



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

Committee Draft 06 / Public Review Draft 01

04 November 2009

## Specification URIs:

### This Version:

<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-06.html>  
<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-06.doc> (Authoritative format)  
<http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-06.pdf>

### Previous Version:

N/A

### 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® 2009. 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 names "OASIS", are trademarks 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.	Non-Normative Text .....	56
G.	Revision History.....	57

---

# 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://ifr.sap.com/bpel4ws/>

### [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>

### [WS-Addr-Core]

Web Services Addressing 1.0 - Core, W3C Recommendation, May 2006, available via <http://www.w3.org/TR/ws-addr-core>

- 46 **[WS-Addr-SOAP]**  
47 [Web Services Addressing 1.0 – SOAP Binding](http://www.w3.org/TR/ws-addr-soap), W3C Recommendation, May 2006, available via  
48 <http://www.w3.org/TR/ws-addr-soap>
- 49 **[WS-Addr-WSDL]**  
50 [Web Services Addressing 1.0 – WSDL Binding](http://www.w3.org/TR/ws-addr-wsdl), W3C Working Draft, February 2006, available via  
51 <http://www.w3.org/TR/ws-addr-wsdl>
- 52 **[WS-BPEL 2.0]**  
53 OASIS Standard, “Web Service Business Process Execution Language Version 2.0”, 11 April  
54 2007, <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- 55 **[WSDL 1.1]**  
56 Web Services Description Language (WSDL) Version 1.1, W3C Note, available via  
57 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>
- 58 **[WS-HumanTask]**  
59 OASIS Committee Draft, “Web Services – Human Task (WS-HumanTask) Specification Version  
60 1.1, CD-06”, 04 November 2009, [http://docs.oasis-open.org/bpel4people/ws-humantask-1.1-](http://docs.oasis-open.org/bpel4people/ws-humantask-1.1-spec-cd-06.doc)  
61 [spec-cd-06.doc](http://docs.oasis-open.org/bpel4people/ws-humantask-1.1-spec-cd-06.doc)
- 62 **[XML Infoset]**  
63 [XML Information Set](http://www.w3.org/TR/2001/REC-xml-infoset-20011024/), W3C Recommendation, available via [infoset-20011024/](http://www.w3.org/TR/2001/REC-xml-</a><br/>64 <a href=)
- 65 **[XML Namespaces]**  
66 Namespaces in XML 1.0 (Second Edition), W3C Recommendation, available via  
67 <http://www.w3.org/TR/REC-xml-names/>
- 68 **[XML Schema Part 1]**  
69 XML Schema Part 1: Structures, W3C Recommendation, October 2004, available via  
70 <http://www.w3.org/TR/xmlschema-1/>
- 71 **[XML Schema Part 2]**  
72 XML Schema Part 2: Datatypes, W3C Recommendation, October 2004, available via  
73 <http://www.w3.org/TR/xmlschema-2/>
- 74 **[XMLSpec]**  
75 XML Specification, W3C Recommendation, February 1998, available via  
76 <http://www.w3.org/TR/1998/REC-xml-19980210>
- 77 **[XPath 1.0]**  
78 XML Path Language (XPath) Version 1.0, W3C Recommendation, November 1999, available via  
79 <http://www.w3.org/TR/1999/REC-xpath-19991116>

## 80 **1.3 Non-Normative References**

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

## 82 **1.4 Conformance Targets**

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

- 84 • BPEL4People Definition  
85 A BPEL4People Definition is a WS-BPEL 2.0 process definition that uses the BPEL4People  
86 extensions to WS-BPEL 2.0 specified in this document.
- 87 • BPEL4People Processor  
88 A BPEL4People Processor is any implementation that accepts a BPEL4People definition and  
89 executes the semantics defined in this document.

90

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



---

## 97 2 Language Design

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

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

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

### 107 2.1 Dependencies on Other Specifications

108 BPEL4People utilizes the following specifications:

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

#### 116 2.1.1 Namespaces Referenced

117 BPEL4People references these namespaces:

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

### 125 2.2 Language Extensibility

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

128 Attributes from other namespaces to appear on any BPEL4People element

129 Elements from other namespaces to appear within BPEL4People elements

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

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

### 134 2.3 Overall Language Structure

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

### 137 2.3.1 Syntax

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

```
140 <bpel:process b4p:shareComments="xsd:boolean"? ...  
141 ...  
142 xmlns:b4p="http://docs.oasis-open.org/ns/bpel4people/bpel4people/200803"  
143 xmlns:htd="http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803">  
144 ...  
145 <bpel:extensions>  
146 <bpel:extension  
147 namespace="http://docs.oasis-  
148 open.org/ns/bpel4people/bpel4people/200803"  
149 mustUnderstand="yes"/>  
150 <bpel:extension  
151 namespace="http://docs.oasis-open.org/ns/bpel4people/ws-  
152 humantask/200803"  
153 mustUnderstand="yes"/>  
154 </bpel:extensions>  
155  
156 <bpel:import  
157 importType="http://docs.oasis-open.org/ns/bpel4people/ws-  
158 humantask/200803" .../>  
159  
160 ...  
161 <b4p:humanInteractions>?  
162  
163 <htd:logicalPeopleGroups/>?  
164 <htd:logicalPeopleGroup name="NCName" reference="QName"?>+  
165 ...  
166 </htd:logicalPeopleGroup>  
167 </htd:logicalPeopleGroups>  
168  
169 <htd:tasks>?  
170 <htd:task name="NCName">+  
171 ...  
172 </htd:task>  
173 </htd:tasks>  
174  
175 <htd:notifications>?  
176 <htd:notification name="NCName">+  
177 ...  
178 </htd:notification>  
179 </htd:notifications>  
180  
181 </b4p:humanInteractions>  
182  
183 <b4p:peopleAssignments>?  
184 ...  
185 </b4p:peopleAssignments>  
186  
187 ...  
188 <bpel:extensionActivity>  
189 <b4p:peopleActivity name="NCName" ...>  
190 ...  
191 </b4p:peopleActivity>  
192 </bpel:extensionActivity>  
193 ...
```

194 `</bpel:process>`

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

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

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

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

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

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

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

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

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

229

```

230
231 <bpel:scope b4p:shareComments="xsd:boolean"? ...>
232   ...
233   <b4p:humanInteractions>?
234     ...
235   </b4p:humanInteractions>
236   ...
237   <bpel:extensionActivity>
238     <b4p:peopleActivity name="NCName" dontShareComments="xsd:boolean" ...>
239       ...
240     </b4p:peopleActivity>
241   </bpel:extensionActivity>
242   ...
243 </bpel:scope>

```

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

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

```

249 <b4p:documentation xml:lang="xsd:language">
250   ...
251 </b4p:documentation>

```

## 252 2.4 Default use of XPath 1.0 as an Expression Language

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

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

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

268

## 3 Concepts

269  
270

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

271

### 3.1 Generic Human Roles

272  
273  
274

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

275  
276  
277

- Process initiator
- Process stakeholders
- Business administrators

278  
279  
280  
281  
282

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

283  
284  
285  
286  
287  
288  
289

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

290  
291  
292  
293  
294  
295

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

296

#### 3.1.1 Syntax

297  
298  
299  
300  
301  
302  
303

```
<b4p:peopleAssignments>?  
  
  <htd:genericHumanRole>+  
    <htd:from>...</htd:from>  
  </htd:genericHumanRole>  
  
</b4p:peopleAssignments>
```

304  
305

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

306  
307  
308  
309  
310  
311  
312  
313  
314  
315

```
<b4p:peopleAssignments>?  
  
  <b4p:processInitiator>?  
    <htd:from ...>...</htd:from>  
  </b4p:processInitiator>  
  
  <b4p:processStakeholders>?  
    <htd:from ...>...</htd:from>  
  </b4p:processStakeholders>
```

```
316 <b4p:businessAdministrators>?  
317   <htd:from ...>...</htd:from>  
318 </b4p:businessAdministrators>  
319  
320 </b4p:peopleAssignments>
```

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

### 323 3.1.2 Initialization Behavior

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

## 329 3.2 Assigning People

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

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

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

### 338 3.2.1 Using Logical People Groups

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

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

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

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

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

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

361  
362

## 363 Assignment of Logical People Groups

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

368 BPEL4People extends the from-spec and to-spec forms introduced in [WS-BPEL 2.0] as shown below:

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

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

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

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

- 385 • To copy to a variable

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

391

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

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

394

- 395 • To copy to a property

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

397

- 398 • To copy to a logical people group

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

400

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

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

- 406 • To copy from a variable

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

412

- 413 • To copy from a property

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

415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460

- 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" />
```

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

```
<bpel:assign name="getVoters">  
  <bpel:copy>  
    <bpel:from b4p:logicalPeopleGroup="voters">  
      <b4p:argument name="region">  
        $selectionRequest/region  
      </b4p:argument>  
    </bpel:from>  
    <bpel:to variable="voters" />  
  </bpel:copy>  
</bpel:assign>
```

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

```
<bpel:assign>  
  <bpel:copy>  
    <bpel:from>  
      <bpel:literal>  
        <htt:tOrganizationalEntity>  
          <htt:user>Alan</htt:user>  
          <htt:user>Dieter</htt:user>  
          <htt:user>Frank</htt:user>  
          <htt:user>Gerhard</htt:user>  
          <htt:user>Ivana</htt:user>  
          <htt:user>Karsten</htt:user>  
          <htt:user>Matthias</htt:user>  
          <htt:user>Patrick</htt:user>  
        </htt:tOrganizationalEntity>  
      </bpel:literal>  
    </bpel:from>  
    <bpel:to b4p:logicalPeopleGroup="bpel4peopleAuthors" />  
  </bpel:copy>  
</bpel:assign>
```



461

462 The third example shows assigning the results of one logical people group to another logical people  
463 group.

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

### 470 3.2.2 Computed Assignment

471 All computed assignment variants described in [WS-HumanTask] (see section 3.2 “Assigning People” for  
472 more details) are supported. In addition, the following variant is possible:

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

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

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

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

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

### 487 3.3 Ad-hoc Attachments

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

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

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

495

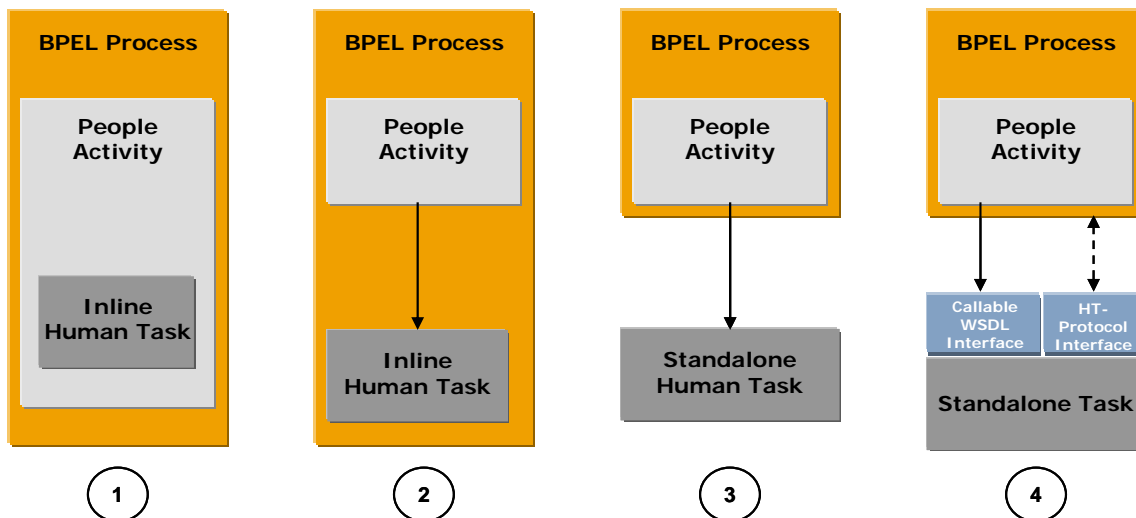
## 4 People Activity

496

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.

498

499



500

501

Figure 1: Constellations

502

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

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

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

## 526 4.1 Overall Syntax

527 Definition of people activity:

```
528 <bpel:extensionActivity>
529
530   <b4p:peopleActivity name="NCName" inputVariable="NCName"?
531     outputVariable="NCName"? isSkipable="xsd:boolean"?
532     dontShareComments="xsd:boolean"?
533     standard-attributes>
534
535     standard-elements
536
537     ( <htd:task>...</htd:task>
538       | <b4p:localTask>...</b4p:localTask>
539       | <b4p:remoteTask>...</b4p:remoteTask>
540       | <htd:notification>...</htd:notification>
541       | <b4p:localNotification>...</b4p:localNotification>
542       | <b4p:remoteNotification>...</b4p:remoteNotification>
543     )
544
545     <b4p:scheduledActions>? ...</b4p:scheduledActions>
546
547     <bpel:toParts>?
548       <bpel:toPart part="NCName" fromVariable="BPELVariableName" />+
549     </bpel:toParts>
550
551     <bpel:fromParts>?
552       <bpel:fromPart part="NCName" toVariable="BPELVariableName" />+
553     </bpel:fromParts>
554
555     <b4p:attachmentPropagation fromProcess="all|none"
556       toProcess="all|newOnly|none" />?
557
558   </b4p:peopleActivity>
559 </bpel:extensionActivity>
```

### 561 4.1.1 Properties

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

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

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

## 618 4.2 Standard Overriding Elements

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

625 The following elements can be overridden:

- 626 • people assignments

- 627 • priority

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

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

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

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

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

## 651 4.3 People Activities Using Local Human Tasks

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

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

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

### 661 4.3.1 Syntax

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

673

### 674 Properties

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

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

### 685 4.3.2 Examples

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

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

707

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

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

## 716 4.4 People Activities Using Local Notifications

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

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

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

## 726 4.4.1 Syntax

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

737

### 738 Properties

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

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

## 750 4.4.2 Examples

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

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

## 761 4.5 People Activities Using Remote Human Tasks

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

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

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

## 773 4.5.1 Syntax

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

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

## 789 4.5.2 Example

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

## 807 4.5.3 Passing Endpoint References for Callbacks

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

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

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



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

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

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

## 838 4.6 People Activities Using Remote Notifications

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

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

### 847 4.6.1 Syntax

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

### 855 4.6.2 Example

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

## 870 4.7 Elements for Scheduled Actions

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

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

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

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

### 889 Syntax:

```
890 <b4p:scheduledActions>?  
891  
892   <b4p:deferActivation>?  
893     ( <b4p:for expressionLanguage="anyURI"?>  
894       duration-expression  
895     </b4p:for>  
896     | <b4p:until expressionLanguage="anyURI"?>  
897       deadline-expression  
898     </b4p:until>  
899   )  
900 </b4p:deferActivation>  
901  
902   <b4p:expiration>?  
903     ( <b4p:for expressionLanguage="anyURI"?>  
904       duration-expression  
905     </b4p:for>  
906     | <b4p:until expressionLanguage="anyURI"?>  
907       deadline-expression  
908     </b4p:until>  
909   )  
910 </b4p:expiration>  
911 </b4p:scheduledActions>
```

### 914 Properties

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

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

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

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

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

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

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

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

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

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

#### 940 **Example:**

```
941 <b4p:scheduledActions>  
942  
943   <b4p:deferActivation>  
944     <b4p:documentation xml:lang="en-US">  
945       Activation of this task is deferred until the time specified  
946       in its input data.  
947     </b4p:documentation>  
948     <b4p:until>htd:getInput()/activateAt</b4p:until>  
949   </b4p:deferActivation>  
950  
951   <b4p:expiration>  
952     <b4p:documentation xml:lang="en-US">  
953       This task expires when not completed within 14 days after  
954       having been activated.  
955     </b4p:documentation>  
956     <b4p:for>P14D</b4p:for>  
957   </b4p:expiration>  
958  
959 </b4p:scheduledActions>
```

## 960 **4.8 People Activity Behavior and State Transitions**

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

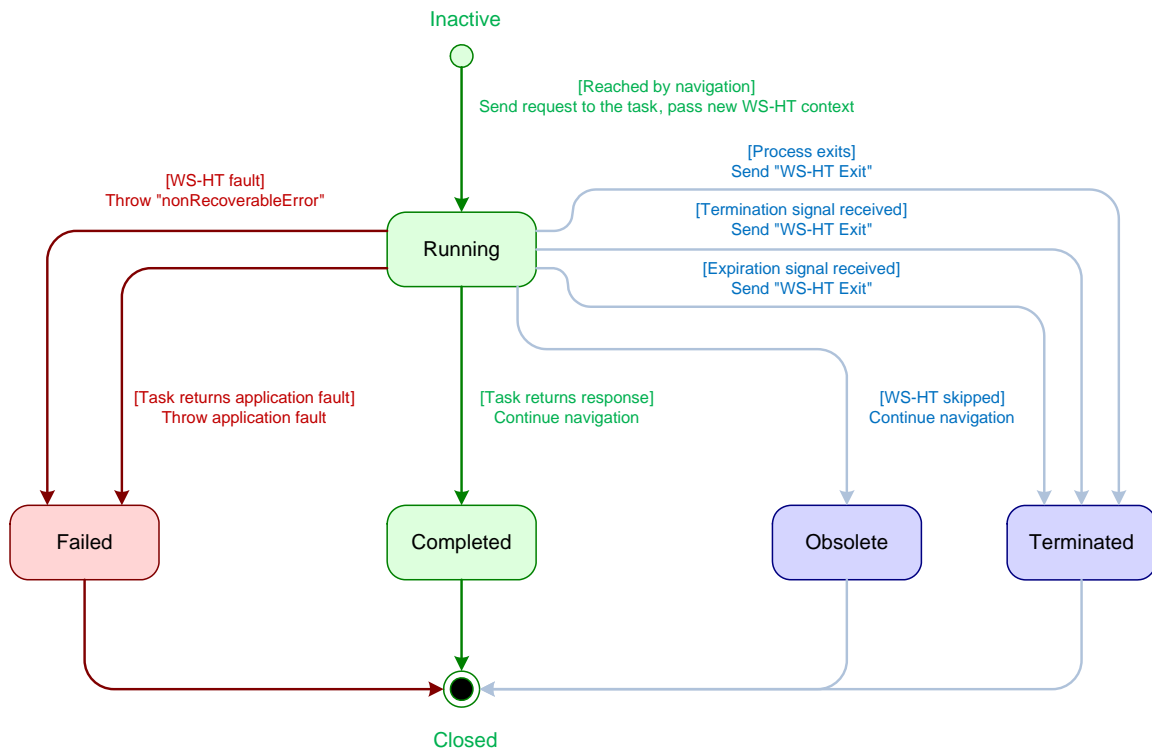


Figure 2: State diagram of the people activity

963

964

965

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

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

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

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

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

## 980 4.9 Task Instance Data

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

### 984 4.9.1 Presentation Data

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

## 987 **4.9.2 Context Data**

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

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

## 994 **4.9.3 Operational Data**

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

997

## 5 XPath Extension Functions

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

1003

Operation Name	Description	Parameters
getProcessStakeholders	Returns the stakeholders of the process. It MUST return an empty <code>htt:organization alEntity</code> in case of an error.	Out <ul style="list-style-type: none"> <li>organizational entity (<code>htt:organization alEntity</code>)</li> </ul>
getBusinessAdministrators	Returns the business administrators of the process. It MUST return an empty <code>htt:organization alEntity</code> in case of an error.	Out <ul style="list-style-type: none"> <li>organizational entity (<code>htt:organization alEntity</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:organization alEntity</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:organization alEntity</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>

1004

1005 XPath functions accessing data of a human task only guarantee to return data once the corresponding  
 1006 task has reached a final state.



1007  
1008  
1009  
1010  
1011  
1012

## 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.

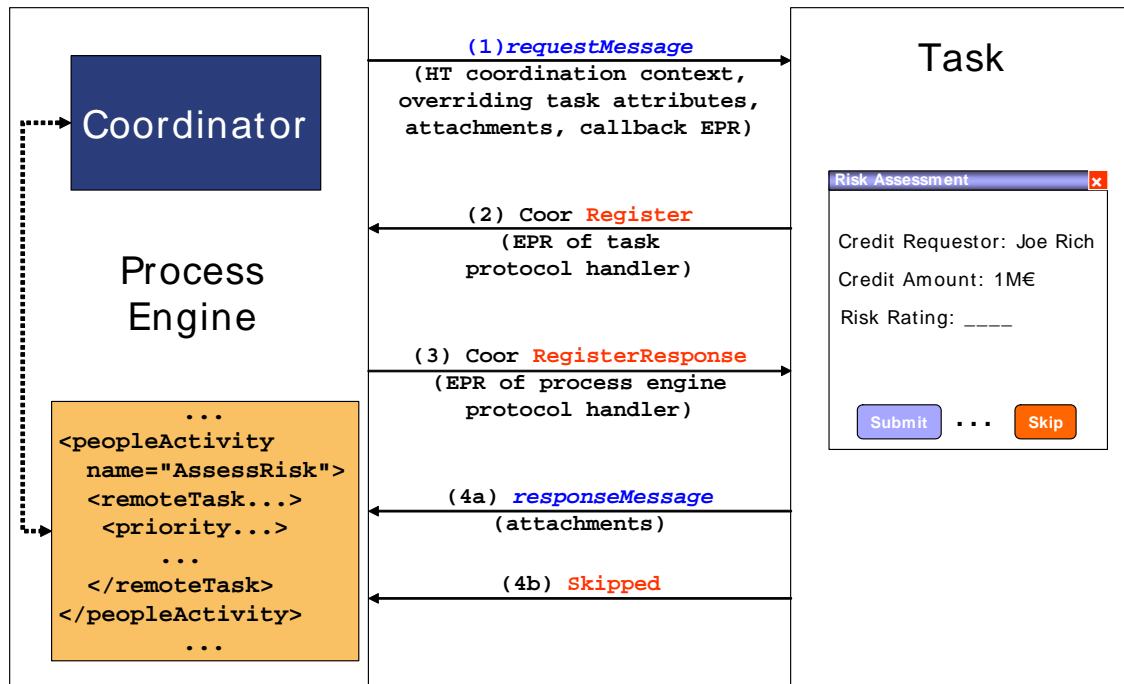


Figure 3: *Message exchange between a people activity and a human task*

1013

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].

- 1029 2. When a response message is received from the task that indicates the successful completion of  
 1030 the task, the people activity completes. This response MAY include all new ad-hoc attachments  
 1031 from the human task.
- 1032 3. When a response message is received from the task that indicates a fault of the task, the people  
 1033 activity faults. The fault MUST be thrown in the scope of the people activity.
- 1034 4. When protocol message fault is received, the fault nonRecoverableError MUST be thrown in the  
 1035 scope enclosing the people activity.
- 1036 5. When protocol message skipped is received, the people activity MUST move to state *Obsolete*.
- 1037 6. If the task does not reach one of the final states by the expiration deadline, the people activity  
 1038 MUST be terminated. Protocol message exit is sent to the task.
- 1039 7. When the people activity is terminated, protocol message exit MUST be sent to the task.
- 1040 8. When the process encounters an <exit> activity, protocol message exit MUST be sent to the task.  
 1041

1042 The following table summarizes this behavior, the protocol messages sent, and their direction, i.e.,  
 1043 whether a message is sent from the people activity to the task (“out” in the column titled Direction) or vice  
 1044 versa (“in”).  
 1045

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

---

## 1046 7 BPEL Abstract Processes

1047 BPEL abstract processes are indicated by the namespace "`http://docs.oasis-`  
1048 `open.org/wsbpel/2.0/process/abstract`". All constructs defined in BPEL4People extension  
1049 namespaces MAY appear in abstract processes.

### 1050 7.1 Hiding Syntactic Elements

1051 Opaque tokens defined in BPEL (activities, expressions, attributes and from-specs) MAY be used in  
1052 BPEL4People extension constructs. The syntactic validity constraints of BPEL MUST apply in the same  
1053 way to an Executable Completion of an abstract process containing BPEL4People extensions.

#### 1054 7.1.1 Opaque Activities

1055 BPEL4people does not change the way opaque activities can be replaced by an executable activity in an  
1056 executable completion of an abstract process, that is, an `<abstract:opaqueActivity>` MAY also  
1057 serve as a placeholder for a `<bpel:extensionActivity>` containing a `<b4p:peopleActivity>`.

#### 1058 7.1.2 Opaque Expressions

1059 Any expression introduced by BPEL4People MAY be made opaque. In particular, the following  
1060 expressions MAY have the `opaque="yes"` attribute:

```
1061 <htd:argument name="NCName" expressionLanguage="anyURI"? opaque="yes" />  
1062 <htd:priority expressionLanguage="anyURI" opaque="yes" />  
1063 <b4p:for expressionLanguage="anyURI"? opaque="yes" />  
1064 <b4p:until expressionLanguage="anyURI"? opaque="yes" />
```

#### 1065 7.1.3 Opaque Attributes

1066 Any attribute introduced by BPEL4People MAY have an opaque value "`##opaque`" in an abstract  
1067 process.

#### 1068 7.1.4 Opaque From-Spec

1069 In BPEL, any from-spec in an executable process can be replaced by an opaque from-spec  
1070 `<opaqueFrom/>` in an abstract process. This already includes any BPEL from-spec extended with the  
1071 BPEL4People `b4p:logicalPeopleGroup="NCName"` attribute. In addition, the extension from-spec  
1072 `<htd:from>` MAY also be replaced by an opaque from-spec in an abstract process.

#### 1073 7.1.5 Omission

1074 In BPEL, omissible tokens are all attributes, activities, expressions and from-specs which are both (1)  
1075 syntactically required by the Executable BPEL XML Schema, and (2) have no default value. This rule also  
1076 applies to BPEL4People extensions in abstract processes. For example, `<b4p:localTask`  
1077 `reference="##opaque">` is equivalent to `<b4p:localTask>`.

## 1078 7.2 Abstract Process Profile for Observable Behavior

1079 The Abstract Process Profile for Observable Behavior, indicated by the process attribute  
1080 `abstractProcessProfile="http://docs.oasis-`  
1081 `open.org/wsbpel/2.0/process/abstract/ap11/2006/08"`, provides a means to create precise  
1082 and predictable descriptions of observable behavior of the service(s) provided by an executable process.

1083 The main application of this profile is the definition of business process contracts; that is, the behavior  
1084 followed by one business partner in the context of Web services exchanges. A valid completion has to  
1085 follow the same interactions as the abstract process, with the partners that are specified by the abstract  
1086 process. The executable process can, however, perform additional interaction steps relating to other  
1087 partners. Likewise, the executable process can perform additional human interactions. Beyond the  
1088 restrictions defined in WS-BPEL 2.0, the use of opacity is not restricted in any way for elements and  
1089 attributes introduced by BPEL4People.

### 1090 **7.3 Abstract Process Profile for Templates**

1091 The Abstract Process Profile for Templates, indicated by the process attribute  
1092 `abstractProcessProfile="http://docs.oasis-`  
1093 `open.org/wsbpel/2.0/process/abstract/simple-template/2006/08"`, allows the definition  
1094 of Abstract Processes which hide almost any arbitrary execution details and have explicit opaque  
1095 extension points for adding behavior.

1096 This profile does not allow the use of omission shortcuts but the use of opacity is not restricted in any  
1097 way. For abstract processes belonging to this profile, this rule is extended to the elements and attributes  
1098 introduced by BPEL4People.

---

## 1099 **8 Conformance**

1100 The XML schema pointed to by the RDDDL document at the namespace URI, defined by this specification,  
1101 are considered to be authoritative and take precedence over the XML schema defined in the appendix of  
1102 this document.

1103

1104 There are four conformance targets defined as part of this specification: a BPEL4People Definition, a  
1105 BPEL4People Processor, a WS-HumanTask Definition and a WS-HumanTask Processor (see section  
1106 2.3). In order to claim conformance with BPEL4People 1.1, the conformance targets **MUST** comply with  
1107 all normative statements in the BPEL4People and the WS-HumanTask specification, notably all **MUST**  
1108 statements have to be implemented.

---

1109 **A. Standard Faults**

1110 The following list specifies the standard faults defined within the BPEL4People specification. All standard  
1111 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.

1112

---

## B. Portability and Interoperability Considerations

1113 The following section illustrates the portability and interoperability aspects of the various usage  
1114 constellations of BPEL4People with WS-HumanTask as described in Figure 1:

1115

1116 Portability - The ability to take design-time artifacts created in one vendor's environment and use them in  
1117 another vendor's environment. Constellations one and two provide portability of BPEL4People processes  
1118 with embedded human interactions in. Constellations three and four provide portability of BPEL4People  
1119 processes with referenced human interactions.

1120

1121 Interoperability - The capability for multiple components (process engine, task engine and task list client)  
1122 to interact using well-defined messages and protocols. This enables to combine components from  
1123 different vendors allowing seamless execution.

1124 Constellation four achieves interoperability between process and tasks from different vendor  
1125 implementations.

1126

1127 Constellation 1

1128 Task definitions are defined inline of the people activities. Usage in this manner is typically for self-  
1129 contained people activities, whose tasks definitions are not intended to be reused elsewhere in the  
1130 process or across multiple processes. This format will also provide scoping of the task definition since it  
1131 will not be visible or accessible outside the people activity in which it is contained. Portability for this  
1132 constellation requires support of both WS-HumanTask and BPEL4People artifacts using the inline task  
1133 definition format. Since the process and task interactions are combined in one component, interoperability  
1134 requirements are limited to those between the task list client and the infrastructure.

1135

1136 Constellation 2

1137 Similar to constellation 1, but tasks are defined at the process level. This allows task definitions to be  
1138 referenced from within people activities enabling task reuse. Portability for this constellation requires  
1139 support of both WS-HumanTask and BPEL4People artifacts using the process level scoped task  
1140 definition format. Since the process and task interactions are combined in one component, interoperability  
1141 requirements are limited to those between the task list client and the infrastructure.

1142

1143 Constellation 3

1144 In this constellation, the task and people activity definitions are defined as separate artifacts and execute  
1145 in different infrastructure components but provided by the same vendor. Portability for this constellation  
1146 requires support of both WS-HumanTask and BPEL4People as separate artifacts. Since the process and  
1147 task components are implemented by the same vendor, interoperability requirements are limited to those  
1148 between the task list client and the infrastructure.

1149

1150 Constellation 4

1151 Identical to constellation 3 in terms of the task and people activity definitions, but in this case the process  
1152 and task infrastructure are provided by different vendors. Portability for this constellation requires support  
1153 of both WS-HumanTask and BPEL4People as separate artifacts. Interoperability between task and  
1154 process infrastructures from different vendors is achieved using the WS-HumanTask coordination  
1155 protocol.

1156

## C. BPEL4People Schema

```

1157 <?xml version="1.0" encoding="UTF-8"?>
1158 <!--
1159   Copyright (c) OASIS Open 2009. All Rights Reserved.
1160 -->
1161 <xsd:schema
1162   targetNamespace="http://docs.oasis-
1163 open.org/ns/bpel4people/bpel4people/200803"
1164   xmlns="http://docs.oasis-open.org/ns/bpel4people/bpel4people/200803"
1165   xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
1166   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1167   xmlns:htd="http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803"
1168   elementFormDefault="qualified"
1169   blockDefault="#all">
1170
1171   <xsd:annotation>
1172     <xsd:documentation>
1173       XML Schema for BPEL4People 1.1 - WS-BPEL 2.0 Extension for Human Task
1174 Interactions
1175     </xsd:documentation>
1176   </xsd:annotation>
1177
1178   <!-- other namespaces -->
1179   <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
1180     schemaLocation="http://www.w3.org/2001/xml.xsd" />
1181   <xsd:import namespace="http://docs.oasis-open.org/ns/bpel4people/ws-
1182 humantask/200803"
1183     schemaLocation="ws-humantask.xsd" />
1184   <xsd:import namespace="http://docs.oasis-
1185 open.org/wsbpel/2.0/process/executable"
1186     schemaLocation="http://docs.oasis-
1187 open.org/wsbpel/2.0/OS/process/executable/ws-bpel_executable.xsd" />
1188
1189   <!-- base types for extensible elements -->
1190   <xsd:complexType name="tExtensibleElements">
1191     <xsd:sequence>
1192       <xsd:element name="documentation" type="tDocumentation"
1193         minOccurs="0" maxOccurs="unbounded" />
1194       <xsd:any namespace="##other" processContents="lax" minOccurs="0"
1195         maxOccurs="unbounded" />
1196     </xsd:sequence>
1197     <xsd:anyAttribute namespace="##other" processContents="lax" />
1198   </xsd:complexType>
1199   <xsd:complexType name="tExtensibleMixedNamespaceElements">
1200     <xsd:sequence>
1201       <xsd:element name="documentation" type="tDocumentation"
1202         minOccurs="0" maxOccurs="unbounded" />
1203       <xsd:element name="extensions" type="tExtensions" minOccurs="0" />
1204     </xsd:sequence>
1205     <xsd:anyAttribute namespace="##other" processContents="lax" />
1206   </xsd:complexType>
1207   <xsd:complexType name="tDocumentation" mixed="true">
1208     <xsd:sequence>
1209     <xsd:any namespace="##other" processContents="lax" minOccurs="0"

```



```

1210     maxOccurs="unbounded" />
1211   </xsd:sequence>
1212   <xsd:attribute ref="xml:lang" />
1213 </xsd:complexType>
1214 <xsd:complexType name="tExtensions">
1215   <xsd:sequence>
1216     <xsd:any namespace="##other" processContents="lax" minOccurs="0"
1217       maxOccurs="unbounded" />
1218   </xsd:sequence>
1219 </xsd:complexType>
1220
1221 <!-- element "humanInteractions" to be used within "bpel:process" -->
1222 <xsd:element name="humanInteractions" type="tHumanInteractions" />
1223 <xsd:complexType name="tHumanInteractions">
1224   <xsd:complexContent>
1225     <xsd:extension base="tExtensibleMixedNamespaceElements">
1226       <xsd:sequence>
1227         <xsd:element ref="htd:logicalPeopleGroups" minOccurs="0" />
1228         <xsd:element ref="htd:tasks" minOccurs="0" />
1229         <xsd:element ref="htd:notifications" minOccurs="0" />
1230       </xsd:sequence>
1231     </xsd:extension>
1232   </xsd:complexContent>
1233 </xsd:complexType>
1234
1235 <!-- element "peopleAssignments" to be used within "bpel:process" -->
1236 <xsd:element name="peopleAssignments" type="tPeopleAssignments" />
1237 <xsd:complexType name="tPeopleAssignments">
1238   <xsd:complexContent>
1239     <xsd:extension base="tExtensibleElements">
1240       <xsd:sequence>
1241         <xsd:element ref="genericHumanRole" minOccurs="1"
1242 maxOccurs="unbounded" />
1243       </xsd:sequence>
1244     </xsd:extension>
1245   </xsd:complexContent>
1246 </xsd:complexType>
1247
1248 <!-- element "genericHumanRole" within BPEL4People -->
1249 <xsd:element name="genericHumanRole"
1250 type="htd:tGenericHumanRoleAssignmentBase" abstract="true" block="" />
1251
1252   <xsd:element name="processStakeholders"
1253 type="htd:tGenericHumanRoleAssignment" substitutionGroup="genericHumanRole" />
1254   <xsd:element name="businessAdministrators"
1255 type="htd:tGenericHumanRoleAssignment" substitutionGroup="genericHumanRole" />
1256   <xsd:element name="processInitiator" type="htd:tGenericHumanRoleAssignment"
1257 substitutionGroup="genericHumanRole" />
1258
1259 <!-- element "argument" to be used within "bpel:from" -->
1260 <xsd:element name="argument" type="tArgument" />
1261 <xsd:complexType name="tArgument">
1262   <xsd:complexContent>
1263     <xsd:extension base="bpel:tExpression">
1264       <xsd:attribute name="name" type="xsd:NCName" />
1265     </xsd:extension>
1266   </xsd:complexContent>
1267 </xsd:complexType>

```

```

1268
1269     <!-- attribute "logicalPeopleGroup" to be used within "bpel:from" and
1270 "bpel:to" -->
1271     <xsd:attribute name="logicalPeopleGroup" type="xsd:NCName" />
1272
1273     <!-- attribute "shareComments" to be used within "bpel:process" and
1274 "bpel:scope" -->
1275     <xsd:attribute name="shareComments" type="xsd:boolean" />
1276
1277     <!-- element "peopleActivity" to be used within "bpel:extensionActivity" --
1278 >
1279     <xsd:element name="peopleActivity" type="tPeopleActivity" />
1280     <xsd:complexType name="tPeopleActivity">
1281         <xsd:complexContent>
1282             <xsd:extension base="tExtensibleMixedNamespaceElements">
1283                 <xsd:sequence>
1284                     <xsd:element ref="bpel:targets" minOccurs="0" />
1285                     <xsd:element ref="bpel:sources" minOccurs="0" />
1286                     <xsd:choice>
1287                         <xsd:element ref="htd:task" />
1288                         <xsd:element ref="localTask" />
1289                         <xsd:element ref="remoteTask" />
1290                         <xsd:element ref="htd:notification" />
1291                         <xsd:element ref="localNotification" />
1292                         <xsd:element ref="remoteNotification" />
1293                     </xsd:choice>
1294                     <xsd:element ref="scheduledActions" minOccurs="0" />
1295                     <xsd:element ref="toParts" minOccurs="0" />
1296                     <xsd:element ref="fromParts" minOccurs="0" />
1297                     <xsd:element ref="attachmentPropagation" minOccurs="0" />
1298                     <xsd:any namespace="##other" processContents="lax"
1299                         minOccurs="0" maxOccurs="unbounded" />
1300                 </xsd:sequence>
1301                 <xsd:attribute name="name" type="xsd:NCName" />
1302                 <xsd:attribute name="suppressJoinFailure" type="tBoolean"
1303                     use="optional" />
1304                 <xsd:attribute name="inputVariable" type="xsd:QName" />
1305                 <xsd:attribute name="outputVariable" type="xsd:QName" />
1306                 <xsd:attribute name="isSkipable" type="tBoolean"
1307                     use="optional" default="no" />
1308                 <xsd:attribute name="dontShareComments" type="tBoolean"
1309                     use="optional" default="no" />
1310             </xsd:extension>
1311         </xsd:complexContent>
1312     </xsd:complexType>
1313     <xsd:complexType name="tOverridableTaskElements">
1314         <xsd:complexContent>
1315             <xsd:extension base="tExtensibleMixedNamespaceElements">
1316                 <xsd:sequence>
1317                     <xsd:element ref="htd:priority" minOccurs="0" />
1318                     <xsd:element ref="htd:peopleAssignments" minOccurs="0" />
1319                 </xsd:sequence>
1320             </xsd:extension>
1321         </xsd:complexContent>
1322     </xsd:complexType>
1323     <xsd:element name="localTask" type="tLocalTask" />
1324     <xsd:complexType name="tLocalTask">
1325         <xsd:complexContent>

```

```

1326     <xsd:extension base="tOverridableTaskElements">
1327         <xsd:attribute name="reference" type="xsd:QName"
1328             use="required" />
1329     </xsd:extension>
1330 </xsd:complexContent>
1331 </xsd:complexType>
1332 <xsd:element name="remoteTask" type="tRemoteTask" />
1333 <xsd:complexType name="tRemoteTask">
1334     <xsd:complexContent>
1335         <xsd:extension base="tOverridableTaskElements">
1336             <xsd:attribute name="partnerLink" type="xsd:NCName"
1337                 use="required" />
1338             <xsd:attribute name="operation" type="xsd:NCName"
1339                 use="required" />
1340             <xsd:attribute name="responseOperation" type="xsd:NCName" />
1341         </xsd:extension>
1342     </xsd:complexContent>
1343 </xsd:complexType>
1344 <xsd:complexType name="tOverridableNotificationElements">
1345     <xsd:complexContent>
1346         <xsd:extension base="tExtensibleMixedNamespaceElements">
1347             <xsd:sequence>
1348                 <xsd:element ref="htd:priority" minOccurs="0" />
1349                 <xsd:element ref="htd:peopleAssignments" minOccurs="0" />
1350             </xsd:sequence>
1351         </xsd:extension>
1352     </xsd:complexContent>
1353 </xsd:complexType>
1354 <xsd:element name="localNotification" type="tLocalNotification" />
1355 <xsd:complexType name="tLocalNotification">
1356     <xsd:complexContent>
1357         <xsd:extension base="tOverridableNotificationElements">
1358             <xsd:attribute name="reference" type="xsd:QName"
1359                 use="required" />
1360         </xsd:extension>
1361     </xsd:complexContent>
1362 </xsd:complexType>
1363 <xsd:element name="remoteNotification" type="tRemoteNotification" />
1364 <xsd:complexType name="tRemoteNotification">
1365     <xsd:complexContent>
1366         <xsd:extension base="tOverridableNotificationElements">
1367             <xsd:attribute name="partnerLink" type="xsd:NCName"
1368                 use="required" />
1369             <xsd:attribute name="operation" type="xsd:NCName"
1370                 use="required" />
1371         </xsd:extension>
1372     </xsd:complexContent>
1373 </xsd:complexType>
1374 <xsd:element name="scheduledActions" type="tScheduledActions" />
1375 <xsd:complexType name="tScheduledActions">
1376     <xsd:complexContent>
1377         <xsd:extension base="tExtensibleElements">
1378             <xsd:sequence>
1379                 <xsd:element name="deferActivation"
1380                     type="tScheduledActionsDetails" minOccurs="0" />
1381                 <xsd:element name="expiration"
1382                     type="tScheduledActionsDetails" minOccurs="0" />
1383             </xsd:sequence>

```

```

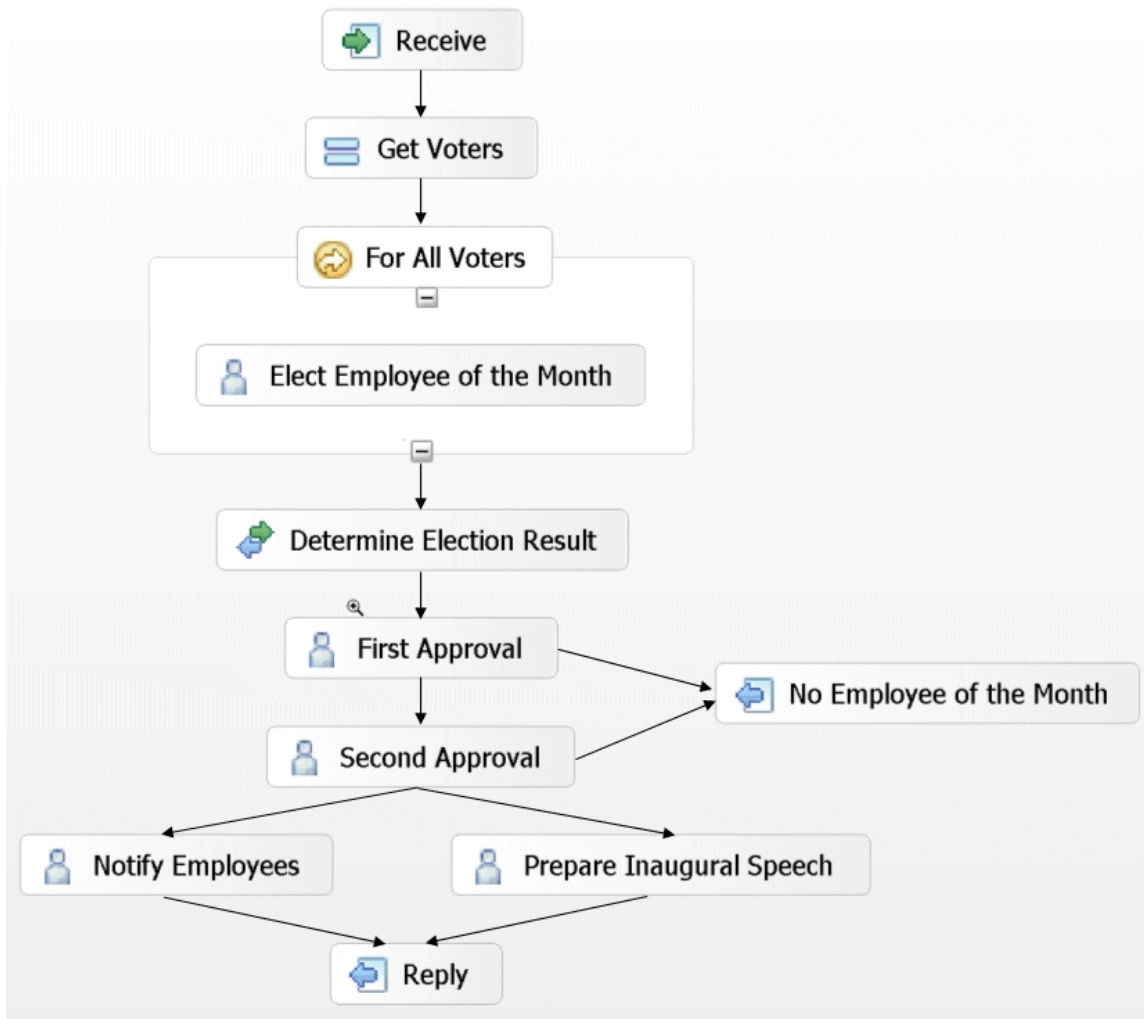
1384     </xsd:extension>
1385   </xsd:complexContent>
1386 </xsd:complexType>
1387 <xsd:complexType name="tScheduledActionsDetails">
1388   <xsd:complexContent>
1389     <xsd:extension base="tExtensibleElements">
1390       <xsd:sequence>
1391         <xsd:choice>
1392           <xsd:element name="for" type="bpel:tDuration-expr" />
1393           <xsd:element name="until" type="bpel:tDeadline-expr" />
1394         </xsd:choice>
1395       </xsd:sequence>
1396     </xsd:extension>
1397   </xsd:complexContent>
1398 </xsd:complexType>
1399 <xsd:element name="fromParts" type="tFromParts" />
1400 <xsd:complexType name="tFromParts">
1401   <xsd:complexContent>
1402     <xsd:extension base="tExtensibleElements">
1403       <xsd:sequence>
1404         <xsd:element ref="fromPart" maxOccurs="unbounded" />
1405       </xsd:sequence>
1406     </xsd:extension>
1407   </xsd:complexContent>
1408 </xsd:complexType>
1409 <xsd:element name="fromPart" type="tFromPart" />
1410 <xsd:complexType name="tFromPart">
1411   <xsd:complexContent>
1412     <xsd:extension base="tExtensibleElements">
1413       <xsd:attribute name="part" type="xsd:NCName" use="required" />
1414       <xsd:attribute name="toVariable" type="bpel:BPELVariableName"
1415         use="required" />
1416     </xsd:extension>
1417   </xsd:complexContent>
1418 </xsd:complexType>
1419 <xsd:element name="toParts" type="tToParts" />
1420 <xsd:complexType name="tToParts">
1421   <xsd:complexContent>
1422     <xsd:extension base="tExtensibleElements">
1423       <xsd:sequence>
1424         <xsd:element ref="toPart" maxOccurs="unbounded" />
1425       </xsd:sequence>
1426     </xsd:extension>
1427   </xsd:complexContent>
1428 </xsd:complexType>
1429 <xsd:element name="toPart" type="tToPart" />
1430 <xsd:complexType name="tToPart">
1431   <xsd:complexContent>
1432     <xsd:extension base="tExtensibleElements">
1433       <xsd:attribute name="part" type="xsd:NCName" use="required" />
1434       <xsd:attribute name="fromVariable"
1435         type="bpel:BPELVariableName" use="required" />
1436     </xsd:extension>
1437   </xsd:complexContent>
1438 </xsd:complexType>
1439 <xsd:element name="attachmentPropagation"
1440   type="tAttachmentPropagation" />
1441 <xsd:complexType name="tAttachmentPropagation">

```

```
1442 <xsd:complexContent>
1443   <xsd:extension base="tExtensibleElements">
1444     <xsd:attribute name="fromProcess" type="tFromProcess"
1445       default="all" />
1446     <xsd:attribute name="toProcess" type="tToProcess"
1447       default="newOnly" />
1448   </xsd:extension>
1449 </xsd:complexContent>
1450 </xsd:complexType>
1451 <xsd:simpleType name="tFromProcess">
1452   <xsd:restriction base="xsd:string">
1453     <xsd:enumeration value="all" />
1454     <xsd:enumeration value="none" />
1455   </xsd:restriction>
1456 </xsd:simpleType>
1457 <xsd:simpleType name="tToProcess">
1458   <xsd:restriction base="xsd:string">
1459     <xsd:enumeration value="all" />
1460     <xsd:enumeration value="newOnly" />
1461     <xsd:enumeration value="none" />
1462   </xsd:restriction>
1463 </xsd:simpleType>
1464
1465 <!-- miscellaneous helper elements and types -->
1466 <xsd:simpleType name="tBoolean">
1467   <xsd:restriction base="xsd:string">
1468     <xsd:enumeration value="yes" />
1469     <xsd:enumeration value="no" />
1470   </xsd:restriction>
1471 </xsd:simpleType>
1472
1473 </xsd:schema>
```

1474 **D. Sample**

1475 This appendix contains a sample that outlines the basic concepts of this specification. The sample  
1476 process implements the election of the “Employee of the month” in a fictitious company. The structure of  
1477 the business process is shown in the figure below:



1478  
1479 The process is started and as a first step, the people are determined that qualify as voters for the  
1480 “Employee of the month”. Next, all the voters identified before get a chance to cast their votes. After that,  
1481 the election result is determined by counting the votes casted. After the result is clear, two different  
1482 people from the set of people entitled to approve the election either accept or reject the voting result. In  
1483 case any of the two rejects, then there is no “Employee of the month” elected in the given month, and the  
1484 process ends. In case all approvals are obtained successfully, the employees are notified about the  
1485 outcome of the election, and a to-do is created for the elected “Employee of the month” to prepare an  
1486 inaugural speech. Once this is completed, the process completes successfully.

1487 The sections below show the definition of the BPEL process implementing the “Employee of the month”  
1488 process.

## 1489 D.1 BPEL Definition

```
1490 <?xml version="1.0" encoding="UTF-8"?>
1491 <!--
1492 Copyright (c) OASIS Open 2009. All Rights Reserved.
1493 -->
1494 <process name="EmployeeOfTheMonthProcess"
1495 targetNamespace="http://www.example.com"
1496 xmlns:tns="http://www.example.com"
1497 xmlns:hr="http://www.example.com/approval"
1498 xmlns:el="http://www.example.com/election"
1499 xmlns:ty="http://www.example.com/types"
1500 xmlns:ta="http://www.example.com/tasks"
1501 xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
1502 xmlns:b4p="http://docs.oasis-open.org/ns/bpel4people/bpel4people/200803"
1503 xmlns:htd="http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803"
1504 xmlns:htt="http://docs.oasis-open.org/ns/bpel4people/ws-
1505 humantask/types/200803"
1506 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1507 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1508 xsi:schemaLocation="http://docs.oasis-
1509 open.org/ns/bpel4people/bpel4people/200803 ../../xml/bpel4people.xsd
1510 http://docs.oasis-open.org/ns/bpel4people/ws-humantask/200803 ../../xml/ws-
1511 humantask.xsd http://docs.oasis-open.org/ns/bpel4people/ws-
1512 humantask/types/200803 ../../xml/ws-humantask-types.xsd">
1513
1514 <documentation>
1515 Example for BPEL4People 1.1 - WS-BPEL 2.0 Process with BPEL4People
1516 Extensions
1517 </documentation>
1518
1519 <b4p:humanInteractions>
1520
1521 <htd:logicalPeopleGroups>
1522
1523 <htd:logicalPeopleGroup name="voters">
1524 <htd:documentation xml:lang="en-US">
1525 The group entitled to vote the employee of the month for the
1526 given region.
1527 </htd:documentation>
1528 <htd:parameter name="region" type="xsd:string" />
1529 </htd:logicalPeopleGroup>
1530
1531 <htd:logicalPeopleGroup name="approvers">
1532 <htd:documentation xml:lang="en-US">
1533 The group entitled to approve the elected employee of the
1534 month for the given region.
1535 </htd:documentation>
1536 <htd:parameter name="region" type="xsd:string" />
1537 </htd:logicalPeopleGroup>
1538
1539 <htd:logicalPeopleGroup name="employees">
1540 <htd:documentation xml:lang="en-US">
1541 The group of employees to be notified about the election
1542 result of the employee of the month election for the given
1543 region.
1544 </htd:documentation>
1545 <htd:parameter name="region" type="xsd:string" />
```

```

1546 </htd:logicalPeopleGroup>
1547
1548 <htd:logicalPeopleGroup name="regionalElectionCommittee">
1549   <htd:documentation xml:lang="en-US">
1550     The group who is in charge for the election of the
1551     employee of the month election for the given region.
1552   </htd:documentation>
1553   <htd:parameter name="region" type="xsd:string" />
1554 </htd:logicalPeopleGroup>
1555
1556 </htd:logicalPeopleGroups>
1557
1558 <htd:tasks>
1559   <htd:task name="approveEmployeeOfTheMonth">
1560     <htd:documentation xml:lang="en-US">
1561       The reusable definition of the task used to approve the
1562       election of the employee of the month.
1563     </htd:documentation>
1564     <htd:interface operation="approve" portType="hr:approvalPT" />
1565     <htd:peopleAssignments>
1566       <htd:potentialOwners>
1567         <htd:from logicalPeopleGroup="approvers">
1568           <!-- variables used here need to be defined on the
1569           enclosing scope or above -->
1570           <htd:argument name="region">
1571             $selectionRequest/region
1572           </htd:argument>
1573         </htd:from>
1574       </htd:potentialOwners>
1575     </htd:peopleAssignments>
1576     <htd:presentationElements/>
1577   </htd:task>
1578 </htd:tasks>
1579
1580 </b4p:humanInteractions>
1581
1582 <b4p:peopleAssignments>
1583
1584   <b4p:processStakeholders>
1585     <htd:from logicalPeopleGroup="regionalElectionCommittee">
1586       <htd:argument name="region">
1587         $selectionRequest/region
1588       </htd:argument>
1589     </htd:from>
1590   </b4p:processStakeholders>
1591
1592   <b4p:businessAdministrators>
1593     <htd:from>
1594       <htd:literal>
1595         <htt:organizationalEntity>
1596           <htt:user>Peter</htt:user>
1597           <htt:user>Paul</htt:user>
1598           <htt:user>Mary</htt:user>
1599         </htt:organizationalEntity>
1600       </htd:literal>
1601     </htd:from>
1602   </b4p:businessAdministrators>
1603

```



```

1604 </b4p:peopleAssignments>
1605
1606 <extensions>
1607   <extension
1608     namespace="http://docs.oasis-
1609 open.org/ns/bpel4people/bpel4people/200803"
1610     mustUnderstand="yes" />
1611   <extension
1612     namespace="http://docs.oasis-open.org/ns/bpel4people/ws-
1613 humantask/200803"
1614     mustUnderstand="yes" />
1615 </extensions>
1616
1617 <import
1618   importType="http://www.w3.org/2001/XMLSchema"
1619   namespace="http://www.example.com/types" />
1620 <import
1621   importType="http://www.example.org/WS-HT"
1622   namespace="http://www.example.com/tasks" />
1623 <import
1624   importType="http://schemas.xmlsoap.org/wsdl/"
1625   namespace="http://www.example.com/election"
1626   location="bpel4people-example-election.wsdl" />
1627 <import
1628   importType="http://schemas.xmlsoap.org/wsdl/"
1629   namespace="http://www.example.com/approval"
1630   location="bpel4people-example-approval.wsdl" />
1631
1632 <partnerLinks>
1633   <partnerLink partnerLinkType="electionPLT"
1634     name="electionPL" />
1635 </partnerLinks>
1636
1637 <variables>
1638   <variable name="candidates" type="htt:users" />
1639   <variable name="voters" type="htd:tOrganizationalEntity" />
1640   <variable name="electionRequest" type="ty:electionRequestData" />
1641   <variable name="electionResult" type="ty:electionResultData" />
1642   <variable name="decision" type="xsd:boolean" />
1643   <variable name="speech" type="ty:document" />
1644 </variables>
1645
1646 <sequence>
1647
1648   <receive partnerLink="electionPL"
1649     portType="el:electionPT"
1650     operation="elect"
1651     variable="electionRequest"
1652     createInstance="yes" />
1653
1654   <assign name="getVoters">
1655     <copy>
1656       <from>$selectionRequests/candidates</from>
1657       <to variable="candidates" />
1658     </copy>
1659     <copy>
1660       <from b4p:logicalPeopleGroup="voters">
1661         <b4p:argument name="region">

```

```

1662     $SelectionRequest/region
1663     </b4p:argument>
1664 </from>
1665     <to variable="voters" />
1666 </copy>
1667 </assign>
1668
1669 <forEach counterName="i" parallel="yes">
1670     <startCounterValue> 1 </startCounterValue>
1671     <finalCounterValue>
1672         count($voters/users/user)
1673     </finalCounterValue>
1674
1675     <scope>
1676         <variables>
1677             <variable name="vote" type="htt:user"/>
1678         </variables>
1679
1680         <sequence>
1681             <!-- Constellation 1 -->
1682             <extensionActivity>
1683                 <b4p:peopleActivity name="electEmployeeOfTheMonth"
1684                     inputVariable="candidates"
1685                     outputVariable="vote"
1686                     isSkipable="yes">
1687                     <htd:task name="votingTask">
1688                         <htd:interface operation="vote"
1689                             portType="el:votingPT" />
1690                     <htd:peopleAssignments>
1691                         <htd:potentialOwners>
1692                             <htd:from>$voters/users/user[i]</htd:from>
1693                         </htd:potentialOwners>
1694                     </htd:peopleAssignments>
1695                     <htd:presentationElements/>
1696                 </htd:task>
1697                 <b4p:scheduledActions>
1698                     <b4p:expiration>
1699                         <b4p:documentation xml:lang="en-US">
1700                             This people activity expires when not completed
1701                             within 2 days after having been activated.
1702                         </b4p:documentation>
1703                         <b4p:for>P2D</b4p:for>
1704                     </b4p:expiration>
1705                 </b4p:scheduledActions>
1706                 </b4p:peopleActivity>
1707             </extensionActivity>
1708
1709             <assign>
1710                 <copy>
1711                     <from>$vote</from>
1712                     <to>$SelectionResult/votes[i]</to>
1713                 </copy>
1714             </assign>
1715
1716         </sequence>
1717     </scope>
1718 </forEach>
1719

```

```

1720 <!-- Might be Constellation 5 - standard WS-BPEL 2.0 invoke -->
1721 <!--
1722 <invoke name="determineElectionResult" partnerLink="..." operation="..."
1723 />
1724 -->
1725
1726 <!-- Constellation 2 -->
1727 <extensionActivity>
1728   <b4p:peopleActivity name="firstApproval"
1729     inputVariable="electionResult"
1730     outputVariable="decision">
1731     <b4p:localTask reference="tns:approveEmployeeOfTheMonth" />
1732   </b4p:peopleActivity>
1733 </extensionActivity>
1734
1735 <!-- Constellation 2 with override specifications -->
1736 <extensionActivity>
1737   <b4p:peopleActivity name="secondApproval"
1738     inputVariable="electionResult"
1739     outputVariable="decision">
1740     <b4p:localTask reference="tns:approveEmployeeOfTheMonth">
1741       <htd:peopleAssignments>
1742         <htd:excludedOwners>
1743           <htd:from>
1744             b4p:getActualOwner("tns:firstApproval")
1745           </htd:from>
1746         </htd:excludedOwners>
1747       </htd:peopleAssignments>
1748     </b4p:localTask>
1749   </b4p:peopleActivity>
1750 </extensionActivity>
1751
1752 <!-- Constellation 3 -->
1753 <extensionActivity>
1754   <b4p:peopleActivity name="notifyEmployees"
1755     inputVariable="electionResult">
1756     <b4p:localNotification reference="ta:employeeBroadcast" />
1757     <!-- notification is not defined as part of this document,
1758         but within a separate one
1759     -->
1760   </b4p:peopleActivity>
1761 </extensionActivity>
1762
1763 <!-- Constellation 4 -->
1764 <extensionActivity>
1765   <b4p:peopleActivity name="prepareInauguralSpeech"
1766     inputVariable="electionResult"
1767     outputVariable="speech"
1768     isSkipable="no">
1769     <b4p:remoteTask partnerLink="author"
1770       operation="prepareSpeech"
1771       responseOperation="receiveSpeech">
1772       <htd:priority>0</htd:priority> <!-- assign highest priority -->
1773     <htd:peopleAssignments>
1774       <htd:potentialOwners>
1775         <htd:from>$electionResult/winner</htd:from>
1776       </htd:potentialOwners>
1777     </htd:peopleAssignments>

```

```
1778     </b4p:remoteTask>
1779     </b4p:peopleActivity>
1780     </extensionActivity>
1781
1782 </sequence>
1783
1784 </process>
```

## 1785 D.2 WSDL Definitions

```
1786 <?xml version="1.0" encoding="UTF-8"?>
1787 <!--
1788 Copyright (c) OASIS Open 2009. All Rights Reserved.
1789 -->
1790 <wsdl:definitions
1791   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1792   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1793   xmlns:tns="http://www.example.com/approval"
1794   targetNamespace="http://www.example.com/approval">
1795
1796   <wsdl:documentation>
1797     Example for BPEL4People 1.1 - PeopleActivity Interface Definition
1798   </wsdl:documentation>
1799
1800   <!-- Messages -->
1801   <wsdl:message name="approvalInput">
1802     <wsdl:part name="parameters" type="xsd:string" />
1803   </wsdl:message>
1804   <wsdl:message name="approvalOutput">
1805     <wsdl:part name="parameters" type="xsd:string" />
1806   </wsdl:message>
1807
1808   <!-- Port Type -->
1809   <wsdl:portType name="approvalPT">
1810     <wsdl:operation name="approve">
1811       <wsdl:input message="tns:approvalInput" />
1812       <wsdl:output message="tns:approvalOutput" />
1813     </wsdl:operation>
1814   </wsdl:portType>
1815
1816 </wsdl:definitions>
```

```
1817
1818 <?xml version="1.0" encoding="UTF-8"?>
1819 <!--
1820 Copyright (c) OASIS Open 2009. All Rights Reserved.
1821 -->
1822 <wsdl:definitions
1823   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
1824   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1825   xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
1826   xmlns:tns="http://www.example.com/election"
1827   targetNamespace="http://www.example.com/election">
1828
1829   <wsdl:documentation>
1830     Example for BPEL4People 1.1 - PeopleActivity Interface Definition
1831   </wsdl:documentation>
1832
```

```
1833 <!-- WS-BPEL 2.0 Partner Link Type -->
1834 <plnk:partnerLinkType name="electionPLT">
1835   <plnk:role name="electionService" portType="tns:electionPT" />
1836 </plnk:partnerLinkType>
1837
1838 <!-- Messages -->
1839 <wsdl:message name="electionInput">
1840   <wsdl:part name="parameters" type="xsd:string" />
1841 </wsdl:message>
1842 <wsdl:message name="votingInput">
1843   <wsdl:part name="parameters" type="xsd:string" />
1844 </wsdl:message>
1845
1846 <!-- Port Types -->
1847 <wsdl:portType name="electionPT">
1848   <wsdl:operation name="elect">
1849     <wsdl:input message="tns:electionInput" />
1850   </wsdl:operation>
1851 </wsdl:portType>
1852 <wsdl:portType name="votingPT">
1853   <wsdl:operation name="vote">
1854     <wsdl:input message="tns:votingInput" />
1855   </wsdl:operation>
1856 </wsdl:portType>
1857
1858 </wsdl:definitions>
```

1859

---

## E. Acknowledgements

1860 The following individuals have participated in the creation of this specification and are gratefully  
1861 acknowledged:

1862

1863 **Members of the BPEL4People Technical Committee:**

1864 Phillip Allen, Microsoft Corporation

1865 Ashish Agrawal, Adobe Systems

1866 Mike Amend, BEA Systems, Inc.

1867 Stefan Baeuerle, SAP AG

1868 Charlton Barreto, Adobe Systems

1869 Justin Brunt, TIBCO Software Inc.

1870 Martin Chapman, Oracle Corporation

1871 Luc Clément, Active Endpoints, Inc.

1872 Manoj Das, Oracle Corporation

1873 Alireza Farhoush, TIBCO Software Inc.

1874 Mark Ford, Active Endpoints, Inc.

1875 Sabine Holz, SAP AG

1876 Dave Ings, IBM

1877 Gershon Janssen, Individual

1878 Diane Jordan, IBM

1879 Anish Karmarkar, Oracle Corporation

1880 Ulrich Keil, SAP AG

1881 Oliver Kieselbach, SAP AG

1882 Matthias Kloppmann, IBM

1883 Dieter König, IBM

1884 Marita Kruempelmann, SAP AG

1885 Frank Leymann, IBM

1886 Mark Little, Red Hat

1887 Alexander Malek, Microsoft Corporation

1888 Ashok Malhotra, Oracle Corporation

1889 Mike Marin, IBM

1890 Vinkesh Mehta, Deloitte Consulting LLP

1891 Jeff Mischkinsky, Oracle Corporation

1892 Ralf Mueller, Oracle Corporation

1893 Krasimir Nedkov, SAP AG

1894 Benjamin Notheis, SAP AG

1895 Michael Pellegrini, Active Endpoints, Inc.

1896 Hannah Petereit, SAP AG

1897 Gerhard Pfau, IBM

1898 Karsten Ploesser, SAP AG

1899 Ravi Rangaswamy, Oracle Corporation  
1900 Alan Rickayzen, SAP AG  
1901 Michael Rowley, BEA Systems, Inc.  
1902 Ron Ten-Hove, Sun Microsystems  
1903 Ivana Trickovic, SAP AG  
1904 Alessandro Triglia, OSS Nokalva  
1905 Claus von Riegen, SAP AG  
1906 Peter Walker, Sun Microsystems  
1907 Franz Weber, SAP AG  
1908 Prasad Yendluri, Software AG, Inc.

1909

1910 **BPEL4People 1.0 Specification Contributors:**

1911 Ashish Agrawal, Adobe  
1912 Mike Amend, BEA  
1913 Manoj Das, Oracle  
1914 Mark Ford, Active Endpoints  
1915 Chris Keller, Active Endpoints  
1916 Matthias Kloppmann, IBM  
1917 Dieter König, IBM  
1918 Frank Leymann, IBM  
1919 Ralf Müller, Oracle  
1920 Gerhard Pfau, IBM  
1921 Karsten Plösser, SAP  
1922 Ravi Rangaswamy, Oracle  
1923 Alan Rickayzen, SAP  
1924 Michael Rowley, BEA  
1925 Patrick Schmidt, SAP  
1926 Ivana Trickovic, SAP  
1927 Alex Yiu, Oracle  
1928 Matthias Zeller, Adobe

1929

1930 In addition, the following individuals have provided valuable input into the design of this specification:  
1931 Dave Ings, Diane Jordan, Mohan Kamath, Ulrich Keil, Matthias Kruse, Kurt Lind, Jeff Mischkinsky, Bhagat  
1932 Nainani, Michael Pellegrini, Lars Rueter, Frank Ryan, David Shaffer, Will Stallard, Cyrille Waguët, Franz  
1933 Weber, and Eric Wittmann.

---

## F. Non-Normative Text



1935

## G. Revision History

1936

[optional; should not be included in OASIS Standards]

1937

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