WS-BPEL Extension for People (BPEL4People), Version 1.0

June 2007

Authors

Ashish Agrawal, Adobe
Mike Amend, BEA
Manoj Das, Oracle
Mark Ford, Active Endpoints
Chris Keller, Active Endpoints
Matthias Kloppmann, IBM
Dieter König, IBM
Frank Leymann, IBM
Ralf Müller, Oracle
Gerhard Pfau, IBM
Karsten Plösser, SAP
Ravi Rangaswamy, Oracle
Alan Rickayzen, SAP
Michael Rowley, BEA
Patrick Schmidt, SAP
Ivana Trickovic, SAP
Alex Yiu, Oracle
Matthias Zeller, Adobe

Copyright Notice

© 2007 Active Endpoints Inc., Adobe Systems Inc., BEA Systems Inc., International Business Machines Corporation, Oracle Inc., and SAP AG. All rights reserved.

Licence

Permission to copy and display the WS-BPEL Extension for People Specification (the “Specification”, which includes WSDL and schema documents), in any medium without fee or royalty is hereby granted, provided that you include the following on ALL copies of the WS-BPEL Extension for People Specification, or portions thereof, that you make:

1. A link or URL to the Specification at one of the Authors’ websites.
2. The copyright notice as shown in the Specification.

Active Endpoints, Adobe Systems, BEA Systems, IBM, Oracle and SAP (collectively, the “Authors”) each agree to grant you a license, under royalty-free and otherwise reasonable, non-discriminatory terms and conditions, to their respective essential patent claims that they deem necessary to implement the Specification.

THE SPECIFICATION IS PROVIDED "AS IS," AND THE AUTHORS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

THE AUTHORS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE SPECIFICATION.

The name and trademarks of the Authors may NOT be used in any manner, including advertising or publicity pertaining to the Specification or its contents without specific, written prior permission. Title to copyright in the Specification will at all times remain with the Authors.

No other rights are granted by implication, estoppel or otherwise.

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

BPEL4People is provided as-is and for review and evaluation only. The authors hope to solicit your contributions and suggestions in the near future. The authors make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

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.

2 Language Design

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

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

2.1 Dependencies on Other Specifications

BPEL4People utilizes the following specifications:

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

2.2 Notational Conventions

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].

2.3 Namespaces

This specification uses a number of namespace prefixes throughout; they are listed in Table 1. Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [XML Namespaces]).

Prefix	**Namespace**
b4p	http://www.example.org/BPEL4People
bpel	http://docs.oasis-open.org/wsbpel/2.0/process/executable
htd	http://www.example.org/WS-HT
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance

Table 1 Prefixes and namespaces used in this specification

All information items defined by BPEL4People are identified by the XML namespace URI [XML Namespaces] `http://www.example.org/BPEL4People`. A normative XML Schema [XML Schema Part 1, Part 2] document can be obtained by dereferencing the XML namespace URI.

2.4 Language Extensibility

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

- Attributes from other namespaces to appear on any BPEL4People element

- Elements from other namespaces to appear within BPEL4People elements

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

Standard BPEL element `<extension>` must be used to declare mandatory and optional extensions of BPEL4People.

2.5 Overall Language Structure

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

2.5.1 Syntax

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

```
<bpel:process ...
  ...

  xmlns:b4p="http://www.example.org/BPEL4People"
  xmlns:htd="http://www.example.org/WS-HT">
  ...
  <bpel:extensions>
    <bpel:extension
      namespace="http://www.example.org/BPEL4People"
      mustUnderstand="yes"/>
    <bpel:extension
      namespace="http://www.example.org/WS-HT"
      mustUnderstand="yes"/>
  </bpel:extensions>

  <bpel:import
    importType="http://www.example.org/WS-HT" …/>

  ...
  <b4p:humanInteractions>?

    <htd:logicalPeopleGroups/>?
      <htd:logicalPeopleGroup name="NCName">+
        ...
      </htd:logicalPeopleGroup>
    </htd:logicalPeopleGroups>

    <htd:tasks>?
      <htd:task name="NCName">+
        ...
      </htd:task>
    </htd:tasks>

    <htd:notifications>?
      <htd:notification name="NCName">+
        ...
      </htd:notification>
    </htd:notifications>
  </b4p:humanInteractions>

  <b4p:peopleAssignments>
    ...
  </b4p:peopleAssignments>

  ...
  <bpel:extensionActivity>
    <b4p:peopleActivity name="NCName" ...>
      ...
    </b4p:peopleActivity>
  </bpel:extensionActivity>
  ...

</bpel:process>
```

A BPEL4People process must use BPEL4People extension elements and elements from WS-HumanTask namespace. Therefore elements from namespaces BPEL4People and WS-HumanTask MUST be understood.

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

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

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

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

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

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

```
<bpel:scope ...>
  ...
  <b4p:humanInteractions>?
    ...
  </b4p:humanInteractions>
  ...
  <bpel:extensionActivity>
```

```
    <b4p:peopleActivity name="NCName" ...>
      ...
    </b4p:peopleActivity>
  </bpel:extensionActivity>
  ...
</bpel:scope>
```

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

All BPEL4People elements may use the element `<b4p:documentation>` to provide annnotation for users. The content could be a plain text, HTML, and so on. The `<b4p:documentation>` element is optional and has the following syntax:

```
<b4p:documentation xml:lang="xsd:language">

  ...

</b4p:documentation>
```

3 Concepts

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

3.1 Generic Human Roles

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:

- Process initiator
- Process stakeholders
- Business administrators

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 overriden by specifying a people assignment for process initiator. Compliant implementations MUST ensure that at runtime at least one person is associated with this role.

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. Compliant implementations MUST ensure that at runtime at least one person is associated with this role

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. Compliant implementations MUST ensure that at runtime at least one person is associated with this role

3.1.1 Syntax

```
<b4p:peopleAssignments>

  <htd:genericHumanRole>+
    <htd:from>…</htd:from>
  </htd:genericHumanRole>

<b4p:peopleAssignments>
```

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

```
<b4p:peopleAssignments>

  <b4p:processInitiator>?
    <htd:from ...>...</htd:from>
  </b4p:processInitiator>

  <b4p:processStakeholders>?
    <htd:from ...>...</htd:from>
  </b4p:processStakeholders>

  <b4p:businessAdministrators>?
    <htd:from ...>...</htd:from>
  </b4p:businessAdministrators>

</b4p:peopleAssignments>
```

Only process-related human roles MAY be used within the `<b4p:peopleAssignments>` element. People are assigned to these roles as described in the following section.

Assigning people to process-related generic human roles happens during the process initialization. As such it is part of the scope initialization (which occurs when a `<bpel:process>` or `<bpel:scope>` is entered) and has impact on the scope initialization outcome. That is, if an assignment fails the entire process is treated as faulted.

3.2 Assigning People

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

- Via logical people groups (see 3.2.1 "Using Logical People Groups")
- Via literals (as introduced section 3.2.2 in [WS-HumanTask])
- Via expressions (see 3.2.2 "Computed Assignment").

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

3.2.1 Using Logical People Groups

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

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

Logical people group definitions can be referenced by multiple people activities. Each logical people group is bound to a people query during deployment. Using the same logical people group does not mean that the result of a people query is re-used, but that the same query is used to obtain a result. If the result of a previous people query needs to be re-used, then this result needs to be referenced explicitly from the process context. Please refer to section 5 "XPath Extension Functions" for a description of the syntax.

Assignment of Logical People Groups

BPEL <assign> activity (see [WS-BPEL 2.0] section 8.4 for more details) is used for manipulating values of logical people group. A mechanism to assign to a logical people group or to assign from a logical people group using BPEL copy assignments is provided. The semantics of the `<copy>` activity introduced in [WS-BPEL 2.0] (see sections 8.4.1, 8.4.2 and 8.4.3 for more details) applies.

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

```
<bpel:from b4p:logicalPeopleGroup="NCName">
  <b4p:argument name="NCName" expressionLanguage="anyURI"?>*
    value
  </b4p:argument>
</bpel:from>

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

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

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

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

- To copy to a variable

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

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

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

- To copy to a property

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

- To copy to a logical people group

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

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

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

- To copy from a variable

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

- To copy from a property

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

- 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">
        $electionRequest/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>
        <myns:entity xsi:type="htd:tOrganizationalEntity">
          <htd:users>
            <htd:user>Alan</htd:user>
            <htd:user>Dieter</htd:user>
            <htd:user>Frank</htd:user>
            <htd:user>Gerhard</htd:user>
            <htd:user>Ivana</htd:user>
            <htd:user>Karsten</htd:user>
            <htd:user>Matthias</htd:user>
            <htd:user>Patrick</htd:user>
          </htd:users>
        </myns:entity>
      </bpel:literal>
    </bpel:from>
    <bpel:to b4p:logicalPeopleGroup="bpel4peopleAuthors" />
  </bpel:copy>
</bpel:assign>
```

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

```
<bpel:assign>
  <bpel:copy>
    <bpel:from b4p:logicalPeopleGroup="bpel4peopleAuthors" />
    <bpel:to b4p:logicalPeopleGroup="approvers" />
  </bpel:copy>
</bpel:assign>
```

3.2.2 Computed Assignment

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

```
<htd:genericHumanRole>
  <bpel:from variable="NCName" part="NCName"?>
   …
  </bpel:from>
</htd:genericHumanRole>
```

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

All other process context may be accessed using expressions of the following style

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

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

3.3 Ad-hoc Attachments

Processes can have ad-hoc attachments. It is possible to exchange ad-hoc attachments between people activities of a process, and even propagate ad-hoc attachments to and from the process level.

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

In all cases, if several attachments of the same name are propagated, they are combined into a list of attachments with that name; no attachment is lost or overwritten.

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

4 People Activity

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.

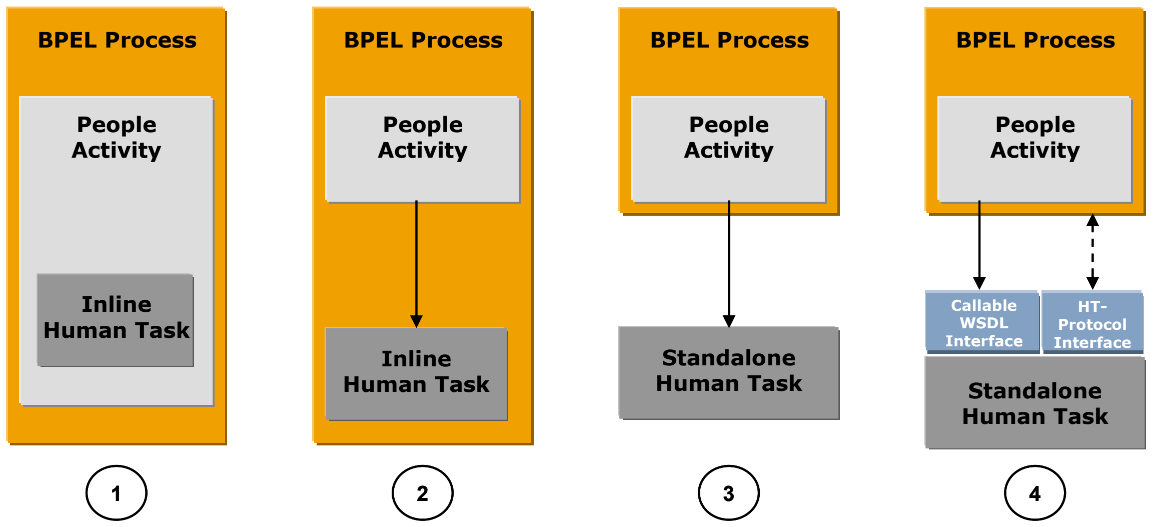


Figure 1: Constellations

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 process and task (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.

4.1 Overall Syntax

Definition of people activity:

```
<bpel:extensionActivity>

  <b4p:peopleActivity name="NCName"
    inputVariable="NCName"?
    outputVariable="NCName"?
    isSkipable="xsd:boolean"?
    standard-attributes>

    standard-elements

    ( <htd:task>...</htd:task>
    | <b4p:localTask>...</b4p:localTask>
    | <b4p:remoteTask>...</b4p:remoteTask>
    | <htd:notification>...</htd:notification>
    | <b4p:localNotification>...</b4p:localNotification>
    | <b4p:remoteNotification>...</b4p:remoteNotification>
    )

    <b4p:scheduledActions>?
      ...
    </b4p:scheduledActions>

    <bpel:toParts>?
      <bpel:toPart part="NCName" fromVariable="BPELVariableName"/>+
    </bpel:toParts>
    <bpel:fromParts>?
      <bpel:fromPart part="NCName" toVariable="BPELVariableName"/>+
    </bpel:fromParts>

    <b4p:attachmentPropagation fromProcess="all|none"
                               toProcess="all|newOnly|none"/>?
  </b4p:peopleActivity>

</bpel:extensionActivity>
```

4.1.1 Properties

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

- `inputVariable`: This attribute refers to a process variable which is used as input of the WSDL operation of a task or notification. The process variable MUST have a WSDL message type. This attribute is optional. If this attribute is not present the `<bpel:toParts>` element MUST be used.
- `outputVariable`: This attribute refers to a process variable which is used as output of the WSDL operation of a task. The process variable MUST have a WSDL message type. This attribute is optional. If the people activity uses a human task and this attribute is not present the `<bpel:fromParts>` element MUST be used. The `outputVariable` attribute MUST not be used if the people activity uses a notification.
- `isSkipable`: This attribute indicates whether the task associated with the activity can be skipped at runtime or not. This is propagated to the task level. This attribute is optional. The default for this attribute is "no".
- `standard-attributes`: The activity makes available all BPEL's standard attributes.

- `standard-elements`: The activity makes available all BPEL's standard elements.
- `htd:task`: This element is used to define an inline task within the people activity (constellation 1 in the figure above). This element is optional. Its syntax and semantics are introduced in section 4.3 "People Activities Using Local Human Tasks".
- `b4p:localTask`: This element is used to refer to a standalone task with no callable Web service interface (constellations 2 or 3). This element is optional. Its syntax and semantics are introduced in section 4.3 "People Activities Using Local Human Tasks".
- `b4p:remoteTask`: This element is used to refer to a standalone task offering callable Web service interface (constellation 4). This element is optional. Its syntax and semantics are introduced in section 4.5 "People Activities Using Remote Human Tasks".
- `htd:notification`: This element is used to define an inline notification within the people activity (constellation 1 in the figure above). This element is optional. Its semantics is introduced in section 4.4 "People Activities Using Local Notifications".
- `b4p:localNotification`: This element is used to refer to a standalone notification with no callable Web service interface (constellations 2 or 3). This element is optional. Its semantics is introduced in section 4.4 "People Activities Using Local Notifications".
- `b4p:remoteNotification`: This element is used to refer to a standalone notification offering callable Web service interface (constellation 4). This element is optional. Its syntax and semantics are introduced in section 4.6 "People Activities Using Remote Notifications".
- `b4p:scheduledActions`: This element specifies when the task must change the state. Its syntax and semantics are introduced in section 4.7 "Elements for Scheduled Actions".
- `bpel:toParts`: This element is used to explicilty create multi-part WSDL message from multiple BPEL variables. The element is optional. Its syntax and semantics are introduced in the WS-BPEL 2.0 specification, section 10.3.1. The `<bpel:toParts>` element and the `inputVariable` attribute are mutually exclusive.
- `bpel:fromParts`: This element is used to assign values to multiple BPEL variables from an incoming multi-part WSDL message. The element is optional. Its syntax and semantics are introduced in the WS-BPEL 2.0 specification, section 10.3.1. The `<bpel:fromParts>` element and the `outputVariable` attribute are mutually exclusive. This element MUST not be used if the people activity uses a notification.
- `b4p:attachmentPropagation`: This element is used to describe the propagation behavior of ad-hoc attachments to and from the people activity. On activation of the people activity, either all ad-hoc attachments from the process are propagated to the people activity, so they become available to the corresponding task, or none. The `fromProcess` attribute is used to specify this. On completion of a people activity, all ad-hoc attachments are propagated to its process, or only newly created ones (but not those that

were modified), or none. The `toProcess` attribute is used to specify this. The element is optional. The default value for this element is that all attachments are propagated from the process to the people activity and only new attachments are propagated back to the process.

4.2 Standard Overriding Elements

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

The following elements can be overriden:

- people assignments
- priority

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

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

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

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

```
<htd:priority expressionLanguage="anyURI"?>
  integer-expression
</htd:priority>

<htd:peopleAssignments>?
  <htd:genericHumanRole>
    <htd:from>…</htd:from>
  </htd:genericHumanRole>
</htd:peopleAssignments>
```

4.3 People Activities Using Local Human Tasks

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

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

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

4.3.1 Syntax

```
<b4p:peopleActivity
      inputVariable="NCName"?
      outputVariable="NCName"?
      isSkipable="xsd:boolean"?
      standard-attributes>
      standard-elements

      ( <htd:task>...</htd:task>
      | <b4p:localTask reference="QName">
          standard-overriding-elements
        </b4p:localTask>
      )

</b4p:peopleActivity>
```

Properties

Element `<htd:task>` is used to define an inline task within the people activity. The syntax and semantics of the element are given in the WS-HumanTask specification. In addition, XPath expressions used in enclosed elements may refer to process variables.

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

4.3.2 Examples

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

```
<b4p:peopleActivity inputVariable="candidates"
                    outputVariable="vote"
                    isSkipable="yes">
  <htd:task>
    <htd:peopleAssignments>
      <htd:potentialOwners>
        <htd:from>$voters/users/user[i]</htd:from>
      </htd:potentialOwners>
    </htd:peopleAssignments>
  </htd:task>
  <b4p:scheduledActions>
    <b4p:expiration>
      <b4p:documentation xml:lang="en-US">
        This people activity expires when not completed
        within 2 days after having been activated.
      </b4p:documentation>
      <b4p:for>P2D</b4p:for>
    </b4p:expiration>
  </b4p:scheduledActions>
</b4p:peopleActivity>
```

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

```
<extensionActivity>
  <b4p:peopleActivity name="firstApproval"
                      inputVariable="electionResult"
                      outputVariable="decision">
    <b4p:localTask reference="tns:approveEmployeeOfTheMonth"/>
  </b4p:peopleActivity>
</extensionActivity>
```

4.4 People Activities Using Local Notifications

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

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

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

4.4.1 Syntax

```
<b4p:peopleActivity name="NCName"?
      inputVariable="NCName"?
      standard-attributes>
      standard-elements

      ( <htd:notification>...</htd:notification>
      | <b4p:localNotification reference="QName">
          standard-overriding-elements
        </b4p:localNotification>
      )
```

```
</b4p:peopleActivity>
```

Properties

Element `<htd:notification>` is used to define an inline notification within the people activity. The syntax and semantics of the element are given in the WS-HumanTask specification. In addition, XPath expressions used in enclosed elements may refer to process variables.

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

4.4.2 Examples

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

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

4.5 People Activities Using Remote Human Tasks

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

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

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

4.5.1 Syntax

```
<b4p:remoteTask
  partnerLink="NCName"
  operation="NCName"
  responseOperation="NCName"?>

  standard-overriding-elements

</b4p:remoteTask>
```

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

4.5.2 Example

```
<bpel:extensionActivity>
  <b4p:peopleActivity name="prepareInauguralSpeech"
                      inputVariable="electionResult"
                      outputVariable="speech"
                      isSkipable="no">
    <b4p:remoteTask partnerLink="author"
                    operation="prepareSpeech"
                    responseOperation="receiveSpeech">
      <htd:priority>0</htd:priority> <!-- assign highest prio -->
      <htd:peopleAssignments>
        <htd:potentialOwners>
          <htd:from>$electionResult/winner</htd:from>
        </htd:potentialOwners>
      </htd:peopleAssignments>
    </b4p:remoteTask>
  </b4p:peopleActivity>
</bpel:extensionActivity>
```

4.5.3 Passing Endpoint References For Callbacks

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

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

The additional metadata may consist of the name of the port type of the port as well as binding information about how to reach the port (see [WS-Addr-Core]) in order to support the replying activity of the human task to send its response to the port. In addition, the name of the receiving operation at the calling process side is required. This name MUST be provided as value of the `responseOperation` attribute of the `<b4p:remoteTask>` element (discussed in the previous section) and is passed together with an appropriate endpoint reference.

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

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

4.6 People Activities Using Remote Notifications

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

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

4.6.1 Syntax

```
<b4p:remoteNotification
  partnerLink="NCName"
  operation="NCName">

  standard-overriding-elements

</b4p:remoteNotification>
```

4.6.2 Example

```
<bpel:extensionActivity>
  <b4p:peopleActivity name="notifyEmployees"
                      inputVariable="electionResult">
    <b4p:remoteNotification partnerLink="employeeNotification"
                            operation="receiveElectionResult">
      <htd:priority>42</htd:priority> <!-- assign moderate prio -->
      <htd:peopleAssignments>
```

```
        <htd:recipients>
          <htd:from>$voters</htd:from>
        </htd:recipients>
      </htd:peopleAssignments>
    </b4p:remoteNotification>
  </b4p:peopleActivity>
</bpel:extensionActivity>
```

4.7 Elements for Scheduled Actions

Scheduled actions allow specification determining when a task must change the state. The following scheduled actions are defined:

- **DeferActivation**: Specifies the activation time of the task. It is defined as either the period of time after which the task reaches state *Ready* (in case of explicit claim) or state *Reserved* (in case of implicit claim), or the point in time when the task reaches state *Ready* or state *Reserved*. The default value is zero, i.e. the task is immediately activated. If the activation time is defined as a point in time and the task is created after that point in time then the task will be activated immediately.
- **Expiration**: Specifies the expiration time of the task when the task becomes obsolete. It is defined as either the period of time after which the task expires or the point in time when the task expires. The time starts to be measured when the task enters state *Created*. If the task does not reach one of the final states (*Completed*, *Failed*, *Error*, *Exited, Obsolete*) by the expiration time it will change to state *Exited* and no additional user-defined action will be performed. The default value is infinity, i.e. the task never expires. If the expiration time is defined as a point in time and the task is created after that point in time then the task will immediately change to state *Exited*. Note that deferred activation does not impact expiration. Therefore the task may expire even before being activated.

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

Syntax:

```
<b4p:scheduledActions>?

  <b4p:deferActivation>?
  ( <b4p:for expressionLanguage="anyURI"?>
      duration-expression
    </b4p:for>
  | <b4p:until expressionLanguage="anyURI"?>
     deadline-expression
    </b4p:until>
  )
  </b4p:deferActivation>

  <b4p:expiration>?
  ( <b4p:for expressionLanguage="anyURI"?>
      duration-expression
    </b4p:for>
  | <b4p:until expressionLanguage="anyURI"?>
      deadline-expression
    </b4p:until>
  )
  </b4p:expiration>

</b4p:scheduledActions>
```

Properties

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

- `b4p:deferActivation`: The element is used to specify activation time of the task. It includes the following elements:
    - `b4p:for`: The element is an expression which specifies the period of time (duration) after which the task reaches state *Ready* (in case of explicit claim) or state *Reserved* (in case of implicit claim).
    - `b4p:until`: The element is an expression which specifies the point in time when the task reaches state *Ready* or state *Reserved*.

  Elements `<b4p:for>` and `<b4p:until>` are mutually exclusive. There MUST be at least one `<b4p:for>` or `<b4p:until>` element.
- `b4p:expiration`: The element is used to specify the expiration time of the task when the task becomes obsolete:
    - `b4p:for`: The element is an expression which specifies the period of time (duration) after which the task expires.
    - `b4p:until`: The element is an expression which specifies the point in time when the task expires.

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

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

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

Example:

```
<b4p:scheduledActions>

  <b4p:deferActivation>
    <b4p:documentation xml:lang="en-US">
      Activation of this task is deferred until the time specified
      in its input data.
    </b4p:documentation>
     <b4p:until>htd:getInput()/activateAt</b4p:until>
  </b4p:deferActivation>

  <b4p:expiration>
    <b4p:documentation xml:lang="en-US">
      This task expires when not completed within 14 days after
      having been activated.
    </b4p:documentation>
    <b4p:for>P14D</b4p:for>
  </b4p:expiration>

</b4p:scheduledActions>
```

4.8 People Activity Behavior and State Transitions

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

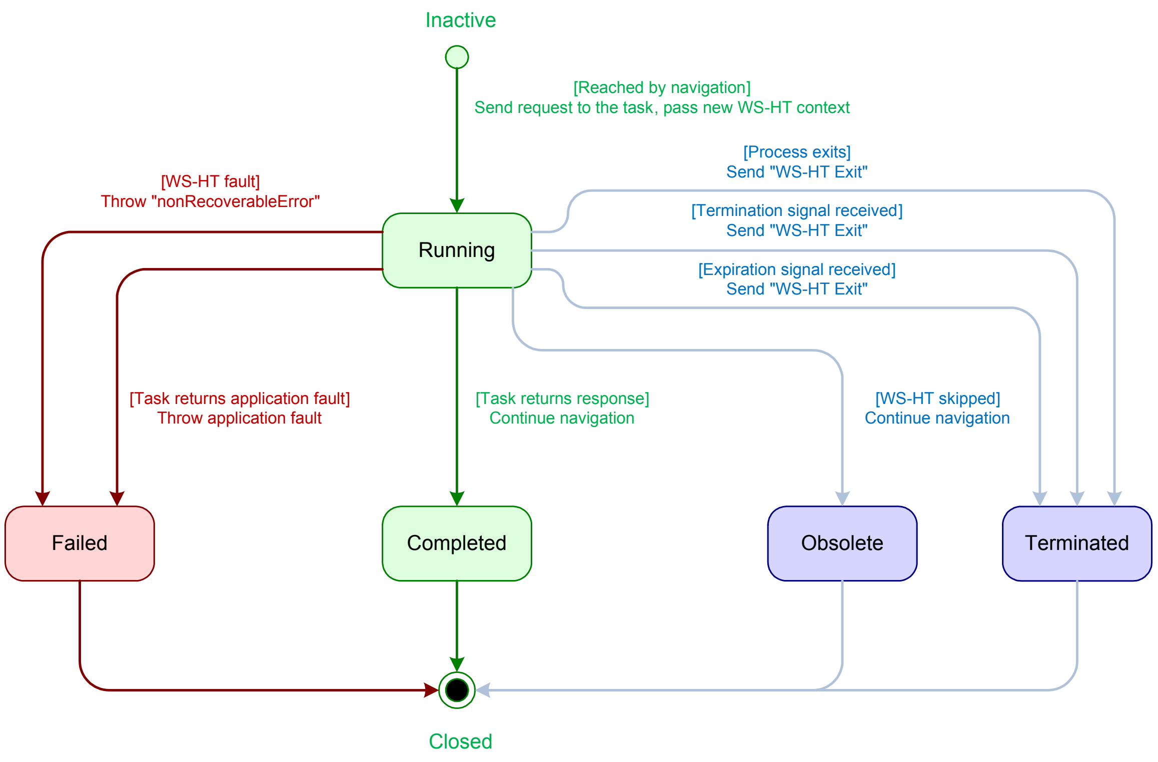


Figure 2: State diagram of the people activity

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

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

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

If the termination of the enclosed scope is triggered while the people activity is still running, the people acitivty is terminated prematurely and the associated running task is exited. When a reponse is received for a terminated people activity it MUST be ignored.

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

4.9 Task Instance Data

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

4.9.1 Presentation Data

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

4.9.2 Context Data

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

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

4.9.3 Operational Data

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

5 XPath Extension Functions

The following XPath extension functions are provided to be used within the definition of a BPEL4People business process to access process context. Because XPath 1.0 functions do not support returning faults, an empty node set is returned in the event of an error.

Operation Name	**Description**	**Parameters**
getProcessStakeholders	Returns the stakeholders of the	Out • organizational entity

	process.	(htd:organizationalEntity)
getBusinessAdministrators	Returns the business administrators of the process.	Out • organizational entity (htd:organizationalEntity)
getProcessInitiator	Returns the initiator of the process.	Out • the process initiator (htd:tUser)
getLogicalPeopleGroup	Returns the value of a logical people group.	In • name of the logical people group (xsd:string) Out • the value of the logical people group (htd:organizationalEntity)
getActualOwner	Returns the actual owner of the task associated with the people activity.	In • people activity name (xsd:string) Out • the actual owner (htd:tUser)
getTaskInitiator	Returns the initiator of the task. Evaluates to an empty htd:user in case there is no initiator.	In • people activity name (xsd:string) Out • the task initiator (user id as htd:user)
getTaskStakeholders	Returns the stakeholders of the task. Evaluates to an empty htd:organizationalE ntity in case of an error.	In • people activity name (xsd:string) Out • task stakeholders (htd:organizationalEntity)
getPotentialOwners	Returns the potential owners of the task associated with the people activity.	In • people activity name (xsd:string) Out • potential owners (htd:organizationalEntity)
getAdministrators	Returns the	In

	administrators of the task associated with the people activity.	• people activity name (xsd:string) Out • business administrators (htd:organizationalEntity)
getTaskPriority	Returns the priority of the task associated with the people activity.	In • people activity name (xsd:string) Out • priority (xsd:nonNegativeInteger)

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

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.

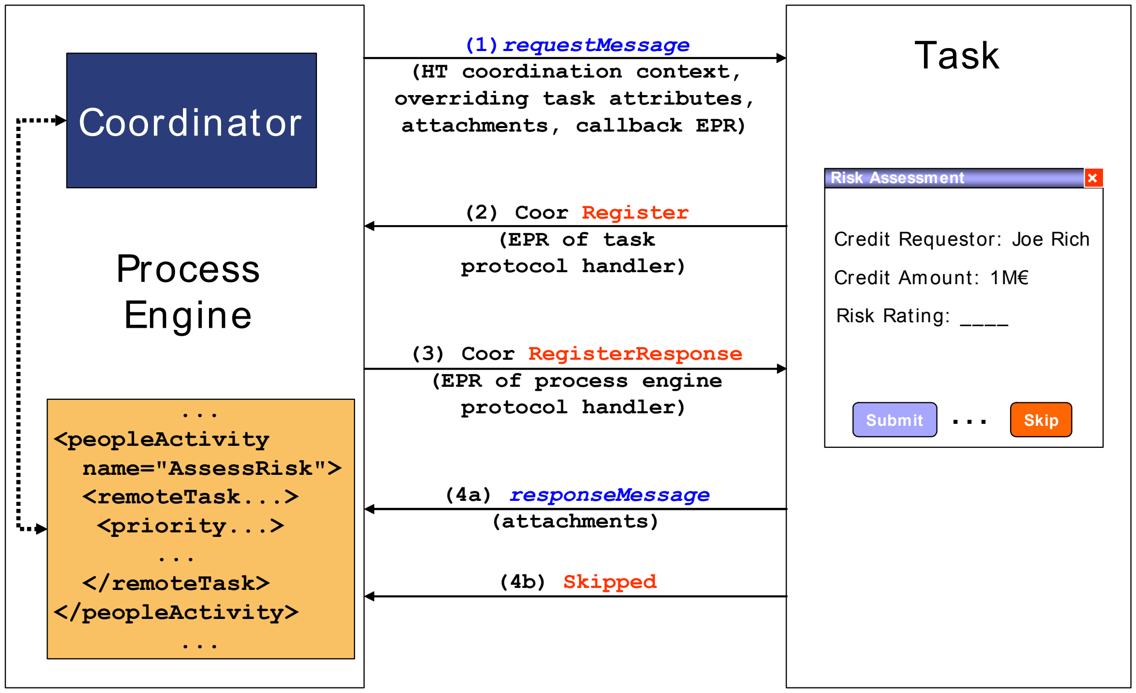


Figure 3: Message exchange between a people activiy and a human task

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 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, a WS-HT coordination context associated with the activity is created. 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 ad-hoc attachments may be propagated 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].

2. When a response message is received from the task that indicates the successful completion of the the task, the people activity completes. This response may include all new ad-hoc attachments from the human task.
3. When a response message is received from the task that indicates a fault of the task, the people activity faults. The fault is thrown in the scope of the people activity.
4. When protocol message `fault` is received, the fault `nonRecoverableError` is thrown in the scope enclosing the people activity.
5. When protocol message skipped is received, the people acitivity moves to state *Obsolete*.
6. If the task does not reach one of the final states by the expiration deadline, the people activity will be terminated. Protocol message `exit` is sent to the task.
7. When the people activity is terminated, protocol message `exit` is sent to the task.
8. When the process encounters an `<exit>` activity, protocol message `exit` is sent to the task.

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

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

7 BPEL Abstract Processes

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

7.1 Hiding Syntactic Elements

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

7.1.1 Opaque Activities

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

7.1.2 Opaque Expressions

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

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

7.1.3 Opaque Attributes

Any attribute introduced by BPEL4People can have an opaque value `"##opaque"` in an abstact process.

7.1.4 Opaque From-Spec

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

7.1.5 Omission

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

7.2 Abstract Process Profile for Observable Behavior

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

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

7.3 Abstract Process Profile for Templates

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

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

8 Acknowledgements

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

9 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

[WS-Addr-SOAP]

Web Services Addressing 1.0 – SOAP Binding, W3C Recommendation, May 2006, available via http://www.w3.org/TR/ws-addr-soap

[WS-Addr-WSDL]

Web Services Addressing 1.0 – WSDL Binding, W3C Working Draft, February 2006, available via http://www.w3.org/TR/ws-addr-wsdl

[WS-BPEL 2.0]

Web Service Business Process Execution Language Version 2.0, OASIS Standard, April 2007, OASIS Technical Committee, available via http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

[WSDL 1.1]

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

[WS-HumanTask]

Published simultaneously with this specification.

[XML Infoset]

XML Information Set, W3C Recommendation, available via http://www.w3.org/TR/2001/REC-xml-infoset-20011024/

[XML Namespaces]

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

[XML Schema Part 1]

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

[XML Schema Part 2]

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

[XMLSpec]

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

[XPATH 1.0]

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

Appendix A – Standard Faults

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

Appendix B – Portability and Interoperability Considerations

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

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

- Interoperability - The capability for multiple components (process engine, task engine and task list client) to interact using well-defined messages and protocols. This enables to combine components from different vendors allowing seamless execution.
  Constellation four achieves interoperability between process and tasks from different vendor implementations.

Constellation 1

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

Constellation 2

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

Constellation 3

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

Constellation 4

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

Appendix C – BPEL4People Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.example.org/BPEL4People"
  xmlns:htd="http://www.example.org/WS-HT"
  xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.example.org/BPEL4People"
  elementFormDefault="qualified" blockDefault="#all">

  <!-- other namespaces -->
  <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd" />
  <xsd:import namespace="http://www.example.org/WS-HT"
    schemaLocation="ws-humantask.xsd" />
  <xsd:import
    namespace="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    schemaLocation="ws-bpel_executable.xsd" />

  <!-- base types for extensible elements -->
  <xsd:complexType name="tExtensibleElements">
    <xsd:sequence>
      <xsd:element name="documentation" type="tDocumentation"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:any namespace="##other" processContents="lax" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other" processContents="lax" />
  </xsd:complexType>
  <xsd:complexType name="tExtensibleMixedNamespaceElements">
    <xsd:sequence>
      <xsd:element name="documentation" type="tDocumentation"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="extensions" type="tExtensions" minOccurs="0" />
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other" processContents="lax" />
  </xsd:complexType>
  <xsd:complexType name="tDocumentation" mixed="true">
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute ref="xml:lang" />
  </xsd:complexType>
  <xsd:complexType name="tExtensions">
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <!-- element "humanInteractions" to be used within "bpel:process" or
"bpel:scope" -->
  <xsd:element name="humanInteractions" type="tHumanInteractions" />
  <xsd:complexType name="tHumanInteractions">
```

```
    <xsd:complexContent>
      <xsd:extension base="tExtensibleMixedNamespaceElements">
        <xsd:sequence>
          <xsd:element ref="htd:logicalPeopleGroups" minOccurs="0" />
          <xsd:element ref="htd:tasks" minOccurs="0" />
          <xsd:element ref="htd:notifications" minOccurs="0" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- element "peopleAssignments" to be used within "bpel:process" or
"bpel:scope" -->
  <xsd:element name="peopleAssignments" type="tPeopleAssignments" />
  <xsd:complexType name="tPeopleAssignments">
    <xsd:complexContent>
      <xsd:extension base="tExtensibleElements">
        <xsd:sequence>
          <xsd:group ref="genericHumanRole" minOccurs="1"
            maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- element "genericHumanRole" within BPEL4People -->
  <xsd:group name="genericHumanRole">
    <xsd:choice>
      <xsd:element ref="processStakeholders" />
      <xsd:element ref="businessAdministrators" />
      <xsd:element ref="processInitiator" />
    </xsd:choice>
  </xsd:group>
  <xsd:element name="processStakeholders"
    type="htd:tGenericHumanRole" />
  <xsd:element name="businessAdministrators"
    type="htd:tGenericHumanRole" />
  <xsd:element name="processInitiator" type="htd:tGenericHumanRole" />

  <!-- element "argument" to be used within "bpel:from" -->
  <xsd:element name="argument" type="tArgument" />
  <xsd:complexType name="tArgument">
    <xsd:complexContent>
      <xsd:extension base="bpel:tExpression">
        <xsd:attribute name="name" type="xsd:NCName" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- attribute "logicalPeopleGroup" to be used within "bpel:from" and
"bpel:to" -->
  <xsd:attribute name="logicalPeopleGroup" type="xsd:NCName" />

  <!-- element "peopleActivity" to be used within
"bpel:extensionActivity" -->
  <xsd:element name="peopleActivity" type="tPeopleActivity" />
  <xsd:complexType name="tPeopleActivity">
  <xsd:complexContent>
    <xsd:extension base="tExtensibleMixedNamespaceElements">
      <xsd:sequence>
        <xsd:element ref="bpel:targets" minOccurs="0" />
        <xsd:element ref="bpel:sources" minOccurs="0" />
        <xsd:choice>
          <xsd:element ref="htd:task" />
          <xsd:element ref="localTask" />
          <xsd:element ref="remoteTask" />
          <xsd:element ref="htd:notification" />
          <xsd:element ref="localNotification" />
          <xsd:element ref="remoteNotification" />
        </xsd:choice>
        <xsd:element ref="scheduledActions" minOccurs="0" />
        <xsd:element ref="toParts" minOccurs="0" />
        <xsd:element ref="fromParts" minOccurs="0" />
        <xsd:element ref="attachmentPropagation" minOccurs="0" />
        <xsd:any namespace="##other" processContents="lax"
          minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NCName" />
      <xsd:attribute name="suppressJoinFailure" type="tBoolean"
        use="optional" />
      <xsd:attribute name="inputVariable" type="xsd:QName" />
      <xsd:attribute name="outputVariable" type="xsd:QName" />
      <xsd:attribute name="isSkipable" type="tBoolean"
        use="optional" default="no" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="tOverridableTaskElements">
  <xsd:complexContent>
    <xsd:extension base="tExtensibleMixedNamespaceElements">
      <xsd:sequence>
        <xsd:element ref="htd:priority" minOccurs="0" />
        <xsd:element ref="htd:peopleAssignments" minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="localTask" type="tLocalTask" />
<xsd:complexType name="tLocalTask">
  <xsd:complexContent>
    <xsd:extension base="tOverridableTaskElements">
      <xsd:attribute name="reference" type="xsd:QName"
        use="required" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="remoteTask" type="tRemoteTask" />
<xsd:complexType name="tRemoteTask">
  <xsd:complexContent>
    <xsd:extension base="tOverridableTaskElements">
      <xsd:attribute name="partnerLink" type="xsd:NCName"
        use="required" />
      <xsd:attribute name="operation" type="xsd:NCName"
        use="required" />
        <xsd:attribute name="responseOperation" type="xsd:NCName" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="tOverridableNotificationElements">
    <xsd:complexContent>
      <xsd:extension base="tExtensibleMixedNamespaceElements">
        <xsd:sequence>
          <xsd:element ref="htd:priority" minOccurs="0" />
          <xsd:element ref="htd:peopleAssignments" minOccurs="0" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="localNotification" type="tLocalNotification" />
  <xsd:complexType name="tLocalNotification">
    <xsd:complexContent>
      <xsd:extension base="tOverridableNotificationElements">
        <xsd:attribute name="reference" type="xsd:QName"
          use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="remoteNotification" type="tRemoteNotification" />
  <xsd:complexType name="tRemoteNotification">
    <xsd:complexContent>
      <xsd:extension base="tOverridableNotificationElements">
        <xsd:attribute name="partnerLink" type="xsd:NCName"
          use="required" />
        <xsd:attribute name="operation" type="xsd:NCName"
          use="required" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="scheduledActions" type="tScheduledActions" />
  <xsd:complexType name="tScheduledActions">
    <xsd:complexContent>
      <xsd:extension base="tExtensibleElements">
        <xsd:sequence>
          <xsd:element name="deferActivation"
            type="tScheduledActionsDetails" minOccurs="0" />
          <xsd:element name="expiration"
            type="tScheduledActionsDetails" minOccurs="0" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="tScheduledActionsDetails">
    <xsd:complexContent>
      <xsd:extension base="tExtensibleElements">
        <xsd:sequence>
          <xsd:choice>
            <xsd:element name="for" type="bpel:tDuration-expr" />
            <xsd:element name="until" type="bpel:tDeadline-expr" />
          </xsd:choice>
        </xsd:sequence>
      </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="fromParts" type="tFromParts" />
<xsd:complexType name="tFromParts">
  <xsd:complexContent>
    <xsd:extension base="tExtensibleElements">
      <xsd:sequence>
        <xsd:element ref="fromPart" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="fromPart" type="tFromPart" />
<xsd:complexType name="tFromPart">
  <xsd:complexContent>
    <xsd:extension base="tExtensibleElements">
      <xsd:attribute name="part" type="xsd:NCName" use="required" />
      <xsd:attribute name="toVariable" type="bpel:BPELVariableName"
        use="required" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="toParts" type="tToParts" />
<xsd:complexType name="tToParts">
  <xsd:complexContent>
    <xsd:extension base="tExtensibleElements">
      <xsd:sequence>
        <xsd:element ref="toPart" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="toPart" type="tToPart" />
<xsd:complexType name="tToPart">
  <xsd:complexContent>
    <xsd:extension base="tExtensibleElements">
      <xsd:attribute name="part" type="xsd:NCName" use="required" />
      <xsd:attribute name="fromVariable"
        type="bpel:BPELVariableName" use="required" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="attachmentPropagation"
  type="tAttachmentPropagation" />
<xsd:complexType name="tAttachmentPropagation">
  <xsd:complexContent>
    <xsd:extension base="tExtensibleElements">
      <xsd:attribute name="fromProcess" type="tFromProcess"
        default="all" />
      <xsd:attribute name="toProcess" type="tToProcess"
        default="newOnly" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="tFromProcess">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="all" />
```

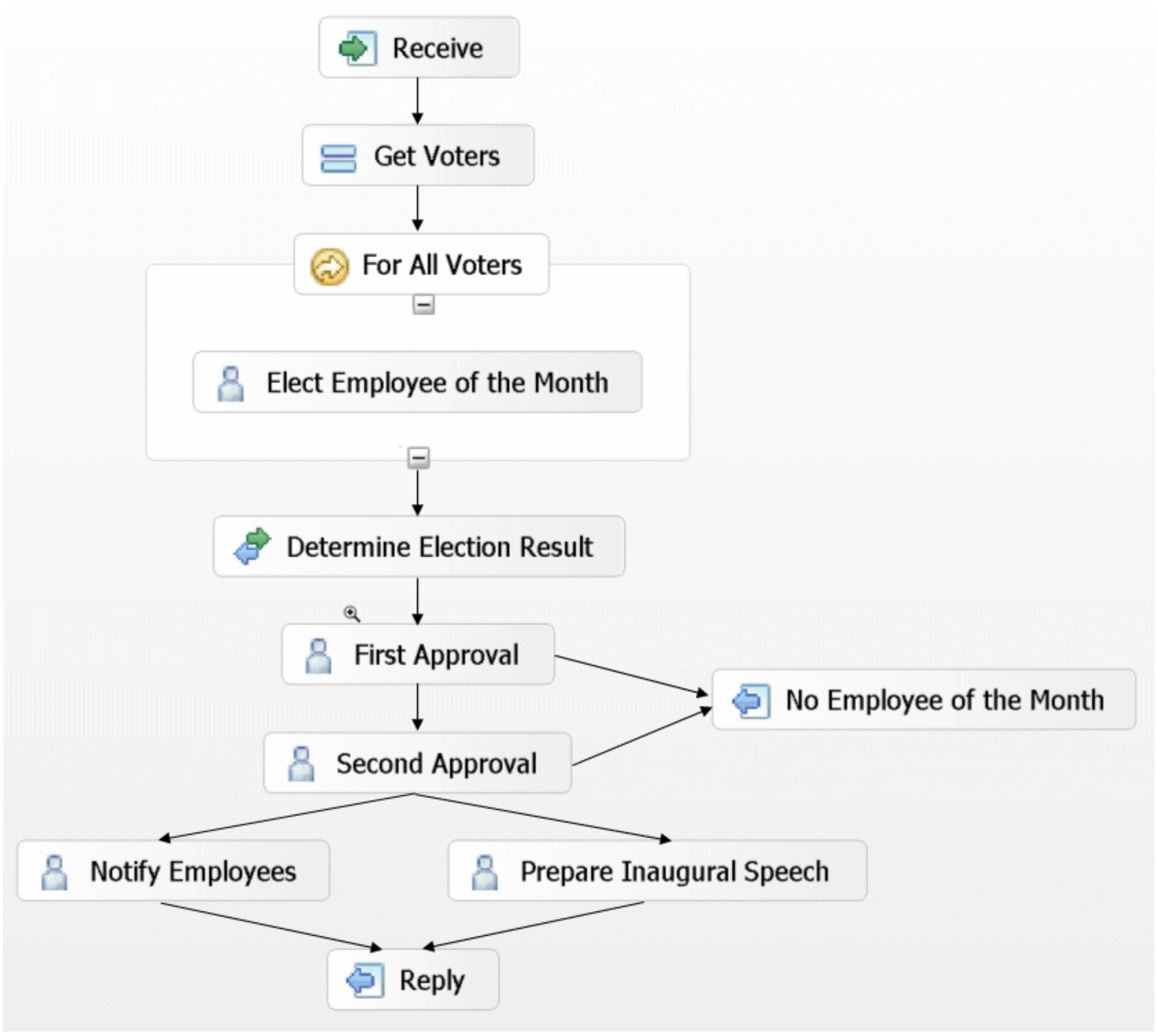
```
      <xsd:enumeration value="none" />
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="tToProcess">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="all" />
      <xsd:enumeration value="newOnly" />
      <xsd:enumeration value="none" />
    </xsd:restriction>
  </xsd:simpleType>

  <!-- miscellaneous helper elements and types -->
  <xsd:simpleType name="tBoolean">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="yes" />
      <xsd:enumeration value="no" />
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

Appendix D – Sample

This appendix contains a sample that outlines the basic concepts of this specification. The sample process implements the election of the "Employee of the month" in a fictious company. The structure of the business process is shown in the figure below:



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

The sections below show the definition of the BPEL process implementing the "Employee of the month" process.

BPEL Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<process
  name="EmployeeOfTheMonthProcess"
  targetNamespace="http://www.example.com"
  xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
  xmlns:b4p="http://www.example.org/BPEL4People"
  xmlns:htd="http://www.example.org/WS-HT"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:tns="http://www.example.com"
  xmlns:hr="http://www.example.com/approval"
  xmlns:el="http://www.example.com/election"
  xmlns:ty="http://www.example.com/types"
  xmlns:ta="http://www.example.com/tasks"
  xsi:schemaLocation="http://www.example.org/BPEL4People
                      ws-bpel4people.xsd">

  <b4p:humanInteractions>

    <htd:logicalPeopleGroups>

      <htd:logicalPeopleGroup name="voters">
        <htd:documentation xml:lang="en-US">
          The group entitled to vote the employee of the month for the
          given region.
        </htd:documentation>
        <htd:parameter name="region" type="xsd:string" />
      </htd:logicalPeopleGroup>

      <htd:logicalPeopleGroup name="approvers">
        <htd:documentation xml:lang="en-US">
          The group entitled to approve the elected employee of the
          month for the given region.
        </htd:documentation>
        <htd:parameter name="region" type="xsd:string" />
      </htd:logicalPeopleGroup>

      <htd:logicalPeopleGroup name="employees">
        <htd:documentation xml:lang="en-US">
          The group of employees to be notified about the election
          result of the employee of the month election for the given
          region.
        </htd:documentation>
        <htd:parameter name="region" type="xsd:string" />
      </htd:logicalPeopleGroup>

      <htd:logicalPeopleGroup name="regionalElectionCommittee">
        <htd:documentation xml:lang="en-US">
          The group who is in charge for the election of the
          employee of the month election for the given region.
        </htd:documentation>
        <htd:parameter name="region" type="xsd:string" />
      </htd:logicalPeopleGroup>
    </htd:logicalPeopleGroups>

    <htd:tasks>
      <htd:task name="approveEmployeeOfTheMonth">
        <htd:documentation xml:lang="en-US">
           The reusable definition of the task used to approve the
           election of the employee of the month.
        </htd:documentation>
              <htd:interface operation="approve"
                             portType="hr:approvalPT"/>
        <htd:peopleAssignments>
          <htd:potentialOwners>
            <htd:from logicalPeopleGroup="approvers">
              <!-- variables used here need to be defined on the
                   enclosing scope or above -->
              <htd:argument name="region">
                $electionRequest/region
              </htd:argument>
            </htd:from>
          </htd:potentialOwners>
        </htd:peopleAssignments>
              <htd:presentationElements/>
      </htd:task>
    </htd:tasks>

  </b4p:humanInteractions>

  <b4p:peopleAssignments>

    <b4p:processStakeholders>
      <htd:from logicalPeopleGroup="regionalElectionCommittee">
        <htd:argument name="region">
          $electionRequest/region
        </htd:argument>
      </htd:from>
    </b4p:processStakeholders>

    <b4p:businessAdministrators>
      <htd:from>
        <htd:literal>
          <htd:entity xsi:type="htd:organizationalEntity">
            <htd:users>
              <htd:user>Peter</htd:user>
              <htd:user>Paul</htd:user>
              <htd:user>Mary</htd:user>
            </htd:users>
          </htd:entity>
        </htd:literal>
      </htd:from>
    </b4p:businessAdministrators>

  </b4p:peopleAssignments>

  <extensions>
    <extension
      namespace="http://www.example.org/BPEL4People"
      mustUnderstand="yes"/>
    <extension
      namespace="http://www.example.org/WS-HT"
      mustUnderstand="yes"/>
  </extensions>

  <import
    importType="http://www.w3.org/2001/XMLSchema"
    namespace="http://www.example.com/types"/>
  <import
    importType="http://www.example.org/WS-HT"
    namespace="http://www.example.com/tasks"/>
  <import
    importType="http://schemas.xmlsoap.org/wsdl/"
    namespace="http://www.example.com/election"
    location="election.wsdl"/>
  <import
    importType="http://schemas.xmlsoap.org/wsdl/"
    namespace="http://www.example.com/approval"
    location="approval.wsdl"/>

  <partnerLinks>
    <partnerLink partnerLinkType="electionPLT"
                 name="electionPL"/>
  </partnerLinks>

  <variables>
    <variable name="candidates" type="htd:users"/>
    <variable name="voters" type="htd:tOrganizationalEntity"/>
    <variable name="electionRequest" type="ty:electionRequestData"/>
    <variable name="electionResult" type="ty:electionResultData"/>
    <variable name="decision" type="xsd:boolean"/>
    <variable name="speech" type="ty:document"/>
  </variables>

  <sequence>
    <receive partnerLink="electionPL"
             portType="el:electionPT"
             operation="elect"
             variable="electionRequest"
             createInstance="yes"/>

    <assign name="getVoters">
      <copy>
        <from>$electionRequests/candidates</from>
        <to variable="candidates"/>
      </copy>
      <copy>
        <from b4p:logicalPeopleGroup="voters">
          <b4p:argument name="region">
            $electionRequest/region
          </b4p:argument>
        </from>
        <to variable="voters" />
      </copy>
    </assign>

    <forEach counterName="i" parallel="yes">
      <startCounterValue> 1 </startCounterValue>
      <finalCounterValue>
        count($voters/users/user)
      </finalCounterValue>

      <scope>
        <variables>
           <variable name="vote" type="htd:user"/>
        </variables>

        <sequence>
          <!-- Scenario 1 -->
          <extensionActivity>
            <b4p:peopleActivity name="electEmployeeOfTheMonth"
                                inputVariable="candidates"
                                outputVariable="vote"
                                isSkipable="yes">
              <htd:task name="votingTask">
                <htd:interface operation="vote"
                               portType="el:votingPT"/>
                <htd:peopleAssignments>
                  <htd:potentialOwners>
                    <htd:from>$voters/users/user[i]</htd:from>
                  </htd:potentialOwners>
                </htd:peopleAssignments>
                <htd:presentationElements/>
              </htd:task>
              <b4p:scheduledActions>
                <b4p:expiration>
                  <b4p:documentation xml:lang="en-US">
                    This people activity expires when not completed
                    within 2 days after having been activated.
                  </b4p:documentation>
                  <b4p:for>P2D</b4p:for>
                </b4p:expiration>
              </b4p:scheduledActions>
            </b4p:peopleActivity>
          </extensionActivity>

          <assign>
            <copy>
              <from>$vote</from>
              <to>$electionResult/votes[i]</to>
            </copy>
          </assign>

        </sequence>
      </scope>
    </forEach>

    <!-- Might be scenario 5 ... -->
    <invoke name="determineElectionResult"
            partnerLink=""
            portType=""
            operation=""
            inputVariable=""
            outputVariable=""/>
```

```
<!-- Scenario 2 -->
<extensionActivity>
  <b4p:peopleActivity name="firstApproval"
                      inputVariable="electionResult"
                      outputVariable="decision">
    <b4p:localTask reference="tns:approveEmployeeOfTheMonth"/>
  </b4p:peopleActivity>
</extensionActivity>

<!-- Scenario 2 with override specifications -->
<extensionActivity>
  <b4p:peopleActivity name="secondApproval"
                      inputVariable="electionResult"
                      outputVariable="decision">
    <b4p:localTask reference="tns:approveEmployeeOfTheMonth">
      <htd:peopleAssignments>
        <htd:excludedOwners>
          <htd:from>
            b4p:getActualOwner("tns:firstApproval")
          </htd:from>
        </htd:excludedOwners>
      </htd:peopleAssignments>
    </b4p:localTask>
  </b4p:peopleActivity>
</extensionActivity>

<!-- Scenario 3 -->
<extensionActivity>
  <b4p:peopleActivity name="notifyEmployees"
                      inputVariable="electionResult">
    <b4p:localNotification reference="task:employeeBroadcast"/>
    <!-- notification is not defined as part of this document,
         but within a separate one
    -->
  </b4p:peopleActivity>
</extensionActivity>

<!-- Scenario 4 -->
<extensionActivity>
  <b4p:peopleActivity name="prepareInauguralSpeech"
                      inputVariable="electionResult"
                      outputVariable="speech"
                      isSkipable="no">
    <b4p:remoteTask partnerLink="author"
                    operation="prepareSpeech"
                    responseOperation="receiveSpeech">
      <htd:priority>0</htd:priority> <!-- assign highest prio -->
      <htd:peopleAssignments>
          <htd:potentialOwners>
            <htd:from>$electionResult/winner</htd:from>
          </htd:potentialOwners>
      </htd:peopleAssignments>
    </b4p:remoteTask>
  </b4p:peopleActivity>
</extensionActivity>
```

```
  </sequence>

</process>
```

WSDL Definitions

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
  xmlns:tns="http://www.example.com/election"
  targetNamespace="http://www.example.com/election">

  <plnk:partnerLinkType name="electionPLT">
    <plnk:role name="electionService" portType="tns:electionPT" />
  </plnk:partnerLinkType>

  <wsdl:message name="electionInput">
    <wsdl:part name="parameters" element="xsd:string" />
  </wsdl:message>
  <wsdl:message name="votingInput">
    <wsdl:part name="parameters" element="xsd:string" />
  </wsdl:message>

  <wsdl:portType name="electionPT">
    <wsdl:operation name="elect">
      <wsdl:input message="tns:electionInput" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="votingPT">
    <wsdl:operation name="vote">
      <wsdl:input message="tns:votingInput" />
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>

<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://www.example.com/approval"
  targetNamespace="http://www.example.com/approval">

  <!-- Messages -->
  <wsdl:message name="approvalInput">
    <wsdl:part name="parameters" element="xsd:string" />
  </wsdl:message>
  <wsdl:message name="approvalOutput">
    <wsdl:part name="parameters" element="xsd:string" />
  </wsdl:message>

  <!-- Mandatory Asynchronous Messaging PortTypes -->
  <wsdl:portType name="approvalPT">
    <wsdl:operation name="approve">
      <wsdl:input message="tns:approvalInput" />
      <wsdl:output message="tns:approvalOutput" />
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```