedFault

- 564 • The publisher registration is rejected by the NotificationBroker. The NotificationBroker
565 MAY provide a hint in the fault message indicating why the registration is rejected.

566 PublisherRegistrationFailedFault

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

569 UnacceptableInitialTerminationTimeFault

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

6.1.1 Example SOAP Encoding of the RegisterPublisher Message Exchange

573
574

575 The following is a non-normative example of a RegisterPublisher request message using SOAP
576 1.1 [SOAP 1.1] or SOAP 1.2 [SOAP 1.2]:

```
577 <s:Envelope ... >
578   <s:Header>
579     <wsa:Action>
580       http://docs.oasis-open.org/wsn/brw-
581 2/RegisterPublisher/RegisterPublisherRequest
582     </wsa:Action>
583     ...
584   </s:Header>
585   <s:Body>
586     <wsn-br:RegisterPublisher>
587       <wsn-br:PublisherReference>
588         <wsa:Address>
589           http://www.example.org/PublisherEndpoint
590         </wsa:Address>
591         <wsa:ReferenceParameters>
592           <npex: NPResourceDisambiguator>
593             uuid:84decd55-7d3f-65ad-ac44-675d9fce5d22
594           </npex: NPResourceDisambiguator>
595         </wsa:ReferenceParameters>
596       </wsn-br:PublisherReference>
597       <wsn-br:Topic Dialect="http://docs.oasis-open.org/wsn/t-
598 1/TopicExpression/Simple">
599         npex:SomeTopic
600       </wsn-br:Topic>
601       <wsn-br:Demand>
602         true
603       </wsn-br:Demand>
604       <wsn-br:InitialTerminationTime>
605         2003-12-25T00:00:00.000000Z
606       </wsn-br:InitialTerminationTime>
607     </wsn-br:RegisterPublisher>
608   </s:Body>
609 </s:Envelope>
```

610

611 The following is a non-normative example of a RegisterPublisher response message using
612 SOAP:

```
613 <s:Envelope ... >
614   <s:Header>
615     <wsa:Action>
616       http://docs.oasis-open.org/wsn/brw-
617 2/RegisterPublisher/RegisterPublisherResponse
618     </wsa:Action>
619     ...
620   </s:Header>
```

```
621 <s:Body>
622   <wsn-br:RegisterPublisherResponse>
623     <wsn-br:PublisherRegistrationReference>
624       <wsa:Address>
625         http://www.example.org/PublisherRegistrationManager
626       </wsa:Address>
627       <wsa:ReferenceParameters>
628         <npex:NPubResourceId>
629           uuid:95fefeb3-f37d-5dfe-44fe-221d9fceed99
630         </npex:NPubResourceId>
631       </wsa:ReferenceParameters>
632     </wsn-br:PublisherRegistrationReference>
633     <wsn-br:ConsumerReference>
634       <wsa:Address>
635         http://www.example.org/NotificationConsumer
636       </wsa:Address>
637       ...
638     </wsn-br:ConsumerReference>
639   </wsn-br:RegisterPublisherResponse>
640 </s:Body>
641 </s:Envelope>
```

642

7 PublisherRegistrationManager Interface

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

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

7.1 PublisherRegistration Resource Properties

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

658

659

660

661

662

663

664

665

666

```
.....
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" />
...
```

667

668

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

669

670

671

672

673

674

675

676

```
<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" />
```

677

These resource property elements are further constrained as follows:

678

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

679

680

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

681 /wsn-br:CreationTime

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

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

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

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

692 **7.2 DestroyRegistration**

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

```
696 <wsn-br:DestroyRegistration>  
697   {any} *  
698 </wsn-br:DestroyRegistration>  
700
```

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

703 The DestroyRegistration request message allows for open content and contains an extension
704 component

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

706

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

```
711 <wsn-br:DestroyRegistrationResponse>  
712   {any} *  
713 </wsn-br:DestroyRegistrationResponse>  
715
```

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

717 [http://docs.oasis-open.org/wsn/brw-
718 2/PublisherRegistrationManager/DestroyRegistrationResponse](http://docs.oasis-open.org/wsn/brw-2/PublisherRegistrationManager/DestroyRegistrationResponse).

719 If the PublisherRegistrationManager does not respond to the DestroyRegistration request
720 message with the DestroyRegistrationResponse message, then it MUST send a fault. This

721 specification defines the following faults associated with failure to process the
722 DestroyRegistration request message:

723 ResourceUnknownFault

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

727 ResourceNotDestroyedFault

- 728 • The PublisherRegistrationManager was unable to destroy the PublisherRegistration
729 resource for some reason.

730 **7.2.1 Example SOAP Encoding of the DestroyRegistration Message** 731 **Exchange**

732 The following is a non-normative example of a DestroyRegistration request message using
733 SOAP:

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

746 The following is a non-normative example of a DestroyRegistration response message using
747 SOAP:

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

760

8 Security Considerations

761

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

762

763

8.1 Securing PublisherRegistration

764

765

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

766

767

1. only authorized Publishers can register with a NotificationBroker

768

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

769

770

3. only authorized principals can modify or delete PublisherRegistration resource

771

772

773

774

775

776

777

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.

778

779

780

781

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:

782

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

783

- Refusing to relay messages from unauthorized publishers

784

- Imposing security measurements on all messaging routing through the broker

785

- Providing convenience in messaging security management based on topics.

786

787

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

788

9 References

789

9.1 Normative

790

[RFC2119]

791

S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", IETF

792

RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>

793

[XML]

794

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

795

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

796

[XML-InfoSet]

797

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

798

[WS-Addressing]

799

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

800

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

801

[WS-BaseNotification]

802

"Web Services Base Notification 1.3".

803

http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-cs-01.pdf

804

[WS-Topics]

805

"Web Services Topics 1.3".

806

http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-cs-01.pdf

807

[WS-Resource]

808

"Web Services Resource 1.2", OASIS Standard.

809

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

810

[WS-ResourceLifetime]

811

"Web Services Resource Lifetime 1.2", OASIS Standard.

812

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

813

[WS-ResourceProperties]

814

"Web Services Resource Properties 1.2", OASIS Standard.

815

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

816

[WS-BaseFaults]

817

"Web Services Base Faults 1.2", OASIS Standard.

818

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

819

820

9.2 Non-Normative

821

[SOAP 1.1]

822

"Simple Object Access Protocol (SOAP) 1.1"

823

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

824 **[SOAP 1.2]**
825 “SOAP Version 1.2 Part 1: Messaging Framework”, W3C Recommendation.
826 <http://www.w3.org/TR/soap12-part1/>
827 **[WS-Security]**
828 “Web Services Security: SOAP Message Security 1.0”, OASIS Standard.
829 [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)
830 1.0.pdf

831

Appendix A. Acknowledgments

832 The following individuals were members of the committee during the development of this
833 specification:

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

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

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

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

Appendix B. XML Schema

854 The XML types and elements used in WS-BrokeredNotification are defined in the following XML
855 Schema

```
856 <?xml version="1.0" encoding="UTF-8"?>
857 <!--
858 OASIS takes no position regarding the validity or scope of any
859 intellectual property or other rights that might be claimed to pertain
860 to the implementation or use of the technology described in this
861 document or the extent to which any license under such rights might or
862 might not be available; neither does it represent that it has made any
863 effort to identify any such rights. Information on OASIS's procedures
864 with respect to rights in OASIS specifications can be found at the OASIS
865 website. Copies of claims of rights made available for publication and
866 any assurances of licenses to be made available, or the result of an
867 attempt made to obtain a general license or permission for the use of
868 such proprietary rights by implementors or users of this specification,
869 can be obtained from the OASIS Executive Director.
870
871 OASIS invites any interested party to bring to its attention any
872 copyrights, patents or patent applications, or other proprietary rights
873 which may cover technology that may be required to implement this
874 specification. Please address the information to the OASIS Executive
875 Director.
876
877 Copyright (C) OASIS Open (2004-2006). All Rights Reserved.
878
879 This document and translations of it may be copied and furnished to
880 others, and derivative works that comment on or otherwise explain it or
881 assist in its implementation may be prepared, copied, published and
882 distributed, in whole or in part, without restriction of any kind,
883 provided that the above copyright notice and this paragraph are included
884 on all such copies and derivative works. However, this document itself
885 may not be modified in any way, such as by removing the copyright notice
886 or references to OASIS, except as needed for the purpose of developing
887 OASIS specifications, in which case the procedures for copyrights
888 defined in the OASIS Intellectual Property Rights document must be
889 followed, or as required to translate it into languages other than
890 English.
891
892 The limited permissions granted above are perpetual and will not be
893 revoked by OASIS or its successors or assigns.
894
895 This document and the information contained herein is provided on an "AS
896 IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
897 INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
898 INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
899 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
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
951
952
953

```
<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"
          minOccurs="0" maxOccurs="unbounded" />

```



```

954         <xsd:element ref="wstop:TopicSet"
955             minOccurs="0" maxOccurs="1" />
956         <!-- NotificationBroker specific -->
957         <xsd:element ref="wsn-br:RequiresRegistration"
958             minOccurs="1" maxOccurs="1" />
959     </xsd:sequence>
960 </xsd:complexType>
961 </xsd:element>
962
963 <!-- ===== Resource Properties for PublisherRegistration ===== -->
964 <xsd:element name="PublisherRegistrationRP">
965     <xsd:complexType>
966         <xsd:sequence>
967             <xsd:element ref="wsn-br:PublisherReference"
968                 minOccurs="0" maxOccurs="1" />
969             <xsd:element ref="wsn-br:Topic"
970                 minOccurs="0" maxOccurs="unbounded" />
971             <xsd:element ref="wsn-br:Demand"
972                 minOccurs="1" maxOccurs="1" />
973             <xsd:element ref="wsn-br:CreationTime"
974                 minOccurs="0" maxOccurs="1" />
975         </xsd:sequence>
976     </xsd:complexType>
977 </xsd:element>
978
979 <!-- ===== Message Types for NotificationBroker ===== -->
980 <xsd:element name="RegisterPublisher">
981     <xsd:complexType>
982         <xsd:sequence>
983             <xsd:element name="PublisherReference"
984                 type="wsa:EndpointReferenceType"
985                 minOccurs="0" maxOccurs="1" />
986             <xsd:element name="Topic"
987                 type="wsn-b:TopicExpressionType"
988                 minOccurs="0" maxOccurs="unbounded" />
989             <xsd:element name="Demand"
990                 type="xsd:boolean" default="false"
991                 minOccurs="0" maxOccurs="1" />
992             <xsd:element name="InitialTerminationTime"
993                 type="xsd:dateTime"
994                 minOccurs="0" maxOccurs="1" />
995             <xsd:any namespace="##other" processContents="lax"
996                 minOccurs="0" maxOccurs="unbounded" />
997         </xsd:sequence>
998     </xsd:complexType>
999 </xsd:element>
1000
1001 <xsd:element name="RegisterPublisherResponse">
1002     <xsd:complexType>
1003         <xsd:sequence>
1004             <xsd:element name="PublisherRegistrationReference"
1005                 type="wsa:EndpointReferenceType"
1006                 minOccurs="1" maxOccurs="1" />

```

```

1007         <xsd:element name="ConsumerReference"
1008                     type="wsa:EndpointReferenceType"
1009                     minOccurs="0" maxOccurs="1" />
1010
1011         </xsd:sequence>
1012     </xsd:complexType>
1013 </xsd:element>
1014
1015 <xsd:complexType name="PublisherRegistrationRejectedFaultType">
1016     <xsd:complexContent>
1017         <xsd:extension base="wsrf-bf:BaseFaultType" />
1018     </xsd:complexContent>
1019 </xsd:complexType>
1020 <xsd:element name="PublisherRegistrationRejectedFault"
1021             type="wsn-br:PublisherRegistrationRejectedFaultType" />
1022
1023 <xsd:complexType name="PublisherRegistrationFailedFaultType">
1024     <xsd:complexContent>
1025         <xsd:extension base="wsrf-bf:BaseFaultType" />
1026     </xsd:complexContent>
1027 </xsd:complexType>
1028 <xsd:element name="PublisherRegistrationFailedFault"
1029             type="wsn-br:PublisherRegistrationFailedFaultType" />
1030
1031
1032
1033 <xsd:element name="DestroyRegistration">
1034     <xsd:complexType>
1035         <xsd:sequence>
1036             <xsd:any namespace="##other" processContents="lax"
1037                 minOccurs="0" maxOccurs="unbounded" />
1038         </xsd:sequence>
1039         <xsd:anyAttribute />
1040     </xsd:complexType>
1041 </xsd:element>
1042
1043 <xsd:element name="DestroyRegistrationResponse">
1044     <xsd:complexType>
1045         <xsd:sequence>
1046             <xsd:any namespace="##other" processContents="lax"
1047                 minOccurs="0" maxOccurs="unbounded" />
1048         </xsd:sequence>
1049         <xsd:anyAttribute />
1050     </xsd:complexType>
1051 </xsd:element>
1052
1053 <xsd:complexType name="ResourceNotDestroyedFaultType">
1054     <xsd:complexContent>
1055         <xsd:extension base="wsrf-bf:BaseFaultType" />
1056     </xsd:complexContent>
1057 </xsd:complexType>
1058 <xsd:element name="ResourceNotDestroyedFault"
1059             type="wsn-br:ResourceNotDestroyedFaultType" />

```

1060
1061

```
</xsd:schema>
```

1062

Appendix C. WSDL 1.1

1063 The following illustrates the WSDL 1.1 for the Web service methods described in this
1064 specification:

```
1065 <?xml version="1.0" encoding="utf-8"?>
1066 <!--
1067 OASIS takes no position regarding the validity or scope of any
1068 intellectual property or other rights that might be claimed to pertain
1069 to the implementation or use of the technology described in this
1070 document or the extent to which any license under such rights might or
1071 might not be available; neither does it represent that it has made any
1072 effort to identify any such rights. Information on OASIS's procedures
1073 with respect to rights in OASIS specifications can be found at the OASIS
1074 website. Copies of claims of rights made available for publication and
1075 any assurances of licenses to be made available, or the result of an
1076 attempt made to obtain a general license or permission for the use of
1077 such proprietary rights by implementors or users of this specification,
1078 can be obtained from the OASIS Executive Director.
1079
1080 OASIS invites any interested party to bring to its attention any
1081 copyrights, patents or patent applications, or other proprietary rights
1082 which may cover technology that may be required to implement this
1083 specification. Please address the information to the OASIS Executive
1084 Director.
1085
1086 Copyright (C) OASIS Open (2004-2006). All Rights Reserved.
1087
1088 This document and translations of it may be copied and furnished to
1089 others, and derivative works that comment on or otherwise explain it or
1090 assist in its implementation may be prepared, copied, published and
1091 distributed, in whole or in part, without restriction of any kind,
1092 provided that the above copyright notice and this paragraph are included
1093 on all such copies and derivative works. However, this document itself
1094 may not be modified in any way, such as by removing the copyright notice
1095 or references to OASIS, except as needed for the purpose of developing
1096 OASIS specifications, in which case the procedures for copyrights
1097 defined in the OASIS Intellectual Property Rights document must be
1098 followed, or as required to translate it into languages other than
1099 English.
1100
1101 The limited permissions granted above are perpetual and will not be
1102 revoked by OASIS or its successors or assigns.
1103
1104 This document and the information contained herein is provided on an "AS
1105 IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
1106 INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
1107 INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
1108 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
1109 -->
```

wsn-ws_brokered_notification-1.3-spec-cs-01

7/31/2006

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

Page 36 of 43

```

1110 <wsdl:definitions name="WS-BrokeredNotification"
1111   xmlns="http://schemas.xmlsoap.org/wsdl/"
1112   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1113   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1114   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1115   xmlns:wsa="http://www.w3.org/2005/08/addressing"
1116   xmlns:wsn-br="http://docs.oasis-open.org/wsn/br-2"
1117   xmlns:wsn-brw="http://docs.oasis-open.org/wsn/brw-2"
1118   xmlns:wsn-b="http://docs.oasis-open.org/wsn/b-2"
1119   xmlns:wsn-bw="http://docs.oasis-open.org/wsn/bw-2"
1120   xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-2"
1121   xmlns:wsrf-rw="http://docs.oasis-open.org/wsrf/rw-2"
1122   targetNamespace="http://docs.oasis-open.org/wsn/brw-2">
1123
1124
1125 <!-- ===== Imports ===== -->
1126 <wsdl:import namespace="http://docs.oasis-open.org/wsrf/rw-2"
1127   location="http://docs.oasis-open.org/wsrf/rw-2.wsdl"/>
1128
1129 <wsdl:import namespace="http://docs.oasis-open.org/wsn/bw-2"
1130   location="http://docs.oasis-open.org/wsn/bw-2.wsdl"/>
1131
1132 <!-- ===== Types Definitions ===== -->
1133 <wsdl:types>
1134   <xsd:schema>
1135     <xsd:import
1136       namespace="http://docs.oasis-open.org/wsn/br-2"
1137       schemaLocation="http://docs.oasis-open.org/wsn/br-2.xsd"/>
1138   </xsd:schema>
1139 </wsdl:types>
1140
1141 <!-- ===== NotificationBroker::RegisterPublisher =====
1142 RegisterPublisher(PublisherReference, TopicExpression* ,
1143   [Demand], [InitialTerminationTime])
1144 returns: WS-Resource qualified EPR to a PublisherRegistration -->
1145 <wsdl:message name="RegisterPublisherRequest">
1146   <wsdl:part name="RegisterPublisherRequest"
1147     element="wsn-br:RegisterPublisher"/>
1148 </wsdl:message>
1149
1150 <wsdl:message name="RegisterPublisherResponse">
1151   <wsdl:part name="RegisterPublisherResponse"
1152     element="wsn-br:RegisterPublisherResponse"/>
1153 </wsdl:message>
1154
1155 <wsdl:message name="PublisherRegistrationRejectedFault">
1156   <wsdl:part name="PublisherRegistrationRejectedFault"
1157     element="wsn-br:PublisherRegistrationRejectedFault"/>
1158 </wsdl:message>
1159
1160 <wsdl:message name="PublisherRegistrationFailedFault">
1161   <wsdl:part name="PublisherRegistrationFailedFault"
1162     element="wsn-br:PublisherRegistrationFailedFault"/>

```

```

1163 </wsdl:message>
1164
1165 <wsdl:message name="DestroyRegistrationRequest">
1166   <wsdl:part name="DestroyRegistrationRequest"
1167     element="wsn-br:DestroyRegistration" />
1168 </wsdl:message>
1169
1170 <wsdl:message name="DestroyRegistrationResponse">
1171   <wsdl:part name="DestroyRegistrationResponse"
1172     element="wsn-br:DestroyRegistrationResponse" />
1173 </wsdl:message>
1174
1175 <wsdl:message name="ResourceNotDestroyedFault">
1176   <wsdl:part name="ResourceNotDestroyedFault"
1177     element="wsn-br:ResourceNotDestroyedFault" />
1178 </wsdl:message>
1179
1180 <!-- ===== PortType Definitions ===== -->
1181
1182 <!-- ===== RegisterPublisher ===== -->
1183 <wsdl:portType name="RegisterPublisher">
1184   <wsdl:operation name="RegisterPublisher">
1185     <wsdl:input message="wsn-brw:RegisterPublisherRequest" />
1186     <wsdl:output message="wsn-brw:RegisterPublisherResponse" />
1187     <wsdl:fault name="ResourceUnknownFault"
1188       message="wsrf-rw:ResourceUnknownFault" />
1189     <wsdl:fault name="InvalidTopicExpressionFault"
1190       message="wsn-bw:InvalidTopicExpressionFault" />
1191     <wsdl:fault name="TopicNotSupportedFault"
1192       message="wsn-bw:TopicNotSupportedFault" />
1193     <wsdl:fault name="PublisherRegistrationRejectedFault"
1194       message="wsn-brw:PublisherRegistrationRejectedFault" />
1195     <wsdl:fault name="PublisherRegistrationFailedFault"
1196       message="wsn-brw:PublisherRegistrationFailedFault" />
1197     <wsdl:fault name="UnacceptableInitialTerminationTimeFault"
1198       message="wsn-bw:UnacceptableInitialTerminationTimeFault" />
1199   </wsdl:operation>
1200 </wsdl:portType>
1201
1202 <!-- ===== NotificationBroker PortType Definition ===== -->
1203 <wsdl:portType name="NotificationBroker">
1204   <!-- ===== extends NotificationConsumer ===== -->
1205   <wsdl:operation name="Notify">
1206     <wsdl:input message="wsn-bw:Notify" />
1207   </wsdl:operation>
1208
1209   <!-- ===== extends NotificationProducer ===== -->
1210   <wsdl:operation name="Subscribe">
1211     <wsdl:input message="wsn-bw:SubscribeRequest" />
1212     <wsdl:output message="wsn-bw:SubscribeResponse" />
1213     <wsdl:fault name="ResourceUnknownFault"
1214       message="wsrf-rw:ResourceUnknownFault" />
1215     <wsdl:fault name="InvalidFilterFault"

```

```

1216         message="wsn-bw:InvalidFilterFault" />
1217     <wsdl:fault name="TopicExpressionDialectUnknownFault"
1218         message="wsn-bw:TopicExpressionDialectUnknownFault" />
1219     <wsdl:fault name="InvalidTopicExpressionFault"
1220         message="wsn-bw:InvalidTopicExpressionFault" />
1221     <wsdl:fault name="TopicNotSupportedFault"
1222         message="wsn-bw:TopicNotSupportedFault" />
1223     <wsdl:fault name="InvalidProducerPropertiesExpressionFault"
1224         message="wsn-bw:InvalidProducerPropertiesExpressionFault" />
1225     <wsdl:fault name="InvalidMessageContentExpressionFault"
1226         message="wsn-bw:InvalidMessageContentExpressionFault" />
1227     <wsdl:fault name="UnacceptableInitialTerminationTimeFault"
1228         message="wsn-bw:UnacceptableInitialTerminationTimeFault" />
1229     <wsdl:fault name="UnrecognizedPolicyRequestFault"
1230         message="wsn-bw:UnrecognizedPolicyRequestFault" />
1231     <wsdl:fault name="UnsupportedPolicyRequestFault"
1232         message="wsn-bw:UnsupportedPolicyRequestFault" />
1233     <wsdl:fault name="NotifyMessageNotSupportedFault"
1234         message="wsn-bw:NotifyMessageNotSupportedFault" />
1235     <wsdl:fault name="SubscribeCreationFailedFault"
1236         message="wsn-bw:SubscribeCreationFailedFault" />
1237 </wsdl:operation>
1238 <wsdl:operation name="GetCurrentMessage">
1239     <wsdl:input message="wsn-bw:GetCurrentMessageRequest" />
1240     <wsdl:output message="wsn-bw:GetCurrentMessageResponse" />
1241     <wsdl:fault name="ResourceUnknownFault"
1242         message="wsrf-rw:ResourceUnknownFault" />
1243     <wsdl:fault name="TopicExpressionDialectUnknownFault"
1244         message="wsn-bw:TopicExpressionDialectUnknownFault" />
1245     <wsdl:fault name="InvalidTopicExpressionFault"
1246         message="wsn-bw:InvalidTopicExpressionFault" />
1247     <wsdl:fault name="TopicNotSupportedFault"
1248         message="wsn-bw:TopicNotSupportedFault" />
1249     <wsdl:fault name="NoCurrentMessageOnTopicFault"
1250         message="wsn-bw:NoCurrentMessageOnTopicFault" />
1251     <wsdl:fault name="MultipleTopicsSpecifiedFault"
1252         message="wsn-bw:MultipleTopicsSpecifiedFault" />
1253 </wsdl:operation>
1254
1255 <!-- ===== extends RegisterPublisher ===== -->
1256 <wsdl:operation name="RegisterPublisher">
1257     <wsdl:input message="wsn-brw:RegisterPublisherRequest" />
1258     <wsdl:output message="wsn-brw:RegisterPublisherResponse" />
1259     <wsdl:fault name="ResourceUnknownFault"
1260         message="wsrf-rw:ResourceUnknownFault" />
1261     <wsdl:fault name="InvalidTopicExpressionFault"
1262         message="wsn-bw:InvalidTopicExpressionFault" />
1263     <wsdl:fault name="TopicNotSupportedFault"
1264         message="wsn-bw:TopicNotSupportedFault" />
1265     <wsdl:fault name="PublisherRegistrationRejectedFault"
1266         message="wsn-brw:PublisherRegistrationRejectedFault" />
1267     <wsdl:fault name="PublisherRegistrationFailedFault"
1268         message="wsn-brw:PublisherRegistrationFailedFault" />

```

```

1269         <wsdl:fault name="UnacceptableInitialTerminationTimeFault"
1270             message="wsn-bw:UnacceptableInitialTerminationTimeFault" />
1271     </wsdl:operation>
1272
1273 </wsdl:portType>
1274
1275 <!-- ===== PublisherRegistrationManager PortType Definition ===== -->
1276 <wsdl:portType name="PublisherRegistrationManager">
1277
1278     <!--====DestroyRegistration:ImmediateResourceTermination====-->
1279     <wsdl:operation name="DestroyRegistration">
1280         <wsdl:input name="DestroyRegistrationRequest"
1281             message="wsn-brw:DestroyRegistrationRequest" />
1282         <wsdl:output name="DestroyRegistrationResponse"
1283             message="wsn-brw:DestroyRegistrationResponse" />
1284         <wsdl:fault name="ResourceUnknownFault"
1285             message="wsrf-rw:ResourceUnknownFault" />
1286         <wsdl:fault name="ResourceNotDestroyedFault"
1287             message="wsn-brw:ResourceNotDestroyedFault" />
1288     </wsdl:operation>
1289 </wsdl:portType>
1290 </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

1292

Appendix E. Notices

1293 OASIS takes no position regarding the validity or scope of an intellectual property or other rights
1294 that might be claimed to pertain to the implementation or use of the technology described in this
1295 document or the extent to which any license under such rights might or might not be available;
1296 neither does it represent that it has made any effort to identify any such rights. Information on
1297 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1298 website. Copies of claims of rights made available for publication and any assurances of licenses
1299 to be made available, or the result of an attempt made to obtain a general license or permission
1300 for the use of such proprietary rights by implementers or users of this specification, can be
1301 obtained from the OASIS Executive Director.

1302 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1303 applications, or other proprietary rights which may cover technology that may be required to
1304 implement this specification. Please address the information to the OASIS Executive Director.

1305 Copyright © OASIS Open 2004-2006. All Rights Reserved.

1306 This document and translations of it may be copied and furnished to others, and derivative works
1307 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1308 published and distributed, in whole or in part, without restriction of any kind, provided that the
1309 above copyright notice and this paragraph are included on all such copies and derivative works.
1310 However, this document itself does not be modified in any way, such as by removing the
1311 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
1312 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1313 Property Rights document must be followed, or as required to translate it into languages other
1314 than English.

1315 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1316 successors or assigns.

1317 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1318 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1319 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY
1320 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1321 PARTICULAR PURPOSE.