

1 Simple Events Push for WS-Manageability 1.0

2

3 Authors

4 Igor Sedukhin, Computer Associates (igor.sedukhin@ca.com)

5 Mark Potts, Talking Blocks (mark.potts@talkingblocks.com)

6

7 Copyright

8 This document is a technical note sent to the [open discussion list](#) of the OASIS WSDM TC
9 (wsdm@oasis-open.org). The authors provide this document as input and to aid future
10 discussions within the TC. This document and its contents can be freely distributed and
11 copied provided that reference to the authors and source of the document is included with
12 the copy.

13

14 Status and Scope of this Document

15 The document is provided as a clarification to the [submission of the WS-Manageability 1.0](#)
16 specification to the [OASIS Web Services Distributed Management Technical Committee](#).
17 This document merely explains how to apply existing standards, such as WSDL, to
18 describe a simple push mechanism applicable to event aspects defined within the WS-
19 Manageability 1.0 and their implementation. The authors believe that a general event
20 mechanism for Web services should be standardized outside of the WSDM TC, and this
21 document is not a proposal for general Web services events mechanism.

22

1 Introduction

23 The [WS-Manageability 1.0 – Specification](#) document presented a Web service endpoint
24 manageability model in terms of topics and aspects. One of the aspects is events. Particular
25 topic model may define particular event information. For example, the state topic defines
26 the StateChangeInformation data type. The information is then contained in an event
27 description along with other common description of the situation ([Common Base Event](#)).
28 The topic model may define named events with particular situational semantics. Named
29 events imply under which circumstances, what kinds of information needs to be emitted in
30 an event description. For example the state topic defines the stateChanged named event.
31

32

33 However the model does not imply any particular mechanism for the delivery of event
34 descriptions or the registration of interest in particular events. With the absence of such
35 standard mechanism defined for WS-I 1.0 compliant Web services today, the
36 complimentary [WS-Manageability 1.0 – Representation](#) document presented a simple event
37 polling approach defined in a WSDL 1.1 portType (CommonEventsAccess in
38 urn:wsdm:common namespace). The polling mechanism is assumed to be replaced by a
39 more prominent notification mechanism when it becomes available for WS-I compliant
40 Web services (pure Web services, per se).

41

42 In certain situations, the event polling mechanism is not sufficient or even applicable at all.
43 With that, it may need to be addressed today for WS-I 1.0 compliant Web services
44 implementations. It may not be sufficient to wait until event push is addressed by some
 standard specification and adopted by Web services platform vendors and industry in

45 general. In the mean time, WS-I 1.0 compliant Web services have to be made manageable
46 in a standard way and events push may be required for efficient and working
47 implementations.

48
49 The document introduces a clarification to an earlier WS-Manageability 1.0 submission to
50 the OASIS WSDM TC by IBM, CA and Talking Blocks. The clarification is a very simple
51 event push mechanism described using WSDL 1.1, WS-Addressing and CBE and is
52 compliant with WS-I 1.0 Web services and does not require modifications or augmentation
53 of the Web services platforms and tools. The intention of this clarification is to allow
54 efficient and capable implementations of the WS-Manageability 1.0 specification.

55
56 This document uses certain constructs defined in the [WS-Manageability 1.0](#) specification.
57

58 This document does not try to address event management generally. That could be
59 introduced later by an extension or otherwise externally. This document describes a simple
60 peer-to-peer notification approach applicable to WS-Manageability 1.0 implementations.

61 **2 Simple Events Push**

62 To briefly recap the requirements and definitions:

- 63 • Events are named elements of the model.
- 64 • Events can be identified. Identifiers include, but not limited to: a property QName,
65 a transition URI, a name string.
- 66 • Named events may have specific information defined in the model
- 67 • Event information is packaged into a Common Base Event description as an
68 extended data element.
- 69 • Need a WSDL 1.1 description of the events push approach: registration of the
70 interest and a deliverable recipient.

71
72 There are two WSDL 1.1 portTypes defined in the sections below: one for registration of
73 the interest in the events and another one for a deliverable recipient. These portTypes are
74 intended to be aggregated with other manageability portTypes and then either consolidated
75 (composed) into the functional portType offered by the manageable functional Web service
76 endpoint, or offered by the dedicated manageability endpoint associated with the functional
77 Web Service endpoint (see [WS-Manageability 1.0 – Concepts](#) section 5.1.1).

78
79 The deliverable recipient portType has to be composed (consolidated) into the Web service
80 endpoint exposed by the consumer of events. For example, a manager could compose
81 deliverable recipient operations and messages into the “manager’s Web service”.

82 **2.1 How does it work**

83 Following are XML message exchange examples that demonstrate what do the event
84 producer and event consumer do to make the simple events push mechanism work.

85
86 Assume the following namespace declarations.

87
88 `xmlns:p="urn:wsdm:common:events:access:push:types"`

Simple Events Push for WS-Manageability 1.0

```
89    xmlns:c="urn:wsdm:common:events:recipient:types"
90    xmlns:cmn="urn:wsdm:common"
91    xmlns:cbe="http://www.ibm.com/AC/commonbaseevent1_1"
92    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
93
94 To register an interest, the event consumer sends the following XML in a SOAP envelope
95 to the event producer.
96
97 <p:RegisterInterestInEvents>
98 <p:eventIdentifiers>
99     <cmn:eventIdentifier>
100    <cmn:transition>urn:wsdm:webservice:endpoint:lifecycle:up:TO:down
101    </cmn:transition>
102    <cmn:eventIdentifier>
103 </p:eventIdentifiers>
104 <p:recipientEndpoint>
105    <wsa:Address>http://myRecipient.mycorp.com/listener</wsa:Address>
106 </p:recipientEndpoint>
107 </p:RegisterInterestInEvents>
108
109 The response to the above request is the following XML in a SOAP envelope or any SOAP
110 fault.
111
112 <p:RegisterInterestInEventsResponse>
113 <p:registration> [...some XML...] </p:registration>
114 </p:RegisterInterestInEventsResponse>
115
116 To verify that registration is in place, the event consumer sends the following XML in a
117 SOAP envelope to the event producer.
118
119 <p:VerifyInterestInEvents>
120 <p:registration> [...some XML...] </p:registration>
121 </p:VerifyInterestInEvents>
122
123 The response to the above request is the following XML in a SOAP envelope or any SOAP
124 fault.
125
126 <p:VerifyInterestInEventsResponse>
127 <p:eventIdentifiers>
128     <cmn:eventIdentifier>
129     <cmn:transition>urn:wsdm:webservice:endpoint:lifecycle:up:TO:down
130     </cmn:transition>
131     <cmn:eventIdentifier>
132 </p:eventIdentifiers>
133 </p:VerifyInterestInEventsResponse>
134
135 To cancel the interest, the event consumer sends the following XML in a SOAP envelope
136 to the event producer.
137
138 <p:CancelInterestInEvents>
139 <p:registration> [...some XML...] </p:registration>
140 </p:CancelInterestInEvents>
```

141
142 To cancel the interest in particular events, the event consumer sends the following XML in
143 a SOAP envelope to the event producer.

144
145 <p:CancelInterestInEvents>
146 <p:registration> [...some XML...] </p:registration>
147 <p:eventIdentifiers>
148 <cmn:eventIdentifier>
149 <cmn:transition>urn:wsdm:webservice:endpoint:lifecycle:up:TO:down
150 </cmn:transition>
151 <cmn:eventIdentifier>
152 </p:eventIdentifiers>
153 </p:CancelInterestInEvents>

154
155 The response to either of the above requests is the following XML in a SOAP envelope or
156 any SOAP fault.

157
158 <p:CancelInterestInEventsResponse>
159 <p:OK>true</p:OK>
160 </p:CancelInterestInEventsResponse>

161
162 To receive an event notification, the event consumer implements the EventsPushRecipient
163 portType. That means the event consumer must be capable to accept the following XML in
164 a SOAP envelope.

165
166 <c:ReceiveEvent>
167 <c:registration> [...some XML...] </c:registration>
168 <c:eventIdentifier>
169 <cmn:transition>urn:wsdm:webservice:endpoint:lifecycle:up:TO:down
170 </cmn:transition>
171 <c:eventIdentifier>
172 <c:eventDescription>
173 [...CBE data...]
174 </c:eventDescription>
175 </c:ReceiveEvent>

176
177 The event consumer should respond with the following XML in a SOAP envelope or any
178 SOAP fault.

179
180 <c:ReceiveEventResponse>
181 <c:OK>true</c:OK>
182 </c:ReceiveEventResponse>

183
184 To understand that registration was cancelled by the producer, the events consumer needs
185 to process the following XML in a SOAP envelope. Essentially scan XML for the
186 RegistrationCancelled element QName.

187
188 <c:ReceiveEvent>
189 <c:registration> [...some XML...] </c:registration>
190 <c:eventIdentifier>
191 <cmn:transition>urn:wsdm:webservice:endpoint:lifecycle:up:TO:down
192 </cmn:transition>

```
193 <c:eventIdentifier>
194 <c:eventDescription>
195 [...CBE data...]
196 <cbe:extendedDataElement>
197 <p:RegistrationCancelled> ... </p:RegistrationCancelled>
198 </cbe:extendedDataElement>
199 [...CBE data...]
200 </c:eventDescription>
201 </c:ReceiveEvent>
202
```

203 2.2 Description of the Registration of the Interest

```
204 <definitions name="EventsPushAccess"
205   targetNamespace="urn:wsdm:common:events:access:push"
206   xmlns:tns="urn:wsdm:common:events:access:push"
207   xmlns:tns-typ="urn:wsdm:common:events:access:push:types"
208   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
209   xmlns="http://schemas.xmlsoap.org/wsdl/">
210
211 <types>
212 <xsd:schema
213   targetNamespace="urn:wsdm:common:events:access:push:types"
214   xmlns:cmn="urn:wsdm:common"
215   xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
216   elementFormDefault="qualified">
217
218 <xsd:import
219   namespace="urn:wsdm:common"
220   schemaLocation="../../schema/Common.xsd"/>
221
222 <xsd:import
223   namespace="http://schemas.xmlsoap.org/ws/2003/03/addressing"
224   schemaLocation="../../schema/wsa.xsd"/>
225
226 <xsd:element name="RegisterInterestInEvents">
227   <xsd:sequence>
228     <xsd:element name="eventIdentifiers"
229       type="cmn:ArrayOfEventIdentifiers"/>
230     <!-- the recipientEndpoint must be able to accept
231         messages as defined in the EventsPushRecipient
232         portType in the urn:wsdm:common:events:recipient
233         namespace -->
234     <xsd:element name="recipientEndpoint"
235       type="wsa:EndpointReferenceType"/>
236   </xsd:sequence>
237 </xsd:element>
238
239 <xsd:element name="RegisterInterestInEventsResponse">
240   <xsd:sequence>
241     <xsd:element name="registration" type="xsd:anyType" />
242   </xsd:sequence>
243 </xsd:element>
244
245 <xsd:element name="CancelInterestInEvents">
246   <xsd:sequence>
```

Simple Events Push for WS-Manageability 1.0

```
248         <xsd:element name="registration" type="xsd:anyType" />
249         <xsd:element name="eventIdentifiers"
250             type="cmn:ArrayOfEventIdentifiers"
251             minOccurs="0" />
252     </xsd:sequence>
253 </xsd:element>
254
255 <xsd:element name="CancelInterestInEventsResponse">
256     <xsd:sequence>
257         <xsd:element name="OK" type="xsd:boolean" />
258     </xsd:sequence>
259 </xsd:element>
260
261 <xsd:element name="VerifyInterestInEvents">
262     <xsd:sequence>
263         <xsd:element name="registration" type="xsd:anyType" />
264     </xsd:sequence>
265 </xsd:element>
266
267 <xsd:element name="VerifyInterestInEventsResponse">
268     <xsd:sequence>
269         <xsd:element name="eventIdentifiers"
270             type="cmn:ArrayOfEventIdentifiers" />
271     </xsd:sequence>
272 </xsd:element>
273
274 <!-- this element appears in the CBE extendedDataElement
275     it indicates that a resource will not be able to meet
276     the specified registration for the interest in events
277     anymore -->
278 <xsd:element name="RegistrationCancelled">
279     <xsd:sequence>
280         <xsd:element name="registration" type="xsd:anyType" />
281     </xsd:sequence>
282 </xsd:element>
283
284 </xsd:schema>
285 </types>
286
287 <message name="RegisterInterestInEventsIn">
288     <part name="parameters"
289         element="tns-typ:RegisterInterestInEvents" />
290 </message>
291
292 <message name="RegisterInterestInEventsOut">
293     <part name="parameters"
294         element="tns-typ:RegisterInterestInEventsResponse" />
295 </message>
296
297 <message name="CancelInterestInEventsIn">
298     <part name="parameters"
299         element="tns-typ:CancelInterestInEvents" />
300 </message>
301
302 <message name="CancelInterestInEventsOut">
303     <part name="parameters"
304         element="tns-typ:CancelInterestInEventsResponse" />
```

```
305 </message>
306
307 <message name="VerifyInterestInEventsIn">
308     <part name="parameters"
309         element="tns-typ:VerifyInterestInEvents" />
310 </message>
311
312 <message name="VerifyInterestInEventsOut">
313     <part name="parameters"
314         element="tns-typ:VerifyInterestInEventsResponse" />
315 </message>
316
317 <portType name="EventsPushAccess">
318
319     <operation name="RegisterInterestInEvents">
320         <input message="tns:RegisterInterestInEventsIn" />
321         <output message="tns:RegisterInterestInEventsOut" />
322     </operation>
323
324     <operation name="CancelInterestInEvents">
325         <input message="tns:CancelInterestInEventsIn" />
326         <output message="tns:CancelInterestInEventsOut" />
327     </operation>
328
329     <operation name="VerifyInterestInEvents">
330         <input message="tns:VerifyInterestInEventsIn" />
331         <output message="tns:VerifyInterestInEventsOut" />
332     </operation>
333
334 </portType>
335
336 </definitions>
337
```

338 **2.3 Description of the Deliverable Recipient**

```
339
340 <definitions name="EventsPushRecipient"
341     targetNamespace="urn:wsdm:common:events:recipient"
342     xmlns:tns="urn:wsdm:common:events:recipient"
343     xmlns:tns-typ="urn:wsdm:common:events:recipient:types"
344     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
345     xmlns="http://schemas.xmlsoap.org/wsdl/">
346
347 <types>
348 <xsd:schema
349     targetNamespace="urn:wsdm:common:events:recipient:types"
350     xmlns:cmn="urn:wsdm:common"
351     xmlns:cbe="http://www.ibm.com/AC/commonbaseevent1_1"
352     elementFormDefault="qualified">
353
354 <xsd:import
355     namespace="urn:wsdm:common"
356     schemaLocation="../../schema/Common.xsd"/>
357
358 <xsd:import
359     namespace="http://www.ibm.com/AC/commonbaseevent1_1"
```

Simple Events Push for WS-Manageability 1.0

```
360     schemaLocation="cbe.xsd" />
361
362 <xsd:element name="ReceiveEvent">
363     <xsd:sequence>
364         <xsd:element name="registration"
365             type="xsd:anyType" />
366         <xsd:element name="eventIdentifier"
367             type="cmn:EventIdentifier" minOccurs="0" />
368         <xsd:element name="eventDescription"
369             type="cbe:CommonBaseEventType" />
370     </xsd:sequence>
371 </xsd:element>
372
373 <xsd:element name="ReceiveEventResponse">
374     <xsd:sequence>
375         <xsd:element name="OK" type="xsd:boolean" />
376     </xsd:sequence>
377 </xsd:element>
378
379 </xsd:schema>
380 </types>
381
382 <message name="ReceiveEventIn">
383     <part name="parameters" element="tns-typ:ReceiveEvent" />
384 </message>
385
386 <message name="ReceiveEventOut">
387     <part name="parameters" element="tns-typ:ReceiveEventResponse" />
388 </message>
389
390 <portType name="EventsPushRecipient">
391
392     <operation name="ReceiveEvent">
393         <input message="tns:ReceiveEventIn" />
394         <output message="tns:ReceiveEventOut" />
395     </operation>
396
397 </portType>
398
399 </definitions>
400
```