



---

# WS-BPEL Extension for People (BPEL4People) Specification Version 1.1

## Committee Draft 09 / Public Review Draft 03

### 14 April 2010

#### Specification URIs:

##### This Version:

<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-09.html>  
<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-09.doc> (Authoritative format)  
<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-09.pdf>

##### Previous Version:

<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-08.html>  
<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-08.doc>  
<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-08.pdf>

##### Latest Version:

<http://docs.oasis-open.org/bpel4people/bpel4people-1.1.html>  
<http://docs.oasis-open.org/bpel4people/bpel4people-1.1.doc>  
<http://docs.oasis-open.org/bpel4people/bpel4people-1.1.pdf>

#### Technical Committee:

[OASIS BPEL4People TC](#)

#### Chair:

Dave Ings, IBM

#### Editors:

Luc Clément, Active Endpoints, Inc.  
Dieter König, IBM  
Vinkesh Mehta, Deloitte Consulting LLP  
Ralf Mueller, Oracle Corporation  
Ravi Rangaswamy, Oracle Corporation  
Michael Rowley, Active Endpoints, Inc.  
Ivana Trickovic, SAP

#### Related work:

This specification is related to:

- BPEL4People – WS-HumanTask Specification – Version 1.1 - <http://docs.oasis-open.org/bpel4people/ws-humantask-1.1.html>
- Web Services – Business Process Execution Language – Version 2.0 – <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>

**Declared XML Namespace:**

**b4p** – <http://docs.oasis-open.org/ns/bpel4people/bpel4people/200803>

**Abstract:**

Web Services Business Process Execution Language, version 2.0 (WS-BPEL 2.0 or BPEL for brevity) introduces a model for business processes based on Web services. A BPEL process orchestrates interactions among different Web services. The language encompasses features needed to describe complex control flows, including error handling and compensation behavior. In practice, however many business process scenarios require human interactions. A process definition should incorporate people as another type of participants, because humans may also take part in business processes and can influence the process execution.

This specification introduces a BPEL extension to address human interactions in BPEL as a first-class citizen. It defines a new type of basic activity which uses human tasks as an implementation, and allows specifying tasks local to a process or use tasks defined outside of the process definition. This extension is based on the WS-HumanTask specification.

**Status:**

This document was last revised or approved by the OASIS WS-BPEL Extension for People Technical Committee on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at <http://www.oasis-open.org/committees/bpel4people/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<http://www.oasis-open.org/committees/bpel4people/ipr.php>).

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/bpel4people/>.

---

## Notices

Copyright © OASIS® 2010. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

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

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

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "**OASIS**" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

---

# Table of Contents

1	Introduction .....	6
1.1	Terminology.....	6
1.2	Normative References.....	6
1.3	Non-Normative References .....	7
1.4	Conformance Targets.....	7
2	Language Design .....	9
2.1	Dependencies on Other Specifications.....	9
2.1.1	Namespaces Referenced.....	9
2.2	Language Extensibility.....	9
2.3	Overall Language Structure.....	9
2.3.1	Syntax .....	10
2.4	Default use of XPath 1.0 as an Expression Language .....	12
3	Concepts .....	13
3.1	Generic Human Roles .....	13
3.1.1	Syntax .....	13
3.1.2	Initialization Behavior .....	14
3.2	Assigning People.....	14
3.2.1	Using Logical People Groups.....	14
3.2.2	Computed Assignment .....	17
3.3	Ad-hoc Attachments .....	17
4	People Activity.....	18
4.1	Overall Syntax .....	19
4.1.1	Properties.....	19
4.2	Standard Overriding Elements.....	20
4.3	People Activities Using Local Human Tasks .....	21
4.3.1	Syntax .....	21
4.3.2	Examples .....	22
4.4	People Activities Using Local Notifications .....	22
4.4.1	Syntax .....	23
4.4.2	Examples .....	23
4.5	People Activities Using Remote Human Tasks .....	23
4.5.1	Syntax .....	24
4.5.2	Example .....	24
4.5.3	Passing Endpoint References for Callbacks .....	24
4.6	People Activities Using Remote Notifications .....	25
4.6.1	Syntax .....	25
4.6.2	Example .....	25
4.7	Elements for Scheduled Actions.....	26
4.8	People Activity Behavior and State Transitions .....	27
4.9	Task Instance Data.....	28
4.9.1	Presentation Data.....	28
4.9.2	Context Data .....	29
4.9.3	Operational Data .....	29

5	XPath Extension Functions .....	30
6	Coordinating Standalone Human Tasks .....	33
6.1	Protocol Messages from the People Activity's Perspective .....	33
7	BPEL Abstract Processes .....	35
7.1	Hiding Syntactic Elements .....	35
7.1.1	Opaque Activities .....	35
7.1.2	Opaque Expressions .....	35
7.1.3	Opaque Attributes .....	35
7.1.4	Opaque From-Spec .....	35
7.1.5	Omission .....	35
7.2	Abstract Process Profile for Observable Behavior .....	35
7.3	Abstract Process Profile for Templates .....	36
8	Conformance .....	37
A.	Standard Faults .....	38
B.	Portability and Interoperability Considerations .....	39
C.	BPEL4People Schema .....	40
D.	Sample .....	46
D.1	BPEL Definition .....	47
D.2	WSDL Definitions .....	52
E.	Acknowledgements .....	54
F.	Revision History .....	56

---

# 1 Introduction

This specification introduces an extension to BPEL in order to support a broad range of scenarios that involve people within business processes.

The BPEL specification focuses on business processes the activities of which are assumed to be interactions with Web services, without any further prerequisite behavior. But the spectrum of activities that make up general purpose business processes is much broader. People often participate in the execution of business processes introducing new aspects such as interaction between the process and user interface, and taking into account human behavior. This specification introduces a set of elements which extend the standard BPEL elements and enable the modeling of human interactions, which may range from simple approvals to complex scenarios such as separation of duties, and interactions involving ad-hoc data.

The specification introduces the people activity as a new type of basic activity which enables the specification of human interaction in processes in a more direct way. The implementation of a people activity could be an inline task or a standalone human task defined in the WS-HumanTask specification [WS-HumanTask]. The syntax and state diagram of the people activity and the coordination protocol that allows interacting with human tasks in a more integrated way is described. The specification also introduces XPath extension functions needed to access the process context.

The goal of this specification is to enable portability and interoperability:

**Portability** - The ability to take design-time artifacts created in one vendor's environment and use them in another vendor's environment.

**Interoperability** - The capability for multiple components (process infrastructure, task infrastructures and task list clients) to interact using well-defined messages and protocols. This enables combining components from different vendors allowing seamless execution.

Out of scope of this specification is how processes with human interactions are deployed or monitored. Usually people assignment is accomplished by performing queries on a people directory which has a certain organizational model. The mechanism of how an implementation evaluates people assignments, as well as the structure of the data in the people directory is also out of scope.

## 1.1 Terminology

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

## 1.2 Normative References

### [BPEL4WS 1.1]

Business Process Execution Language for Web Services Version 1.1, BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems, May 2003, available via <http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>, <http://fr.sap.com/bpel4ws/>, <http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>

### [RFC 2119]

Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, available via <http://www.ietf.org/rfc/rfc2119.txt>

### [RFC 3066]

Tags for the Identification of Languages, H. Alvestrand, IETF, January 2001, available via <http://www.isi.edu/in-notes/rfc3066.txt>, <http://www.ietf.org/rfc/rfc3066.txt>

### [WS-Addr-Core]

45 [Web Services Addressing 1.0 – Core](http://www.w3.org/TR/ws-addr-core), W3C Recommendation, May 2006, available via  
46 <http://www.w3.org/TR/ws-addr-core>

47 **[WS-Addr-SOAP]**

48 [Web Services Addressing 1.0 – SOAP Binding](http://www.w3.org/TR/ws-addr-soap), W3C Recommendation, May 2006, available via  
49 <http://www.w3.org/TR/ws-addr-soap>

50 **[WS-Addr-WSDL]**

51 [Web Services Addressing 1.0 – WSDL Binding](http://www.w3.org/TR/ws-addr-wsdl), W3C Working Draft, February 2006, available via  
52 <http://www.w3.org/TR/ws-addr-wsdl>

53 **[WS-BPEL 2.0]**

54 OASIS Standard, “Web Service Business Process Execution Language Version 2.0”, 11 April  
55 2007, <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>

56 **[WSDL 1.1]**

57 Web Services Description Language (WSDL) Version 1.1, W3C Note, available via  
58 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

59 **[WS-HumanTask]**

60 OASIS Committee Draft, “Web Services – Human Task (WS-HumanTask) Specification Version  
61 1.1, CD-07”, 03 March 2010, <http://docs.oasis-open.org/bpel4people/ws-humantask-1.1-spec-cd-07.doc>

62

63 **[XML Infoset]**

64 [XML Information Set](http://www.w3.org/TR/2001/REC-xml-infoset-20011024/), W3C Recommendation, available via <http://www.w3.org/TR/2001/REC-xml-infoset-20011024/>

65

66 **[XML Namespaces]**

67 Namespaces in XML 1.0 (Second Edition), W3C Recommendation, available via  
68 <http://www.w3.org/TR/REC-xml-names/> <http://www.w3.org/TR/REC-xml-names/>

69 **[XML Schema Part 1]**

70 XML Schema Part 1: Structures, W3C Recommendation, October 2004, available via  
71 <http://www.w3.org/TR/xmlschema-1/>

72 **[XML Schema Part 2]**

73 XML Schema Part 2: Datatypes, W3C Recommendation, October 2004, available via  
74 <http://www.w3.org/TR/xmlschema-2/>

75 **[XMLSpec]**

76 XML Specification, W3C Recommendation, February 1998, available via  
77 <http://www.w3.org/TR/1998/REC-xml-19980210>

78 **[XPath 1.0]**

79 XML Path Language (XPath) Version 1.0, W3C Recommendation, November 1999, available via  
80 <http://www.w3.org/TR/1999/REC-xpath-19991116>

## 81 **1.3 Non-Normative References**

82 There are no non-normative references made by this specification.

## 83 **1.4 Conformance Targets**

84 As part of this specification, the following conformance targets are specified

- 85 • BPEL4People Definition
- 86 A BPEL4People Definition is a WS-BPEL 2.0 process definition that uses the BPEL4People  
87 extensions to WS-BPEL 2.0 specified in this document.

- 88
- BPEL4People Processor
- 89 A BPEL4People Processor is any implementation that accepts a BPEL4People definition and  
90 executes the semantics defined in this document.
- 91
- 92
- WS-HumanTask Definition
- 93
- 94 A WS-HumanTask Definition is any artifact that complies with the human interaction schema  
95 and additional constraints as defined by the WS-HumanTask 1.1 specification.
- WS-HumanTask Processor
- 96
- 97 A WS-HumanTask Processor is any implementation that accepts a WS-HumanTask  
98 definition and executes the semantics as defined by the WS-HumanTask 1.1 specification.



---

## 99 2 Language Design

100 The BPEL4People extension is defined in a way that it is layered on top of BPEL so that its features can  
101 be composed with BPEL features whenever needed. All elements and attributes introduced in this  
102 extension are made available to both BPEL executable processes and abstract processes.

103 This extension introduces a set of elements and attributes to cover different complex human interaction  
104 patterns, such as separation of duties, which are not defined as first-class elements.

105 Throughout this specification, WSDL and schema elements may be used for illustrative or convenience  
106 purposes. However, in a situation where those elements or other text within this document contradict the  
107 separate BPEL4People, WS-HumanTask, WSDL or schema files, it is those files that have precedence  
108 and not this document.

### 109 2.1 Dependencies on Other Specifications

110 BPEL4People utilizes the following specifications:

- 111 • WS-BPEL 2.0: BPEL4People extends the WS-BPEL 2.0 process model and uses existing WS-  
112 BPEL 2.0 capabilities, such as those for data manipulation.
- 113 • WS-HumanTask 1.1: BPEL4People uses the definition of human tasks and, notifications, and  
114 extends generic human roles and people assignments introduced in WS-HumanTask 1.1.
- 115 • WSDL 1.1: BPEL4People uses WSDL for service interface definitions.
- 116 • XML Schema 1.0: BPEL4People utilizes XML Schema data model.
- 117 • XPath 1.0: BPEL4People uses XPath as default query and expression language.

#### 118 2.1.1 Namespaces Referenced

119 BPEL4People references these namespaces:

- 120 • **htd** – <http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803>
- 121 • **htt** – <http://docs.oasis-open.org/ns/bpel4people/ws-humantask/types/200803>
- 122 • **bpel** – <http://docs.oasis-open.org/wsbpel/2.0/process/executable>
- 123 • **abstract** – <http://docs.oasis-open.org/wsbpel/2.0/process/abstract>
- 124 • **wSDL** – <http://schemas.xmlsoap.org/wSDL/>
- 125 • **xsd** – <http://www.w3.org/2001/XMLSchema>
- 126 • **xsi** – <http://www.w3.org/2001/XMLSchema-instance>

### 127 2.2 Language Extensibility

128 The BPEL4People specification extends the reach of the standard BPEL extensibility mechanism to  
129 BPEL4People elements. This allows:

130 Attributes from other namespaces to appear on any BPEL4People element

131 Elements from other namespaces to appear within BPEL4People elements

132 Extension attributes and extension elements MUST NOT contradict the semantics of any attribute or  
133 element from the BPEL4People namespace.

134 The standard BPEL element `<extension>` MUST be used to declare mandatory and optional  
135 extensions of BPEL4People.

### 136 2.3 Overall Language Structure

137 This section explains the structure of BPEL4People extension elements, including the new activity type  
138 people activity, inline human tasks and people assignments.

### 139 2.3.1 Syntax

140 Informal syntax of a BPEL process and scope containing logical people groups, inline human tasks, and  
141 people activity follows.

```
142 <bpel:process b4p:shareComments="xsd:boolean"? ...  
143   ...  
144   xmlns:b4p="http://docs.oasis-open.org/ns/bpel4people/bpel4people/200803"  
145   xmlns:htd="http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803">  
146   ...  
147   <bpel:extensions>  
148     <bpel:extension  
149       namespace="http://docs.oasis-  
150 open.org/ns/bpel4people/bpel4people/200803"  
151       mustUnderstand="yes"/>  
152     <bpel:extension  
153       namespace="http://docs.oasis-open.org/ns/bpel4people/ws-  
154 humantask/200803"  
155       mustUnderstand="yes"/>  
156   </bpel:extensions>  
157  
158   <bpel:import  
159     importType="http://docs.oasis-open.org/ns/bpel4people/ws-  
160 humantask/200803" .../>  
161  
162   ...  
163   <b4p:humanInteractions>?  
164  
165     <htd:logicalPeopleGroups/>?  
166     <htd:logicalPeopleGroup name="NCName" reference="QName"?>+  
167     ...  
168     </htd:logicalPeopleGroup>  
169   </htd:logicalPeopleGroups>  
170  
171   <htd:tasks>?  
172     <htd:task name="NCName">+  
173     ...  
174     </htd:task>  
175   </htd:tasks>  
176  
177   <htd:notifications>?  
178     <htd:notification name="NCName">+  
179     ...  
180     </htd:notification>  
181   </htd:notifications>  
182  
183 </b4p:humanInteractions>  
184  
185 <b4p:peopleAssignments>?  
186   ...  
187 </b4p:peopleAssignments>  
188  
189   ...  
190 <bpel:extensionActivity>  
191   <b4p:peopleActivity name="NCName" ...>  
192   ...  
193   </b4p:peopleActivity>  
194 </bpel:extensionActivity>  
195   ...
```

196 `</bpel:process>`

197 A BPEL4People Definition MUST use BPEL4People extension elements and elements from WS-  
198 HumanTask namespace. Therefore elements from namespaces BPEL4People and WS-HumanTask  
199 MUST be understood.

200 The element `<b4p:humanInteractions>` is optional and contains declarations of elements from WS-  
201 HumanTask namespace, that is `<htd:logicalPeopleGroups>`, `<htd:tasks>` and  
202 `<htd:notifications>`.

203 The element `<htd:logicalPeopleGroup>` specifies a logical people group used in an inline human  
204 task or a people activity. The name attribute specifies the name of the logical people group. The name  
205 MUST be unique among the names of all logical people groups defined within the  
206 `<b4p:humanInteractions>` element.

207 The `<htd:task>` element is used to provide the definition of an inline human task. The syntax and  
208 semantics of the element are provided in the WS-HumanTask specification. The name attribute specifies  
209 the name of the task. The name MUST be unique among the names of all tasks defined within the  
210 `<htd:tasks>` element.

211 The `<htd:notification>` element is used to provide the definition of an inline notification. The syntax  
212 and semantics of the element are provided in the WS-HumanTask specification. The name attribute  
213 specifies the name of the notification. The name MUST be unique among the names of all notifications  
214 defined within the `<htd:notifications>` element.

215 The element `<b4p:peopleAssignments>` is used to assign people to process-related generic human  
216 roles. This element is optional. The syntax and semantics are introduced in section 3.1 “Generic Human  
217 Roles”.

218 New activity type `<b4p:peopleActivity>` is used to model human interactions within BPEL  
219 processes. The new activity is included in the BPEL activity `<bpel:extensionActivity>` which is  
220 used as wrapper. The syntax and semantics of the people activity are introduced in section 4 “People  
221 Activity”.

222 Any scope (or the process itself) can specify `@b4p:shareComments="true"` to specify that the  
223 comments that are added to any task executed within the scope (or a child scope) should be propagated  
224 to any other task within the same scope that is started after the first task completes. When comments  
225 propagate to later tasks, all metadata for the comment MUST also be propagated.

226 Note that, when a scope specifies the sharing of comments, it is not possible to override that sharing for  
227 child or descendent scopes. When a scope specifies `@b4p:shareComments="true"` then child and  
228 descendent scopes MUST NOT specify `@b4p:shareComments="false"`. However, an individual  
229 people activity can prevent its tasks’ comments from being propagated by specifying  
230 `@dontShareComments="true"`.

231

232

```
233 <bpel:scope b4p:shareComments="xsd:boolean"? ...>
234   ...
235   <b4p:humanInteractions>?
236     ...
237   </b4p:humanInteractions>
238   ...
239   <bpel:extensionActivity>
240     <b4p:peopleActivity name="NCName" dontShareComments="xsd:boolean" ...>
241       ...
242     </b4p:peopleActivity>
243   </bpel:extensionActivity>
244   ...
245 </bpel:scope>
```

246 BPEL scopes can also include elements from BPEL4People and WS-HumanTask namespaces except for  
247 the <b4p:peopleAssignments> element.

248 All BPEL4People Definition elements MAY use the element <b4p:documentation> to provide  
249 annotation for users. The content could be a plain text, HTML, and so on. The <b4p:documentation>  
250 element is optional and has the following syntax:

```
251 <b4p:documentation xml:lang="xsd:language">
252   ...
253 </b4p:documentation>
```

## 254 2.4 Default use of XPath 1.0 as an Expression Language

255 The XPath 1.0 specification [XPath 1.0] defines the context in which an XPath expression is evaluated.  
256 When XPath 1.0 is used as an Expression Language in BPEL4People or inlined WS-HumanTask  
257 language elements then the XPath context is initialized as follows:

- 258 • Context node: none
- 259 • Context position: none
- 260 • Context size: none
- 261 • Variable bindings: all WS-BPEL variables visible to the enclosing element as defined by the WS-  
262 BPEL scope rules
- 263 • Function library: Core XPath 1.0, WS-BPEL, BPEL4People and WS-HumanTask functions MUST  
264 be available and processor-specific functions MAY be available
- 265 • Namespace declaration: all in-scope namespace declarations from the enclosing element

266 Note that XPath 1.0 explicitly requires that any element or attribute used in an XPath expression that  
267 does not have a namespace prefix must be treated as being namespace unqualified. As a result, even if  
268 there is a default namespace defined on the enclosing element, the default namespace will not be  
269 applied.

---

## 270 3 Concepts

271 Many of the concepts in BPEL4People are inherited from the WS-HumanTask specification so familiarity  
272 with this specification is assumed.

### 273 3.1 Generic Human Roles

274 Process-related generic human roles define what a person or a group of people resulting from a people  
275 assignment can do with the process instance. The process-related human roles complement the set of  
276 generic human roles specified in [WS-HumanTask]. There are three process-related generic human roles:

- 277 • Process initiator
- 278 • Process stakeholders
- 279 • Business administrators

280 *Process initiator* is the person associated with triggering the process instance at its creation time. The  
281 initiator is typically determined by the infrastructure automatically. This can be overridden by specifying a  
282 people assignment for process initiator. A BPEL4People Definition MAY define assignment for this  
283 generic human role. A compliant BPEL4People Processor MUST ensure that at runtime at least one  
284 person is associated with this role.

285 *Process stakeholders* are people who can influence the progress of a process instance, for example, by  
286 adding ad-hoc attachments, forwarding a task, or simply observing the progress of the process instance.  
287 The scope of a process stakeholder is broader than the actual BPEL4People specification outlines. The  
288 process stakeholder is associated with a process instance. If no process stakeholders are specified, the  
289 process initiator becomes the process stakeholder. A BPEL4People Definition MAY define assignment for  
290 this generic human role. A compliant BPEL4People Processor MUST ensure that at runtime at least one  
291 person is associated with this role.

292 *Business administrators* are people allowed to perform administrative actions on the business process,  
293 such as resolving missed deadlines. A business administrator, in contrast to a process stakeholder, has  
294 an interest in all process instances of a particular process type, and not just one. If no business  
295 administrators are specified, the process stakeholders become the business administrators. A  
296 BPEL4People Definition MAY define assignment for this generic human role. A compliant BPEL4People  
297 Processor MUST ensure that at runtime at least one person is associated with this role.

#### 298 3.1.1 Syntax

```
299 <b4p:peopleAssignments>?
```

```
300  
301   <htd:genericHumanRole>+  
302     <htd:from>...</htd:from>  
303   </htd:genericHumanRole>
```

```
304  
305 </b4p:peopleAssignments>
```

306 The *genericHumanRole* abstract element introduced in the WS-HumanTask specification is extended  
307 with the following process-related human roles.

```
308 <b4p:peopleAssignments>?
```

```
309  
310   <b4p:processInitiator>?  
311     <htd:from ...>...</htd:from>  
312   </b4p:processInitiator>
```

```
313  
314   <b4p:processStakeholders>?  
315     <htd:from ...>...</htd:from>  
316   </b4p:processStakeholders>
```

```
317
```

```
318 <b4p:businessAdministrators>?  
319   <htd:from ...>...</htd:from>  
320 </b4p:businessAdministrators>  
321  
322 </b4p:peopleAssignments>
```

323 Only process-related human roles MUST be used within the `<b4p:peopleAssignments>` element.  
324 People are assigned to these roles as described in section 3.2 (“Assigning People”).

### 325 3.1.2 Initialization Behavior

326 Assigning people to process-related generic human roles happens after BPEL process initialization (see  
327 [WS-BPEL 2.0], section 12.1). A BPEL4People Processor MUST initialize process-related generic human  
328 roles after the end of the initial start activity of the process and before processing other activities or links  
329 leaving the start activity. If that initialization fails then the fault `b4p:initializationFailure` MUST be  
330 thrown by a BPEL4People Processor.

## 331 3.2 Assigning People

332 To determine who is responsible for acting on a process, a human task or a notification in a certain  
333 generic human role, people need to be assigned. People assignment can be achieved in different ways:

- 334 • Via logical people groups (see 3.2.1 “Using Logical People Groups”)
- 335 • Via literals (as introduced section 3.2.2 in [WS-HumanTask])
- 336 • Via expressions (see 3.2.2 “Computed Assignment”)

337 When specifying people assignments then the data type `htt:tOrganizationalEntity` defined in  
338 [WS-HumanTask] is used. Using `htt:tOrganizationalEntity` allows to assign either a list of users  
339 or a list of unresolved groups of people (“work queues”).

### 340 3.2.1 Using Logical People Groups

341 This section focuses on describing aspects of logical people groups that are specific to business  
342 processes. Logical people groups define which person or set of people can interact with a human task or  
343 a notification of a people activity. Details about how logical people groups are used with human tasks and  
344 notifications are provided by the WS-HumanTask specification.

345 Logical people groups can be specified as part of the business process definition. They can be defined  
346 either at the process level or on enclosed scopes. Definitions on inner scopes override definitions on  
347 outer scopes or the process respectively.

348 Logical people group definitions can be referenced by multiple people activities. Each logical people  
349 group is bound to a people query during deployment.

350 In the same way as in WS-HumanTask, a logical people group has one instance per set of unique  
351 arguments. Whenever a logical people group is referenced for the first time with a given set of unique  
352 arguments, a new instance MUST be created by the BPEL4People Processor. To achieve that, the  
353 logical people group MUST be evaluated / resolved for this set of arguments. Whenever a logical people  
354 group is referenced for which an in-stance already exists (i.e., it has already referenced before with the  
355 same set of arguments), the logical people group MAY be re-evaluated / re-resolved.

356 In particular, for a logical people group with no parameters, there is a single instance, which MUST be  
357 evaluated / resolved when the logical people group is first referenced, and which MAY be re-evaluated /  
358 re-resolved when referenced again.

359 Hence, using the same logical people group does not necessarily mean that the result of a people query  
360 is re-used, but that the same query is used to obtain a result. If the result of a previous people query  
361 needs to be re-used, then this result needs to be referenced explicitly from the process context. Please  
362 refer to section 5 “XPath Extension Functions” for a description of the syntax.

363  
364

## 365 Assignment of Logical People Groups

366 A BPEL4People Definition MAY use the <assign> activity (see [WS-BPEL 2.0] section 8.4 for more  
367 details) to manipulate values of logical people group. A mechanism to assign to a logical people group or  
368 to assign from a logical people group using BPEL copy assignments is provided. The semantics of the  
369 <copy> activity introduced in [WS-BPEL 2.0] (see sections 8.4.1, 8.4.2 and 8.4.3 for more details) applies.  
370 BPEL4People extends the from-spec and to-spec forms introduced in [WS-BPEL 2.0] as shown below:

```
371 <bpel:from b4p:logicalPeopleGroup="NCName">  
372   <b4p:argument name="NCName" expressionLanguage="anyURI"?>*</b4p:argument>  
373   value</b4p:argument>  
374 </bpel:from>  
375  
376  
377 <to b4p:logicalPeopleGroup="NCName"/>
```

378 In this form of from-spec and to-spec the `b4p:logicalPeopleGroup` attribute provides the name of a  
379 logical people group. The from-spec variant MAY include zero or more `<b4p:argument>` elements in  
380 order to pass values used in the people query. The `expressionLanguage` attribute specifies the  
381 language used in the expression. The attribute is optional. If not specified, the default language as  
382 inherited from the closest enclosing element that specifies the attribute is used.

383 Using a logical people group in the from-spec causes the evaluation of the logical people group. Logical  
384 people groups return data of type `htt:tOrganizationalEntity`. This data can be manipulated and  
385 assigned to other process variables using standard BPEL to-spec variable variants.

386 The new form of the from-spec can be used with the following to-spec variants:

- 387 • To copy to a variable

```
388 <bpel:to variable="BPELVariableName" part="NCName"? >  
389   <bpel:query queryLanguage="anyURI"? >?</bpel:query>  
390   queryContent</bpel:query>  
391 </bpel:to>
```

393

- 394 • To copy to non-message variables and parts of message variables

```
395 <bpel:to expressionLanguage="anyURI"?>expression</bpel:to>
```

396

- 397 • To copy to a property

```
398 <bpel:to variable="BPELVariableName" property="QName"/>
```

399

- 400 • To copy to a logical people group

```
401 <bpel:to b4p:logicalPeopleGroup="NCName"/>
```

402

403 Using a logical people group in the to-spec of a `<bpel:copy>` assignment enables a set of people to be  
404 explicitly assigned. Whenever the logical people group is used after the assignment this assigned set of  
405 people is returned. Assigning values to a logical people group overrides what has been defined during  
406 deployment. This is true irrespective of any parameters specified for the logical people group.

407 The new form of the to-spec can be used with the following from-spec variants:

- 408 • To copy from a variable

```
409 <bpel:from variable="BPELVariableName" part="NCName"? >  
410   <bpel:query queryLanguage="anyURI"? >?</bpel:query>  
411   queryContent</bpel:query>  
412 </bpel:from>
```

414

- 415 • To copy from a property

```
416 <bpel:from variable="BPELVariableName" property="QName"/>
```

417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428

- To copy from non-message variables and parts of message variables

```
<bpel:from expressionLanguage="anyURI"?>expression</bpel:from>
```

- To copy from a literal value

```
<bpel:from>  
  <bpel:literal>literal value</bpel:literal>  
</bpel:from>
```

- To copy from a logical people group

```
<bpel:from b4p:logicalPeopleGroup="NCName" />
```

429 Below are several examples illustrating the usage of logical people groups in copy assignments. The first  
430 example shows assigning the results of the evaluation of a logical people group to a process variable.

```
431 <bpel:assign name="getVoters">  
432   <bpel:copy>  
433     <bpel:from b4p:logicalPeopleGroup="voters">  
434       <b4p:argument name="region">  
435         $selectionRequest/region  
436       </b4p:argument>  
437     </bpel:from>  
438     <bpel:to variable="voters" />  
439   </bpel:copy>  
440 </bpel:assign>
```

441

442 The next example demonstrates assigning a set of people to a logical people group using literal values.

```
443 <bpel:assign>  
444   <bpel:copy>  
445     <bpel:from>  
446       <bpel:literal>  
447         <htt:tOrganizationalEntity>  
448           <htt:user>Alan</htt:user>  
449           <htt:user>Dieter</htt:user>  
450           <htt:user>Frank</htt:user>  
451           <htt:user>Gerhard</htt:user>  
452           <htt:user>Ivana</htt:user>  
453           <htt:user>Karsten</htt:user>  
454           <htt:user>Matthias</htt:user>  
455           <htt:user>Patrick</htt:user>  
456         </htt:tOrganizationalEntity>  
457       </bpel:literal>  
458     </bpel:from>  
459     <bpel:to b4p:logicalPeopleGroup="bpel4peopleAuthors" />  
460   </bpel:copy>  
461 </bpel:assign>
```

462



463

464 The third example shows assigning the results of one logical people group to another logical people  
465 group.

```
466 <bpel:assign>  
467   <bpel:copy>  
468     <bpel:from b4p:logicalPeopleGroup="bpel4peopleAuthors" />  
469     <bpel:to b4p:logicalPeopleGroup="approvers" />  
470   </bpel:copy>  
471 </bpel:assign>
```

## 472 3.2.2 Computed Assignment

473 All computed assignment variants described in [WS-HumanTask] (see section 3.25 “Assigning People” for  
474 more details) are supported. In addition, the following ~~variant is~~ variants are possible:

```
475 <htd:genericHumanRole>  
476   <bpel:from variable="NCName" part="NCName"? >  
477     ...  
478   </bpel:from>  
479 </htd:genericHumanRole>
```

480 The from-spec variant `<bpel:from variable>` is used to assign people that have been specified  
481 using a variable of the business process. The data type of the variable MUST be of type  
482 `htt:tOrganizationalEntity`.

483 All other process context can be accessed using expressions of the following style:

```
484 <bpel:from expressionLanguage="anyURI"?>expression</bpel:from>
```

485 with XPath extension functions defined in section 5 “XPath Extension Functions”. The  
486 `expressionLanguage` attribute specifies the language used in the expression. The attribute is optional.  
487 If not specified, the default language as inherited from the closest enclosing element that specifies the  
488 attribute is used.

## 489 3.3 Ad-hoc Attachments

490 Processes can have ad-hoc attachments. It is possible to exchange ad-hoc attachments between people  
491 activities of a process by propagating ad-hoc attachments to and from the process level.

492 When a people activity is activated, attachments from earlier tasks and from the process can be  
493 propagated to its implementing human task. On completion of the human task, its ad-hoc attachments  
494 can be propagated to the process level, to make them globally available.

495 All manipulations of ad-hoc attachments at the process level are instantaneous, and not subject to  
496 compensation or isolation.

497

## 4 People Activity

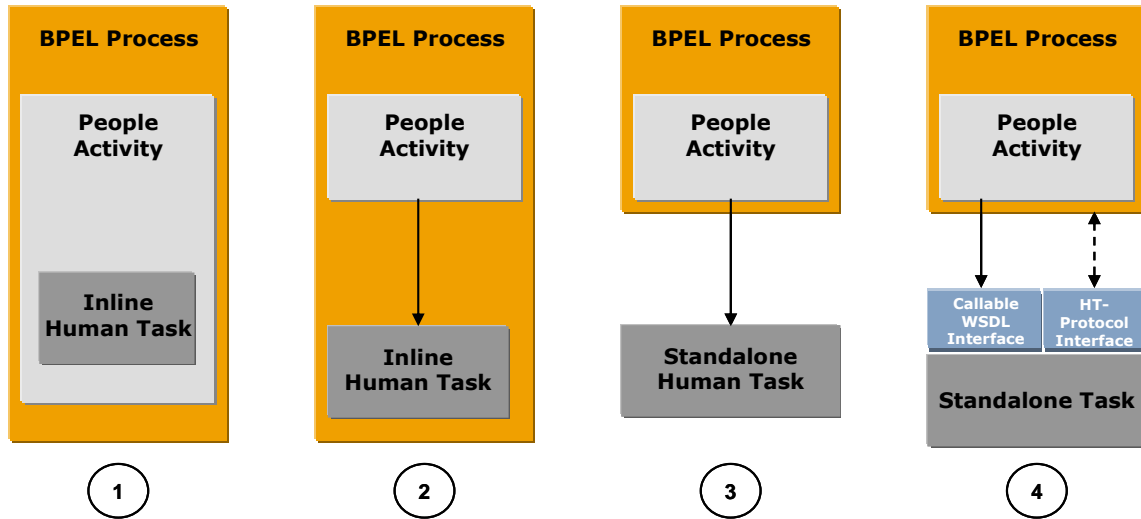
498

People activity is a basic activity used to integrate human interactions within BPEL processes. The following figure illustrates different ways in which human interactions (including human tasks and notifications) could be integrated.

499

500

501



502

503

Figure 1: Constellations

504

505 Constellations 1 and 2 show models of interaction in which tasks are defined inline as part of a BPEL  
506 process. An *inline task* can be defined as part of a people activity (constellation 1). In this case, the use of  
507 the task is limited to the people activity encompassing it. Alternatively, a task can be defined as a top-  
508 level construct of the BPEL process or scope (constellation 2). In this case, the same task can be used  
509 within multiple people activities, which is significant from a reuse perspective. BPEL4People processes  
510 that use tasks in this way are portable among BPEL engines that implement BPEL4People. This also  
511 holds true for notifications.

512 Constellation 3 shows the use of a standalone task within the same environment, without the specification  
513 of a callable Web services interface on the task. Thus the task invocation is implementation-specific. This  
514 constellation is similar to constellation 2, except that the definition of the task is done independently of  
515 any process. As a result, the task has no direct access to process context. This also holds true for  
516 notifications.

517 Constellation 4 shows the use of a standalone task from a different environment. The major difference  
518 when compared to constellation 3 is that the task has a Web services callable interface, which is invoked  
519 using Web services protocols. In addition, the WS-HumanTask coordination protocol is used to  
520 communicate between processes and tasks (see section 6 “Coordinating Standalone Human Tasks” for  
521 more details on the WS-HumanTask coordination protocol). Using this mechanism, state changes are  
522 propagated between task and process activity, and the process can perform life cycle operations on the  
523 task, such as terminating it. BPEL4People processes that use tasks in this way are portable across  
524 different BPEL engines that implement BPEL4People. They are interoperable, assuming that both the  
525 process infrastructures and the task infrastructures implement the coordination protocol. In case of  
526 notifications a simplified protocol is used. For more detail on the relationship of WS-HumanTask and the  
527 BPEL4People specifications refer to section 1.1 of WS-HumanTask.

## 528 4.1 Overall Syntax

529 Definition of people activity:

```
530 <bpel:extensionActivity>
531
532   <b4p:peopleActivity name="NCName" inputVariable="NCName"?
533     outputVariable="NCName"? isSkipable="xsd:boolean"?
534     dontShareComments="xsd:boolean"?
535     standard-attributes>
536
537     standard-elements
538
539     ( <htd:task>...</htd:task>
540     | <b4p:localTask>...</b4p:localTask>
541     | <b4p:remoteTask>...</b4p:remoteTask>
542     | <htd:notification>...</htd:notification>
543     | <b4p:localNotification>...</b4p:localNotification>
544     | <b4p:remoteNotification>...</b4p:remoteNotification>
545     )
546
547     <b4p:scheduledActions>? ...</b4p:scheduledActions>
548
549     <bpel:toParts>?
550       <bpel:toPart part="NCName" fromVariable="BPELVariableName" />+
551     </bpel:toParts>
552
553     <bpel:fromParts>?
554       <bpel:fromPart part="NCName" toVariable="BPELVariableName" />+
555     </bpel:fromParts>
556
557     <b4p:attachmentPropagation fromProcess="all|none"
558       toProcess="all|newOnly|none" />?
559
560   </b4p:peopleActivity>
561 </bpel:extensionActivity>
```

### 563 4.1.1 Properties

564 The `<b4p:peopleActivity>` element is enclosed in the BPEL `extensionActivity` and has the  
565 following attributes and elements:

- 566 • `inputVariable`: This attribute refers to a process variable which is used as input of the WSDL  
567 operation of a task or notification. The process variable in the BPEL4People Definition MUST  
568 have a WSDL message type. This attribute is optional. If this attribute is not present the  
569 `<bpel:toParts>` element MUST be used.
- 570 • `outputVariable`: This attribute refers to a process variable which is used as output of the  
571 WSDL operation of a task. The process variable in the BPEL4People Definition MUST have a  
572 WSDL message type. This attribute is optional. If the people activity uses a human task and this  
573 attribute is not present the `<bpel:fromParts>` element MUST be used. The `outputVariable`  
574 attribute MUST NOT be used if the people activity uses a notification.
- 575 • `isSkipable`: This attribute indicates whether the task associated with the activity can be  
576 skipped at runtime or not. This is propagated to the task level. This attribute is optional. The  
577 default for this attribute is "no".
- 578 • `dontShareComments`: This attribute, if set to "true", indicates that comments that are added to  
579 the task associated with this people activity MUST NOT be propagated to any other task.

- 580 • `standard-attributes`: The activity makes available all BPEL's standard attributes.
- 581 • `standard-elements`: The activity makes available all BPEL's standard elements.
  - 582 ○ `htd:task`: This element is used to define an inline task within the people activity
  - 583 (constellation 1 in the figure above). This element is optional. Its syntax and semantics
  - 584 are introduced in section 4.3 "People Activities Using Local Human Tasks".
  - 585 ○ `b4p:localTask`: This element is used to refer to a standalone task with no callable
  - 586 Web service interface (constellations 2 or 3). This element is optional. Its syntax and
  - 587 semantics are introduced in section 4.3 "People Activities Using Local Human Tasks".
  - 588 ○ `b4p:remoteTask`: This element is used to refer to a standalone task offering callable
  - 589 Web service interface (constellation 4). This element is optional. Its syntax and semantics
  - 590 are introduced in section 4.5 "People Activities Using Remote Human Tasks".
  - 591 ○ `htd:notification`: This element is used to define an inline notification within the
  - 592 people activity (constellation 1 in the figure above). This element is optional. Its
  - 593 semantics is introduced in section 4.4 "People Activities Using Local Notifications".
  - 594 ○ `b4p:localNotification`: This element is used to refer to a standalone notification
  - 595 with no callable Web service interface (constellations 2 or 3). This element is optional. Its
  - 596 semantics is introduced in section 4.4 "People Activities Using Local Notifications".
  - 597 • `b4p:remoteNotification`: This element is used to refer to a standalone notification offering
  - 598 callable Web service interface (constellation 4). This element is optional. Its syntax and semantics
  - 599 are introduced in section 4.6 "People Activities Using Remote Notifications".
  - 600 • `b4p:scheduledActions`: This element specifies when the task changes its state. Its syntax
  - 601 and semantics are introduced in section 4.7 "Elements for Scheduled Actions".
  - 602 • `bpel:toParts`: This element is used to explicitly create multi-part WSDL message from multiple
  - 603 BPEL variables. The element is optional. Its syntax and semantics are introduced in the WS-
  - 604 BPEL 2.0 specification, section 10.3.1. The `<bpel:toParts>` element and the
  - 605 `inputVariable` attribute are mutually exclusive.
  - 606 • `bpel:fromParts`: This element is used to assign values to multiple BPEL variables from an
  - 607 incoming multi-part WSDL message. The element is optional. Its syntax and semantics are
  - 608 introduced in the WS-BPEL 2.0 specification, section 10.3.1. The `<bpel:fromParts>` element
  - 609 and the `outputVariable` attribute are mutually exclusive. This element **MUST NOT** be used in
  - 610 a BPEL4People Definition if the people activity uses a notification.
  - 611 • `b4p:attachmentPropagation`: This element is used to describe the propagation behavior of
  - 612 ad-hoc attachments to and from the people activity. On activation of the people activity, either all
  - 613 ad-hoc attachments from the process are propagated to the people activity, so they become
  - 614 available to the corresponding task, or none. The `fromProcess` attribute is used to specify this.
  - 615 On completion of a people activity, all ad-hoc attachments are propagated to its process, or only
  - 616 newly created ones (but not those that were modified), or none. The `toProcess` attribute is used
  - 617 to specify this. The element is optional. The default value for this element is that all attachments
  - 618 are propagated from the process to the people activity and only new attachments are propagated
  - 619 back to the process.

## 620 4.2 Standard Overriding Elements

621 Certain properties of human tasks and notifications can be specified on the process level as well as on  
 622 local and remote task definitions and notification definitions allowing the process to override the original  
 623 human task and notification definitions respectively. This increases the potential for reuse of tasks and  
 624 notifications. Overriding takes place upon invocation of the Web service implemented by the human task  
 625 (or notification) via the advanced interaction protocol implemented by both the process and the task (or  
 626 notification).

627 The following elements can be overridden:

- 628 • people assignments

- 629 • priority

630 People assignments can be specified on remote and local human tasks and notifications. As a  
631 consequence, the invoked task receives the results of people queries performed by the business process  
632 on a per generic human role base. The result will be of type `tOrganizationalEntity`. The result  
633 needs to be understandable in the context of the task, i.e., the user identifiers and groups need to a)  
634 follow the same scheme and b) there exists a 1:1 relationship between the user identifiers and users. If a  
635 generic human role is specified on both the business process and the task it calls then the people  
636 assignment as determined by the process overrides what is specified on the task. In other words, the  
637 generic human roles defined at the task level provide the default. The same applies to people  
638 assignments on remote and local notifications.

639 The task's originator is set to the process stakeholder.

640 Priority of tasks and notifications can be specified on remote and local human tasks and notifications. If  
641 specified, it overrides the original priority of the human task (or notification).

642 *Standard-overriding-elements* is used in the syntax below as a shortened form of the following list of  
643 elements:

```
644 <htd:priority expressionLanguage="anyURI"? >  
645   integer-expression  
646 </htd:priority>  
647  
648 <htd:peopleAssignments?>  
649   <htd:genericHumanRole>  
650     <htd:from>...</htd:from>  
651   </htd:genericHumanRole>  
652 </htd:peopleAssignments>
```

## 653 4.3 People Activities Using Local Human Tasks

654 People activities can be implemented using local human tasks. A local human task is one of the following:

- 655 • An inline task declared within the people activity. The task can be used only by that people  
656 activity
- 657 • An inline task declared within either the scope containing the people activity or the process  
658 scope. In this case the task can be reused as implementation of multiple people activities  
659 enclosed within the scope containing the task declaration
- 660 • A standalone task identified using a QName. In this case the task can be reused across multiple  
661 BPEL4People processes within the same environment.

662 The syntax and semantics of people activity using local tasks is given below.

### 663 4.3.1 Syntax

```
664 <b4p:peopleActivity inputVariable="NCName"? outputVariable="NCName"?  
665   isSkipable="xsd:boolean"? standard-attributes>  
666   standard-elements  
667  
668   ( <htd:task>...</htd:task>  
669   | <b4p:localTask reference="QName">  
670     standard-overriding-elements  
671   </b4p:localTask>  
672   )  
673  
674 </b4p:peopleActivity>
```

675

### 676 Properties

677 Element `<htd:task>` is used to define an inline task within the people activity. The syntax and  
678 semantics of the element are given in the WS-HumanTask specification. In addition, XPath expressions  
679 used in enclosed elements MAY refer to process variables. Enclosed elements MUST use the current  
680 value of the process variable. Changes to process variables MUST NOT directly cause changes in the  
681 execution of the enclosed elements, but only provide more current values when the enclosed elements  
682 choose to re-evaluate the expressions.

683 Element `<b4p:localTask>` is used to refer to a task enclosed in the BPEL4People process (a BPEL  
684 scope or the process scope) or a standalone task provided by the same environment. Attribute  
685 `reference` provides the QName of the task. The attribute is mandatory. The element MAY contain  
686 standard overriding elements explained in section 4.2 “Standard Overriding Elements”.

### 687 4.3.2 Examples

688 The following code shows a people activity declaring an inline task.

```
689 <b4p:peopleActivity inputVariable="candidates"  
690                   outputVariable="vote"  
691                   isSkipable="yes">  
692   <htd:task>  
693     <htd:peopleAssignments>  
694       <htd:potentialOwners>  
695         <htd:from>$voters/users/user[i]</htd:from>  
696       </htd:potentialOwners>  
697     </htd:peopleAssignments>  
698   </htd:task>  
699   <b4p:scheduledActions>  
700     <b4p:expiration>  
701       <b4p:documentation xml:lang="en-US">  
702         This people activity expires when not completed  
703         within 2 days after having been activated.  
704       </b4p:documentation>  
705       <b4p:for>P2D</b4p:for>  
706     </b4p:expiration>  
707   </b4p:scheduledActions>  
708 </b4p:peopleActivity>
```

709

710 The following code shows a people activity referring to an inline task defined in the BPEL4People  
711 process.

```
712 <extensionActivity>  
713   <b4p:peopleActivity name="firstApproval"  
714                     inputVariable="electionResult" outputVariable="decision">  
715     <b4p:localTask reference="tns:approveEmployeeOfTheMonth" />  
716   </b4p:peopleActivity>  
717 </extensionActivity>
```

## 718 4.4 People Activities Using Local Notifications

719 People activities can be implemented using local notifications. A local notification is one of the following:

- 720 • An inline notification declared within the people activity. The notification can be used only by that  
721 people activity
- 722 • An inline notification declared within either the scope containing the people activity or the process  
723 scope. In this case the notification can be reused as implementation of multiple people activities  
724 enclosed within the scope containing the notification declaration
- 725 • A standalone notification identified using a QName. In this case the notification can be reused  
726 across multiple BPEL4People processes within the same environment.

727 The syntax and semantics of people activity using local notifications is given below.

## 728 4.4.1 Syntax

```
729 <b4p:peopleActivity name="NCName"? inputVariable="NCName"?  
730   standard-attributes>  
731   standard-elements  
732  
733   ( <htd:notification>...</htd:notification>  
734   | <b4p:localNotification reference="QName">  
735     standard-overriding-elements  
736     </b4p:localNotification>  
737   )  
738 </b4p:peopleActivity>
```

739

### 740 Properties

741 Element `<htd:notification>` is used to define an inline notification within the people activity. The  
742 syntax and semantics of the element are given in the WS-HumanTask specification. In addition, XPath  
743 expressions used in enclosed elements MAY refer to process variables. Enclosed elements MUST use  
744 the current value of the process variable. Changes to process variables MUST NOT directly cause  
745 changes in the execution of the enclosed elements, but only provide more current values when the  
746 enclosed elements choose to re-evaluate the expressions.

747 Element `<b4p:localNotification>` is used to refer to a notification enclosed in the BPEL4People  
748 Definition (a BPEL scope or the process scope) or a standalone notification provided by the same  
749 environment. Attribute `reference` provides the QName of the notification. The attribute is mandatory.  
750 The element MAY contain standard overriding elements explained in section 4.2 “Standard Overriding  
751 Elements”.

## 752 4.4.2 Examples

753 The following code shows a people activity using a standalone notification.

```
754 <bpel:extensionActivity>  
755   <b4p:peopleActivity name="notifyEmployees"  
756     inputVariable="electionResult">  
757     <htd:localNotification reference="task:employeeBroadcast"/>  
758     <!-- notification is not defined as part of this document,  
759        but within a separate one  
760     -->  
761   </b4p:peopleActivity>  
762 </bpel:extensionActivity>
```

## 763 4.5 People Activities Using Remote Human Tasks

764 People activities can be implemented using remote human tasks. This variant has been referred to as  
765 constellation 4 in Figure 1. The remote human task is invoked using a mechanism similar to the BPEL  
766 invoke activity: Partner link and operation identify the human task based Web service to be called. In  
767 addition to that, the name of a response operation on the *myRole* of the partner link is specified, allowing  
768 the human task based Web service to provide its result back to the calling business process.

769 Constellation 4 allows interoperability between BPEL4People compliant business processes of one  
770 vendor, and WS-HumanTask compliant human tasks of another vendor. For example, the communication  
771 to propagate state changes between the business process and the remote human task happens in a  
772 standardized way, as described in section 6 “Coordinating Standalone Human Tasks”.

773 The remote human task can also define a priority element and people assignments. The priority and  
774 people assignments specified here override the original priority of the human task.

## 775 4.5.1 Syntax

```
776 <b4p:remoteTask  
777     partnerLink="NCName"  
778     operation="NCName"  
779     responseOperation="NCName"?>  
780  
781     standard-overriding-elements  
782  
783 </b4p:remoteTask>
```

784  
785 The attribute `responseOperation` (of type `xsd:NCName`) specifies the name of the operation to be  
786 used to receive the response message from the remote human task. The `operation` attribute refers to  
787 an operation of the `myRole` port type of the partner link associated with the `<b4p:remoteTask>`. The  
788 attribute MUST be set in the BPEL4People Definition when the `operation` attribute refers to a WSDL  
789 one-way operation. The attribute MUST NOT be set when the `operation` attribute refers to a WSDL  
790 request-response operation.

## 791 4.5.2 Example

```
792 <bpel:extensionActivity>  
793     <b4p:peopleActivity name="prepareInauguralSpeech"  
794         inputVariable="electionResult"  
795         outputVariable="speech"  
796         isSkipable="no">  
797         <b4p:remoteTask partnerLink="author"  
798             operation="prepareSpeech"  
799             responseOperation="receiveSpeech">  
800             <htd:priority>0</htd:priority> <!-- assign highest priority -->  
801             <htd:peopleAssignments>  
802                 <htd:potentialOwners>  
803                     <htd:from>$electionResult/winner</htd:from>  
804                 </htd:potentialOwners>  
805             </htd:peopleAssignments>  
806         </b4p:remoteTask>  
807     </b4p:peopleActivity>  
808 </bpel:extensionActivity>
```

## 809 4.5.3 Passing Endpoint References for Callbacks

810 A WS-HumanTask Processor MUST send a response message back to its calling process. The endpoint  
811 to which the response is to be returned to typically becomes known as late as when the human task is  
812 instantiated. This is no problem in case the human task is invoked synchronously via a request-response  
813 operation: a corresponding session between the calling process and the human task will exist and the  
814 response message of the human task uses this session.

815 But if the human task is called asynchronously via a one-way operation, such a session does not exist  
816 when the response message is sent. In this case, the BPEL4People Processor MUST pass the endpoint  
817 reference of the port expecting the response message of the human task to the WS-HumanTask  
818 Processor hosting the human task. Conceptually, this endpoint reference overrides any deployment  
819 settings for the human task. Besides the address of this port that endpoint reference MUST also specify  
820 additional metadata such that the port receiving the response is able to understand that the incoming  
821 message is in fact the response for an outstanding request (see [WS-HumanTask] section 8.2 for the  
822 definition of the metadata). Finally, such an endpoint reference MUST specify identifying data to allow the  
823 response message to be targeted to the correct instance of the calling process.

824 The additional metadata MAY consist of the name of the port type of the port as well as binding  
825 information about how to reach the port (see [WS-Addr-Core]) in order to support the replying activity of



826 the human task to send its response to the port. In addition, the name of the receiving operation at the  
827 calling process side is REQUIRED. This name MUST be provided as value of the `responseOperation`  
828 attribute of the `<b4p:remoteTask>` element (discussed in the previous section) and is passed together  
829 with an appropriate endpoint reference.

830 The above metadata represents the most generic solution allowing the response to be returned in all  
831 situations supported by WSDL. A simpler solution is supported in the case of the interaction between the  
832 calling process and the human task being based on SOAP: In this case, the metadata of the endpoint  
833 reference simply contains the value of the action header to be set in the response message.

834 In both cases (a request-response `<b4p:remoteTask>` as well as a `<b4p:remoteTask>` using two  
835 one-ways) the `<b4p:remoteTask>` activity is blocking. That is, the normal processing of a  
836 `<b4p:remoteTask>` activity does not end until a response message or fault message has been received  
837 from the human task. If the human task experiences a non-recoverable error, the WS-HumanTask  
838 Processor will signal that to the BPEL4People Processor and an `b4p:nonRecoverableError` fault  
839 MUST be raised in the parent process.

## 840 4.6 People Activities Using Remote Notifications

841 As described in the previous section, people activities can also be implemented using remote  
842 notifications. This variant is also referred to as *constellation 4*. Using remote notifications is very similar to  
843 using remote human tasks. Except for the name of the element enclosed in the people activity the main  
844 difference is that the remote notification is one-way by nature, and thus does not allow the specification of  
845 a response operation.

846 Remote notifications, like remote human tasks allow specifying properties that override the original  
847 properties of the notification Web service. The mechanism used is the same as described above. Like  
848 remote human tasks, remote notifications also allow overriding both people assignments and priority.

### 849 4.6.1 Syntax

```
850 <b4p:remoteNotification  
851   partnerLink="NCName"  
852   operation="NCName">  
853  
854   standard-overriding-elements  
855  
856 </b4p:remoteNotification>
```

### 857 4.6.2 Example

```
858 <bpel:extensionActivity>  
859   <b4p:peopleActivity name="notifyEmployees"  
860     inputVariable="electionResult">  
861     <b4p:remoteNotification partnerLink="employeeNotification"  
862       operation="receiveElectionResult">  
863       <htd:priority>5</htd:priority> <!-- assign moderate priority -->  
864       <htd:peopleAssignments>  
865         <htd:recipients>  
866           <htd:from>$voters</htd:from>  
867         </htd:recipients>  
868       </htd:peopleAssignments>  
869     </b4p:remoteNotification>  
870   </b4p:peopleActivity>  
871 </bpel:extensionActivity>
```

## 872 4.7 Elements for Scheduled Actions

873 Scheduled actions allow the specification of determining when a task needs to change its state. The  
874 following scheduled actions are defined:

875 **DeferActivation:** Specifies the activation time of the task. It is defined as either the period of time after  
876 which the task reaches state *Ready* (in case of explicit claim) or state *Reserved* (in case of implicit claim),  
877 or the point in time when the task reaches state *Ready* or state *Reserved*. The default value is zero, i.e.  
878 the task is immediately activated. If the activation time is defined as a point in time and the task is created  
879 after that point in time then the BPEL4People Processor MUST activate the task immediately.

880 **Expiration:** Specifies the expiration time of the task when the task becomes obsolete. It is defined as  
881 either the period of time after which the task expires or the point in time when the task expires. The time  
882 starts to be measured when the task enters state *Created*. If the task does not reach one of the final  
883 states (*Completed*, *Failed*, *Error*, *Exited*, *Obsolete*) by the expiration time the BPEL4People Processor  
884 MUST change the task state to *Exited*. Additional user-defined actions MUST NOT be performed. The  
885 default value is infinity, i.e. the task never expires. If the expiration time is defined as a point in time and  
886 the task is created after that point in time the BPEL4People Processor MUST change the task state to  
887 *Exited*. Note that deferred activation does not impact expiration. Therefore the task MAY expire even  
888 before being activated.

889 Element `<b4p:scheduledActions>` is used to include the definition of all scheduled actions within the  
890 task definition. If present, at least one scheduled activity MUST be defined in the BPEL4People Definition.

### 891 Syntax:

```
892 <b4p:scheduledActions>?  
893  
894   <b4p:deferActivation>?  
895     ( <b4p:for expressionLanguage="anyURI"?>  
896       duration-expression  
897     </b4p:for>  
898   | <b4p:until expressionLanguage="anyURI"?>  
899     deadline-expression  
900   </b4p:until>  
901   )  
902 </b4p:deferActivation>  
903  
904   <b4p:expiration>?  
905     ( <b4p:for expressionLanguage="anyURI"?>  
906       duration-expression  
907     </b4p:for>  
908   | <b4p:until expressionLanguage="anyURI"?>  
909     deadline-expression  
910   </b4p:until>  
911   )  
912 </b4p:expiration>  
913  
914 </b4p:scheduledActions>
```

### 916 Properties

917 The `<b4p:scheduledActions>` element has the following optional elements:

- 918 • `b4p:deferActivation`: The element is used to specify activation time of the task. It includes  
919 the following elements:
  - 920 ○ `b4p:for`: The element is an expression which specifies the period of time (duration)  
921 after which the task reaches state *Ready* (in case of explicit claim) or state *Reserved* (in  
922 case of implicit claim). The absolute time of this transition is computed by adding the  
923 specified duration to the time at which the people activity begins execution.

924           o b4p:until: The element is an expression which specifies the point in time when the  
925           task reaches state *Ready* or state *Reserved*.

926 Elements <b4p:for> and <b4p:until> are mutually exclusive. There MUST be at least one  
927 <b4p:for> or <b4p:until> element.

928           • b4p:expiration: The element is used to specify the expiration time of the task when the task  
929           becomes obsolete:

930           o b4p:for: The element is an expression which specifies the period of time (duration)  
931           after which the task expires. The absolute time of the expiration is computed by adding  
932           the duration to the time at which the people activity begins execution.

933           o b4p:until: The element is an expression which specifies the point in time when the  
934           task expires.

935 Elements <b4p:for> and <b4p:until> are mutually exclusive. There MUST be at least one  
936 <b4p:for> or <b4p:until> element.

937 The language used in expressions is specified using the `expressionLanguage` attribute. This attribute  
938 is optional. If not specified, the default language as inherited from the closest enclosing element that  
939 specifies the attribute is used.

940 If specified, the `scheduledActions` element MUST NOT be empty, that is one of the elements  
941 `b4p:deferActivation` and `b4p:expiration` MUST be defined.

#### 942 **Example:**

```
943 <b4p:scheduledActions>  
944     <b4p:deferActivation>  
945         <b4p:documentation xml:lang="en-US">  
946             Activation of this task is deferred until the time specified  
947             in its input data.  
948         </b4p:documentation>  
949         <b4p:until>htd:getInput () /activateAt</b4p:until>  
950     </b4p:deferActivation>  
951  
952     <b4p:expiration>  
953         <b4p:documentation xml:lang="en-US">  
954             This task expires when not completed within 14 days after  
955             having been activated.  
956         </b4p:documentation>  
957         <b4p:for>P14D</b4p:for>  
958     </b4p:expiration>  
959 </b4p:scheduledActions>
```

## 962 **4.8 People Activity Behavior and State Transitions**

963 Figure 2 shows the different states of the people activity and state transitions with associated triggers  
964 (events and conditions) and actions to be performed when transitions take place.

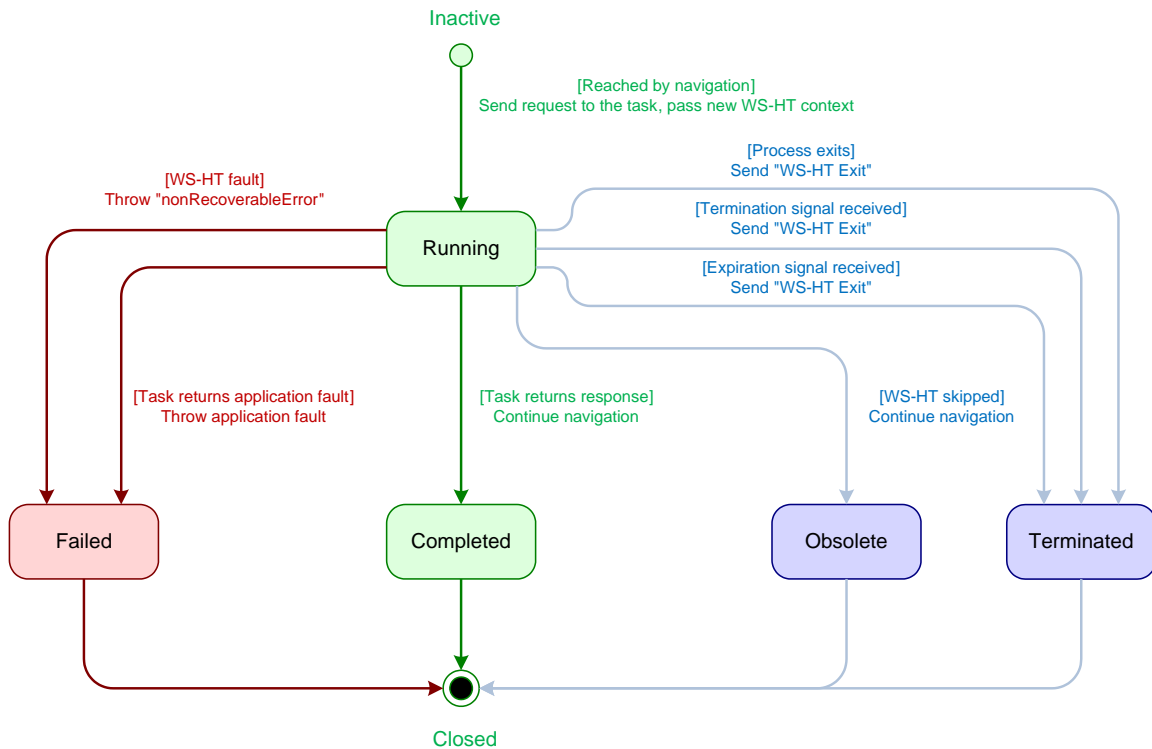


Figure 2: State diagram of the people activity

965  
966  
967

968 When the process execution instantiates a people activity this activity triggers the creation of a task in  
969 state *Running*. Upon receiving a response from the task, the people activity completes successfully and  
970 its state changes into the final state *Completed*.

971 If the task returns a fault, the people activity completes unsuccessfully and moves to final state *Failed* and  
972 the fault is thrown in the scope enclosing the people activity. If the task experiences a non-recoverable  
973 error, the people activity completes unsuccessfully and the standard fault `nonRecoverableError` is  
974 thrown in the enclosing scope.

975 The people activity goes to final state *Obsolete* if the task is skipped.

976 If the termination of the enclosed scope is triggered while the people activity is still running, the people  
977 activity is terminated prematurely and the associated running task is exited. A response for a terminated  
978 people activity **MUST** be ignored by the BPEL4People Processor.

979 If the task expires, the people activity is terminated prematurely and the associated task exits. In this case  
980 the standard fault `b4p:taskExpired` is thrown in the enclosing scope. When the process exits the  
981 people activity will also be terminated and the associated task is exited.

## 982 4.9 Task Instance Data

983 As defined by [WS-HumanTask], task instance data falls into the categories presentation data, context  
984 data, and operational data. Human tasks defined as part of a BPEL4People compliant business process  
985 have a superset of the instance data defined in [WS-HumanTask].

### 986 4.9.1 Presentation Data

987 The presentation data of tasks defined as part of a BPEL4People compliant business process is  
988 equivalent to that of a standalone human task.

## 989 **4.9.2 Context Data**

990 Tasks defined as part of a BPEL4People business process not only have access to the context data of  
991 the task, but also of the surrounding business process. The process context includes

- 992 • Process state like variables and ad-hoc attachments
- 993 • Values for all generic human roles of the business process, i.e. the process stakeholders, the  
994 business administrators of the process, and the process initiator
- 995 • Values for all generic human roles of human tasks running within the same business process

## 996 **4.9.3 Operational Data**

997 The operational data of tasks that is defined as part of a BPEL4People compliant business process is  
998 equivalent to that of a standalone human task.

## 5 XPath Extension Functions

1000 This section introduces XPath extension functions that are provided to be used within the definition of a  
 1001 BPEL4People business process to access process context. Definition of these XPath extension functions  
 1002 is provided in the table below. Input parameters that specify peopleActivity name MUST be literal strings.  
 1003 This restriction does not apply to other parameters. Because XPath 1.0 functions do not support returning  
 1004 faults, an empty node set is returned in the event of an error.  
 1005

Operation Name	Description	Parameters
getProcessStakeholders	Returns the stakeholders of the process. It MUST return an empty <code>htt:organizationalEntity</code> in case of an error.	Out <ul style="list-style-type: none"> <li>organizational entity (<code>htt:organizationalEntity</code>)</li> </ul>
getBusinessAdministrators	Returns the business administrators of the process. It MUST return an empty <code>htt:organizationalEntity</code> in case of an error.	Out <ul style="list-style-type: none"> <li>organizational entity (<code>htt:organizationalEntity</code>)</li> </ul>
getProcessInitiator	Returns the initiator of the process. It MUST return an empty <code>htt:tUser</code> in case of an error.	Out <ul style="list-style-type: none"> <li>the process initiator (<code>htt:tUser</code>)</li> </ul>
getLogicalPeopleGroup	Returns the value of a logical people group. It MUST return an empty <code>htt:organizationalEntity</code> in case of an error.	In <ul style="list-style-type: none"> <li>name of the logical people group (<code>xsd:string</code>)</li> <li>The optional parameters that follow MUST appear in pairs. Each pair is defined as:               <ul style="list-style-type: none"> <li>the qualified name of a logical people group parameter</li> <li>the value for the named logical people group parameter; it can be an XPath expression</li> </ul> </li> </ul> Out <ul style="list-style-type: none"> <li>the value of the logical people group (<code>htt:organizationalEnt</code>)</li> </ul>

Operation Name	Description	Parameters
		ity)
getActualOwner	Returns the actual owner of the task associated with the people activity. It MUST return an empty <code>htt:tUser</code> in case of an error.	In <ul style="list-style-type: none"> <li>people activity name (<code>xsd:string</code>)</li> </ul> Out <ul style="list-style-type: none"> <li>the actual owner (<code>htt:tUser</code>)</li> </ul>
getTaskInitiator	Returns the initiator of the task. Evaluates to an empty <code>htt:user</code> in case there is no initiator. It MUST return an empty <code>htt:tUser</code> in case of an error.	In <ul style="list-style-type: none"> <li>people activity name (<code>xsd:string</code>)</li> </ul> Out <ul style="list-style-type: none"> <li>the task initiator (user id as <code>htt:user</code>)</li> </ul>
getTaskStakeholders	Returns the stakeholders of the task. It MUST evaluate to an empty <code>htt:organizationalEntity</code> in case of an error.	In <ul style="list-style-type: none"> <li>people activity name (<code>xsd:string</code>)</li> </ul> Out <ul style="list-style-type: none"> <li>task stakeholders (<code>htt:organizationalEntity</code>)</li> </ul>
getPotentialOwners	Returns the potential owners of the task associated with the people activity. It MUST return an empty <code>htt:organizationalEntity</code> in case of an error.	In <ul style="list-style-type: none"> <li>people activity name (<code>xsd:string</code>)</li> </ul> Out <ul style="list-style-type: none"> <li>potential owners (<code>htt:organizationalEntity</code>)</li> </ul>
getAdministrators	Returns the administrators of the task associated with the people activity. It MUST return an empty <code>htt:organizationalEntity</code> in case of an error.	In <ul style="list-style-type: none"> <li>people activity name (<code>xsd:string</code>)</li> </ul> Out <ul style="list-style-type: none"> <li>business administrators (<code>htt:organizationalEntity</code>)</li> </ul>
getTaskPriority	Returns the priority of the task associated with the people activity. It MUST evaluate to "5" in case the priority is not explicitly set.	In <ul style="list-style-type: none"> <li>people activity name (<code>xsd:string</code>)</li> </ul> Out <ul style="list-style-type: none"> <li>priority (<code>htt:tPriority</code>)</li> </ul>

Operation Name	Description	Parameters
getOutcome	Returns the task outcome of the task associated with the people activity	In <ul style="list-style-type: none"> <li>people activity name (<code>xsd:string</code>)</li> </ul> Out <ul style="list-style-type: none"> <li>the task outcome (<code>xsd:string</code>)- if the outcome is not present, the empty nodeset MUST be returned.</li> </ul>
getState	Returns the state of the people activity	In <ul style="list-style-type: none"> <li>people activity name (<code>xsd:string</code>)</li> </ul> Out <ul style="list-style-type: none"> <li>the people activity state (<code>xsd:string</code> - see 4.8 People Activity Behavior and State Transitions)</li> </ul>

1006

1007

1008

XPath functions accessing data of a human task only guarantee to return data once the corresponding task has reached a final state.



1009  
1010  
1011  
1012  
1013  
1014

## 6 Coordinating Standalone Human Tasks

Using the *WS-HT coordination protocol* introduced by [WS-HumanTask] (see section 7 “Interoperable Protocol for Advanced Interaction with Human Tasks” for more details) to control the autonomy and life cycle of human tasks, a BPEL process with a people activity can act as the parent application for remote human tasks.

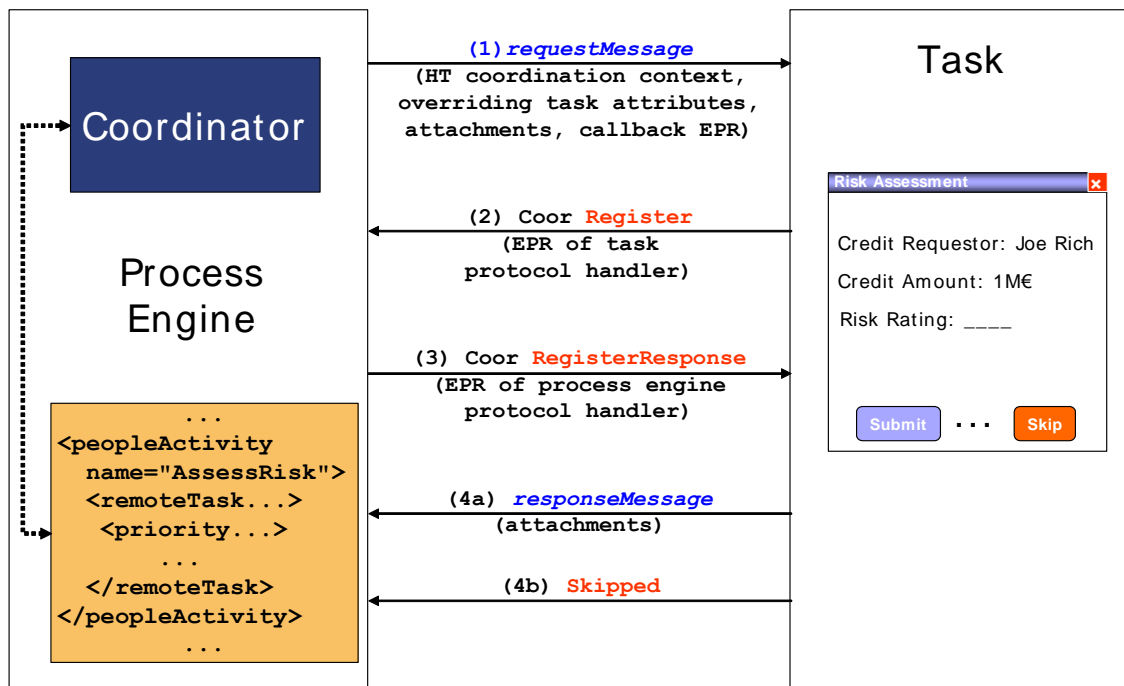


Figure 3: Message exchange between a people activity and a human task

1015  
1016  
1017  
1018  
1019

Figure 3 shows some message exchanges between a BPEL process containing a people activity to perform a task (e.g. risk assessment) implemented by a remote human. The behavior of the people activity is the same as for a people activity with an inline human task. That behavior is achieved by coordinating the remote human task via the WS-HT coordination protocol.

### 6.1 Protocol Messages from the People Activity’s Perspective

The BPEL4People Processor people activity MUST support the following behavior and the protocol messages exchanged with a standalone task. A summary is provided in the table below.

1. When the process execution reaches a people activity and determines that this activity can be executed, the BPEL4People Processor MUST create a WS-HT coordination context associated with the activity. This context is sent together with the request message to the appropriate service associated with the task. In addition, overriding attributes from the people activity, namely priority, people assignments, the skipable indicator and the task’s expiration time, are sent. Also the BPEL4People Processor MAY propagate ad-hoc attachments from the process. All this information is sent as part of the header fields of the requesting message. These header fields as well as a corresponding mapping to SOAP headers are discussed in [WS-HumanTask].

- 1031 2. When a response message is received from the task that indicates the successful completion of  
 1032 the task, the people activity completes. This response MAY include all new ad-hoc attachments  
 1033 from the human task.
- 1034 3. When a response message is received from the task that indicates a fault of the task, the people  
 1035 activity faults. The fault MUST be thrown in the scope of the people activity.
- 1036 4. When protocol message fault is received, the fault nonRecoverableError MUST be thrown in the  
 1037 scope enclosing the people activity.
- 1038 5. When protocol message skipped is received, the people activity MUST move to state *Obsolete*.
- 1039 6. If the task does not reach one of the final states by the expiration deadline, the people activity  
 1040 MUST be terminated. Protocol message exit is sent to the task.
- 1041 7. When the people activity is terminated, protocol message exit MUST be sent to the task.
- 1042 8. When the process encounters an <exit> activity, protocol message exit MUST be sent to the task.  
 1043

1044 The following table summarizes this behavior, the protocol messages sent, and their direction, i.e.,  
 1045 whether a message is sent from the people activity to the task (“out” in the column titled Direction) or vice  
 1046 versa (“in”).  
 1047

Message	Direction	People activity behavior
application request with WS-HT coordination context (and callback information)	Out	People activity reached
task response	In	People activity completes
task fault response	In	People activity faults
Fault	In	People activity faults with b4p:nonRecoverableError
Skipped	In	People activity is set to obsolete
Exit	Out	Expired time-out
Exit	Out	People activity terminated
Exit	Out	<exit> encountered in enclosing process

---

## 1048 7 BPEL Abstract Processes

1049 BPEL abstract processes are indicated by the namespace "`http://docs.oasis-`  
1050 `open.org/wsbpel/2.0/process/abstract`". All constructs defined in BPEL4People extension  
1051 namespaces MAY appear in abstract processes.

### 1052 7.1 Hiding Syntactic Elements

1053 Opaque tokens defined in BPEL (activities, expressions, attributes and from-specs) MAY be used in  
1054 BPEL4People extension constructs. The syntactic validity constraints of BPEL MUST apply in the same  
1055 way to an Executable Completion of an abstract process containing BPEL4People extensions.

#### 1056 7.1.1 Opaque Activities

1057 BPEL4people does not change the way opaque activities can be replaced by an executable activity in an  
1058 executable completion of an abstract process, that is, an `<abstract:opaqueActivity>` MAY also  
1059 serve as a placeholder for a `<bpel:extensionActivity>` containing a `<b4p:peopleActivity>`.

#### 1060 7.1.2 Opaque Expressions

1061 Any expression introduced by BPEL4People MAY be made opaque. In particular, the following  
1062 expressions MAY have the `opaque="yes"` attribute:

```
1063 <htd:argument name="NCName" expressionLanguage="anyURI"? opaque="yes" />  
1064 <htd:priority expressionLanguage="anyURI" opaque="yes" />  
1065 <b4p:for expressionLanguage="anyURI"? opaque="yes" />  
1066 <b4p:until expressionLanguage="anyURI"? opaque="yes" />
```

#### 1067 7.1.3 Opaque Attributes

1068 Any attribute introduced by BPEL4People MAY have an opaque value "`##opaque`" in an abstract  
1069 process.

#### 1070 7.1.4 Opaque From-Spec

1071 In BPEL, any from-spec in an executable process can be replaced by an opaque from-spec  
1072 `<opaqueFrom/>` in an abstract process. This already includes any BPEL from-spec extended with the  
1073 BPEL4People `b4p:logicalPeopleGroup="NCName"` attribute. In addition, the extension from-spec  
1074 `<htd:from>` MAY also be replaced by an opaque from-spec in an abstract process.

#### 1075 7.1.5 Omission

1076 In BPEL, omissible tokens are all attributes, activities, expressions and from-specs which are both (1)  
1077 syntactically required by the Executable BPEL XML Schema, and (2) have no default value. This rule also  
1078 applies to BPEL4People extensions in abstract processes. For example, `<b4p:localTask`  
1079 `reference="##opaque">` is equivalent to `<b4p:localTask>`.

## 1080 7.2 Abstract Process Profile for Observable Behavior

1081 The Abstract Process Profile for Observable Behavior, indicated by the process attribute  
1082 `abstractProcessProfile="http://docs.oasis-`  
1083 `open.org/wsbpel/2.0/process/abstract/ap11/2006/08"`, provides a means to create precise  
1084 and predictable descriptions of observable behavior of the service(s) provided by an executable process.

1085 The main application of this profile is the definition of business process contracts; that is, the behavior  
1086 followed by one business partner in the context of Web services exchanges. A valid completion has to  
1087 follow the same interactions as the abstract process, with the partners that are specified by the abstract  
1088 process. The executable process can, however, perform additional interaction steps relating to other  
1089 partners. Likewise, the executable process can perform additional human interactions. Beyond the  
1090 restrictions defined in WS-BPEL 2.0, the use of opacity is not restricted in any way for elements and  
1091 attributes introduced by BPEL4People.

### 1092 **7.3 Abstract Process Profile for Templates**

1093 The Abstract Process Profile for Templates, indicated by the process attribute  
1094 `abstractProcessProfile="http://docs.oasis-`  
1095 `open.org/wsbpel/2.0/process/abstract/simple-template/2006/08"`, allows the definition  
1096 of Abstract Processes which hide almost any arbitrary execution details and have explicit opaque  
1097 extension points for adding behavior.

1098 This profile does not allow the use of omission shortcuts but the use of opacity is not restricted in any  
1099 way. For abstract processes belonging to this profile, this rule is extended to the elements and attributes  
1100 introduced by BPEL4People.

---

## 1101 **8 Conformance**

1102 The XML schema pointed to by the RDDDL document at the namespace URI, defined by this specification,  
1103 are considered to be authoritative and take precedence over the XML schema defined in the appendix of  
1104 this document.

1105

1106 There are four conformance targets defined as part of this specification: a BPEL4People Definition, a  
1107 BPEL4People Processor, a WS-HumanTask Definition and a WS-HumanTask Processor (see section  
1108 2.3). In order to claim conformance with BPEL4People 1.1, the conformance targets **MUST** comply with  
1109 all normative statements in the BPEL4People and the WS-HumanTask specification, notably all **MUST**  
1110 statements have to be implemented.

1111

---

## A. Standard Faults

1112 The following list specifies the standard faults defined within the BPEL4People specification. All standard  
1113 fault names are qualified with the standard BPEL4People namespace.

Fault name	Description
nonRecoverableError	Thrown if the task experiences a non-recoverable error.
taskExpired	Thrown if the task expired.

1114

---

## B. Portability and Interoperability Considerations

1115 The following section illustrates the portability and interoperability aspects of the various usage  
1116 constellations of BPEL4People with WS-HumanTask as described in Figure 1:

1117

1118 Portability - The ability to take design-time artifacts created in one vendor's environment and use them in  
1119 another vendor's environment. Constellations one and two provide portability of BPEL4People processes  
1120 with embedded human interactions in. Constellations three and four provide portability of BPEL4People  
1121 processes with referenced human interactions.

1122

1123 Interoperability - The capability for multiple components (process engine, task engine and task list client)  
1124 to interact using well-defined messages and protocols. This enables to combine components from  
1125 different vendors allowing seamless execution.

1126 Constellation four achieves interoperability between process and tasks from different vendor  
1127 implementations.

1128

1129 Constellation 1

1130 Task definitions are defined inline of the people activities. Usage in this manner is typically for self-  
1131 contained people activities, whose tasks definitions are not intended to be reused elsewhere in the  
1132 process or across multiple processes. This format will also provide scoping of the task definition since it  
1133 will not be visible or accessible outside the people activity in which it is contained. Portability for this  
1134 constellation requires support of both WS-HumanTask and BPEL4People artifacts using the inline task  
1135 definition format. Since the process and task interactions are combined in one component, interoperability  
1136 requirements are limited to those between the task list client and the infrastructure.

1137

1138 Constellation 2

1139 Similar to constellation 1, but tasks are defined at the process level. This allows task definitions to be  
1140 referenced from within people activities enabling task reuse. Portability for this constellation requires  
1141 support of both WS-HumanTask and BPEL4People artifacts using the process level scoped task  
1142 definition format. Since the process and task interactions are combined in one component, interoperability  
1143 requirements are limited to those between the task list client and the infrastructure.

1144

1145 Constellation 3

1146 In this constellation, the task and people activity definitions are defined as separate artifacts and execute  
1147 in different infrastructure components but provided by the same vendor. Portability for this constellation  
1148 requires support of both WS-HumanTask and BPEL4People as separate artifacts. Since the process and  
1149 task components are implemented by the same vendor, interoperability requirements are limited to those  
1150 between the task list client and the infrastructure.

1151

1152 Constellation 4

1153 Identical to constellation 3 in terms of the task and people activity definitions, but in this case the process  
1154 and task infrastructure are provided by different vendors. Portability for this constellation requires support  
1155 of both WS-HumanTask and BPEL4People as separate artifacts. Interoperability between task and  
1156 process infrastructures from different vendors is achieved using the WS-HumanTask coordination  
1157 protocol.

1158

## C. BPEL4People Schema

```

1159 <?xml version="1.0" encoding="UTF-8"?>
1160 <!--
1161   Copyright (c) OASIS Open 2009. All Rights Reserved.
1162 -->
1163 <xsd:schema
1164   targetNamespace="http://docs.oasis-
1165 open.org/ns/bpel4people/bpel4people/200803"
1166   xmlns="http://docs.oasis-open.org/ns/bpel4people/bpel4people/200803"
1167   xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
1168   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1169   xmlns:htd="http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803"
1170   elementFormDefault="qualified"
1171   blockDefault="#all">
1172
1173   <xsd:annotation>
1174     <xsd:documentation>
1175       XML Schema for BPEL4People 1.1 - WS-BPEL 2.0 Extension for Human Task
1176 Interactions
1177     </xsd:documentation>
1178   </xsd:annotation>
1179
1180   <!-- other namespaces -->
1181   <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
1182     schemaLocation="http://www.w3.org/2001/xml.xsd" />
1183   <xsd:import namespace="http://docs.oasis-open.org/ns/bpel4people/ws-
1184 humantask/200803"
1185     schemaLocation="ws-humantask.xsd" />
1186   <xsd:import namespace="http://docs.oasis-
1187 open.org/wsbpel/2.0/process/executable"
1188     schemaLocation="http://docs.oasis-
1189 open.org/wsbpel/2.0/OS/process/executable/ws-bpel_executable.xsd" />
1190
1191   <!-- base types for extensible elements -->
1192   <xsd:complexType name="tExtensibleElements">
1193     <xsd:sequence>
1194       <xsd:element name="documentation" type="tDocumentation"
1195         minOccurs="0" maxOccurs="unbounded" />
1196       <xsd:any namespace="##other" processContents="lax" minOccurs="0"
1197         maxOccurs="unbounded" />
1198     </xsd:sequence>
1199     <xsd:anyAttribute namespace="##other" processContents="lax" />
1200   </xsd:complexType>
1201   <xsd:complexType name="tExtensibleMixedNamespaceElements">
1202     <xsd:sequence>
1203       <xsd:element name="documentation" type="tDocumentation"
1204         minOccurs="0" maxOccurs="unbounded" />
1205       <xsd:element name="extensions" type="tExtensions" minOccurs="0" />
1206     </xsd:sequence>
1207     <xsd:anyAttribute namespace="##other" processContents="lax" />
1208   </xsd:complexType>
1209   <xsd:complexType name="tDocumentation" mixed="true">
1210     <xsd:sequence>
1211     <xsd:any namespace="##other" processContents="lax" minOccurs="0"

```



```

1212         maxOccurs="unbounded" />
1213     </xsd:sequence>
1214     <xsd:attribute ref="xml:lang" />
1215 </xsd:complexType>
1216 <xsd:complexType name="tExtensions">
1217     <xsd:sequence>
1218         <xsd:any namespace="##other" processContents="lax" minOccurs="0"
1219             maxOccurs="unbounded" />
1220     </xsd:sequence>
1221 </xsd:complexType>
1222
1223 <!-- element "humanInteractions" to be used within "bpel:process" -->
1224 <xsd:element name="humanInteractions" type="tHumanInteractions" />
1225 <xsd:complexType name="tHumanInteractions">
1226     <xsd:complexContent>
1227         <xsd:extension base="tExtensibleMixedNamespaceElements">
1228             <xsd:sequence>
1229                 <xsd:element ref="htd:logicalPeopleGroups" minOccurs="0" />
1230                 <xsd:element ref="htd:tasks" minOccurs="0" />
1231                 <xsd:element ref="htd:notifications" minOccurs="0" />
1232             </xsd:sequence>
1233         </xsd:extension>
1234     </xsd:complexContent>
1235 </xsd:complexType>
1236
1237 <!-- element "peopleAssignments" to be used within "bpel:process" -->
1238 <xsd:element name="peopleAssignments" type="tPeopleAssignments" />
1239 <xsd:complexType name="tPeopleAssignments">
1240     <xsd:complexContent>
1241         <xsd:extension base="tExtensibleElements">
1242             <xsd:sequence>
1243                 <xsd:element ref="genericHumanRole" minOccurs="1"
1244 maxOccurs="unbounded" />
1245             </xsd:sequence>
1246         </xsd:extension>
1247     </xsd:complexContent>
1248 </xsd:complexType>
1249
1250 <!-- element "genericHumanRole" within BPEL4People -->
1251 <xsd:element name="genericHumanRole"
1252 type="htd:tGenericHumanRoleAssignmentBase" abstract="true" block=""/>
1253
1254     <xsd:element name="processStakeholders"
1255 type="htd:tGenericHumanRoleAssignment" substitutionGroup="genericHumanRole"/>
1256     <xsd:element name="businessAdministrators"
1257 type="htd:tGenericHumanRoleAssignment" substitutionGroup="genericHumanRole"/>
1258     <xsd:element name="processInitiator" type="htd:tGenericHumanRoleAssignment"
1259 substitutionGroup="genericHumanRole"/>
1260
1261 <!-- element "argument" to be used within "bpel:from" -->
1262 <xsd:element name="argument" type="tArgument" />
1263 <xsd:complexType name="tArgument">
1264     <xsd:complexContent>
1265         <xsd:extension base="bpel:tExpression">
1266             <xsd:attribute name="name" type="xsd:NCName" />
1267         </xsd:extension>
1268     </xsd:complexContent>
1269 </xsd:complexType>

```

```

1270
1271     <!-- attribute "logicalPeopleGroup" to be used within "bpel:from" and
1272 "bpel:to" -->
1273     <xsd:attribute name="logicalPeopleGroup" type="xsd:NCName" />
1274
1275     <!-- attribute "shareComments" to be used within "bpel:process" and
1276 "bpel:scope" -->
1277     <xsd:attribute name="shareComments" type="xsd:boolean" />
1278
1279     <!-- element "peopleActivity" to be used within "bpel:extensionActivity" --
1280 >
1281     <xsd:element name="peopleActivity" type="tPeopleActivity" />
1282     <xsd:complexType name="tPeopleActivity">
1283         <xsd:complexContent>
1284             <xsd:extension base="tExtensibleMixedNamespaceElements">
1285                 <xsd:sequence>
1286                     <xsd:element ref="bpel:targets" minOccurs="0" />
1287                     <xsd:element ref="bpel:sources" minOccurs="0" />
1288                     <xsd:choice>
1289                         <xsd:element ref="htd:task" />
1290                         <xsd:element ref="localTask" />
1291                         <xsd:element ref="remoteTask" />
1292                         <xsd:element ref="htd:notification" />
1293                         <xsd:element ref="localNotification" />
1294                         <xsd:element ref="remoteNotification" />
1295                     </xsd:choice>
1296                     <xsd:element ref="scheduledActions" minOccurs="0" />
1297                     <xsd:element ref="toParts" minOccurs="0" />
1298                     <xsd:element ref="fromParts" minOccurs="0" />
1299                     <xsd:element ref="attachmentPropagation" minOccurs="0" />
1300                     <xsd:any namespace="##other" processContents="lax"
1301                         minOccurs="0" maxOccurs="unbounded" />
1302                 </xsd:sequence>
1303                 <xsd:attribute name="name" type="xsd:NCName" />
1304                 <xsd:attribute name="suppressJoinFailure" type="tBoolean"
1305                     use="optional" />
1306                 <xsd:attribute name="inputVariable" type="xsd:QName" />
1307                 <xsd:attribute name="outputVariable" type="xsd:QName" />
1308                 <xsd:attribute name="isSkipable" type="tBoolean"
1309                     use="optional" default="no" />
1310                 <xsd:attribute name="dontShareComments" type="tBoolean"
1311                     use="optional" default="no" />
1312             </xsd:extension>
1313         </xsd:complexContent>
1314     </xsd:complexType>
1315     <xsd:complexType name="tOverridableTaskElements">
1316         <xsd:complexContent>
1317             <xsd:extension base="tExtensibleMixedNamespaceElements">
1318                 <xsd:sequence>
1319                     <xsd:element ref="htd:priority" minOccurs="0" />
1320                     <xsd:element ref="htd:peopleAssignments" minOccurs="0" />
1321                 </xsd:sequence>
1322             </xsd:extension>
1323         </xsd:complexContent>
1324     </xsd:complexType>
1325     <xsd:element name="localTask" type="tLocalTask" />
1326     <xsd:complexType name="tLocalTask">
1327         <xsd:complexContent>

```

```

1328     <xsd:extension base="tOverridableTaskElements">
1329         <xsd:attribute name="reference" type="xsd:QName"
1330             use="required" />
1331     </xsd:extension>
1332 </xsd:complexContent>
1333 </xsd:complexType>
1334 <xsd:element name="remoteTask" type="tRemoteTask" />
1335 <xsd:complexType name="tRemoteTask">
1336     <xsd:complexContent>
1337         <xsd:extension base="tOverridableTaskElements">
1338             <xsd:attribute name="partnerLink" type="xsd:NCName"
1339                 use="required" />
1340             <xsd:attribute name="operation" type="xsd:NCName"
1341                 use="required" />
1342             <xsd:attribute name="responseOperation" type="xsd:NCName" />
1343         </xsd:extension>
1344     </xsd:complexContent>
1345 </xsd:complexType>
1346 <xsd:complexType name="tOverridableNotificationElements">
1347     <xsd:complexContent>
1348         <xsd:extension base="tExtensibleMixedNamespaceElements">
1349             <xsd:sequence>
1350                 <xsd:element ref="htd:priority" minOccurs="0" />
1351                 <xsd:element ref="htd:peopleAssignments" minOccurs="0" />
1352             </xsd:sequence>
1353         </xsd:extension>
1354     </xsd:complexContent>
1355 </xsd:complexType>
1356 <xsd:element name="localNotification" type="tLocalNotification" />
1357 <xsd:complexType name="tLocalNotification">
1358     <xsd:complexContent>
1359         <xsd:extension base="tOverridableNotificationElements">
1360             <xsd:attribute name="reference" type="xsd:QName"
1361                 use="required" />
1362         </xsd:extension>
1363     </xsd:complexContent>
1364 </xsd:complexType>
1365 <xsd:element name="remoteNotification" type="tRemoteNotification" />
1366 <xsd:complexType name="tRemoteNotification">
1367     <xsd:complexContent>
1368         <xsd:extension base="tOverridableNotificationElements">
1369             <xsd:attribute name="partnerLink" type="xsd:NCName"
1370                 use="required" />
1371             <xsd:attribute name="operation" type="xsd:NCName"
1372                 use="required" />
1373         </xsd:extension>
1374     </xsd:complexContent>
1375 </xsd:complexType>
1376 <xsd:element name="scheduledActions" type="tScheduledActions" />
1377 <xsd:complexType name="tScheduledActions">
1378     <xsd:complexContent>
1379         <xsd:extension base="tExtensibleElements">
1380             <xsd:sequence>
1381                 <xsd:element name="deferActivation"
1382                     type="tScheduledActionsDetails" minOccurs="0" />
1383                 <xsd:element name="expiration"
1384                     type="tScheduledActionsDetails" minOccurs="0" />
1385             </xsd:sequence>

```

```

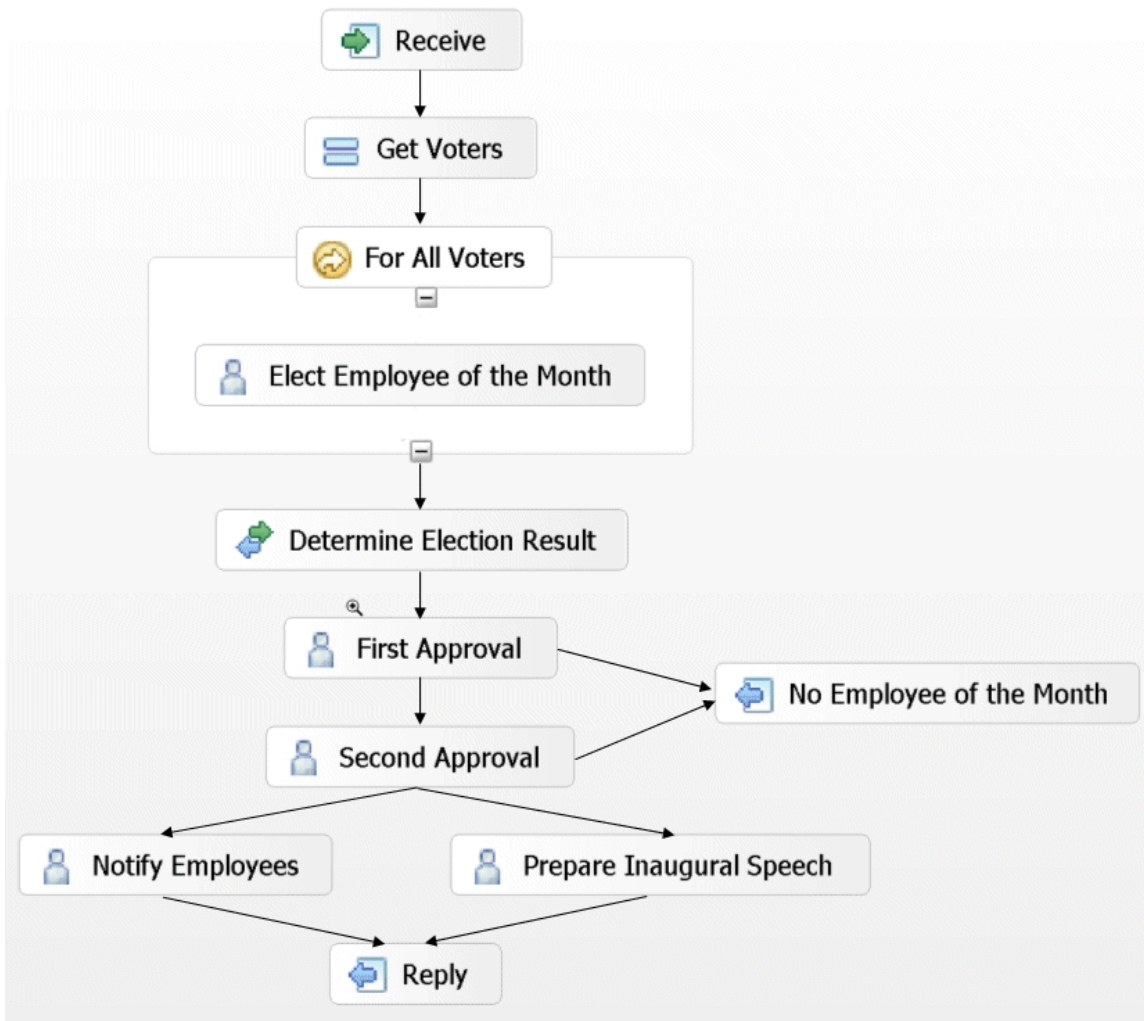
1386     </xsd:extension>
1387   </xsd:complexContent>
1388 </xsd:complexType>
1389 <xsd:complexType name="tScheduledActionsDetails">
1390   <xsd:complexContent>
1391     <xsd:extension base="tExtensibleElements">
1392       <xsd:sequence>
1393         <xsd:choice>
1394           <xsd:element name="for" type="bpel:tDuration-expr" />
1395           <xsd:element name="until" type="bpel:tDeadline-expr" />
1396         </xsd:choice>
1397       </xsd:sequence>
1398     </xsd:extension>
1399   </xsd:complexContent>
1400 </xsd:complexType>
1401 <xsd:element name="fromParts" type="tFromParts" />
1402 <xsd:complexType name="tFromParts">
1403   <xsd:complexContent>
1404     <xsd:extension base="tExtensibleElements">
1405       <xsd:sequence>
1406         <xsd:element ref="fromPart" maxOccurs="unbounded" />
1407       </xsd:sequence>
1408     </xsd:extension>
1409   </xsd:complexContent>
1410 </xsd:complexType>
1411 <xsd:element name="fromPart" type="tFromPart" />
1412 <xsd:complexType name="tFromPart">
1413   <xsd:complexContent>
1414     <xsd:extension base="tExtensibleElements">
1415       <xsd:attribute name="part" type="xsd:NCName" use="required" />
1416       <xsd:attribute name="toVariable" type="bpel:BPELVariableName"
1417         use="required" />
1418     </xsd:extension>
1419   </xsd:complexContent>
1420 </xsd:complexType>
1421 <xsd:element name="toParts" type="tToParts" />
1422 <xsd:complexType name="tToParts">
1423   <xsd:complexContent>
1424     <xsd:extension base="tExtensibleElements">
1425       <xsd:sequence>
1426         <xsd:element ref="toPart" maxOccurs="unbounded" />
1427       </xsd:sequence>
1428     </xsd:extension>
1429   </xsd:complexContent>
1430 </xsd:complexType>
1431 <xsd:element name="toPart" type="tToPart" />
1432 <xsd:complexType name="tToPart">
1433   <xsd:complexContent>
1434     <xsd:extension base="tExtensibleElements">
1435       <xsd:attribute name="part" type="xsd:NCName" use="required" />
1436       <xsd:attribute name="fromVariable"
1437         type="bpel:BPELVariableName" use="required" />
1438     </xsd:extension>
1439   </xsd:complexContent>
1440 </xsd:complexType>
1441 <xsd:element name="attachmentPropagation"
1442   type="tAttachmentPropagation" />
1443 <xsd:complexType name="tAttachmentPropagation">

```

```
1444 <xsd:complexContent>
1445   <xsd:extension base="tExtensibleElements">
1446     <xsd:attribute name="fromProcess" type="tFromProcess"
1447       default="all" />
1448     <xsd:attribute name="toProcess" type="tToProcess"
1449       default="newOnly" />
1450   </xsd:extension>
1451 </xsd:complexContent>
1452 </xsd:complexType>
1453 <xsd:simpleType name="tFromProcess">
1454   <xsd:restriction base="xsd:string">
1455     <xsd:enumeration value="all" />
1456     <xsd:enumeration value="none" />
1457   </xsd:restriction>
1458 </xsd:simpleType>
1459 <xsd:simpleType name="tToProcess">
1460   <xsd:restriction base="xsd:string">
1461     <xsd:enumeration value="all" />
1462     <xsd:enumeration value="newOnly" />
1463     <xsd:enumeration value="none" />
1464   </xsd:restriction>
1465 </xsd:simpleType>
1466
1467 <!-- miscellaneous helper elements and types -->
1468 <xsd:simpleType name="tBoolean">
1469   <xsd:restriction base="xsd:string">
1470     <xsd:enumeration value="yes" />
1471     <xsd:enumeration value="no" />
1472   </xsd:restriction>
1473 </xsd:simpleType>
1474
1475 </xsd:schema>
```

1476 **D. Sample**

1477 This appendix contains a sample that outlines the basic concepts of this specification. The sample  
1478 process implements the election of the “Employee of the month” in a fictitious company. The structure of  
1479 the business process is shown in the figure below:



1480  
1481 The process is started and as a first step, the people are determined that qualify as voters for the  
1482 “Employee of the month”. Next, all the voters identified before get a chance to cast their votes. After that,  
1483 the election result is determined by counting the votes casted. After the result is clear, two different  
1484 people from the set of people entitled to approve the election either accept or reject the voting result. In  
1485 case any of the two rejects, then there is no “Employee of the month” elected in the given month, and the  
1486 process ends. In case all approvals are obtained successfully, the employees are notified about the  
1487 outcome of the election, and a to-do is created for the elected “Employee of the month” to prepare an  
1488 inaugural speech. Once this is completed, the process completes successfully.

1489 The sections below show the definition of the BPEL process implementing the “Employee of the month”  
1490 process.

## 1491 D.1 BPEL Definition

```
1492 <?xml version="1.0" encoding="UTF-8"?>
1493 <!--
1494   Copyright (c) OASIS Open 2009. All Rights Reserved.
1495 -->
1496 <process name="EmployeeOfTheMonthProcess"
1497   targetNamespace="http://www.example.com"
1498   xmlns:tns="http://www.example.com"
1499   xmlns:hr="http://www.example.com/approval"
1500   xmlns:el="http://www.example.com/election"
1501   xmlns:ty="http://www.example.com/types"
1502   xmlns:ta="http://www.example.com/tasks"
1503   xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
1504   xmlns:b4p="http://docs.oasis-open.org/ns/bpel4people/bpel4people/200803"
1505   xmlns:htd="http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803"
1506   xmlns:htt="http://docs.oasis-open.org/ns/bpel4people/ws-
1507   humantask/types/200803"
1508   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1509   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1510   xsi:schemaLocation="http://docs.oasis-
1511   open.org/ns/bpel4people/bpel4people/200803 ../../xml/bpel4people.xsd
1512   http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803 ../../xml/ws-
1513   humantask.xsd http://docs.oasis-open.org/ns/bpel4people/ws-
1514   humantask/types/200803 ../../xml/ws-humantask-types.xsd">
1515
1516   <documentation>
1517     Example for BPEL4People 1.1 - WS-BPEL 2.0 Process with BPEL4People
1518     Extensions
1519   </documentation>
1520
1521   <b4p:humanInteractions>
1522
1523     <htd:logicalPeopleGroups>
1524
1525       <htd:logicalPeopleGroup name="voters">
1526         <htd:documentation xml:lang="en-US">
1527           The group entitled to vote the employee of the month for the
1528           given region.
1529         </htd:documentation>
1530         <htd:parameter name="region" type="xsd:string" />
1531       </htd:logicalPeopleGroup>
1532
1533       <htd:logicalPeopleGroup name="approvers">
1534         <htd:documentation xml:lang="en-US">
1535           The group entitled to approve the elected employee of the
1536           month for the given region.
1537         </htd:documentation>
1538         <htd:parameter name="region" type="xsd:string" />
1539       </htd:logicalPeopleGroup>
1540
1541       <htd:logicalPeopleGroup name="employees">
1542         <htd:documentation xml:lang="en-US">
1543           The group of employees to be notified about the election
1544           result of the employee of the month election for the given
1545           region.
1546         </htd:documentation>
1547         <htd:parameter name="region" type="xsd:string" />

```

```

1548 </htd:logicalPeopleGroup>
1549
1550 <htd:logicalPeopleGroup name="regionalElectionCommittee">
1551   <htd:documentation xml:lang="en-US">
1552     The group who is in charge for the election of the
1553     employee of the month election for the given region.
1554   </htd:documentation>
1555   <htd:parameter name="region" type="xsd:string" />
1556 </htd:logicalPeopleGroup>
1557
1558 </htd:logicalPeopleGroups>
1559
1560 <htd:tasks>
1561   <htd:task name="approveEmployeeOfTheMonth">
1562     <htd:documentation xml:lang="en-US">
1563       The reusable definition of the task used to approve the
1564       election of the employee of the month.
1565     </htd:documentation>
1566     <htd:interface operation="approve" portType="hr:approvalPT"/>
1567     <htd:peopleAssignments>
1568       <htd:potentialOwners>
1569         <htd:from logicalPeopleGroup="approvers">
1570           <!-- variables used here need to be defined on the
1571           enclosing scope or above -->
1572           <htd:argument name="region">
1573             $selectionRequest/region
1574           </htd:argument>
1575         </htd:from>
1576       </htd:potentialOwners>
1577     </htd:peopleAssignments>
1578     <htd:presentationElements/>
1579   </htd:task>
1580 </htd:tasks>
1581
1582 </b4p:humanInteractions>
1583
1584 <b4p:peopleAssignments>
1585
1586   <b4p:processStakeholders>
1587     <htd:from logicalPeopleGroup="regionalElectionCommittee">
1588       <htd:argument name="region">
1589         $selectionRequest/region
1590       </htd:argument>
1591     </htd:from>
1592   </b4p:processStakeholders>
1593
1594   <b4p:businessAdministrators>
1595     <htd:from>
1596       <htd:literal>
1597         <htt:organizationalEntity>
1598           <htt:user>Peter</htt:user>
1599           <htt:user>Paul</htt:user>
1600           <htt:user>Mary</htt:user>
1601         </htt:organizationalEntity>
1602       </htd:literal>
1603     </htd:from>
1604   </b4p:businessAdministrators>
1605

```



```

1606 </b4p:peopleAssignments>
1607
1608 <extensions>
1609   <extension
1610     namespace="http://docs.oasis-
1611 open.org/ns/bpel4people/bpel4people/200803"
1612     mustUnderstand="yes"/>
1613   <extension
1614     namespace="http://docs.oasis-open.org/ns/bpel4people/ws-
1615 humantask/200803"
1616     mustUnderstand="yes"/>
1617 </extensions>
1618
1619 <import
1620   importType="http://www.w3.org/2001/XMLSchema"
1621   namespace="http://www.example.com/types"/>
1622 <import
1623   importType="http://www.example.org/WS-HT"
1624   namespace="http://www.example.com/tasks"/>
1625 <import
1626   importType="http://schemas.xmlsoap.org/wsd/"
1627   namespace="http://www.example.com/election"
1628   location="bpel4people-example-election.wsd"/>
1629 <import
1630   importType="http://schemas.xmlsoap.org/wsd/"
1631   namespace="http://www.example.com/approval"
1632   location="bpel4people-example-approval.wsd"/>
1633
1634 <partnerLinks>
1635   <partnerLink partnerLinkType="electionPLT"
1636     name="electionPL"/>
1637 </partnerLinks>
1638
1639 <variables>
1640   <variable name="candidates" type="htt:users"/>
1641   <variable name="voters" type="htd:tOrganizationalEntity"/>
1642   <variable name="electionRequest" type="ty:electionRequestData"/>
1643   <variable name="electionResult" type="ty:electionResultData"/>
1644   <variable name="decision" type="xsd:boolean"/>
1645   <variable name="speech" type="ty:document"/>
1646 </variables>
1647
1648 <sequence>
1649
1650   <receive partnerLink="electionPL"
1651     portType="el:electionPT"
1652     operation="elect"
1653     variable="electionRequest"
1654     createInstance="yes"/>
1655
1656   <assign name="getVoters">
1657     <copy>
1658       <from>$electionRequests/candidates</from>
1659       <to variable="candidates"/>
1660     </copy>
1661     <copy>
1662       <from b4p:logicalPeopleGroup="voters">
1663         <b4p:argument name="region">

```

```

1664     $SelectionRequest/region
1665     </b4p:argument>
1666 </from>
1667     <to variable="voters" />
1668 </copy>
1669 </assign>
1670
1671 <forEach counterName="i" parallel="yes">
1672     <startCounterValue> 1 </startCounterValue>
1673     <finalCounterValue>
1674         count($voters/users/user)
1675     </finalCounterValue>
1676
1677     <scope>
1678         <variables>
1679             <variable name="vote" type="htt:user"/>
1680         </variables>
1681
1682         <sequence>
1683             <!-- Constellation 1 -->
1684             <extensionActivity>
1685                 <b4p:peopleActivity name="electEmployeeOfTheMonth"
1686                     inputVariable="candidates"
1687                     outputVariable="vote"
1688                     isSkipable="yes">
1689                     <htd:task name="votingTask">
1690                         <htd:interface operation="vote"
1691                             portType="el:votingPT"/>
1692                     <htd:peopleAssignments>
1693                         <htd:potentialOwners>
1694                             <htd:from>$voters/users/user[i]</htd:from>
1695                         </htd:potentialOwners>
1696                     </htd:peopleAssignments>
1697                     <htd:presentationElements/>
1698                 </htd:task>
1699                 <b4p:scheduledActions>
1700                     <b4p:expiration>
1701                         <b4p:documentation xml:lang="en-US">
1702                             This people activity expires when not completed
1703                             within 2 days after having been activated.
1704                         </b4p:documentation>
1705                         <b4p:for>P2D</b4p:for>
1706                     </b4p:expiration>
1707                 </b4p:scheduledActions>
1708                 </b4p:peopleActivity>
1709             </extensionActivity>
1710
1711             <assign>
1712                 <copy>
1713                     <from>$vote</from>
1714                     <to>$SelectionResult/votes[i]</to>
1715                 </copy>
1716             </assign>
1717
1718         </sequence>
1719     </scope>
1720 </forEach>
1721

```

```

1722 <!-- Might be Constellation 5 - standard WS-BPEL 2.0 invoke -->
1723 <!--
1724 <invoke name="determineElectionResult" partnerLink="..." operation="..."
1725 />
1726 -->
1727
1728 <!-- Constellation 2 -->
1729 <extensionActivity>
1730   <b4p:peopleActivity name="firstApproval"
1731     inputVariable="electionResult"
1732     outputVariable="decision">
1733     <b4p:localTask reference="tns:approveEmployeeOfTheMonth"/>
1734   </b4p:peopleActivity>
1735 </extensionActivity>
1736
1737 <!-- Constellation 2 with override specifications -->
1738 <extensionActivity>
1739   <b4p:peopleActivity name="secondApproval"
1740     inputVariable="electionResult"
1741     outputVariable="decision">
1742     <b4p:localTask reference="tns:approveEmployeeOfTheMonth">
1743       <htd:peopleAssignments>
1744         <htd:excludedOwners>
1745           <htd:from>
1746             b4p:getActualOwner("tns:firstApproval")
1747           </htd:from>
1748         </htd:excludedOwners>
1749       </htd:peopleAssignments>
1750     </b4p:localTask>
1751   </b4p:peopleActivity>
1752 </extensionActivity>
1753
1754 <!-- Constellation 3 -->
1755 <extensionActivity>
1756   <b4p:peopleActivity name="notifyEmployees"
1757     inputVariable="electionResult">
1758     <b4p:localNotification reference="ta:employeeBroadcast"/>
1759     <!-- notification is not defined as part of this document,
1760         but within a separate one
1761     -->
1762   </b4p:peopleActivity>
1763 </extensionActivity>
1764
1765 <!-- Constellation 4 -->
1766 <extensionActivity>
1767   <b4p:peopleActivity name="prepareInauguralSpeech"
1768     inputVariable="electionResult"
1769     outputVariable="speech"
1770     isSkipable="no">
1771     <b4p:remoteTask partnerLink="author"
1772       operation="prepareSpeech"
1773       responseOperation="receiveSpeech">
1774       <htd:priority>0</htd:priority> <!-- assign highest priority -->
1775     <htd:peopleAssignments>
1776       <htd:potentialOwners>
1777         <htd:from>$electionResult/winner</htd:from>
1778       </htd:potentialOwners>
1779     </htd:peopleAssignments>

```

```
1780     </b4p:remoteTask>
1781     </b4p:peopleActivity>
1782     </extensionActivity>
1783
1784 </sequence>
1785
1786 </process>
```

## 1787 D.2 WSDL Definitions

```
1788 <?xml version="1.0" encoding="UTF-8"?>
1789 <!--
1790 Copyright (c) OASIS Open 2009. All Rights Reserved.
1791 -->
1792 <wsdl:definitions
1793   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1794   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1795   xmlns:tns="http://www.example.com/approval"
1796   targetNamespace="http://www.example.com/approval">
1797
1798   <wsdl:documentation>
1799     Example for BPEL4People 1.1 - PeopleActivity Interface Definition
1800   </wsdl:documentation>
1801
1802   <!-- Messages -->
1803   <wsdl:message name="approvalInput">
1804     <wsdl:part name="parameters" type="xsd:string" />
1805   </wsdl:message>
1806   <wsdl:message name="approvalOutput">
1807     <wsdl:part name="parameters" type="xsd:string" />
1808   </wsdl:message>
1809
1810   <!-- Port Type -->
1811   <wsdl:portType name="approvalPT">
1812     <wsdl:operation name="approve">
1813       <wsdl:input message="tns:approvalInput" />
1814       <wsdl:output message="tns:approvalOutput" />
1815     </wsdl:operation>
1816   </wsdl:portType>
1817
1818 </wsdl:definitions>
```

```
1819
1820 <?xml version="1.0" encoding="UTF-8"?>
1821 <!--
1822 Copyright (c) OASIS Open 2009. All Rights Reserved.
1823 -->
1824 <wsdl:definitions
1825   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1826   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1827   xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
1828   xmlns:tns="http://www.example.com/election"
1829   targetNamespace="http://www.example.com/election">
1830
1831   <wsdl:documentation>
1832     Example for BPEL4People 1.1 - PeopleActivity Interface Definition
1833   </wsdl:documentation>
1834
```

```
1835 <!-- WS-BPEL 2.0 Partner Link Type -->
1836 <plnk:partnerLinkType name="electionPLT">
1837   <plnk:role name="electionService" portType="tns:electionPT" />
1838 </plnk:partnerLinkType>
1839
1840 <!-- Messages -->
1841 <wsdl:message name="electionInput">
1842   <wsdl:part name="parameters" type="xsd:string" />
1843 </wsdl:message>
1844 <wsdl:message name="votingInput">
1845   <wsdl:part name="parameters" type="xsd:string" />
1846 </wsdl:message>
1847
1848 <!-- Port Types -->
1849 <wsdl:portType name="electionPT">
1850   <wsdl:operation name="elect">
1851     <wsdl:input message="tns:electionInput" />
1852   </wsdl:operation>
1853 </wsdl:portType>
1854 <wsdl:portType name="votingPT">
1855   <wsdl:operation name="vote">
1856     <wsdl:input message="tns:votingInput" />
1857   </wsdl:operation>
1858 </wsdl:portType>
1859
1860 </wsdl:definitions>
```

1861

---

## E. Acknowledgements

1862 The following individuals have participated in the creation of this specification and are gratefully  
1863 acknowledged:

1864

1865 **Members of the BPEL4People Technical Committee:**

1866 Phillip Allen, Microsoft Corporation

1867 Ashish Agrawal, Adobe Systems

1868 Mike Amend, BEA Systems, Inc.

1869 Stefan Baeuerle, SAP AG

1870 Charlton Barreto, Adobe Systems

1871 Justin Brunt, TIBCO Software Inc.

1872 Martin Chapman, Oracle Corporation

1873 Luc Clément, Active Endpoints, Inc.

1874 Manoj Das, Oracle Corporation

1875 Alireza Farhoush, TIBCO Software Inc.

1876 Mark Ford, Active Endpoints, Inc.

1877 Sabine Holz, SAP AG

1878 Dave Ings, IBM

1879 Gershon Janssen, Individual

1880 Diane Jordan, IBM

1881 Anish Karmarkar, Oracle Corporation

1882 Ulrich Keil, SAP AG

1883 Oliver Kieselbach, SAP AG

1884 Matthias Kloppmann, IBM

1885 Dieter König, IBM

1886 Marita Kruempelmann, SAP AG

1887 Frank Leymann, IBM

1888 Mark Little, Red Hat

1889 Alexander Malek, Microsoft Corporation

1890 Ashok Malhotra, Oracle Corporation

1891 Mike Marin, IBM

1892 Vinkesh Mehta, Deloitte Consulting LLP

1893 Jeff Mischkinsky, Oracle Corporation

1894 Ralf Mueller, Oracle Corporation

1895 Krasimir Nedkov, SAP AG

1896 Benjamin Notheis, SAP AG

1897 Michael Pellegrini, Active Endpoints, Inc.

1898 Hannah Petereit, SAP AG

1899 Gerhard Pfau, IBM

1900 Karsten Ploesser, SAP AG

1901 Ravi Rangaswamy, Oracle Corporation  
1902 Alan Rickayzen, SAP AG  
1903 Michael Rowley, BEA Systems, Inc.  
1904 Ron Ten-Hove, Sun Microsystems  
1905 Ivana Trickovic, SAP AG  
1906 Alessandro Triglia, OSS Nokalva  
1907 Claus von Riegen, SAP AG  
1908 Peter Walker, Sun Microsystems  
1909 Franz Weber, SAP AG  
1910 Prasad Yendluri, Software AG, Inc.

1911

1912 **BPEL4People 1.0 Specification Contributors:**

1913 Ashish Agrawal, Adobe  
1914 Mike Amend, BEA  
1915 Manoj Das, Oracle  
1916 Mark Ford, Active Endpoints  
1917 Chris Keller, Active Endpoints  
1918 Matthias Kloppmann, IBM  
1919 Dieter König, IBM  
1920 Frank Leymann, IBM  
1921 Ralf Müller, Oracle  
1922 Gerhard Pfau, IBM  
1923 Karsten Plösser, SAP  
1924 Ravi Rangaswamy, Oracle  
1925 Alan Rickayzen, SAP  
1926 Michael Rowley, BEA  
1927 Patrick Schmidt, SAP  
1928 Ivana Trickovic, SAP  
1929 Alex Yiu, Oracle  
1930 Matthias Zeller, Adobe

1931

1932 In addition, the following individuals have provided valuable input into the design of this specification:  
1933 Dave Ings, Diane Jordan, Mohan Kamath, Ulrich Keil, Matthias Kruse, Kurt Lind, Jeff Mischkinsky, Bhagat  
1934 Nainani, Michael Pellegrini, Lars Rueter, Frank Ryan, David Shaffer, Will Stallard, Cyrille Waguët, Franz  
1935 Weber, and Eric Wittmann.

---

## **F. ~~Non-Normative Text~~**



1937

## G.F. Revision History

1938

[optional; should not be included in OASIS Standards]

1939

Revision	Date	Editor	Changes Made
WD-01	2008-03-12	Dieter König	First working draft created from submitted specification
WD-02	2008-03-13	Dieter König	Added specification editors Moved WSDL and XSD into separate artifacts
WD-02	2008-06-25	Ivana Trickovic	Resolution of Issue #8 incorporated into the document/section 5
WD-02	2008-06-28	Dieter König	Resolution of Issue #13 applied to complete document and all separate XML artifacts
WD-02	2008-06-28	Dieter König	Resolution of Issue #21 applied to section 2 Resolution of Issue #22 applied to sections 2.4.1 and 3.1.1
WD-02	2008-07-06	Vinkesh Mehta	Resolution for Issue #3 applied to sections 2.4.1 (~line 353)
WD-02	2008-07-25	Krasimir Nedkov	Resolution for Issue #18 applied to sections 4.6.2 and 5; Typos correction.
WD-02	2008-07-29	Ralf Mueller	Resolution for Issue #11 applied to section 3.1.2
WD-02	2008-07-29	Luc Clément	Resolution for Issue #10 applied to first paragraph of section 3.3
CD-01-rev-1	2008-10-02	Ralf Mueller	Resolution for Issue #17 and #24 applied to section 2 and 5
CD-01-rev-2	2008-10-07	Michael Rowley	Resolution for Issue #2 applied in section 4.7, and for issue #19 in sections 4.3.1 and 4.4.1.
CD-01-rev-3	2008-10-20	Dieter König	Resolution for Issue #23 applied to section 3.2.1 Resolution of Issue #6 applied to section 5
CD-01-rev-3	2008-10-20	Vinkesh Mehta	Resolution of issue-12, section 3.2.2, 4.2 font changed to italics for htd:genericHumanRole. Also modified XML artifacts for boel4people.xsd, humantask.xsd, humantask-context.xsd

Revision	Date	Editor	Changes Made
CD-01-rev-3	2008-12-03	Ralf Mueller	Resolution for Issue #16 applied to sections 1 – 6
CD-01-rev-3	2008-12-12	Ravi Rangaswamy	Resolution for Issue #16 applied to sections 7 and appendix B
CD-01-rev-3	2008-12-18	Ravi Rangaswamy	Resolution for Issue #16: Undid changes to appendix B
CD-01-rev-4	2008-12-19	Ralf Mueller	Incorporated review comments from Ivana and Luc for Issue BP-16
CD-02	2009-01-18	Luc Clément	Committee Draft 2
CD-02-rev-1	2009-02-20	Dieter König	Issue 47: added getState() in section 5 Issue 48: abstract BPEL ns in 7.1.1 Issue 50, sections 3 and 5 (htd:→htt:)
CD-02-rev-2	2009-03-11	Ralf Mueller	Issue 76: Changes for RFC2119
CD-03	2009-04-15	Luc Clément	Committee Draft 3
CD-03-rev-2	2009-04-29	Luc Clément	Issue 72: add WS-HumanTask and WS-HumanTask Processor definitions to section 2.3
CD-03-rev3	2009-06-01	Luc Clément	Issue 65
CD-03-rev4	2009-06-02	Michael Rowley	Issue 38, 39
CD-04-rev0	2009-06-17	Luc Clément	Committee Draft 4
CD-04-rev1	2009-06-17	Luc Clément	Acknowledgement update
CD-04-rev2	2009-06-26	Dieter König	Formatting
CD-05-rev0	2009-07-15	Luc Clément	Committee Draft 5
CD-05-rev1	2009-08-08	Luc Clément	Editors update
CD-05-rev2	2009-09-28	Dieter König	Issue 125
CD-05-rev3	2009-10-22	Dieter König	Issue 129 XML artifacts copied back to appendix
CD-05-rev4	2009-11-01	Luc Clément	Issue 131 OASIS Spec QA Checklist updates
CD-06-rev0	2009-11-01	Luc Clément	Committee Draft 6
CD-07	2010-03-03	Luc Clément	Copyright date updates, creation of CD07, and cover page annotation as Public Review 02
<a href="#">CD-08</a>	<a href="#">2010-04-14</a>	<a href="#">Luc Clément</a>	<a href="#">CD08</a>
<a href="#">CD-09</a>	<a href="#">2010-04-26</a>	<a href="#">Luc Clément</a>	<a href="#">CD09 / PRD 03</a>