The Workflow Management Coalition Specification

Workflow Management Coalition Workflow Standard

Process Definition Interface -- XML Process Definition Language

Document Number WFMC-TC-1025
Document Status – Final

October 3, 2005
Version 2.00

Workflow Management Coalition
2436 N. Federal Highway #374
Lighthouse Point, Fl 33064
USA

Tel: +1 954 782 3376
Fax: +1 954 782 6365
Email: wfmc@wfmc.org
WWW: http://www.wfmc.org

Table of Content

1. CHANGE HISTORY 4

1.1. ACKNOWLEDGEMENTS 6

2. AUDIENCE 6

3. PURPOSE 6

4. INTRODUCTION 6

4.1. CONFORMANCE 7
4.2. XPDL VERSION 1.0 COMPATIBILITY 7
4.3. REFERENCES 8

5. OVERVIEW OF PROCESS DEFINITION INTERCHANGE 8

5.1. APPROACHES TO PROCESS DEFINITION INTERCHANGE 9

6. META-MODEL 10

6.1. PROCESSES AND PACKAGES 10
6.2. PACKAGE META-MODEL 11
6.2.1. Process Repository 12
6.2.1.1. Redefinition and Scope 12
6.3. PROCESS META-MODEL 13
6.4. ENTITIES OVERVIEW 14
6.4.1. Swimlanes 14
6.4.1.1. Pool 14
6.4.1.2. Lane 14
6.4.2. Process Definition 14
6.4.3. Process Activity 14
6.4.4. Transition Information 15
6.4.5. Participant Declaration 15
6.4.6. Application Declaration 15
6.4.7. Artifact 15
6.4.8. Message Flow 16
6.4.9. Association 16
6.4.10. Relevant data field 16
6.4.11. Data Types and Expressions 16
6.4.12. System and Environmental Data 16
6.4.13. Resource Repository 16
6.4.14. Vendor or User specific Extensions 16
6.4.14.1. Extended Elements and Attributes 16
6.4.14.2. Extended parameter mapping 17

7. XML PROCESS DEFINITION LANGUAGE 18

7.1. ELEMENTS COMMON FOR MULTIPLE ENTITIES 18
7.1.1. Graphic Information 18
7.1.1.1. NodeGraphicsInfo 18
7.1.1.2. ConnectorGraphicsInfo 19
7.1.2. Extended Attributes 20
7.1.2.1. Anonymous Extended Attribute 20
7.1.2.2. Namespace Qualified Extensions 21
7.1.3. Formal Parameters 21
7.1.3.1. Parameter passing semantics 22
7.1.3.2. Concurrency semantics 22
7.1.3.3. Formal-actual parameter mapping 23
7.1.4. External Reference 23
7.1.4.1. Web Services 23

7.1.5. Assignment ... *24*
7.1.6. Category ... *25*
7.1.7. Artifact ... *25*
7.1.7.1. Object ... 26
7.1.7.2. DataObject ... 27
7.2. PACKAGE DEFINITION ... 27
7.2.1. Package definition Header ... *29*
7.2.1.1. Vendor extensions ... 30
7.2.2. Redefinable Header ... *31*
7.2.3. Conformance Class Declaration ... *33*
7.2.4. Script ... *33*
7.2.5. External Package ... *34*
7.3. APPLICATION DECLARATION ... 35
7.3.1. Invocation Parameters ... *36*
7.3.2. Application Types ... *36*
7.3.2.1. EJB ... 36
7.3.2.2. Pojo ... 37
7.3.2.3. XSLT ... 37
7.3.2.4. Script ... 38
7.3.2.5. WebService ... 38
7.3.2.6. BusinessRule ... 39
7.3.2.7. Form ... 39
7.4. SWIMLANES ... 40
7.4.1. Pool ... *40*
7.4.2. Lane ... *41*
7.5. PROCESS DEFINITION ... 42
7.5.1. Process Definition Header ... *45*
7.5.2. Process Redefinable Header ... *48*
7.5.3. Activity Set/Embedded SubProcess ... *49*
7.5.4. ProcessType in BPMN mapping to WS-BPEL ... *50*
7.5.5. Status ... *50*
7.5.6. SuppressJoinFailure ... *51*
7.5.7. EnableInstanceCompensation ... *51*
7.5.8. AdHoc ... *51*
7.6. PROCESS ACTIVITY ... 51
7.6.1. Execution Control Attributes ... *56*
7.6.2. Route Activity ... *56*
7.6.2.1. Gateway Activity ... 57
7.6.2.2. Examples of Gateways and their Representation ... 58
7.6.3. Block Activity ... *59*
7.6.4. Event Activity ... *59*
7.6.4.1. Start Event ... 60
7.6.4.2. Intermediate Event ... 61
7.6.4.3. End Event ... 63
7.6.4.4. Common Elements Used in Start, Intermediate and End Events ... 64
7.6.4.5. Examples of Events and their representation ... 68
7.6.5. Implementation Alternatives ... *69*
7.6.5.1. No Implementation ... 69
7.6.5.2. Tool ... 70
7.6.5.3. Task ... 70
7.6.5.4. SubFlow/ProcessRef ... 74
7.6.5.5. Reference ... 76
7.6.6. Performer Relationship ... *77*
7.6.7. Deadline ... *77*
7.6.8. Simulation Information ... *79*
7.6.9. Transition Restriction ... *80*
7.6.9.1. Join ... 81
7.6.9.2. Split ... 81
7.6.10. Conformance Classes ... *84*
7.6.11. InputSets ... *84*

7.6.12. OutputSets ... *85*
7.6.13. Transaction ... *86*
7.6.14. Loop ... *87*
7.7. TRANSITION INFORMATION ... 89
7.7.1. Condition ... *90*
7.7.1.1. Exception Conditions ... 91
7.8. PARTNER LINKS ... 92
7.8.1. Partner Link Type ... *92*
7.8.2. Partner Link ... *93*
7.9. MESSAGING ... 94
7.9.1. Message Flow ... *94*
7.9.2. Message Type ... *95*
7.9.3. End Point ... *96*
7.9.4. Web ServiceOperation ... *96*
7.9.5. Web Service Fault Catch ... *97*
7.10. ASSOCIATION ... 98
7.11. PARTICIPANTS ... 99
7.11.1. Participant Entity Types ... *100*
7.12. RELEVANT DATA FIELD ... 101
7.13. DATA TYPES ... 102
7.13.1. Basic Data Types ... *103*
7.13.2. Complex Data Types ... *103*
7.13.2.1. Schema Type ... 104
7.13.2.2. Record Type ... 104
7.13.2.3. Union Type ... 104
7.13.2.4. Enumeration Type ... 105
7.13.2.5. Array Type ... 105
7.13.2.6. List Type ... 106
7.13.3. Declared Data Types ... *106*
7.13.3.1. Type Declaration ... 106
7.13.3.2. Declared Type ... 107

8. SAMPLES ... 109

8.1. SAMPLE PROCESS ... 109
8.1.1. The Processes ... *109*
8.1.1.1. The EOrder Main Process ... 109
8.1.1.2. The CreditCheck Subprocess ... 110
8.1.1.3. The FillOrder Subprocess ... 110
8.1.2. Type Declarations ... *110*
8.1.3. ExtendedAttributes ... *112*
8.1.4. External References ... *113*
8.1.5. Sample XPDL ... *113*
8.2. EXTENDING XPDL SCHEMA ... 128
8.2.1. Xyz Schema ... *128*
8.2.2. Xyz Test XPDL Document ... *129*

9. XPDL SCHEMA ... 130

10. FIGURES AND TABLES ... 160

10.1. FIGURES ... 160
10.2. TABLES ... 160

1. Change History

Version 1.14 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Backed out schema change that deprecated SubFlow element. SubFlow is the name of the element.
- Fixed minor errors in specification.

Version 1.13 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Updated the Specification to include discussion of new properties that determine starting ActivitySet and Activity.
- Fixed minor errors in schema. (NodeGraphicsInfos, ConnectorGraphicsInfos, starting ActivitySet and Activity properties.

Version 1.12 – Editor: Mike Marin (mmarin@filenet.com)

- Worked on the web services area and introduced PartnerLink, PartnerLinkType, EndPoint, and Catch.
- Implemented a new extensibility model and reverted extensible attributes (anonymous extensions) to the XPDL 1.0 model.
- Introduced NodeGraphicInfos and ConnectorGraphInfos.
- Added an example of extending the XPDL schema.

Version 1.11 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Added Discussion and attribute tables for the seven application types. (Contributed by TIBCO).

Version 1.10 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Added the Application Type and seven elements under Type..
- Changed DiagramRef to PackageRef.
- Both Tool and SubFlow now have PackageRef.
- Both Tool and SubFlow now have a choice of ActualParameters or DataMappings.
- Tool is now under Task, and it was renamed to TaskApplication.
- SubFlow was deprecated and renamed to ProcessRef in XPDL 2.0 (Note that TriggerResultLink has an attribute called also ProcessRef, so this may introduce some confusion).
- PoolId and LaneId were removed from Activity and Artifact.
- LaneId was added to NodeGraphicsInfo.
- NodeGraphicsInfo now has Extended Attributes.
- StartMode/FinishMode moved to activity attributes.
- Metamodels updated to reflect changes (e.g. Tool now under task)
- Removed initial 't' from all schema type definitions
- Added Length, Precision andScale Attributes to BasicType element.

Version 1.09 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Minor Typos fixed.
- toolId attribute replaced by ToolId

- Target attribute type changed to NMTOKEN
- All schema inserts updated for new approach to Deprecation using name spaces.

Version 1.08 – Editor: Mike Marin (mmarin@filenet.com)

- Enhanced document and schema with new extended attributes model.
- Schema types are now prefixed with 't'.
- Changed ExternalPackage id to Id to be compatible with all other Ids.
- Normalized Name and Id attributes for all entities that require them.
- Changed Assignment to have Target and Expression entities.
- Changed TaskScript to be any arbitrary expression.
- Introduced a ExpressionType type and used for all expressions
- Updated all the sections with schema
- Added a section on XPDL version 1.0 compatibility

Version 1.07 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Includes changes based on XPDL 1.0 requests
- Formal parameter index deprecated.
- AccessLevel attribute of Process made optional with default PUBLIC.
- Additional attribute for SubFlow provides name of datafield in which to store instantiation id.
- Execution attribute for SubFlow is optional with default SYNCHR.
- GraphConformance attribute for Conformance class is optional with default NON_BLOCKED.
- Transition condition may have at most one child element expression.
- Tool attribute 'type' deprecated: no longer needed since there are no constructs for declaring Procedures or any information about their formal parameters or parameter passing. Hence all Tools should be Applications.
- DeadlineCondition deprecated. DeadlineDuration element introduced with type = string.
- . Example redone using BPMN notation

Version 1.06 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Includes all editing changes from Justin Brunt (JBrunt@staffware.com).

Version 1.05 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Minor Changes to Schema: ProcessType – order of elements to preserve compatibility with XPDL 1.0
- Minor typos.
- Instantiate attribute removed from TaskSend and TaskService. TaskSend Message documentation fixed.
- Explanation of OTHERWISE clause changed for SPLITS. Corrected documentation on Splits and Joins to reflect BPMN extensions. Added documentation in Route Activity section.
- Package reference extended to include **type declarations**.
- Added attribute to External Package Reference.
- Added reference to external packages for Participant identifiers.

Version 1.04 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Minor Changes to Schema: order of elements and some elements made optional.
- Examples of Gateways and Events

Version 1.03 – Editor: Robert Shapiro (rshapiro@capevisions.com)

- Inserted text explanations of BPMN constructs used in meta-models.
- Updated individual schema sections and tables, adding sections for all elements

Version 1.02 – Editor: Mike Marin (mmarin@filenet.com)

- Reorganized document and added new meta-models

Version 1.01 – Editors: Mike Marin (mmarin@filenet.com) and Robert Shapiro (rshapiro@capevisions.com)

- Initial draft of bidirectional mapping to BPMN 1.0

Version 1.0 – Editor: Roberta Norin (robertan@attbi.com).

1.1. Acknowledgements

XPDL2.0 required many hours of work by individuals who had to find time to contribute while carrying out their normal duties for the companies that employ them.

Robert Shapiro (Global 360) and Mike Marin (FileNET) did the bulk of the work.

Justin Brunt, Wojciech Zurek, Tim Stephenson (Staffware/TIBCO) , Sasa Bojanic (Prozone)and Gangadhar Gouri (Fujitsu Software) made significant contributions.

Keith Swensen (Fujitsu Software) provided invaluable organizational support and encouragement.

2. Audience

The intended audience for this document is primarily vendor organizations who seek to implement the XML Process Definition Language (XPDL) of the Workflow Management Coalition (WfMC), or using it as a file format for the Business Process Modeling Notation (BPMN) of the Business Process Management Initiative (BPMI). It may also be of interest to those seeking to assess conformance claims made by vendors for their products. Comments should be addressed to the Workflow Management Coalition.

3. Purpose

XPDL version 2.0 is back compatible with XPDL version 1.0, and is intended to be used as a file format for BPMN. The original purpose of XPDL is maintained and enhanced by this second version of the specification. The XPDL and the BPMN specifications address the same modeling problem from different perspectives. XPDL provides an XML file format that can be used to interchange process models between tools. BPMN provides a graphical notation to facilitate human communication between business users and technical users, of complex business processes.

There are a number of elements that are present in BPMN version 1.0 that were not present in XPDL version 1.0. Those had been incorporated into this version of XPDL.

The WfMC has identified five functional interfaces to a process or workflow service as part of its standardization program. This specification forms part of the documentation relating to “Interface one” - supporting Process Definition Import and Export. This interface includes a common meta-model for describing the process definition (this specification) and also a companion XML schema for the interchange of process definitions.

4. Introduction

A variety of different tools may be used to analyse, model, describe and document a business process. The process definition interface defines a common interchange format, which supports the transfer of process definitions between separate products.

The interface also defines a formal separation between the development and run-time environments, enabling a process definition, generated by one modelling tool, to be used as input to a number of different run-time products.

A process definition, generated by a build-time tool, is capable of interpretation in different run-time products. Process definitions transferred between these products or stored in a separate repository are accessible via the XPDL common interchange format.

To provide a common method to access and describe process definitions, a process definition meta-data model has been established. This meta-data model identifies commonly used entities within a process definition. A variety of attributes describe the characteristics of this limited set of entities. Based on this model, vendor specific tools can transfer models via a common exchange format.

One of the key elements of XPDL is its extensibility to handle information used by a variety of different tools. XPDL may never be capable of supporting all additional information requirements in all tools. Based upon a limited number of entities that describe a process definition (the "Minimum Meta Model"), XPDL supports a number of differing approaches.

One of the most important elements of XPDL is a generic construct that supports vendor specific attributes for use within the common representation. We recommend that any missing attributes be proposed to the WfMC interface one workgroup for inclusion in future releases.

This document describes the meta-model, which is used to define the objects and attributes contained within a process definition. The XPDL grammar is directly related to these objects and attributes. This approach needs two operations to be provided by a vendor:

- Import a process definition from XPDL.
- Export a process definition from the vendor's internal representation to XPDL.

A vendor can use an XSL style sheet to accomplish these two operations.

All keywords and terms used within this specification are based upon the WfMC Glossary, or terminology used by BPMN.

For the purpose of this document, the terms process definition, business process model, and workflow model are all considered to represent the same concept, and therefore, they are used interchangeably.

4.1. Conformance

A vendor can not claim conformance to this or any other WfMC specification unless specifically authorised to make that claim by the WfMC. WfMC grants this permission only upon the verification of the particular vendor's implementation of the published specification, according to applicable test procedures defined by WfMC.

Conformance for process definition import / export is essentially based upon conformance to the XPDL grammar. However, there is a mandatory minimum set of objects, as specified within this document, which must be supported within XPDL. But, given the wide variation of capabilities in modelling tools, it is reasonable to assume that an individual tool might conform to this specification but not be able to swap complete definitions with all other conforming products. A product that claims conformance must generate valid, syntactically correct XPDL, and must be able to read valid XPDL files. A valid, syntactically correct XPDL file, must conform and validate against the XPDL schema.

4.2. XPDL Version 1.0 Compatibility

XPDL version 2.0 is compatible with XPDL version 1.0, with minor exceptions. The XPDL schema version 2.0 has a different namespace, and tools wishing to be compatible with XPDL version 1.0 need to understand both XPDL 1.0 and XPDL 2.0 namespaces.

The following XPDL version 1.0 elements have been deprecated in version 2.0:

- Automatic element. Replaced by StartMode and FinishMode attributes of Activity.
- BlockId attribute of BlockActivity element. Replaced by ActivitySetId.
- DeadlineCondition element. Replaced with DeadlineDuration.
- Index attribute in FormalParameter element. Because, FormalParameters must match the order in the

declaration, and so there is no need for Index.

- Manual element. Replaced by StartMode and FinishMode attributes of Activity.
- Type attribute of Tool element. Because tools are applications and so this attribute is not necessary. (Tools cannot be procedures - from WPDL - because XPDL does not provide any way of defining the formal parameters for procedures.) Note also that Tool is a type of Task: TaskAplication.
- Xpression element in Condition element. Replaced by Expression.
- The order in a WorkflowProcess changed from DataFields, Participants, and Applications to be Participants, Applications, and DataFields. This makes the order in Process and package consistent.

4.3. References

The following documents are associated with this document and should be used as a reference.

General background information:

WfMC, Terminology & Glossary (WfMC-TC-1011)

WfMC, Reference Model (WfMC-TC-1003)

WfMC, API specifications, which include process definition manipulation APIs:

WfMC, Client Application API Specifications (WAPI) (WfMC-TC-1009)

WfMC, Process Definition Interchange – Process Model (WfMC-TC-1016-P)

BPMI, Business Process Modeling Notation (BPMN), version 1.0 – May 3, 2004

Process interoperability, used to support process invocation on a remote workflow service:

Workflow Interoperability - Abstract Specifications (WfMC-TC-1012)

Interoperability - Internet E-mail MIME Binding (WfMC-TC-1018)

Accompanying documents:

The Resource Model (Organizational Model: WfMC TC-1016-O)

BPMN icons and tables used in this document are copyright © by the Business Process Management Initiative [BPMI.org], May 3, 2004. All Rights Reserved by BPMI.org.

5. Overview of Process Definition Interchange

An XPDL package corresponds to a Business Process Diagram (BPD) in BPMN, and consists of a set of Process Definitions. The WfMC defines a process as:

The representation of a business process in a form that supports automated manipulation, such as modeling, or enactment by a workflow [or business] management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc. (WfMC Glossary - WfMC-TC-1011)

The process definition provides an environment for a rich description of a process that can be used for the following,

- Act as a template for the creation and control of instances of that process during process enactment.
- For simulation and forecasting.
- As a basis to monitor and analyse enacted processes.
- For documentation, visualization, and knowledge management.

The process definition may contain references to subflows, separately defined, which make up part of the overall process definition.

An initial process definition will contain at least the minimal set of objects and attributes necessary to initiate and support process execution. Some of these objects and attributes will be inherited by each created instance of the process.

The WfMC Glossary also contains descriptions of, and common terminology for, the basic concepts embodied within a process definition such as activities, transitions, relevant data and participants, etc.

5.1. Approaches to Process Definition Interchange

This specification uses XML as the mechanism for process definition interchange. XPDL forms a common interchange standard that enables products to continue to support arbitrary internal representations of process definitions with an import/export function to map to/from the standard at the product boundary.

A variety of different mechanisms may be used to transfer process definition data between systems according to the characteristics of the various business scenarios. In all cases the process definition must be expressed in a consistent form, which is derived from the common set of objects, relationships and attributes expressing its underlying concepts.

The principles of process definition interchange are illustrated in Figure 5-1: The Concept of the Process Definition Interchange.

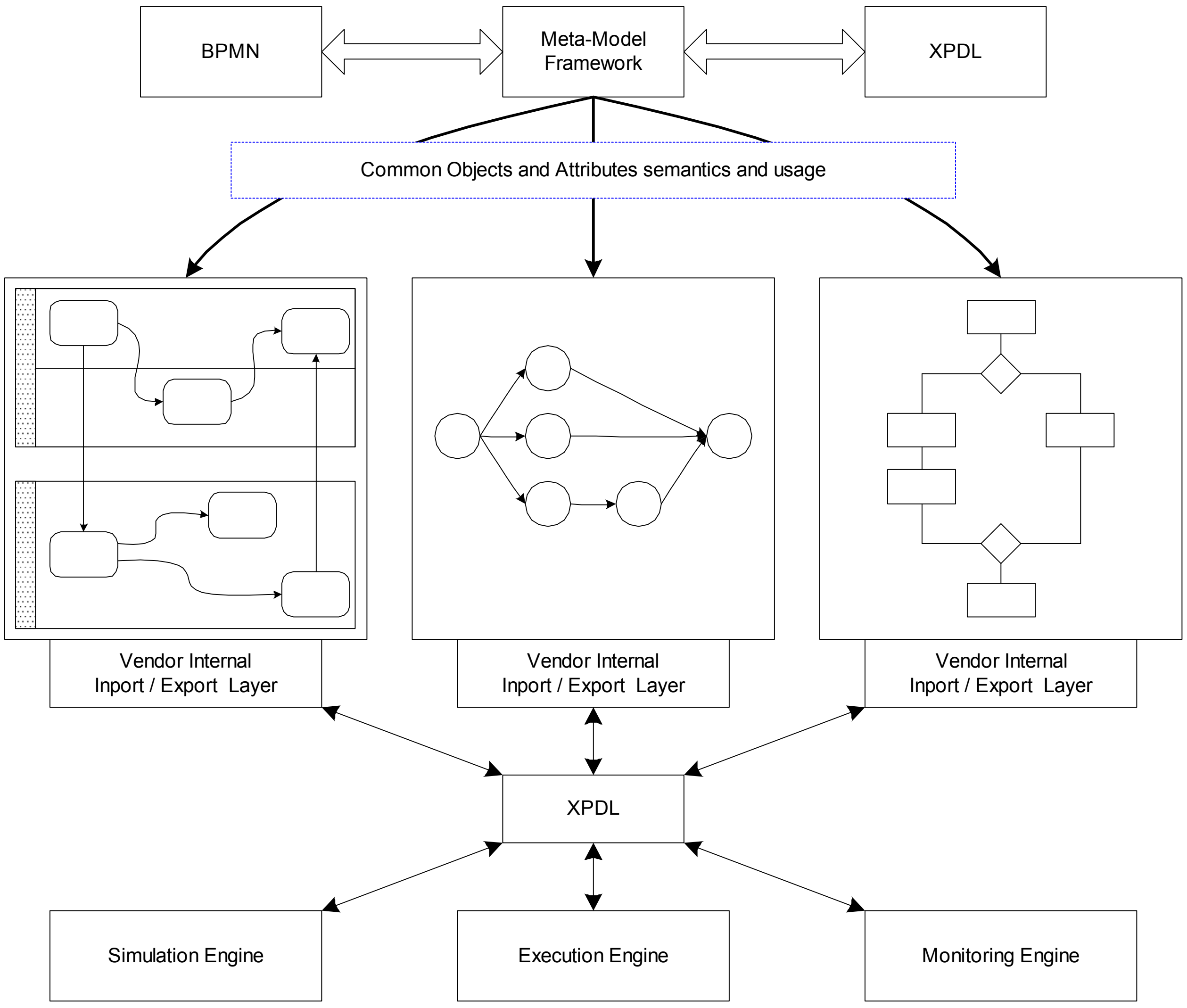

BPMN
Meta-Model Framework
XPDL
Common Objects and Attributes semantics and usage
Vendor Internal Inport / Export Layer
Vendor Internal Inport / Export Layer
Vendor Internal Inport / Export Layer
XPDL
Simulation Engine
Execution Engine
Monitoring Engine

Figure 5-1: The Concept of the Process Definition Interchange

6. Meta-Model

The Meta-Model describes the top-level entities contained within a Process Definition, their relationships and attributes (including some which may be defined for simulation or monitoring purposes rather than for enactment). It also defines various conventions for grouping process definitions into related process models and the use of common definition data across a number of different process definitions or models.

6.1. Processes and Packages

The process model includes various entities whose scope may be wider than a single process definition. In particular the definitions of participants, applications and relevant data may be referenced from a number of process definitions. The meta-model assumes the use of a common process definition repository to hold the various entity types comprising the process definition. Within the repository itself and to support the efficient transfer of process definition data to/from the repository, the concept of a package is introduced, which acts as a container for the grouping of common data entities from a number of different process definitions, to avoid redefinition within each individual process definition.

The package provides a container to hold a number of common attributes from the process definition entity (author, version, status, etc.). Each process definition contained within the package will automatically inherit any common attributes from the package, unless they are separately re-specified locally within the process definition

An XPDL Package corresponds to a BPMN Business Process Diagram. At the level below Package, there are four new elements which are discussed in later sections of this document:

1. **Pools** (and their lanes) are associated with processes and are used in layout and to define participants for the sequence flow elements contained within.
2. **Message flows** are used to represent communication between processes, based on Web Services Description Language (WSDL) protocols.
3. **Associations** and **Artifacts** are used to document the process definitions. Associations and the Artifacts they connect to provide additional information for the reader of a BPMN Diagram, but do not directly affect the execution of the Process.

Within a package, the scope of the definitions of some entities is global and these entities can be referenced from all process definitions (and associated activities and transitions) contained within the package. Those entities are:

- participant specification
- application declaration, and
- relevant data field

The package reference allows the use within the package or its contained objects of references to top-level entities in the referenced external package:

- Process ids for SubFlow reference
- participant specifications
- application declarations
- type declarations

Conventions on name and identifier management across different packages within the same repository address space to achieve any necessary global uniqueness are for user/vendor definition. The assumed convention during process enactment is that name reference searches follow the sequence:

- Process ids - firstly within the same model (including any references to process definitions for remote execution on a different service), then within any externally referenced model
- Applications / participants - firstly within the same model, then within any externally referenced model

Relevant data naming must be unique within a package; where such data is passed between processes as parameters the convention at this version of specification is that copy semantics will be used. Responsibility rests with process designers / administrators to ensure consistent name / data type usage within process definitions / models to support subflow operations (including any required remote process interoperability).

6.2. Package Meta-Model

- Multiple process definitions are bound together in a model definition. The Package acts as a container for grouping together a number of individual process definitions and associated entity data, which is applicable to all the contained process definitions (and hence requires definition only once).

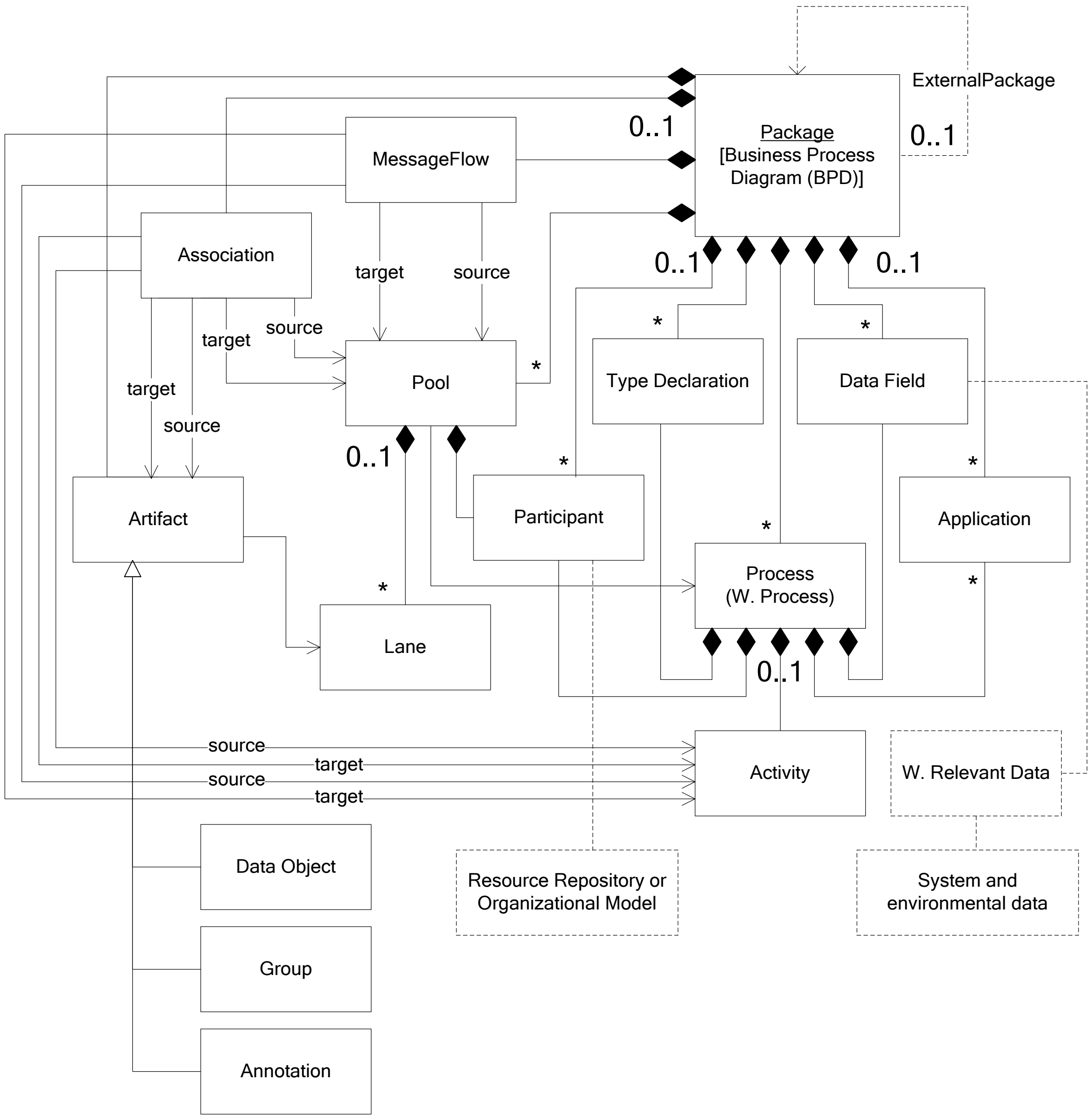


Figure 6-1 Package Definition Meta Model

The meta-model for the Package identifies the entities and attributes for the exchange, or storage, of process models. It defines various rules of inheritance to associate an individual process definition with entity definitions for participant specification, application declaration and relevant data field, which may be defined at the package level rather than at the level of individual process definitions.

The Package Definition allows the specification of a number of common process definition attributes, which will then apply to all individual process definitions contained within the package. Such attributes may then be omitted from the individual process definitions. (If they are re-specified at the level of an individual process definition this local attribute value takes precedence over the global value defined at the package level.

6.2.1. Process Repository

The process definition import/export interface is assumed to operate to/from a definition repository of some form associated with the process or workflow management system. The import/export interface is realized by the transfer of files containing XPDL into or out of such repository. This interface specification allows the import or export of process definition data at the level of individual process definitions and packages.

The internal interface between the repository and control functions is specific to individual vendor products and does not form part of this standard. It is assumed that separation is provided (for example by version control) between repository usage as a static repository (for persistent, ongoing storage of process definition data) and any dynamic usage (for managing changes to the process execution of extant process instances).

The local storage structure of the process definition repository is not part of the WfMC standard. The use of a package is defined only as an aid to simplify the import/export of reusable data structures. Where a simple process repository structure is used, operating at a single level of process definition, shared information within an imported package may be replicated into each of the individual process definitions at the import interface (and similarly repacked, if required, for process definition export).

6.2.1.1. Redefinition and Scope

The possibility of redefining attributes and meta-model entities and referencing external packages introduces the principles of scope and hierarchy into the XPDL (and process repository) structures.

(i) Relevant data field

Process relevant data field has a scope that is defined by the directly surrounding meta-model entity and is not nested. The visibility of its identifier is also defined by that entity.

(ii) Attributes

Attributes including extended attributes have a scope that is defined by the directly surrounding meta-model entity and are nested, i.e. may be redefined at a lower level. Example: The name attribute is redefined in each entity definition. The visibility of extended attribute identifiers is within the particular entity and all sub-entities unless the identifier is redefined in a sub-entity.

(iii) Participants and applications

- Participants and applications have a scope and visibility equivalent to extended attributes. All referenced relevant data field and extended attributes have to be defined in the scope where they are used, at least in the same package.

For a referenced external package entity that needs itself reference to entities and their identifiers defined in its external package clause the mechanism is started with the root in that package. That guarantees that no conflict takes place if the invoking process has an entity with the same id, which the definer of the referenced package cannot be aware of.

The described mechanism of external package provides high flexibility for designers and administrators. One can separate organization descriptions (participant entities) and process definitions in separate models, one can add a new release of a process description or add a new process definition sharing the rest of the definition of previously defined and exchanged models without resubmitting the whole context etc.

6.3. Process Meta-Model

The meta-model identifies the basic set of entities and attributes for the exchange of process definitions. For a Process Definition the following entities must be defined, either explicitly at the level of the process definition, or by inheritance directly or via cross reference from a surrounding package.

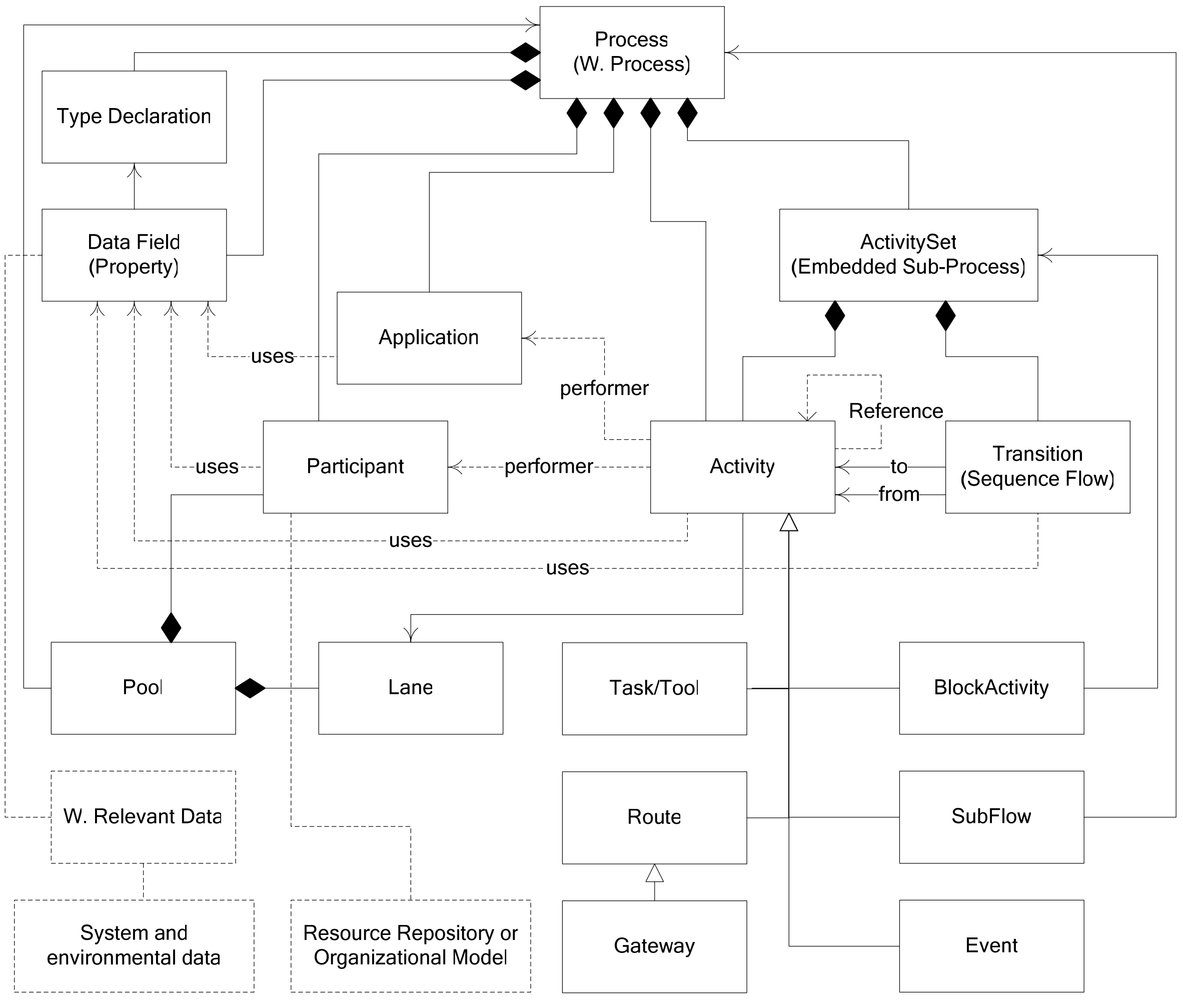


Figure 6-2: Process Definition Meta Model

These entities contain attributes that support a common description mechanism for processes. They are described in the subsequent document sections.

The XPDL Process and WorkflowProcess correspond to the BPMN Process. At the level below Process, there are two new elements: Assignments and Categories; these are discussed in later sections. Assignments allow Data Fields (Properties) to be assigned values in the course of execution of a Process. Categories are useful in reporting and analysis of running Processes but do not affect the behavior of the processes.

6.4. Entities Overview

The meta-model identifies the basic set of entities used in the exchange of process definitions. The top-level entities are as follows:

6.4.1. Swimlanes

Swimlanes are used to facilitate the graphical layout of a collection of processes and may designate participant information at the process level and performer information at the activity level. The Swimlane structure is depicted by a collection of non-overlapping rectangles called Pools. Each Pool may be further subdivided into a number of Lanes.

6.4.1.1. Pool

A Pool acts as the container for flow objects (activities) and Sequence Flow (transitions) between them. The Sequence Flow may cross the boundaries between Lanes of a Pool, but cannot cross the boundaries of a Pool. The interaction between Pools, e.g., in a B2B context, is shown through Message Flow.

Another aspect of Pools is whether or not there is any activity detailed within the Pool. Thus, a given Pool may be shown as a "White Box," with all details exposed, or as a "Black Box," with all details hidden. No Sequence Flow is associated with a "Black Box" Pool, but Message Flow can attach to its boundaries.

Each page of the diagram is regarded as an invisible 'background pool'. Activities and sequence flow not within an explicit Pool are treated as contained by the background pool.

6.4.1.2. Lane

Lanes are used to subdivide a Pool. All the activities within a Lane may inherit one or more properties from the Lane. A typical use of this is to give the Lanes 'role names' and have the Activities inherit these role names as 'Participant assignment/Performer expressions'.

6.4.2. Process Definition

The Process Definition entity provides contextual information that applies to other entities within the process. It is a container for the process itself and provides information associated with administration (creation date, author, etc.) or to be used during process execution (initiation parameters to be used, execution priority, time limits to be checked, person to be notified, simulation information, etc.).

6.4.3. Process Activity

A process definition consists of one or more activities, each comprising a logical, self-contained unit of work within the process. An activity represents work, which will be performed by a combination of resource (specified by participant assignment) and/or computer applications (specified by application assignment). Other optional information may be associated with the activity such as information on whether it is to be started / finished automatically by the process or workflow management system or its priority relative to other activities where contention for resource or system services occurs. Usage of specific relevant data field items by the activity may also be specified. The scope of an activity is local to a specific process definition (although see the description of a subflow activity below).

In addition to the tools explicitly declared as applications, an activity may be implemented as one of a number of built-in BPMN **tasks**. Most of these concern B2B messaging. Their attribute details are discussed in section 7.6.5.3.

An activity may be a subflow - in this case it is a container for the execution of a (separately specified) process definition, which may be executed locally within the same service, or (possibly using the process interoperability interface) on a remote service. The process definition identified within the subflow contains its own definition of activities, internal transitions, resource, and application assignments (although these may be inherited from a common source). In- and out-parameters permit the exchange of any necessary relevant data field between calling and called process (and, where necessary, on return). Such definitions are equivalent to BPMN **independent subprocesses**.

An activity may be a block activity that executes an activity set, or map of activities and transitions. Activities and transitions within an activity set share the name space of the containing process. An activity set is equivalent to a

BPMN **embedded subprocess**.

An activity may be a route activity, which performs no work processing (and therefore has no associated resource or applications), but simply supports routing decisions among the incoming transitions and/or among the outgoing transitions. BPMN **gateways** are represented by Route activities.

Finally, an activity may represent a BPMN **event**. An event is something that "happens" during the course of a business process. These events affect the flow of the process and usually have a cause (trigger) or an impact (result). There are three types of Events, based on when they affect the flow: Start, Intermediate, and End. Their attribute details are discussed in section 7.6.4.

6.4.4. Transition Information

Activities are related to one another via flow control conditions (transition information). Each individual transition has three elementary properties, the from-activity, the to-activity and the condition under which the transition is made. Transition from one activity to another may be conditional (involving expressions which are evaluated to permit or inhibit the transition) or unconditional. The transitions within a process may result in the sequential or parallel operation of individual activities within the process. The information related to associated split or join conditions is defined within the appropriate activity, split as a form of "post activity" processing in the from-activity, join as a form of "pre-activity" processing in the to- activity. This approach allows the control processing associated with process instance thread splitting and synchronization to be managed as part of the associated activity, and retains transitions as simple route assignment functions. The scope of a particular transition is local to the process definition, which contains it and the associated activities.

More complex transitions, which cannot be expressed using the simple elementary transition and the split and join functions associated with the from- and to- activities, are formed using route activities, which can be specified as intermediate steps between real activities allowing additional combinations of split and/or join operations. Using the basic transition entity plus route activities, routing structures of arbitrary complexity can be specified. Since several different approaches to transition control exist within the industry, several conformance classes are specified within XPDL. These are described later in the document.

6.4.5. Participant Declaration

This provides descriptions of resources that can act as the performer of the various activities in the process definition. The particular resources, which can be assigned to perform a specific activity, are specified as an attribute of the activity, participant assignment, which links the activity to the set of resources (within the participant declaration) which may be allocated to it. The participant declaration does not necessarily refer to a human or a single person, but may also identify a set of people of appropriate skill or responsibility, or machine automata resource rather than human. The meta-model includes some simple types of resource that may be defined within the participant declaration.

6.4.6. Application Declaration

This provides descriptions of the IT applications or interfaces which may be invoked by the service to support, or wholly automate, the processing associated with each activity, and identified within the activity by an application assignment attribute (or attributes). Such applications may be generic industry tools, specific departmental or enterprise services, or localized procedures implemented within the framework of the process or workflow management system. The application definition reflects the interface between the engine and the application or interface, including any parameters to be passed.

6.4.7. Artifact

To satisfy additional modeling concepts that are not part of the basic set of flow elements (activities, sequence and message flow), BPMN provides the concept of Artifacts that can be linked to the existing Flow Objects through Associations (see below). Thus, Artifacts do not affect the basic Sequence or Message Flow, nor do they affect mappings to execution languages.

At this point, BPMN provides three standard Artifacts: A Data Object, a Group, and an Annotation. Additional standard Artifacts may be added to the BPMN specification in later versions. A modeler or modeling tool may extend a BPD and add new types of Artifacts to a Diagram. Any new Artifact must follow the specified Sequence Flow and Message Flow connection rules.

6.4.8. Message Flow

A Message Flow is used to show the flow of messages between two participants/processes that are prepared to send and receive them. In BPMN, two separate Pools in the Diagram will represent the two participants/processes (e.g., business entities or business roles). All Message Flow must connect two separate Pools. They can connect to the Pool boundary or to Flow Objects within the Pool boundary. They cannot connect two objects within the same Pool.

6.4.9. Association

An Association is used to associate information and Artifacts with Flow Objects. Text and graphical non-Flow Objects can be associated with the Flow Objects and Flow. An Association is also used to show the activities used to compensate for an activity.

An Association does not have a specific mapping to an execution language element. These objects and the Artifacts they connect to provide additional information for the reader of the BPMN Diagram, but do not directly affect the execution of the Process.

6.4.10. Relevant data field

This defines the data that is created and used within each process instance during process execution. The data is made available to activities or applications executed during the process and may be used to pass persistent information or intermediate results between activities and/or for evaluation in conditional expressions such as in transitions or participant assignment. Relevant data field is of particular type. XPDL includes definition of various basic and complex data types, (including date, string, etc.) Activities, invoked applications and/or transition conditions may refer to process relevant data field.

6.4.11. Data Types and Expressions

The meta-model (and associated XPDL) assumes a number of standard data types (string, reference, integer, float, date/time, etc.); such data types are relevant to data fields, system or environmental data or participant data. Expressions may be formed using such data types to support conditional evaluations and assignment of new values to data fields. Data types may be extended using an XML schema or a reference to data defined in an external source.

6.4.12. System and Environmental Data

This is data which is maintained by the process or workflow management system or the local system environment, but which may be accessed by activities or used by the process or workflow management system in the evaluation of conditional expressions and assignments in the same way as relevant data fields.

6.4.13. Resource Repository

The resource repository accounts for the fact that participants can be humans, programs, or machines. In more sophisticated scenarios the participant declaration may refer to a resource repository, which may be an Organizational Model in the case of human participants. Note that this specification does not define or require a resource repository.

6.4.14. Vendor or User specific Extensions

Although the meta-model and associated XPDL contain most of the constructs, which are likely to be required in the exchange of process definitions, there may be circumstances under which additional information (user or vendor specific) will need to be included within a process definition. Users and vendors are encouraged to work as far as possible within the standard entity / attribute sets; however, when extensions are needed the XPDL schema provides a standard way to extend it with vendor or user specific extensions.

6.4.14.1. Extended Elements and Attributes

The primary method to support such extensions is by the use of extended attributes. Extended attributes are those defined by the user or vendor, where necessary, to express any additional entity characteristics. XPDL schema supports namespace-qualified extensions to all the XPDL elements. The XPDL elements can be extended by adding new child elements, and new attributes.

6.4.14.2. Extended parameter mapping

No specific details of the scheme for encoding and passing parameter data are defined within this specification. Where parameters are passed on remote subflow invocation using the workflow Interoperability Specification (interface four), specifications are provided for the mapping of such parameters (for example into wf-XML exchanges) using the operations within the concrete syntax specification for interoperability. Any local scheme for parameter mapping and encoding is vendor defined on a product-by-product basis and lies outside the scope of this specification.

7. XML Process Definition Language

7.1. Elements Common for Multiple Entities

7.1.1. Graphic Information

Graphic information is optional and tool dependent. The graphic information (NodeGraphicsInfo and ConnectorGraphicsInfo) may appear multiple times on each XPDL element, depending on the number of tools that have added the graphical information to the XPDL file. Each tool, identificated by ToolId, can add its own graphical information. Therefore, each tool can display and represent the same XPDL in totally different ways, because each tool uses its own graphical information, but retains the graphical information from other tools.

7.1.1.1. NodeGraphicsInfo

NodeGraphicsInfo is an optional entity that can be used by a tool to describe graphical information. Each graphical node in XPDL (Activity, Pool, Lane, Artifact) has a list of NodeGraphicsInfo, one for each tool that has saved the corresponding information in the XPDL file. If a tool chooses to use the NodeGraphicsInfo, it should select a tool id, which can be the name of the tool. Therefore multiple tools can work using the same XPDL file and displaying the process with a different presentation layout.

```
<xsd:element name="NodeGraphicsInfo">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Coordinates" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="ToolId" type="xsd:string" use="optional"/>
      <xsd:attribute name="IsVisible" type="xsd:boolean" use="optional"
         default="true"/>
      <xsd:attribute name="Page" type="xsd:NMTOKEN" use="optional"/>
      <xsd:attribute name="LaneId" type="xsd:NMTOKEN" use="optional"/>
      <xsd:attribute name="Height" type="xsd:double" use="optional"/>
      <xsd:attribute name="Width" type="xsd:double" use="optional"/>
      <xsd:attribute name="BorderColor" type="xsd:string" use="optional"/>
      <xsd:attribute name="FillColor" type="xsd:string" use="optional"/>
      <xsd:attribute name="Shape" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="NodeGraphicsInfos">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:NodeGraphicsInfo"
            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:element>

<xsd:element name="Coordinates">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
```

```
        <xsd:attribute name="XCoordinate" type="xsd:double" use="optional"/>
        <xsd:attribute name="YCoordinate" type="xsd:double" use="optional"/>
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
     </xsd:complexType>
  </xsd:element>
```

	Description
Border Color	Color of the border, expressed as a string. Tool specific and depends on ToolId.
Coordinates	X and Y coordinates of node's upper left corner (bounding box). Tool specific and depends on ToolId. Usual implementation based on upper left corner of page being (0,0) and all coordinates >= 0.
Fill Color	Color of the fill, expressed as a string. Tool specific and depends on ToolId.
Height	Height of the node. Tool specific and depends on ToolId.
LaneId	If the node is in a Lane, this is the Id of the Lane.
Tool Id	Tool id. This may correspond to the name of the tool generating the XPDL file. Note that multiple NodeGraphicsInfo elements may appear in an element. This allows each tool to have its NodeGraphicsInfo, allowing for different tools using the same XPDL file to represent the element in totaly different ways. Each tool read and writes its own NodeGraphicsInfo based on its ToolId, and it leave untouch the NodeGraphicsInfo from oter tools.
Is Visible	True indicates node should be shown.
Page	The name of the page on which this node should be displayed.
Shape	Shape, expressed as a string: used to override BPMN shapes.
Width	Width of the node.). Tool specific and depends on ToolId.

Table 1: Node Graphics Info

7.1.1.2. ConnectorGraphicsInfo

ConnectorGraphicsInfo is an optional entity that can be used by a tool to describe graphical information for connecting objects (SequenceFlow, MessageFlow, Associations). Each connector in XPDL has a list of ConnectorGraphicsInfo, one for each tool that has saved the corresponding information in the XPDL file. If a tool chooses to use the ConnectorGraphicsInfo, it should select a tool id, which can be the name of the tool. Therefore multiple tools can work using the same XPDL file and displaying the process with a different presentation layout.

```
<xsd:element name="ConnectorGraphicsInfo">
   <xsd:complexType>
      <xsd:sequence minOccurs="0">
         <xsd:element ref="xpdl:Coordinates" minOccurs="0"
            maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="ToolId" type="xsd:NMTOKEN" use="optional"/>
      <xsd:attribute name="IsVisible" type="xsd:boolean" use="optional"
         default="true"/>
      <xsd:attribute name="Page" type="xsd:NMTOKEN" use="optional"/>
      <xsd:attribute name="Style" type="xsd:string" use="optional"/>
      <xsd:attribute name="BorderColor" type="xsd:string" use="optional"/>
      <xsd:attribute name="FillColor" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="ConnectorGraphicsInfos">
   <xsd:complexType>
```

```
            <xsd:sequence>
               <xsd:element ref="xpdl:ConnectorGraphicsInfo"
                  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:element>
```

	Description
BorderColor	Color of the border, expressed as a string. Tool specific and depends on ToolId.
Coordinates	X and Y coordinates of points in the path. Tool specific and depends on ToolId.
FillColor	Color of the border, expressed as a string.
ToolId	Tool id. This may correspond to the name of the tool generating the XPDL file. Note that multiple NodeGraphicsInfo elements may appear in an element. This allows each tool to have its NodeGraphicsInfo, allowing for different tools using the same XPDL file to represent the element in totaly different ways. Each tool read and writes its own NodeGraphicsInfo based on its ToolId, and it leave untouch the NodeGraphicsInfo from oter tools.
IsVisible	True indicates node should be shown.
Page	The name of the page on which this node should be displayed.
Shape	Shape, expressed as a string: used to override BPMN shapes
Style	LineStyle. Tool specific and depends on ToolId.

Table 2: ConnnectorGraphicsInfo

7.1.2. Extended Attributes

Extended Attributes can be used in all entities. They allow vendors to extend the functionality of this specification to meet individual product needs. A vendor can use extended attributes in two maners:

a- The anonymous extended attribute provides a name and a value for the extension without the need to qualify the extension with a namespace.

b- Namespace qualified extensions can be used to extend XPDL elements by adding child elements or by adding attributes. Namespace qualified extensions can be validated against a vendor provided schema (see 7.2.1.1 Vendor extensions).

7.1.2.1. Anonymous Extended Attribute

The anonymous extended attributes use the ExtendedAttribute element that can be used to add an extension with a name and a value.

```
      <xsd:element name="ExtendedAttribute">
         <xsd:complexType mixed="true">
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:choice>
            <xsd:attribute name="Name" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="Value" type="xsd:string"/>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="ExtendedAttributes">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element ref="xpdl:ExtendedAttribute"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
```

	Description
Name	Used to identify the Extended Attribute
Value	Value of the extension.

Table 3: Extended Attributes

7.1.2.2. Namespace Qualified Extensions

In order for a tool to add namespace-qualified extensions, it first needs to extend the XPDL schema to add the extensions in the places in which the XPDL schema allows the tool to add extensions. The new generated schema should contain in addition to the XPDL namespace the tool namespace. The resulting schema can be used in addition to the XPDL schema to validate XPDL 2.0 files. The extension places are marked in the XPDL schema by [namespace="##other" processContents="lax"].

In addition, the tool should include the namespace, and it should add the VendorExtension section in XPDL (see 7.2.1.1 Vendor extensions), which contains a URI reference to the extended schema.

A child element can be added to a XPDL element, by adding a child element to the ExtendAttributes under the XPDL element. An attribute to a XPDL element can also be added by just qualifying it by the vendor namespace.

An example on how a vendor can create a schema with XPDL extensions is provided in section 8.2 Extending XPDL Schema.

7.1.3. Formal Parameters

Formal parameters can be used as attributes in process and application. They are passed during invocation and return of control (e.g. of an invoked application). These are the invocation parameters.

```
   <xsd:element name="FormalParameter">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="xpdl:DataType"/>
            <xsd:element ref="xpdl:Description" minOccurs="0"/>
            <xsd:element ref="xpdl:Length" minOccurs="0"/>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
         <xsd:attribute name="Mode" default="IN">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="IN"/>
                  <xsd:enumeration value="OUT"/>
                  <xsd:enumeration value="INOUT"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:attribute>
         <xsd:attribute name="Name" type="xsd:string" use="optional"/>
         <xsd:attribute name="IsArray" type="xsd:boolean" use="optional"
            default="false"/>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="FormalParameters">
```

```
      <xsd:complexType>
         <xsd:sequence>
            <xsd:choice minOccurs="0">
               <xsd:element ref="deprecated:FormalParameter"
                  minOccurs="0" maxOccurs="unbounded"/>
               <xsd:element ref="xpdl:FormalParameter"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:choice>
            <xsd:choice minOccurs="0">
               <xsd:sequence>
                  <xsd:element name="Extensions"/>
                  <xsd:any namespace="##other" processContents="lax"
                     minOccurs="0" maxOccurs="unbounded"/>
               </xsd:sequence>
            </xsd:choice>
         </xsd:sequence>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>
```

	Description	
Data type	Data type of the formal parameter. See Section 7.13	
Description	Textual description of the formal parameter	
Length	The length of the data. Used only for strings, to declare maximum length.	
Id	Identifier for the parameter	
Name	Name for the parameter	
Is Array	Indicates if the parameter is an array or a single value parameter	
Index	Index of the parameter (Deprecated)	
Mode	IN	Input Parameters
	OUT	Output Parameters
	INOUT	Parameters used as input and output

Table 4: *Formal Parameters*

7.1.3.1. Parameter passing semantics

The parameter passing semantics is defined as:

(a) Any read-only formal parameters (IN) are initialised by the value of the corresponding actual parameter in the call (an expression). This is pass-by-value semantics.

(b) Any read/write formal parameters (INOUT) are initialised by the value of the corresponding actual (passed) parameter, which must be the identifier of a relevant data field entity. On completion of the process, the value of the formal out parameter is copied back to the original actual parameter (which must be the identifier of a relevant data field entity). This is copy-restore semantics.

(c) Any write-only formal parameters (OUT) are initialised to zero (strings will be set to the empty string, complex data will have each element set to zero). On completion of the process, the value of the formal out parameter is copied back to the original actual parameter (which must be the identifier of a relevant data field entity). This is zero-restore semantics.

7.1.3.2. Concurrency semantics

Copying and restoring of parameters are treated as atomic operations; to avoid access conflicts from concurrent operations on relevant data field within the process instance these operations are serialized. Between copy and restore of (c) no locking is assumed and the returned parameter value will overwrite the local value (of the particular relevant data field item) at the time of the return call.

7.1.3.3. Formal-actual parameter mapping

The mapping of actual to formal parameters during invocation is defined by a parameter map list. The actual parameters are mapped 1:1 to the formal parameters in sequence, i.e. the first actual maps to the first formal, the second actual maps to the second formal etc. Type compatibility is required within the definitions and may be enforced by the run-time system. The effects of violation are locally defined and do not form part of this specification

In case the actual parameter is an expression, the expression is evaluated and buffered by the engine, and the content of this buffer is used for formal-actual mapping. How the buffering and mapping is performed is outside the scope if this document.

7.1.4. External Reference

ExternalReference is a reference to an external definition of an entity. It can be used in DataTypes, Participant, and Application.

```
<xsd:element name="ExternalReference">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="xref" type="xsd:NMTOKEN" use="optional"/>
      <xsd:attribute name="location" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Location	It specifies the URI of the document that defines the type.
Namespace	It allows specification of the scope in which the entity is defined.
xref	It specifies the identity of the entity within the external document.

Table 5: External Reference

Example 1: A FormalParameter that is defined by an XML schema:

```
<FormalParameter Id="PO">
   <DataType>
      <ExternalReference location="http://abc.com/schemas/po.xsd"/>
   </DataType>
   <Description>PO specification for abc.com</Description>
</FormalParameter>
```

Example 2: A DataField defined by a Java class:

```
<DataField Id="PO" Name="PurchaseOrder" IsArray="FALSE">
   <DataType>
      <ExternalReference location="com.abc.purchases.PO"/>
   </DataType>
   <Description>PO specification for abc.com</Description>
</DataField>
```

7.1.4.1. Web Services

An activity in a process may invoke a web service. The ExternalReference element may be used as a reference to applications and data types that are defined in Web Service (WSDL) documents.

Example 3: A DataField whose data type is defined in a WSDL document:

```
<DataField Id="abcPO" Name="abcPurchaseOrder" IsArray="False">
   <DataType>
      <ExternalReference xref="PO"
location="http://abc.com/services/poService.wsdl"
namespace="poService/definitions/types"/>
   </DataType>
</DataField>
```

Example 4: An Application that is defined as an operation in a WSDL document:

```
<Application Id="placeOrder">
   <ExternalReference location="http://abc.com/PO/services/poService.wsdl"
      xref="PlaceOrder" namespace=
         "http://abc.com/services/poService.wsdl/definitions/portType"/>
</Application>
```

7.1.5. Assignment

Data fields may be explicitly assigned new values during the execution of a process. This may happen at the start or termination of a process or an activity. Assignments may also be associated with transitions/sequence flow, in which case the assignments are performed when the particular transition/outgoing sequence flow is chosen.

```
<xsd:element name="Assignment">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Target" type="xpdl:ExpressionType"/>
         <xsd:element name="Expression" type="xpdl:ExpressionType"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="AssignTime" use="optional" default="Start">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="Start"/>
               <xsd:enumeration value="End"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Assignments">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Assignment" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Target	Lvalue expression, normally the name of Data Field
Expression	Rvalue expression in the scripting language specified for the package.
AssignTime	Specifies time of evaluation and assignment: Start or End of Process or Activity. Not relevant to transition/sequence flow.

Table 6: Assignment

7.1.6. Category

The modeler MAY add one or more Categories to various elements, such as Process, Activity and Transition; which can then be used for purposes such as reporting and analysis. In OLAP-based reports the categories would be members of the category dimension and allow filtering and slice-and-dice queries.

```
<xsd:element name="Category">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Categories">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Category" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Id	Id of the category
Name	Name of Category

Table 7: Category

7.1.7. Artifact

An Artifact is a graphical object that provides supporting information about the Process or elements within the Process. However, it does not directly affect the flow of the Process. Other examples of Artifacts include critical success factors and milestones.

```
<xsd:element name="Artifact">
   <xsd:complexType>
      <xsd:sequence minOccurs="0">
         <xsd:element ref="xpdl:Object" minOccurs="0"/>
         <xsd:element ref="xpdl:DataObject" minOccurs="0"/>
         <xsd:element ref="xpdl:NodeGraphicsInfos"
            minOccurs="0" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="ArtifactType" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="DataObject"/>
               <xsd:enumeration value="Group"/>
               <xsd:enumeration value="Annotation"/>
            </xsd:restriction>
         </xsd:simpleType>
```

```
            </xsd:attribute>
            <xsd:attribute name="TextAnnotation" type="xsd:string" use="optional"/>
            <xsd:attribute name="Group" type="xsd:string" use="optional"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="Artifacts">
         <xsd:complexType>
            <xsd:sequence maxOccurs="unbounded">
               <xsd:element ref="xpdl:Artifact"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
Id	Id of the artifact
Name	Name of the artifact
ArtifactType	DataObject | Group | Annotation
DataObject	See section 7.1.7.2
Group	Name of the Group
NodeGraphicsInfos	See section 7.1.1.
Object	See section 7.1.7.1
TextAnnotation	Visible textual description.

Table 8: Artifact

7.1.7.1. Object

Referred to by numerous other graphical elements.

```
      <xsd:element name="Object">
         <xsd:complexType>
            <xsd:sequence minOccurs="0">
               <xsd:element ref="xpdl:Categories" minOccurs="0"/>
               <xsd:element ref="xpdl:Documentation" minOccurs="0"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="Name" type="xsd:string" use="optional"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
Name	Name of the Object
Categories	A list of categories. See section 7.1.6.
Documentation	Textual Documentation
Id	Id of the object

Table 9: Object

7.1.7.2. DataObject

In BPMN, a Data Object is considered an Artifact and not a Flow Object. They are considered an Artifact because they do not have any direct affect on the Sequence Flow or Message Flow of the Process, but they do provide information about what the Process does. That is, how documents, data, and other objects are used and updated during the Process. While the name "Data Object" may imply an electronic document, they can be used to represent many different types of objects, both electronic and physical.

As an Artifact, Data Objects generally will be associated with Flow Objects. An Association will be used to make the connection between the Data Object and the Flow Object. This means that the behavior of the Process can be modeled without Data Objects for modelers who want to reduce clutter. The same Process can be modeled with Data Objects for modelers who want to include more information without changing the basic behavior of the Process.

In some cases, the Data Object will be shown being sent from one activity to another, via a Sequence Flow. Data Objects will also be associated with Message Flow. They are not to be confused with the message itself, but could be thought of as the "payload" or content of some messages.

In other cases, the same Data Object will be shown as being an input, then an output of a Process. Directionality added to the Association will show whether the Data Object is an input or an output. Also, the state attribute of the Data Object can change to show the impact of the Process on the Data Object.

```
<xsd:element name="DataObject">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:DataFields" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="State" type="xsd:string" use="optional"/>
      <xsd:attribute name="RequiredForStart" type="xsd:boolean" use="required"/>
      <xsd:attribute name="ProducedAtCompletion" type="xsd:boolean"
         use="required"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Id	Id of the data object.
Name	Name is an attribute that is text description of the object.
ProducedAtCompletion	The default value for this attribute is True. This means that the Output will be produced when an activity has been completed. If set to False, then the activity MAY produce the output (more than once; i.e. zero or more times) before it has completed.
DataFields	A list of data fields.
RequiredForStart	The default value for this attribute is True. This means that the Input is required for an activity to start. If set to False, then the activity MAY start without the input, but MAY accept the input (more than once) after the activity has started.
State	State is an optional attribute that indicates the impact the Process has had on the Data Object. Multiple Data Objects with the same name MAY share the same state within one Process.

Table 10: Data Object

7.2. Package Definition

It is possible to define several processes within one package, which may share the same tools and participants. We recommend creating one package per business process which should contain all the necessary processes as well as all the associated tools and participants, although it is not required. It is also possible to define just parts of one process definition or common parts of several processes within one package (e.g. a participant list or a application list).

```
<xsd:element name="Package" type="xpdl:PackageType"/>

<xsd:complexType name="PackageType">
   <xsd:sequence>
      <xsd:element ref="xpdl:PackageHeader"/>
      <xsd:element ref="xpdl:RedefinableHeader" minOccurs="0"/>
      <xsd:element ref="xpdl:ConformanceClass" minOccurs="0"/>
      <xsd:element ref="xpdl:Script" minOccurs="0"/>
      <xsd:element ref="xpdl:ExternalPackages" minOccurs="0"/>
      <xsd:element ref="xpdl:TypeDeclarations" minOccurs="0"/>
      <xsd:element ref="xpdl:Participants" minOccurs="0"/>
      <xsd:element ref="xpdl:Applications" minOccurs="0"/>
      <xsd:element ref="xpdl:DataFields" minOccurs="0"/>
      <xsd:element ref="xpdl:PartnerLinkTypes" minOccurs="0"/>
      <xsd:element ref="xpdl:Pools" minOccurs="0"/>
      <xsd:element ref="xpdl:MessageFlows" minOccurs="0"/>
      <xsd:element ref="xpdl:Associations" minOccurs="0"/>
      <xsd:element ref="xpdl:Artifacts" minOccurs="0"/>
      <xsd:element ref="xpdl:WorkflowProcesses" minOccurs="0"/>
      <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
      <xsd:any namespace="##other" processContents="lax"
         minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
   <xsd:attribute name="Name" type="xsd:string" use="optional"/>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
```

	Description
Applications	A list of Application Declarations. See section 7.3
Artifacts	A list of Artifacts that can be linked to the existing Flow Objects through Associations. See section 6.4.7. and 7.1.7.
Associations	A list of Associations which associate information and Artifacts with Flow Objects. See section 7.10
Conformance Class	Structural restriction on process definitions in this package. See section 7.2.3
Data Fields	A list of Relevant data fields defined for the package. See section 7.12
Extended Attributes	A list of vendor-defined extensions that may be added to the package. See section 7.1.1
External Packages	Reference to another Package definition defined in a separate document.
Id	Used to identify the package.
MessageFlows	A list of MessageFlows which go between Pools or activities in two pools. See Section 7.8
Name	Text. Used to identify the package.
Package Header	A set of elements specifying package characteristics.
Participants	A list of resources used in implementing processes in the package. See section 7.11
PartnerLinkTypes	Partner link types for this package (see 7.8.1)
Pools	A list of Pools for the Package. See Section 7.4
Redefinable Header	A set of elements and attributes used by both the Package and Process definitions.
Script	Identifies the scripting language used in expressions.
Type Declarations	A list of Data Types used in the package. See section 7.13
Processes	A list of the Processes that comprise this package. See section 7.5

Table 11: Package Definition

7.2.1. Package definition Header

The package definition header keeps all information central to a package such as XPDL version, source vendor id, etc.

```
<xsd:element name="PackageHeader">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xpdl:XPDLVersion"/>
      <xsd:element ref="xpdl:Vendor"/>
      <xsd:element ref="xpdl:Created"/>
      <xsd:element ref="xpdl:Description" minOccurs="0"/>
      <xsd:element ref="xpdl:Documentation" minOccurs="0"/>
      <xsd:element ref="xpdl:PriorityUnit" minOccurs="0"/>
      <xsd:element ref="xpdl:CostUnit" minOccurs="0"/>
      <xsd:element ref="xpdl:VendorExtensions" minOccurs="0"/>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="XPDLVersion">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Vendor">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Created">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Description">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Documentation">
```

```
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="PriorityUnit">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="CostUnit">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
```

	Description
Cost Unit	Units used in Simulation Data (Usually expressed in terms of a currency)
Created	Creation date of Package Definition.
Description	Textual description of the package
Documentation	Operating System specific path- and filename of help file/description file.
Priority Unit	A text string with user defined semantics.
Vendor	Defines the origin of this model definition and contains vendor's name, vendor's product name and product's release number.
VendorExtensions	List of extensions by vendors. There is a vendor extension entry for each tool that provides extensions in this XPDL content.
XPDL Version	Version of this specification. The current value, for this specification, is “2.0”.

Table 12: *Package Definition Header – Attributes*

7.2.1.1. Vendor extensions

Vendor extension is used for vendors to define extensions and provide a schema and a description for the extensions. For details, see section 7.1.2.2 Namespace Qualified Extensions.

```
<xsd:element name="VendorExtension">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="ToolId" type="xsd:string" use="required"/>
      <xsd:attribute name="schemaLocation" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="extensionDescription" type="xsd:anyURI"
         use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="VendorExtensions">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:VendorExtension"
            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:element>
```

	Description
ToolId	Identification of the tool adding this extension. It is the same value used in NodeGraphicsInfo ToolId. This value may be a URI.
SchemaLocation	An URI indicating the location of the schema that can be used to validate the XPDL containing the extensions for the tool.
ExtensionDescription	An URI indicating the location of a document describing the extensions provided by the schema in schemaLocation.

Table 13: Vendor Extension -- Attributes

7.2.2. Redefinable Header

The redefinable header covers those header attributes that may be defined in the definition header and may be redefined in the header of any process definition. In case of redefinition, the scope rules hold.

```
   <xsd:element name="RedefinableHeader">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="xpdl:Author" minOccurs="0"/>
            <xsd:element ref="xpdl:Version" minOccurs="0"/>
            <xsd:element ref="xpdl:Codepage" minOccurs="0"/>
            <xsd:element ref="xpdl:Countrykey" minOccurs="0"/>
            <xsd:element ref="xpdl:Responsibles" minOccurs="0"/>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="PublicationStatus">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="UNDER_REVISION"/>
                  <xsd:enumeration value="RELEASED"/>
                  <xsd:enumeration value="UNDER_TEST"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:attribute>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="Author">
      <xsd:complexType>
         <xsd:simpleContent>
            <xsd:extension base="xsd:string">
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:extension>
         </xsd:simpleContent>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="Version">
      <xsd:complexType>
         <xsd:simpleContent>
            <xsd:extension base="xsd:string">
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:extension>
         </xsd:simpleContent>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="Codepage">
      <xsd:complexType>
         <xsd:simpleContent>
            <xsd:extension base="xsd:string">
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:extension>
         </xsd:simpleContent>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="Countrykey">
      <xsd:complexType>
         <xsd:simpleContent>
            <xsd:extension base="xsd:string">
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:extension>
         </xsd:simpleContent>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="Responsible">
      <xsd:complexType>
         <xsd:simpleContent>
            <xsd:extension base="xsd:string">
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:extension>
         </xsd:simpleContent>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="Responsibles">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="xpdl:Responsible" 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:element>
```

	Description
Author	Name of the author of this package definition.
Code page	The codepage used for the text parts. if codepage is omitted, then UFT-8 is assumed.
Country key	Country code based on ISO 3166. It could be either the three digits country code number, or the two alpha characters country codes.

	Description
Publication Status	Status of the Process Definition. UNDER_REVISION RELEASED UNDER_TEST
Responsible(s)	Participant, who is responsible for this process; the supervisor during run time Link to entity participant. Participant, who is responsible for this workflow of this Model definition (usually an Organisational Unit or a Human). It is assumed that the responsible is the supervisor during run time. Default: Initiating participant.
Version	Version of this Package Definition.

Table 14: *Redefinable Header*

7.2.3. Conformance Class Declaration

The conformance class declaration allows description of the conformance class to which the definitions in this model definition are restricted. The specified class applies to all the contained process definitions, unless it is re-defined locally at the process definition level.

```
<xsd:element name="ConformanceClass">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="GraphConformance" use="optional" default="NON_BLOCKED">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="FULL_BLOCKED"/>
               <xsd:enumeration value="LOOP_BLOCKED"/>
               <xsd:enumeration value="NON_BLOCKED"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description	
Conformance Class	FULL-BLOCKED	The network structure is restricted to proper nesting of SPLIT/JOIN and loops.
	LOOP-BLOCKED	The network structure is restricted to proper nesting of loops.
	NON-BLOCKED	There is no restriction on the network structure. This is the default.

Table 15: Conformance Class Declaration

7.2.4. Script

The Script element identifies the scripting language used in XPDL expressions. A text expression may be used wherever an element is of type xsd:string. One could, for example, use an expression within Cost elements.

An expression composed of formatted XML (e.g., MathML) may be used within the the ActualParameter or Xpression element (used within a Transition Condition).

```
<xsd:element name="Script">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Type" type="xsd:string" use="required"/>
      <xsd:attribute name="Version" type="xsd:string" use="optional"/>
      <xsd:attribute name="Grammar" type="xsd:anyURI" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Type	Identifies the scripting language used in expressions. For consistency across implementations, when specifying a standard scripting language, it is recommended that the Type be selected from the following strings: text/javascript, text/vbscript, text/tcl, text/ecmascript, text/xml.
Version	This is the version of the scripting language.
Grammar	This is a reference to a document that specifies the grammar of the language. It could be, for example, an XML schema, a DTD, or a BNF.

Table 16: Script

7.2.5. External Package

External package reference allows referencing definitions in another Package definition or in other systems providing an Interface to the Process or Management system (e.g. a legacy Organisation Description Management Tool).

```
<xsd:element name="ExternalPackage">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="href" type="xsd:string"/>
      <xsd:attribute name="Id" type="xsd:NMTOKEN"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="ExternalPackages">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:ExternalPackage"
            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:element>
```

	Description
Extended Attributes	Optional vendor-defined extensions to meet implementation needs. See section 7.1.2
href	A Model Identifier. Logical reference to a Model

	Description
id	The id of externally referenced package
Name	The name given to the external package

Table 17: External Package Reference

7.3. Application Declaration

Application declaration is a list of all applications/services or tools required and invoked by the processes defined within the process definition or surrounding package. Tools may be defined (or, in fact, just named). This means, that the real definition of the tools is not necessary and may be handled by an object manager. The reason for this approach is the handling of multi-platform environments, where a different program (or function) has to be invoked for each platform. XPDL abstracts from the concrete implementation or environment (thus these aspects are not of interest at process definition time).

```
<xsd:element name="Application">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Description" minOccurs="0"/>
         <xsd:element name="Type" type="xpdl:ApplicationType" minOccurs="0"/>
         <xsd:choice>
            <xsd:element ref="xpdl:FormalParameters"/>
            <xsd:element ref="xpdl:ExternalReference" minOccurs="0"/>
         </xsd:choice>
         <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Applications">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Application" 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:element>
```

	Description
Description	Short textual description of the application.
Extended Attributes	Optional vendor-defined extensions to meet implementation needs. See section 7.1.2
External Reference	A reference to an external specification of the application signature. See section 7.1.4
Formal Parameters	A list of parameters that is interchanged with the application via the invocation interface. See section 7.1.3.
Id	Used to identify the application definition
Name	Text used to identify an application (may be interpreted as a generic name of the tool).
Type	There are a number of standard Application Types. See section 7.3.2

Table 18: Application Declaration

7.3.1. Invocation Parameters

An Application declaration may have parameter definitions for the (invocation) parameters and also use them within other entities.

Copying the invocation `IN` is treated as one atomic operation. The same holds for restoring the invocation `OUT`. Between these two operations no assumption is made about concurrency behaviour.

7.3.2. Application Types

Application Type contains several pieces of information required by common applications such as calling an EJB component or invoking a WebService. Support for a particular application type is not mandatory for the engine, but if the engine makes use of the listed technologies the additional information should be provided by Application Type.

7.3.2.1. EJB

Application type that defines information required to call a method of an EJB component. An EJB application has additional restrictions for formal parameters: there can be a maximum of one OUT formal parameter, and if it exists it has to be the last formal parameter, also there should be no INOUT formal parameters. With these restrictions IN formal parameters map directly to arguments of the method and the optional last OUT formal parameter becomes the return value of the method.

```
<xsd:element name="Ejb">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="JndiName">
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                     <xsd:anyAttribute namespace="##other"
                        processContents="lax"/>
                  </xsd:extension>
               </xsd:simpleContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="HomeClass">
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                     <xsd:anyAttribute namespace="##other"
                        processContents="lax"/>
                  </xsd:extension>
               </xsd:simpleContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="Method">
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                     <xsd:anyAttribute namespace="##other"
                        processContents="lax"/>
                  </xsd:extension>
               </xsd:simpleContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
JndiName	JNDI name of the EJB
HomeClass	Home class fully qualified name
Method	Method that will be invoked

Table 19: EJB Application Type

7.3.2.2. Pojo

Application type that defines information required to call method on local Java class. Formal Parameters restrictions are the same as for EJB application type.

```
<xsd:element name="Pojo">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Class">
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                     <xsd:anyAttribute namespace="##other"
                        processContents="lax"/>
                  </xsd:extension>
               </xsd:simpleContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="Method">
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                     <xsd:anyAttribute namespace="##other"
                        processContents="lax"/>
                  </xsd:extension>
               </xsd:simpleContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Class	fully qualified name of the class
Method	Method that will be invoked

Table 20: POJO Application Type

7.3.2.3. XSLT

Application that uses XSL transformation on formal parameters. The Application should have one IN and one OUT formal parameter, or if the transformation transforms the formal parameter into a document in the same schema the application can have one INOUT formal parameter.

```
<xsd:element name="Xslt">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="location" type="xsd:anyURI"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Location	Location of the XSL transformation

Table 21: XSLT Application Type

7.3.2.4. Script

Application that executes a script (expression) using formal parameters. The script should have access only to formal parameters of the application.

```
      <xsd:element name="Script">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="Expression" type="xpdl:ExpressionType"
                  minOccurs="0"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
Expression	The script

Table 22: Script Application Type

7.3.2.5. WebService

Application that invokes a Web Service. All IN formal parameters should be mapped into content of the input message, all OUT formal parameters should be mapped to parts of the output message.

```
      <xsd:element name="WebService">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element ref="xpdl:WebServiceOperation"/>
               <xsd:element ref="xpdl:WebServiceFaultCatch"
                  minOccurs="0" maxOccurs="unbounded"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="InputMsgName" type="xsd:string" use="required"/>
            <xsd:attribute name="OutputMsgName" type="xsd:string" use="optional"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
WebServiceOperation	The web service operation used to invoke this application.
WebServiceFaultCatch	Provides a way to catch faults generated by the application
InputMsgName	The name of inputMessage as defined in the WSDL which will help in uniquely identifying the operation to be invoked.

	Description
OutputMsgName	The name of inputMessage as defined in the WSDL which will help in uniquely identifying the operation to be invoke.

Table 23: WebService Application Type

7.3.2.6. BusinessRule

Application that invokes a Business Rule.

```
<xsd:element name="BusinessRule">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="RuleName">
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                     <xsd:anyAttribute namespace="##other"
                        processContents="lax"/>
                  </xsd:extension>
               </xsd:simpleContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="Location">
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:anyURI">
                     <xsd:anyAttribute namespace="##other"
                        processContents="lax"/>
                  </xsd:extension>
               </xsd:simpleContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
RuleName	Name of the Business Rule
Location	Location of the Rule

Table 24: BusinessRule Application Type

7.3.2.7. Form

The standard does not decide which kind of a form layout should be used. The Form Application Type defines a place where all form related information should be stored.

```
<xsd:element name="Form">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="FormLayout" minOccurs="0">
            <xsd:complexType>
               <xsd:complexContent>
                  <xsd:extension base="xsd:anyType">
                     <xsd:anyAttribute namespace="##other"
                        processContents="lax"/>
                  </xsd:extension>
               </xsd:complexContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
FormLayout	Optional description of form layout

Table 25: Form Application Type

7.4. Swimlanes

7.4.1. Pool

For a general description of SwimLanes, Pools and Lanes refer to section 6.4.1.

```
<xsd:element name="Pool">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Lanes" minOccurs="0"/>
         <xsd:element ref="xpdl:Object" minOccurs="0"/>
         <xsd:element ref="xpdl:NodeGraphicsInfos"
            minOccurs="0" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="Orientation" use="optional" default="HORIZONTAL">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="HORIZONTAL"/>
               <xsd:enumeration value="VERTICAL"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Process" type="xsd:NMTOKEN" use="optional"/>
      <xsd:attribute name="Participant" type="xsd:NMTOKEN" use="optional"/>
      <xsd:attribute name="BoundaryVisible" type="xsd:boolean" use="required"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Pools">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Pool" 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:element>
```

	Description

	Description
BoundaryVisible	This attribute defines if the rectangular boundary for the Pool is visible. Only one Pool on a page MAY have the attribute set to False.
Id	The id of the Pool.
Lanes	The lanes in the pool. See section 7.4.2
Name	The name of the pool
NodeGraphicsInfos	See section 7.1.1.
Object	See section 7.1.7.1
Orientation	HORIZONTAL | VERTICAL
Participant	The Modeler MUST define the Participant for a Pool. The Participant can be either a Role or an Entity. This defines the role that a particular Entity or Role the Pool will play in a Diagram that includes collaboration. <<error>>
Process	The Process attribute defines the Process that is contained within the Pool. Each Pool MAY have a Process.

Table 26: Pools

7.4.2. Lane

```
<xsd:element name="Lane">
   <xsd:complexType>
      <xsd:sequence minOccurs="0">
         <xsd:element ref="xpdl:Object" minOccurs="0"/>
         <xsd:element ref="xpdl:NodeGraphicsInfos"
            minOccurs="0" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="ParentLane" type="xsd:NMTOKEN" use="optional"/>
      <xsd:attribute name="ParentPool" type="xsd:NMTOKEN" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Lanes">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Lane" 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:element>
```

	Description
Id	The id of the Lane.
Name	The name of the Lane.
NodeGraphicsInfos	See section 7.1.1.
Object	See section 7.1.7.1

	Description
ParentLane	ParentLane is an optional attribute that is used if the Lane is nested within another Lane. Nesting can be multi-level, but only the immediate parent is specified.
ParentPool	The Parent Pool MUST be specified. There can be only one Parent.

Table 27: Lane

7.5. Process Definition

The Process Definition defines the elements that make up a process. It contains definitions or declarations, respectively, for Activity and, optionally, for Transition, Application, and Process Relevant Data entities. Attributes may be specified for administration relevant data like author, and version; for runtime relevant data like priority; and for BPR and simulation relevant data.

A Process may run as an implementation of an activity of type SubFlow; in this case parameters may be defined as attributes of the process.

Where a process definition includes input parameters and is instantiated by means other than a SubFlow call (for example by local event) the method for initializing any input parameters is locally defined. In such circumstances any relevant data field associated with the instantiated process definition, which is included within the parameter list will be initialized to the value specified in the “default value” (where specified). Where relevant data field is not passed as an input parameter, or initialized by “default value” the result is undefined. Similarly where a subflow terminates abnormally without returning out parameter values to the calling process, the result is undefined.

In general the scope of the defined entity identifier and name is the surrounding entity. The identifier is unique in this scope. For the Process identifier and name the scope is the surrounding Package.

When a process definition is instantiated it is necessary to determine which activity is the first (start) activity. There are a number of ways to do this.

- There may be only one activity that has no incoming transitions.
- A single activity may have its StartActivity attribute set to true.
- The process attributes DefaultStartActivtySet and/or DefaultStartActivity may be present.
- The process invocation may specify the StartActivitySet and/or StartActivity.

Here we summarize the rules.

The following logic applies:

a. Unless otherwise specified in the process invocation (see ProcessRef 7.6.5.4), DefaultStartActivtySet and/or DefaultStartActivity determine where the process will start executing.
b. If present, DefaultStartActivitySetId must be the id of an Activity Set in the process
c. If present, DefaultStartActivityId must be the id of a start activity
    i. In the Default Activity Set if that’s present
    ii. In the top level process activities otherwise
d. If DefaultStartActivitySetId is present but not DefaultStartActivityId it is assumed that DefaultStartActivitySetId has exactly one start activity
e. If neither is present it is assumed that the top level activities contain exactly one start activity
f. It is assumed that a process invocation can designate StartActivitySetId and StartActivityId and thereby control where process execution starts. In particular sub process invocation can contain the information (see ProcessRef 7.6.5.4).

```
<xsd:element name="WorkflowProcess" type="xpdl:ProcessType"/>

<xsd:complexType name="ProcessType">
   <xsd:sequence>
      <xsd:element ref="xpdl:ProcessHeader"/>
      <xsd:element ref="xpdl:RedefinableHeader" minOccurs="0"/>
      <xsd:element ref="xpdl:FormalParameters" minOccurs="0"/>
      <xsd:choice minOccurs="0">
         <xsd:sequence minOccurs="0">
            <xsd:element ref="xpdl:Participants" minOccurs="0"/>
            <xsd:element ref="xpdl:Applications" minOccurs="0"/>
            <xsd:element ref="xpdl:DataFields" minOccurs="0"/>
         </xsd:sequence>
         <xsd:sequence minOccurs="0">
            <xsd:element ref="deprecated:DataFields" minOccurs="0"/>
            <xsd:element ref="deprecated:Participants" minOccurs="0"/>
            <xsd:element ref="deprecated:Applications" minOccurs="0"/>
         </xsd:sequence>
      </xsd:choice>
      <xsd:element ref="xpdl:ActivitySets" minOccurs="0"/>
      <xsd:element ref="xpdl:Activities" minOccurs="0"/>
      <xsd:element ref="xpdl:Transitions" minOccurs="0"/>
      <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
      <xsd:element ref="xpdl:Assignments" minOccurs="0"/>
      <xsd:element ref="xpdl:PartnerLinks" minOccurs="0"/>
      <xsd:element ref="xpdl:Object" minOccurs="0"/>
      <xsd:choice minOccurs="0">
         <xsd:sequence>
            <xsd:element name="Extensions"/>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
   <xsd:attribute name="Name" type="xsd:string" use="optional"/>
   <xsd:attribute name="AccessLevel" use="optional" default="PUBLIC">
      <xsd:simpleType>
         <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="PUBLIC"/>
            <xsd:enumeration value="PRIVATE"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:attribute>
   <xsd:attribute name="ProcessType" use="optional" default="None">
      <xsd:simpleType>
         <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="None"/>
            <xsd:enumeration value="Private"/>
            <xsd:enumeration value="Abstract"/>
            <xsd:enumeration value="Collaboration"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:attribute>
   <xsd:attribute name="Status" use="optional" default="None">
      <xsd:simpleType>
         <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="None"/>
            <xsd:enumeration value="Ready"/>
            <xsd:enumeration value="Active"/>
            <xsd:enumeration value="Cancelled"/>
            <xsd:enumeration value="Aborting"/>
            <xsd:enumeration value="Aborted"/>
            <xsd:enumeration value="Completing"/>
            <xsd:enumeration value="Completed"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:attribute>
   <xsd:attribute name="SuppressJoinFailure" type="xsd:boolean" use="optional"
      default="false"/>
   <xsd:attribute name="EnableInstanceCompensation" type="xsd:boolean"
      use="optional" default="false"/>
   <xsd:attribute name="AdHoc" type="xsd:boolean" use="optional" default="false"/>
```

```
      <xsd:attribute name="AdHocOrdering" use="optional" default="Parallel">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="Sequential"/>
               <xsd:enumeration value="Parallel"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="AdHocCompletionCondition" type="xsd:string"
         use="optional"/>
      <xsd:attribute name="DefaultStartActivitySetId" type="xsd:NMTOKEN"
         use="optional"/>
      <xsd:attribute name="DefaultStartActivityId" type="xsd:NMTOKEN"
         use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
```

	Description
AccessLevel	The Access level of a process may be either PUBLIC or PRIVATE. If PUBLIC the process may be invoked by an external system or application. A process with private access may only be invoked from a SubFlow Activity (see Section 7.6.5.3.8). Use is optional and default is PUBLIC.
Activities	A list of activities that comprise the process. See section 7.6.
ActivitySets	A list of self contained sets of activities and transitions. Used to represent a BPMN embedded subprocess.
AdHoc	See section 7.5.8.
AdHocOrdering	See section 7.5.8
AdHocCompletionCondition	See section 7.5.8
Applications	A list of Application Declarations. See section 7.3.
Assignments	A list of data field **assignments**. See section 7.1.5
Data Fields	A list of Relevant data fields defined for the process. See section 7.12.
DefaultStartActivityId	If present, DefaultStartActivityId must be the id of a start activity • In the Default StartActivitySet if that's present • In the top level process activities otherwise
DefaultStartActivitySetId	If present, DefaultStartActivitySetId must be the id of an Activity Set in the process.
EnableInstanceCompensation	See section 7.5.7.
Extended Attributes	Optional vendor-defined extensions to meet implementation needs. See section 7.1.1
Formal Parameters	A list of parameters that may be passed to the process. See section 7.1.3.
Id	Used to identify the process.
Name	Text Used to identify the process.
Object	See Section 7.1.7.1
Participants	A list of resources used in implementing the process. See section 7.11.
PartnerLinks	Partner links used by this process (see 7.8.2)
Process Header	A set of elements specifying process characteristics.
ProcessType	BPMN types: None, Private, Abstract, Collaboration. See section 7.5.4.

	Description
Redefinable Header	A set of elements and attributes used by both the Package and Process definitions.
Status	See section 7.5.5.
SuppressJoinFailure	See section 7.5.6.
Transitions	A list of the transitions that connect the process activities. See section 7.7.

Table 28: Process Definition

7.5.1. Process Definition Header

The process definition header keeps all information specific for a process definition such as process version, priority, duration of validity, etc.

```
<xsd:element name="ProcessHeader">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Created" minOccurs="0"/>
         <xsd:element ref="xpdl:Description" minOccurs="0"/>
         <xsd:element ref="xpdl:Priority" minOccurs="0"/>
         <xsd:element ref="xpdl:Limit" minOccurs="0"/>
         <xsd:element ref="xpdl:ValidFrom" minOccurs="0"/>
         <xsd:element ref="xpdl:ValidTo" minOccurs="0"/>
         <xsd:element ref="xpdl:TimeEstimation" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="DurationUnit">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="Y"/>
               <xsd:enumeration value="M"/>
               <xsd:enumeration value="D"/>
               <xsd:enumeration value="h"/>
               <xsd:enumeration value="m"/>
               <xsd:enumeration value="s"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Created">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Description">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
```

```
      </xsd:element>

      <xsd:element name="Limit">
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="Priority">
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="TimeEstimation">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element ref="xpdl:WaitingTime" minOccurs="0"/>
               <xsd:element ref="xpdl:WorkingTime" minOccurs="0"/>
               <xsd:element ref="xpdl:Duration" minOccurs="0"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="WaitingTime">
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="WorkingTime">
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="Duration">
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="ValidFrom">
         <xsd:complexType>
```

```
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="ValidTo">
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>
```

	Description
Created	Creation date of process definition.
Description	Short textual description of the process.
Duration	Expected duration time to perform a task in units of DurationUnit.
Duration Unit	Describes the default unit to be applied to an integer duration value that has no unit tag. Possible units are: Y - year M - month D - day H - hour m - minute s - second
Limit	Expected duration for time management purposes (e.g. starting an escalation procedure etc.) in units of DurationUnit. It is counted from the starting date/time of the Process. The consequences of reaching the limit value are not defined in this document (i.e. vendor specific). It is assumed that in this case at least the Responsible of the current process is notified of this situation.
Priority	The priority of the process type. The units are defined in the Package header priority units.
Time Estimation	Grouping of waiting time, working time, and duration. Used for simulation purposes.
Valid From	The date that the process definition is active from. Empty string means system date. Default: Inherited from Model Definition.
Valid To	The date at which the process definition becomes valid. Empty string means unlimited validity. Default: Inherited from Model Definition.
Waiting Time	Describes the amount of time, which is needed to prepare the performance of the task (time estimation) (waiting time is provided by the analysis environment and may be updated by the runtime environment) in units of DurationUnit.
Working Time	Describes the amount of time the performer of the activity needs to perform the task (time estimation) (working time is needed for analysis purposes and is provided by the evaluation of runtime parameters) in units of DurationUnit.

Table 29: *Process Definition Header*

7.5.2. Process Redefinable Header

Refer to Redefinable Header at the Package level: 7.2.2 .

```
<xsd:element name="RedefinableHeader">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Author" minOccurs="0"/>
         <xsd:element ref="xpdl:Version" minOccurs="0"/>
         <xsd:element ref="xpdl:Codepage" minOccurs="0"/>
         <xsd:element ref="xpdl:Countrykey" minOccurs="0"/>
         <xsd:element ref="xpdl:Responsibles" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="PublicationStatus">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="UNDER_REVISION"/>
               <xsd:enumeration value="RELEASED"/>
               <xsd:enumeration value="UNDER_TEST"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Author">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Codepage">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Countrykey">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Responsible">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
```

```
</xsd:element>

<xsd:element name="Responsibles">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Responsible" 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:element>
```

	Description
Author	Name of the author of this process definition. (The one, who put it into the repository)
Codepage	The codepage used for the text parts. Default: Inherited from Model Definition.
Country key	Country code based on ISO 3166. It could be either the three digits country code number, or the two alpha characters country codes. Default: Inherited from Model Definition.
Publication Status	Status of the Process Definition. Default: Inherited from Model Definition. UNDER_REVISION RELEASED UNDER_TEST
Responsible(s)	Participant, who is responsible for this process (usually an Organisational Unit or a Human). It is assumed that the responsible is the supervisor during execution of the process. Default: Inherited from Model Definition.
Version	Version of this process definition.

Table 30: *Process Redefinable Header*

7.5.3. Activity Set/Embedded SubProcess

An activity set is a self-contained set of activities and transitions. Transitions in the set should refer only to activities in the same set and there should be no transitions into or out of the set. Activity sets can be executed by block activities (see Section 7.6.3). An Activity Set is re-usable; it may be referred to by more than one Block Activity.

```
<xsd:element name="ActivitySet">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Activities" minOccurs="0"/>
         <xsd:element ref="xpdl:Transitions" minOccurs="0"/>
         <xsd:element ref="xpdl:Object" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="AdHoc" type="xsd:boolean" use="optional"
         default="false"/>
      <xsd:attribute name="AdHocOrdering" use="optional" default="Parallel">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="Sequential"/>
```

```
                <xsd:enumeration value="Parallel"/>
             </xsd:restriction>
          </xsd:simpleType>
       </xsd:attribute>
       <xsd:attribute name="AdHocCompletionCondition" type="xsd:string"
          use="optional"/>
       <xsd:attribute name="DefaultStartActivityId" type="xsd:NMTOKEN"
          use="optional"/>
       <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="ActivitySets">
    <xsd:complexType>
       <xsd:sequence>
          <xsd:element ref="xpdl:ActivitySet" 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:element>
```

	Description
Activities	A list of activities that comprise the process. See section 7.6.
AdHoc	See section 7.5.8.
AdHocOrdering	See section 7.5.8
AdHocCompletionCondition	See section 7.5.8
DefaultStartActivityId	Unless otherwise specified in the ActivitySet invocation (BlockActivity 7.6.3), this is where the activity set will start executing. If present, it must be the id of a start activity in the activity set. If not present it is assumed the activity set contains exactly one start activity.
Id	Used to identify the process.
Name	Name of the activity set/embedded sub-process.
Object	See section 7.1.7.1.
Transitions	A list of the transitions that connect the process activities. See section 7.7.

Table 31: ActivitySet

7.5.4. ProcessType in BPMN mapping to WS-BPEL

ProcessType is an attribute that provides information about which lower level language the Pool will be mapped to. By default, the ProcessType is None (or undefined). A Private ProcessType MAY be mapped to an executable WS-BPEL process. An Abstract ProcessType is also called the public interface of a process (or other web services) and MAY be mapped to an abstract WS-BPEL process. A Collaboration ProcessType will have two Lanes that represent business roles (e.g., buyer or seller) and will show the interactions between these roles. These Pools MAY be mapped to languages such as ebXML or WS Choreography. If the Process is to be used to create a WS-BPEL document, then the attribute MUST be set to Private or Abstract.

7.5.5. Status

The Status of a Process is determined when the Process is being executed by a process engine. The Status of a Process can be used within Assignment Expressions.

The following states are recognized:

- None
- Ready,
- Active
- Cancelled
- Aborting
- Aborted
- Completing
- Completed

7.5.6. SuppressJoinFailure

This attribute is included for mapping to WS-BPEL. This specifies whether or not a WS-BPEL joinFailure fault will be suppressed for all activities in the WS-BPEL process.

7.5.7. EnableInstanceCompensation

This attribute is included for mapping to WS-BPEL. It specifies whether or not a compensation can be performed after the Process has completed normally.

7.5.8. AdHoc

AdHoc is a boolean attribute, which has a default of False. This specifies whether the Process is Ad Hoc or not. The activities within an Ad Hoc Process are not controlled or sequenced in a particular order; their performance is determined by the performers of the activities.

If the Process is Ad Hoc (the AdHoc attribute is True), then the AdHocOrdering attribute MUST be included. This attribute defines if the activities within the Process can be performed in Parallel or must be performed sequentially. The default setting is Parallel and the setting of Sequential is a restriction on the performance that may be required due to shared resources.

If the Process is Ad Hoc (the AdHoc attribute is True), then the AdHocCompletionCondition attribute MUST be included. This attribute defines the conditions when the Process will end.

7.6. Process Activity

The Activity Definition is used to define each elementary activity that makes up a process. Attributes may be defined to specify activity control information, implementation alternatives, performer assignment, runtime relevant information like priority, and data used specifically in BPR and simulation situations (and not used within enactment). In addition, restrictions on data access and to transition evaluation (e.g. Split and Join) can be described. Mandatory attributes are used to define the activity identifier and type; a small number of other attributes are optional but have common usage across all activity types. Other attribute usage depends upon the activity type as shown in the table below.

For the Activity identifier and name the scope is the surrounding process.

The activity description is used to describe several different activity types. All these activities share the same (common) general activity attributes, but the usage of other attributes, particularly participant and application assignment and the use of relevant data field may be specialized to the activity type. The following table identifies the usage of other attributes / entity types for the different activity types.

Entity Types (usage within Activity Type)	Activity Type				
	Implementation Type			Route/GateWay/Event	BlockActivity
	None	Application/ Task	SubFlow		
Transition Restriction	Normal	Normal	Normal, plus subflow call / return within activity	Normal; any additional controls implemented within Route activity	Normal; refers to activities within same context, not to activities within ActivitySet
Participant Assignment	Normal	Normal/See Task details	N/A	N/A	N/A
Application Assignment	None	Yes/See task details	N/A	N/A	N/A
Use of relevant data field	Normal	Normal	May be used in parameter passing	May be used in routing control conditions	May be used in routing control conditions

Table 32: Entity type relationships for different Activity types

Notes on usage:

Transition restrictions, subflow, and route activities are described in the section on transitions. In general, normal transition restrictions may be declared at the level of the activity boundary within the surrounding process, whereas specialized flow conditions (subflow, or the internal part of a route activity) operate "internal" to the activity (but may reference activities within the surrounding process definition). The following diagram illustrates the generic structure of an activity and the above variants.

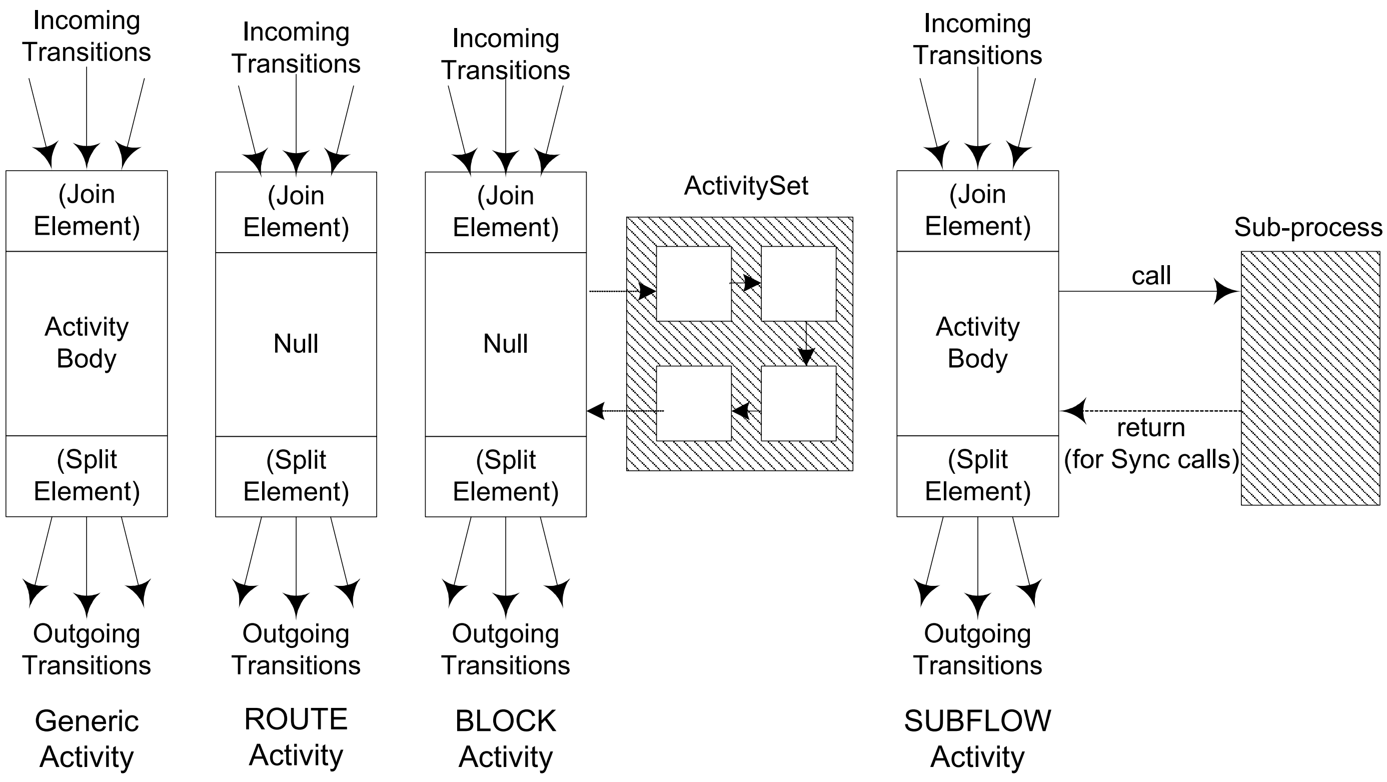


Figure 7-1: Activity Structures & Transition Conditions

Where the implementation type is NONE, the activity is manually controlled and its completion must be explicitly signaled to the process or management system. Such activities might typically comprise instructions to the participant to undertake a non-automated task of some type and inform a supervisor when completed.

Relevant data field may (potentially) be referenced within any activity although its use in manual activities is undefined through the process definition. Where an activity is of type SubFlow any in-parameters passed to the called (sub-) process must have been declared as relevant data field within the calling process / activity definition, or have been inherited from the surrounding package. (Similar requirements apply to any out-parameters returned to the calling process.) Routing and block activities may refer to such data within conditional expressions within the join/split control logic.

```
<xsd:element name="Activity">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Description" minOccurs="0"/>
         <xsd:element ref="xpdl:Limit" minOccurs="0"/>
         <xsd:choice minOccurs="0">
            <xsd:element ref="xpdl:Route"/>
            <xsd:element ref="xpdl:Implementation"/>
            <xsd:choice minOccurs="0">
               <xsd:element ref="deprecated:BlockActivity"/>
               <xsd:element ref="xpdl:BlockActivity"/>
            </xsd:choice>
            <xsd:element ref="xpdl:Event"/>
         </xsd:choice>
         <xsd:element ref="xpdl:Transaction" minOccurs="0"/>
         <xsd:element ref="xpdl:Performer" minOccurs="0"/>
         <xsd:element ref="xpdl:Performers" minOccurs="0"/>
         <xsd:element ref="deprecated:StartMode" minOccurs="0"/>
         <xsd:element ref="deprecated:FinishMode" minOccurs="0"/>
         <xsd:element ref="xpdl:Priority" minOccurs="0"/>
         <xsd:choice minOccurs="0">
            <xsd:element ref="deprecated:Deadline" minOccurs="0"
               maxOccurs="unbounded"/>
            <xsd:element ref="xpdl:Deadline" minOccurs="0" maxOccurs="unbounded"/>
         </xsd:choice>
         <xsd:element ref="xpdl:SimulationInformation" minOccurs="0"/>
         <xsd:element ref="xpdl:Icon" minOccurs="0"/>
         <xsd:element ref="xpdl:Documentation" minOccurs="0"/>
         <xsd:element ref="xpdl:TransitionRestrictions" minOccurs="0"/>
         <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
         <xsd:element ref="xpdl:DataFields" minOccurs="0"/>
         <xsd:element ref="xpdl:InputSets" minOccurs="0"/>
         <xsd:element ref="xpdl:OutputSets" minOccurs="0"/>
         <xsd:element ref="xpdl:IORules" minOccurs="0"/>
         <xsd:element ref="xpdl:Loop" minOccurs="0"/>
         <xsd:element ref="xpdl:Assignments" minOccurs="0"/>
         <xsd:element ref="xpdl:Object" minOccurs="0"/>
         <xsd:element ref="xpdl:NodeGraphicsInfos" minOccurs="0"
            maxOccurs="unbounded"/>
         <xsd:choice minOccurs="0">
            <xsd:sequence>
               <xsd:element name="Extensions"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="StartActivity" type="xsd:boolean" use="optional"/>
      <xsd:attribute name="Status" use="optional" default="None">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="None"/>
               <xsd:enumeration value="Ready"/>
               <xsd:enumeration value="Active"/>
```

```
                <xsd:enumeration value="Cancelled"/>
                <xsd:enumeration value="Aborting"/>
                <xsd:enumeration value="Aborted"/>
                <xsd:enumeration value="Completing"/>
                <xsd:enumeration value="Completed"/>
             </xsd:restriction>
          </xsd:simpleType>
       </xsd:attribute>
       <xsd:attribute name="StartMode">
          <xsd:simpleType>
             <xsd:restriction base="xsd:NMTOKEN">
                <xsd:enumeration value="Automatic"/>
                <xsd:enumeration value="Manual"/>
             </xsd:restriction>
          </xsd:simpleType>
       </xsd:attribute>
       <xsd:attribute name="FinishMode">
          <xsd:simpleType>
             <xsd:restriction base="xsd:NMTOKEN">
                <xsd:enumeration value="Automatic"/>
                <xsd:enumeration value="Manual"/>
             </xsd:restriction>
          </xsd:simpleType>
       </xsd:attribute>
       <xsd:attribute name="StartQuantity" type="xsd:integer"
          use="optional" default="1"/>
       <xsd:attribute name="IsATransaction" type="xsd:boolean"
          use="optional" default="false"/>
       <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="Activities">
    <xsd:complexType>
       <xsd:sequence>
          <xsd:element ref="xpdl:Activity" 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:element>

 <xsd:element name="Performer">
    <xsd:complexType>
       <xsd:simpleContent>
          <xsd:extension base="xsd:string">
             <xsd:anyAttribute namespace="##other" processContents="lax"/>
          </xsd:extension>
       </xsd:simpleContent>
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="Performers">
    <xsd:complexType>
       <xsd:sequence>
          <xsd:element ref="xpdl:Performer" maxOccurs="unbounded"/>
          <xsd:any namespace="##other" processContents="lax"
             minOccurs="0" maxOccurs="unbounded"/>
       </xsd:sequence>
       <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="Icon">
    <xsd:complexType>
       <xsd:simpleContent>
```

```
                <xsd:extension base="xsd:string">
                   <xsd:attribute name="XCOORD" type="xsd:integer" use="optional"/>
                   <xsd:attribute name="YCOORD" type="xsd:integer" use="optional"/>
                   <xsd:attribute name="WIDTH" type="xsd:integer" use="optional"/>
                   <xsd:attribute name="HEIGHT" type="xsd:integer" use="optional"/>
                   <xsd:attribute name="SHAPE" use="optional" default="RoundRectangle">
                      <xsd:simpleType>
                         <xsd:restriction base="xsd:NMTOKEN">
                            <xsd:enumeration value="RoundRectangle"/>
                            <xsd:enumeration value="Rectangle"/>
                            <xsd:enumeration value="Ellipse"/>
                            <xsd:enumeration value="Diamond"/>
                            <xsd:enumeration value="Ellipse"/>
                            <xsd:enumeration value="UpTriangle"/>
                            <xsd:enumeration value="DownTriangle"/>
                         </xsd:restriction>
                      </xsd:simpleType>
                   </xsd:attribute>
                   <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:extension>
             </xsd:simpleContent>
          </xsd:complexType>
       </xsd:element>
```

	Description
Assignments	A list of data field assignments. See section 7.1.5.
BlockActivity	An Activity that executes an ActivitySet. See 7.6.3
Deadline	Specification of a deadline and action to be taken if it is reached.
Description	Textual description of the activity.
Documentation	The address (e.g. path- and filename) for a help file or a description file of the activity.
DataFields	Allows declaration of relevant data local to the activity. See 7.12 .
Event	See section 7.6.4.
Extended Attributes	Optional extensions to meet individual implementation needs
Finish Mode	Describes how the system operates at the end of the Activity.
Icon	Address (path- and filename) for an icon to represent the activity in a graphical modeller. May be used to override the modeller icon for an activity.
Id	Used to identify the process activity.
Implementation	A "regular" Activity. Mandatory if not a Route. Alternative implementations are “no”, or “SubFlow”
InputSets	See section 7.6.11.
IORules	The IORules attribute is an expression that defines the relationship between one InputSet and one OutputSet. That is, if the activity is instantiated with a specified InputSet, then the output of the activity MUST produce the specified OutputSet. Zero or more IORules may be entered.
IsATransaction	If the activity is a block activity or is implemented as a subflow IsATransaction determines whether or not the behavior of the Sub-Process will be treated as a Transaction.
Limit	Expected duration for time management purposes (e.g. starting an escalation procedure etc.) in units of DurationUnit. It is counted from the starting date/time of the Process. The consequences of reaching the limit value are not defined in this document (i.e. vendor specific).
Loop	See Section 7.6.14
Name	Text Used to identify the process activity.
NodeGraphicsInfos	Optional. See section 7.1.1.

	Description
Object	See section 7.1.7.1
OutputSets	See section 7.6.12.
Performer	Link to entity participant. May be an expression. Default: Any Participant.
Priority	A value that describes the initial priority of this activity when it starts execution. If this attribute is not defined but a priority is defined in the Process definition then that is used. By default it is assumed that the priority levels are the natural numbers starting with zero, and that the higher the value the higher the priority (i.e.: 0, 1,…, n).
Route	A "dummy" Activity used for routing. A BPMN Gateway.
Simulation Information	Estimations for simulation of an Activity. No default.
StartActivity	Designates the first activity to be executed when the process is instantiated. Used when there is no other way to determine this. Conflicts with BPMN StartEvent and no process definition should contain both.
Start Mode	Describes how the execution of an Activity is triggered.
StartQuantity	The default value is 1. The value MUST NOT be less than 1. This attribute defines the number of Tokens that must arrive from a single Sequence Flow before the activity can begin.
Status	Status values are assigned during execution. Status can be treated as a property and used in expressions local to an Activity.
Transaction	If the Transaction attribute is False, then a Transaction MUST NOT be Identified. See section 7.6.13.
Transition Restrictions	Provides further restrictions and context-related semantics description of Transitions

Table 33: Process Activity

7.6.1. Execution Control Attributes

These are attributes of an Activity that allow the definition of various activity-specific features for Activity execution control. Refer to the Table for Process Activity.

Automation mode defines the degree of automation when triggering and terminating an activity. There are two automation modes:

- ***Automatic mode*** is fully controlled by the process or workflow engine, i.e. the engine proceeds with execution of the activity within the process automatically, as soon as any incoming transition conditions are satisfied. Similarly, completion of the activity and progression to any post activity conditional logic occurs automatically on termination of the final invoked application.
- ***Manual mode*** requires explicit user interaction to cause activity start or finish. In such systems the activity start and/or completion is as a result of explicit user action.

The automation modes can be specified independently for the ***start*** and ***end*** of an Activity.

7.6.2. Route Activity

The Route activity makes use of Transition restrictions (sub elements of Activity) to implement complex routing logic. Refer to the discussion at the beginning of section 7.6, Figure 7-1 and the section on Transition Restriction: section 7.6.9.

The Route Activity is a "dummy" Activity that permits the expression of "cascading" Transition conditions (e.g. of the type "`IF condition-1 THEN TO Activity-1 ELSE IF condition-2 THEN TO Activity-2 ELSE Activity-3 ENDIF`"). Some vendors might implement "cascading" transition conditions directly without requiring an activity counterpart for a route; others might require it. Wherever possible vendors and process designers are encouraged to structure such cascading conditions as an XOR split from the outgoing activity. Certain transition combinations cannot be expressed within a single transition list from the outgoing activity or a single incoming list to an activity. These cases require the use of one or more dummy activities; examples are:

- Combination of XOR and AND split conditions on outgoing transitions from an activity.
- Combination of XOR and AND join conditions on incoming transitions to an activity
- Transitions involving conditional AND joins of a subset of threads, with continuation of individual threads

A route activity has neither a performer nor an application.

For simulation purposes the following simulation data values should be assumed: `Duration` 0, `Cost` "0", `WorkingTime` 0, `WaitingTime` 0. For `Priority` and `Instantiation` the maximum value should be assumed.

```
<xsd:element name="Route">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="GatewayType" use="optional" default="XOR">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="XOR"/>
               <xsd:enumeration value="OR"/>
               <xsd:enumeration value="Complex"/>
               <xsd:enumeration value="AND"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Instantiate" type="xsd:boolean" use="optional"
         default="false"/>
      <xsd:attribute name="MarkerVisible" type="xsd:boolean" use="optional"
         default="false"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
GatewayType	Used when necessary for BPMN gateways. Note that TransitionRestriction (section 7.6.9) has all the attributes necessary for defining Gateways.
Instantiate	Event-Based Gateways can be defined as the instantiation mechanism for the Process with the Instantiate attribute. This attribute MAY be set to true if the Gateway is the first element after the Start Event or a starting Gateway if there is no Start Event (i.e., there are no incoming Sequence Flow).
MarkerVisible	This attribute determines if the XOR Marker is displayed in the center of the Gateway diamond (an "X"). The marker is displayed if the attribute is True and it is not displayed if the attribute is False. By default, the marker is not displayed.

Table 34: Route Activity

7.6.2.1. Gateway Activity

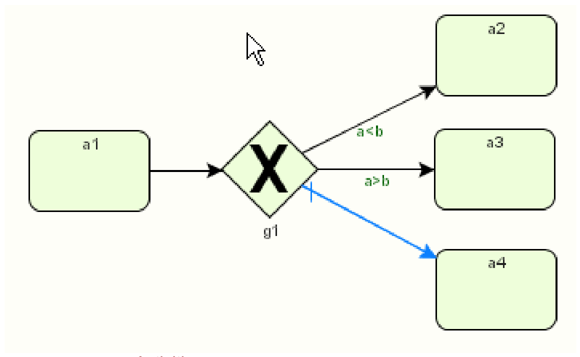
BPMN Gateway activities are represented by Route Activities, using three new attributes:

- GatewayType
- Instantiate
- MarkerVisible

Note that GatewayType is included as an optional supplement to TransitionRestrictions (see section 7.6.9).

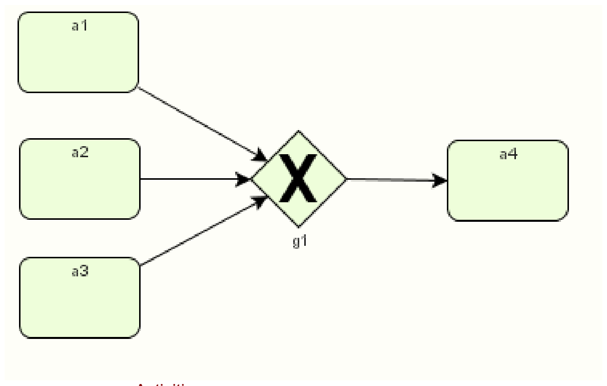
7.6.2.2. Examples of Gateways and their Representation

7.6.2.2.1.XOR Gate – Data based



```
<Activities>
    <Activity Id="3" Name="a1"/>
    <Activity Id="4" Name="g1">
        <Route GatewayType="XOR" XORType="Data" MarkerVisible="TRUE"/>
        <TransitionRestrictions>
            <TransitionRestriction>
                <Split Type="XOR">
                    <TransitionRefs>
                        <TransitionRef Id="9"/>
                        <TransitionRef Id="10"/>
                        <TransitionRef Id="11"/>
                    </TransitionRefs>
                </Split>
            </TransitionRestriction>
        </TransitionRestrictions>
    </Activity>
    <Activity Id="5" Name="a2"/>
    <Activity Id="6" Name="a3"/>
    <Activity Id="7" Name="a4"/>
</Activities>
<Transitions>
    <Transition Id="8" Name="" From="3" To="4" FlowType="SequenceFlow"/>
    <Transition Id="9" Name="" From="4" To="5" FlowType="SequenceFlow">
        <Condition Type="CONDITION">a&lt;b</Condition>
    </Transition>
    <Transition Id="10" Name="" From="4" To="6" FlowType="SequenceFlow">
        <Condition Type="CONDITION">a&gt;b</Condition>
    </Transition>
    <Transition Id="11" Name="" From="4" To="7" FlowType="SequenceFlow">
        <Condition Type="OTHERWISE"/>
    </Transition>
</Transitions>
```

7.6.2.2.2.Merge



```
<Activities>
    <Activity Id="3" Name="a1"/>
    <Activity Id="4" Name="a2"/>
    <Activity Id="5" Name="a3"/>
    <Activity Id="6" Name="g1">
        <Route GatewayType="XOR" XORType="Data" MarkerVisible="TRUE"/>
    </Activity>
    <Activity Id="7" Name="a4">
    </Activity>
</Activities>
<Transitions>
    <Transition Id="8" Name="" From="3" To="6" FlowType="SequenceFlow"/>
    <Transition Id="9" Name="" From="4" To="6" FlowType="SequenceFlow"/>
```

```
        <Transition Id="10" Name="" From="5" To="6" FlowType="SequenceFlow"/>
        <Transition Id="11" Name="" From="6" To="7" FlowType="SequenceFlow"/>
    </Transitions>
```

7.6.3. Block Activity

A block activity executes an ActivitySet or self-contained activities/transitions map. From the Block Activity execution proceeds to the first activity in the set (unless otherwise specified by optional properties) and continues within the set until it reaches an exit activity (an activity with no output transitions). Execution then returns to follow the output transitions of the block activity.

A block activity/embedded subprocess call is transactional if the parent Activity element has been specified as transactional.

```
          <xsd:element name="BlockActivity">
             <xsd:complexType>
                <xsd:sequence>
                   <xsd:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="ActivitySetId" type="xsd:NMTOKEN" use="required"/>
                <xsd:attribute name="StartActivityId" type="xsd:NMTOKEN"
      use="optional"/>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
             </xsd:complexType>
          </xsd:element>
```

	Description
ActivitySetId	The ActivitySet to be executed.
StartActivityId	If present, must be the id of a start activity in the ActivitySet referenced by the ActivitySetId attribute of BlockActivity. If not present then the start activity is deducible by some other means. See ActivitySet 7.5.3

Table 35 BlockActivity

7.6.4. Event Activity

An Event is something that "happens" during the course of a business process. These Events affect the flow of the Process and usually have a cause or an impact. The term "event" is general enough to cover many things in a business process. The start of an activity, the end of an activity, the change of state of a document, a message that arrives, etc., all could be considered events. However, BPMN has restricted the use of events to include only those types of events that will affect the sequence or timing of activities of a process. BPMN further categorizes Events into three main types: Start, Intermediate, and End. Start and most Intermediate Events have "Triggers" that define the cause for the event. There are multiple ways that these events can be triggered. End Events may define a "Result" that is a consequence of a Sequence Flow ending. There are multiple types of Results that can be defined**.**

```
      <xsd:element name="Event">
         <xsd:complexType>
            <xsd:choice>
               <xsd:element ref="xpdl:StartEvent" minOccurs="0"/>
               <xsd:element ref="xpdl:IntermediateEvent" minOccurs="0"/>
               <xsd:element ref="xpdl:EndEvent" minOccurs="0"/>
            </xsd:choice>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

Entity Types and	**StartEvent**	**Intermediate Event**	**End Event**	Reference

Attributes(usage within EventType)				
Activity				Used for Compensation. See 7.6.4.4.1
ErrorCode				See 7.6.4.4.2
Implementation	x	x		WebService | Other | Unspecified
LinkId				See 7.6.4.4.4
Message	x	x	x	Section 7.9.2
SubFlow/ProcessRef				See 7.6.4.4.4
Result			x	See xml in **Error! Reference source not found.**
ResultCompensation		x	x	Section 7.6.4.4.1
ResultError		x	x	Section 7.6.4.4.2
ResultMultiple			x	Section 7.6.4.4.3
RuleName				Section 7.6.4.4.8
Target		x		A Target MAY be included for the Intermediate Event. The Target MUST be an activity. This means that the Intermediate Event is attached to the boundary of the activity and is used to signify an exception or compensation for that activity. Section **Error! Reference source not found.**
TimeCycle				Section 7.6.4.4.9
TimeDate				Section 7.6.4.4.9
Trigger	x	x		
TriggerIntermedateMultiple		x		Section 7.6.4.4.6
TriggerMultiple	x			Section 7.6.4.4.7
TriggerResultLink	x	x	x	Section 7.6.4.4.4
TriggerResultMessage	x	x	x	Section 7.6.4.4.5
TriggerRule	x	x		Section 7.6.4.4.8
TriggerTimer	x	x		Section 7.6.4.4.9
WebService	x	x		Section 7.9.4

Table 36: Event

7.6.4.1. Start Event

As the name implies, the Start Event indicates where a particular Process will start. In terms of Sequence Flow, the Start Event starts the flow of the Process, and thus, will not have any incoming Sequence Flow—no Sequence Flow can connect to a Start Event.

There are many ways that a business process can be started (instantiated). The Trigger for a Start Event is designed to show the general mechanism that will instantiate that particular Process. There are six types of Start Events in BPMN: None, Message, Timer, Rule, Link, and Multiple.

```
<xsd:element name="StartEvent">
   <xsd:complexType>
      <xsd:choice minOccurs="0">
         <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
         <xsd:element ref="xpdl:TriggerTimer" minOccurs="0"/>
         <xsd:element ref="xpdl:TriggerRule" minOccurs="0"/>
         <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
         <xsd:element ref="xpdl:TriggerMultiple" minOccurs="0"/>
      </xsd:choice>
      <xsd:attribute name="Trigger" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="None"/>
               <xsd:enumeration value="Message"/>
               <xsd:enumeration value="Timer"/>
               <xsd:enumeration value="Rule"/>
               <xsd:enumeration value="Link"/>
               <xsd:enumeration value="Multiple"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Implementation" use="optional" default="WebService">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="WebService"/>
               <xsd:enumeration value="Other"/>
               <xsd:enumeration value="Unspecified"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Implementation	WebService | Other | Unspecified See 7.9.4
Trigger	None, Message, Timer. Rule, Link, Multiple “None” is used typically for subflow invocations, including embedded subflows.
TriggerMultiple	This means that there are multiple ways of triggering the Process. Only one of them will be required to start the Process. The attributes of the Start Event will define which of the other types of Triggers apply. See 7.6.4.4.7
TriggerResultLink	A Link is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another. Typically, these are two Sub-Processes within the same parent Process. See 7.6.4.4.4
TriggerResultMessage	A message arrives from a participant and triggers the start of the Process. See 7.6.4.4.5
TriggerRule	This type of event is triggered when the conditions for a rule such as “S&P 500 changes by more than 10% since opening,” or “Temperature above 300C” become true. See 7.6.4.4.8
TriggerTimer	A specific time-date or a specific cycle (e.g., every Monday a 9am) can be set that will trigger the start of the Process. See 7.6.4.4.9

Table 37: Start Event Activity

7.6.4.2. Intermediate Event

Intermediate Events occur between a Start Event and an End Event. This is an event that occurs after a Process has been started. It will affect the flow of the process, but will not start or (directly) terminate the process. Intermediate Events can be used to:

• Show where messages are expected or sent within the Process,
• Show delays are expected within the Process,
• Disrupt the Normal Flow through exception handling, or
• Show the extra work required for compensation.

```
<xsd:element name="IntermediateEvent">
   <xsd:complexType>
      <xsd:choice minOccurs="0">
         <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
         <xsd:element ref="xpdl:TriggerTimer" minOccurs="0"/>
         <xsd:element ref="xpdl:ResultError" minOccurs="0"/>
         <xsd:element ref="xpdl:ResultCompensation" minOccurs="0"/>
         <xsd:element ref="xpdl:TriggerRule" minOccurs="0"/>
         <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
         <xsd:element ref="xpdl:TriggerIntermediateMultiple" minOccurs="0"/>
      </xsd:choice>
      <xsd:attribute name="Trigger" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="None"/>
               <xsd:enumeration value="Message"/>
               <xsd:enumeration value="Timer"/>
               <xsd:enumeration value="Error"/>
               <xsd:enumeration value="Cancel"/>
               <xsd:enumeration value="Rule"/>
               <xsd:enumeration value="Link"/>
               <xsd:enumeration value="Compensation"/>
               <xsd:enumeration value="Multiple"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Implementation" use="optional" default="WebService">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="WebService"/>
               <xsd:enumeration value="Other"/>
               <xsd:enumeration value="Unspecified"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Target" type="xsd:NMTOKEN" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Implementation	WebService | Other | Unspecified See 7.9.4
ResultCompensation	This is used for compensation handling--both setting and performing compensation. It calls for compensation if the Event is part of a Normal Flow. It reacts to a named compensation call when attached to the boundary of an activity. See 7.6.4.4.1
ResultError	This is used for error handling--both to set (throw) and to react to (catch) errors. It sets (throws) an error if the Event is part of a Normal Flow. It reacts to (catches) a named error, or to any error if a name is not specified, when attached to the boundary of an activity. See 7.6.4.4.2
Target	A Target MAY be included for the Intermediate Event. The Target MUST be an activity (Sub-Process or Task). This means that the Intermediate Event is attached to the boundary of the activity and is used to signify an exception or compensation for that activity.

	Description
Trigger	None, Message, Timer. Error, Cancel, Rule, Link, Compensation, Multiple. The None and Link Trigger MUST NOT be used when the Event is attached to the boundary of an Activity. The Multiple, Rule, and Cancel Triggers MUST NOT be used when the Event is part of the Normal Flow of the Process. The Cancel Trigger MUST NOT be used when the Event is attached to the boundary of a Transaction or if the Event is not contained within a Process that is a Transaction. “None” is used typically for Intermediate Events that are in the main flow of the Process. It is used for modeling methodologies that use Events to indicate some change of state in the Process.
TriggerCancel	This type of Intermediate Event is used within a Transaction Sub-Process. This type of Event MUST be attached to the boundary of a Sub-Process. It SHALL be triggered if a Cancel End Event is reached within the Transaction Sub-Process. It also SHALL be triggered if a Transaction Protocol “Cancel” message has been received while the Transaction is being performed.
TriggerIntermediateMultiple	This means that there are multiple triggers listed. See 7.6.4.4.7
TriggerResultLink	If the Trigger is a Link, then the LinkId MUST be supplied.See 7.6.4.4.4
TriggerResultMessage	A message arrives from a participant and triggers the Event. This causes the Process to continue if it was waiting for the message, or changes the flow for exception handling. In Normal Flow, Message Intermediate Events can be used for sending messages to a participant. If used for exception handling it will change the Normal Flow into an Exception Flow.. See 7.6.4.4.5
TriggerRule	This type of event is triggered when the conditions for a rule such as “S&P 500 changes by more than 10% since opening,” or “Temperature above 300C” become true. See 7.6.4.4.8
TriggerTimer	A specific time-date or a specific cycle (e.g., every Monday at 9am) can be set that will trigger the Event. If used within the main flow it acts as a delay mechanism. If used for exception handling it will change the Normal Flow into an Exception Flow.See 7.6.4.4.9

Table 38: Intermediate Event Activity

7.6.4.3. *End Event*

As the name implies, the End Event indicates where a process will end. In terms of Sequence Flow, the End Event ends the flow of the Process, and thus, will not have any outgoing Sequence Flow—no Sequence Flow can connect from an End Event.

```
<xsd:element name="EndEvent">
   <xsd:complexType>
      <xsd:choice>
         <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
         <xsd:element ref="xpdl:ResultError" minOccurs="0"/>
         <xsd:element ref="xpdl:ResultCompensation" minOccurs="0"/>
         <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
         <xsd:element ref="xpdl:ResultMultiple" minOccurs="0"/>
      </xsd:choice>
      <xsd:attribute name="Result" use="optional" default="None">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="None"/>
               <xsd:enumeration value="Message"/>
               <xsd:enumeration value="Error"/>
               <xsd:enumeration value="Cancel"/>
               <xsd:enumeration value="Compensation"/>
               <xsd:enumeration value="Link"/>
               <xsd:enumeration value="Terminate"/>
```

```
                        <xsd:enumeration value="Multiple"/>
                     </xsd:restriction>
                  </xsd:simpleType>
               </xsd:attribute>
               <xsd:attribute name="Implementation" use="optional" default="WebService">
                  <xsd:simpleType>
                     <xsd:restriction base="xsd:NMTOKEN">
                        <xsd:enumeration value="WebService"/>
                        <xsd:enumeration value="Other"/>
                        <xsd:enumeration value="Unspecified"/>
                     </xsd:restriction>
                  </xsd:simpleType>
               </xsd:attribute>
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
         </xsd:element>
```

	Description
Implementation	WebService | Other | Unspecified See 7.9.4
Result	None, Message, Error, Cancel, Compensation, Link, Terminate, Multiple. 'None' is used then the modeler does not display the type of Event. It is also used to show the end of a Sub-Process that ends, which causes the flow goes back to its Parent Process. "Terminate" indicates that all activities in the Process should be immediately ended. This includes all instances of Multi-Instances. The Process is ended without compensation or event handling.
ResultCompensation	This type of End will indicate that a Compensation is necessary. The Compensation identifier will trigger an Intermediate Event when the Process is rolling back.See 7.6.4.4.1
ResultError	This type of End indicates that a named Error should be generated. This Error will be caught by an Intermediate Event within the Event Context.See 7.6.4.4.2
TriggerResultLink	A Link is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another. Typically, these are two Sub-Processes within the same parent Process. A Token arriving at Link End Event will immediately jump to its corresponding target Start or Intermediate Event.See 7.6.4.4.4
ResultMultiple	This means that there are multiple consequences of ending the Process. All of them will occur (e.g., there might be multiple messages sent). The attributes of the End Event will define which of the other types of Results apply.See 7.6.4.4.3

Table 39: End Event Activity

7.6.4.4. Common Elements Used in Start, Intermediate and End Events

7.6.4.4.1. ResultCompensation

For an Intermediate Event within Normal Flow or an End Event:
If the Trigger is a Compensation, then the Id of the Activity that needs to be compensated MUST be supplied. This "throws" the compensation.

For an Intermediate Event attached to the boundary of an Activity: This Event "catches" the compensation. No further information is required. The Id of the activity the Event is attached to will provide the Id necessary to match the compensation event with the event that "threw" the compensation.

Compensation requires specific notation and is a special circumstance that occurs outside the Normal Flow of the Process. For this reason, the Compensation Intermediate Event does not have an outgoing Sequence Flow, but instead has an outgoing directed Association. The target of this Association is the activity that will compensate for the work done in the source activity, and will be referred to as the Compensation Activity. The Compensation Activity is special in that it does not follow the normal Sequence Flow rules--as mentioned, it is outside the Normal Flow of the

Process. This activity cannot have any incoming or outgoing Sequence Flow. The Compensation marker (as is in the Compensation Intermediate Event) will be displayed in the bottom center of the Activity to show this status of the activity.

```
<xsd:element name="ResultCompensation">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ActivityId" type="xsd:NMTOKEN" use="optional"/>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>
</xsd:element>
```

	Description
ActivityId	See Discussion above.

Table 40: Event ResultCompensation

7.6.4.4.2.ResultError

This is used for error handling--both to set (throw) and to react to (catch) errors. It sets (throws) an error if the Event is part of a Normal Flow (either Intermediate or End Event). It reacts to (catches) a named error, or to any error if a name is not specified, when attached to the boundary of an activity.

```
<xsd:element name="ResultError">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ErrorCode" type="xsd:string" use="required"/>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>
</xsd:element>
```

	Description
ErrorCode	String designates the error. Code.

Table 41: Event ResultError

7.6.4.4.3.ResultMultiple

For an End Event:
This means that there are multiple consequences of ending the Process. All of them will occur (e.g., there might be multiple messages sent). The attributes of the End Event will define which of the other types of Results apply.

```
<xsd:element name="ResultMultiple">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
      <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
      <xsd:element ref="xpdl:ResultCompensation" minOccurs="0"/>
      <xsd:element ref="xpdl:ResultError" minOccurs="0"/>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>
</xsd:element>
```

	Description
	List of all applicable result elements.

Table 42: Event ResultMultiple

7.6.4.4.4.TriggerResultLink

For an End Event:
A Token arriving at Link End Event will immediately jump to its corresponding target Start or Intermediate Event, based on the LinkId.

For an Intermediate Event:
A Link is a mechanism for connecting an End Event (Result) of one Process to an Intermediate Event (Trigger) in another Process. Paired Intermediate Events can also be used as “Go To” objects within a Process.

For all Event Types:
The LinkId MUST be entered.
The ProcessRef MUST be entered. The identified Process MAY be the same Process as that of the Link Event.

```
<xsd:element name="TriggerResultLink">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="LinkId" type="xsd:NMTOKEN" use="required"/>
    <xsd:attribute name="ProcessRef" type="xsd:NMTOKEN" use="required"/>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>
</xsd:element>
```

	Description
LinkId	Unique name for this link.
ProcessRef	Id of the parent (embedding) process or Id of the subprocess.

Table 43: Event TriggerResultLink

7.6.4.4.5.TriggerResultMessage

For Start Event:
A message arrives from a participant and triggers the start of the Process.

For End Event:
This type of End indicates that a message is sent to a participant at the conclusion of the Process.

For Intermediate Event:
A message arrives from a participant and triggers the Event. This causes the Process to continue if it was waiting for the message, or changes the flow for exception handling. In Normal Flow, Message Intermediate Events can be used for sending messages to a participant. If used for exception handling it will change the Normal Flow into an Exception Flow.

```
<xsd:element name="TriggerResultMessage">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Message" type="xpdl:MessageType"/>
      <xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
      <xsd:any namespace="##other" processContents="lax"
```

```
            minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:complexType>
</xsd:element>
```

	Description
Message	Describes the message (See 7.9.2)
WebServiceOperation	Describes the web services operation (See 7.9.4)

Table 44: Event TriggerResultMessage

7.6.4.4.6.TriggerIntermediateMultiple

If the Trigger is a Multiple, then each Trigger on the list MUST have the appropriate data as specified for the attributes. The Trigger MUST NOT be of type None or Multiple.

```
<xsd:element name="TriggerIntermediateMultiple">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
            <xsd:element ref="xpdl:TriggerTimer" minOccurs="0"/>
            <xsd:element ref="xpdl:ResultError" minOccurs="0"/>
            <xsd:element ref="xpdl:ResultCompensation" minOccurs="0"/>
            <xsd:element ref="xpdl:TriggerRule" minOccurs="0"/>
            <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
            <xsd:any namespace="##other" processContents="lax"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:complexType>
</xsd:element>
```

	Description
	List of applicable triggers.

Table 45: Event TriggerIntermediat Multiple

7.6.4.4.7.TriggerMultiple

If the Trigger attribute is a Multiple, then a list of two or more Triggers MUST be provided. Each Trigger MUST have the appropriate data. The Trigger MUST NOT be of type None or Multiple.

```
<xsd:element name="TriggerMultiple">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
            <xsd:element ref="xpdl:TriggerTimer" minOccurs="0"/>
            <xsd:element ref="xpdl:TriggerRule" minOccurs="0"/>
            <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
            <xsd:any namespace="##other" processContents="lax"
                minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:complexType>
</xsd:element>
```

	Description
	List of applicable triggers.

Table 46: Event TriggerMultiple

7.6.4.4.8.TriggerRule

If the Trigger is a Rule, then a Rule MUST be entered.

For Intermediate Event:
This is only used for exception handling. This type of event is triggered when a Rule becomes true. A Rule is an expression that evaluates some Process data.

```
<xsd:element name="TriggerRule">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="RuleName" type="xsd:string" use="required"/>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>
</xsd:element>
```

	Description
RuleName	A string which provides a way of locating the rule, or is the rule.

Table 47: Event TriggerRule

7.6.4.4.9.TriggerTimer

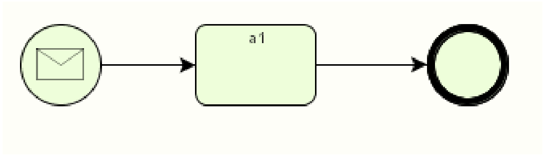
If the Trigger is a Timer, then a TimeDate MAY be entered. If a TimeDate is not entered, then a TimeCycle MUST be entered (see the attribute below).

```
<xsd:element name="TriggerTimer">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="TimeDate" type="xsd:string" use="optional"/>
    <xsd:attribute name="TimeCycle" type="xsd:string" use="optional"/>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>
</xsd:element>
```

	Description
TimeDate	String (Should be Date)
TimeCycle	String (Should be expression designating repetitive occurrence based on time.

Table 48: Event TriggerTimer

7.6.4.5. Examples of Events and their representation



```
<Activities>
    <Activity Id="3" Name="">
        <Event >
            <StartEvent Trigger="Message"></StartEvent>
        </Event>
    </Activity>
    <Activity Id="4" Name="a1"/>
```

```
          <Activity Id="5" Name="">
               <Event >
                    <EndEvent Result="None"/>
               </Event>
          </Activity>
     </Activities>
     <Transitions>
          <Transition Id="6" Name="" From="3" To="4" FlowType="SequenceFlow"/>
          <Transition Id="7" Name="" From="4" To="5" FlowType="SequenceFlow"/>
     </Transitions>
```

7.6.5. Implementation Alternatives

It is assumed that the execution of the Activity is atomic with respect to the data under control of the Process or Workflow engine. That implies that in the case of a system crash, an abort, or a cancellation of the Activity, the Relevant data field and the control data are rolled back (automatically or by other means), or an appropriate compensating activity is applied. (This does not necessarily hold for audit data.) This version of the specification does not include any specific controls over data synchronization or recovery (for example between execution, subflows or applications under execution.

```
<xsd:element name="Implementation">
   <xsd:complexType>
      <xsd:choice minOccurs="0">
         <xsd:element ref="xpdl:No" minOccurs="0"/>
         <xsd:element ref="deprecated:Tool" minOccurs="0" maxOccurs="unbounded"/>
         <xsd:element ref="xpdl:Task" minOccurs="0"/>
         <xsd:element ref="xpdl:SubFlow" minOccurs="0"/>
         <xsd:element ref="xpdl:Reference" minOccurs="0"/>
      </xsd:choice>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

An Activity may be implemented in one of four ways as described in the following table:

	Description
No implementation	Implementation by manual procedures (i.e. not supported by process or workflow)
Tool	Implementation is supported by (one or more) application(s). Deprecated, see Task
Task	Implementation by a BPMN Task or Application(Tool). See section 7.6.5.3.
SubFlow/ ProcessRef	Implementation by another process. Note that BlockActivity is used for ActivitySet/Embedded subprocess.
Reference	Implementation by BPMN reference. See section 7.6.5.5.

Table 49: Implementation Alternatives

7.6.5.1. No Implementation

No Implementation means that the implementation of this Activity is not supported by Workflow using automatically invoked applications or procedures. Two Alternatives have been identified as to how this may be used:

It is a Manual Activity. In this case FinishMode value Manual is required.

It is an "implicit" activity, which is known to the Process Engine (e.g. by vendor-specific Extended Attributes) in terms of any processing requirements. An example is the Pre- and Post-processing Activities in a Process, which generate and clear hidden data when starting and terminating a process (e.g. managing the relationship to imaging system and archive). In this case the StartMode and FinishMode values Automatic are common.

(Note that application initiation may still be handled directly by the participant under local control in a manual activity; this lies outside the scope of the specification.)

```
<xsd:element name="No">
```

```
         <xsd:complexType>
            <xsd:sequence>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

7.6.5.2. Tool

Deprecated. See TaskApplication 7.6.5.3.8

7.6.5.3. Task

The Activity is implemented by one of the seven BPMN tasks or by an Application (Tool).

```
      <xsd:element name="Task">
         <xsd:complexType>
            <xsd:choice>
               <xsd:element ref="xpdl:TaskService"/>
               <xsd:element ref="xpdl:TaskReceive"/>
               <xsd:element ref="xpdl:TaskManual"/>
               <xsd:element ref="xpdl:TaskReference"/>
               <xsd:element ref="xpdl:TaskScript"/>
               <xsd:element ref="xpdl:TaskSend"/>
               <xsd:element ref="xpdl:TaskUser"/>
               <xsd:element ref="xpdl:TaskApplication"/>
            </xsd:choice>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

7.6.5.3.1.TaskManual

```
      <xsd:element name="TaskManual">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element ref="xpdl:Performers"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
Performers	A list of performers that will be performing the Manual Task. The Performers entry could be in the form of a specific individual, a group, or an organization. Similar to the 'Implementation:No" activity (see 7.6.5.1)

Table 50: Task Manual

7.6.5.3.2.TaskReceive

```
      <xsd:element name="TaskReceive">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="Message" type="xpdl:MessageType"/>
               <xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="Instantiate" type="xsd:boolean" use="required"/>
            <xsd:attribute name="Implementation" use="optional" default="WebService">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:NMTOKEN">
                     <xsd:enumeration value="WebService"/>
                     <xsd:enumeration value="Other"/>
                     <xsd:enumeration value="Unspecified"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
Implementation	WebService | Other | Unspecified
Instantiate	Receive Tasks can be defined as the instantiation mechanism for the Process with the Instantiate attribute. This attribute MAY be set to true if the Task is the first activity after the Start Event or a starting Task if there is no Start Event. Multiple Tasks MAY have this attribute set to True.
Message	A Message for the Message attribute MUST be entered. This indicates that the Message will be received by the Task. The Message in this context is equivalent to an in-only message pattern (Web service). A corresponding incoming Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required. Section 7.9.2
WebServiceOperation	Describes the web service operation to be used by this task. (See section 7.9.4)

Table 51: Task Receive

7.6.5.3.3.TaskReference

```
      <xsd:element name="TaskReference">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="TaskRef" type="xsd:NMTOKEN" use="required"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
TaskRef	The Task being referenced MUST be identified.

Table 52: TaskReference

7.6.5.3.4.TaskSend

```
      <xsd:element name="TaskSend">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="Message" type="xpdl:MessageType"/>
               <xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
               <xsd:element ref="xpdl:WebServiceFaultCatch"
                  minOccurs="0" maxOccurs="unbounded"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="Implementation" use="optional" default="WebService">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:NMTOKEN">
                     <xsd:enumeration value="WebService"/>
                     <xsd:enumeration value="Other"/>
                     <xsd:enumeration value="Unspecified"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
Implementation	WebService | Other | Unspecified
Message	A Message for the Message attribute MUST be entered. This indicates that the Message will be sent by the Task. The Message in this context is equivalent to an out-only message pattern (Web service). A corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required. See Section 7.9.2
WebServiceOperation	Describes the web services operation to be used by this task. (See section 7.9.4)
WebServiceFaultCatch	Describes how to process faults generated by the web service operation in this task. (See section 7.9.5)

Table 53: TaskSend

7.6.5.3.5.TaskService

```
      <xsd:element name="TaskService">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="MessageIn" type="xpdl:MessageType"/>
               <xsd:element name="MessageOut" type="xpdl:MessageType"/>
               <xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
               <xsd:element ref="xpdl:WebServiceFaultCatch"
                  minOccurs="0" maxOccurs="unbounded"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="Implementation" use="optional" default="WebService">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:NMTOKEN">
                     <xsd:enumeration value="WebService"/>
                     <xsd:enumeration value="Other"/>
                     <xsd:enumeration value="Unspecified"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
Implementation	WebService | Other | Unspecified
MessageIn	A Message for the InMessage attribute MUST be entered. This indicates that the Message will be sent at the start of the Task, after the availability of any defined InputSets (child element of Activity). A corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required. See Section 7.9.2
MessageOut	A Message for the OutMessage attribute MUST be entered. The arrival of this message marks the completion of the Task, which may cause the production of an OutputSet (child element of Activity). A corresponding incoming Message Flow MAYbe shown on the diagram. However, the display of the Message Flow is not required. See Section 7.9.2

	Description
WebServiceFaultCatch	Describes how to process faults generated by the web service operation in this task. (See section 7.9.5)
WebServiceOperation	Describes the web services operation to be used by this task. (See section 7.9.4)

Table 54: Task Service

7.6.5.3.6.TaskScript

```
<xsd:element name="TaskScript">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Script" type="xpdl:ExpressionType"/>
         <xsd:any namespace="##other" processContents="lax"
           minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Script	The modeler MAY include a script that can be run when the Task is performed. If a script is not included, then the Task will act equivalent to a TaskType of None.

Table 55: Task Script

7.6.5.3.7.TaskUser

```
<xsd:element name="TaskUser">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Performers"/>
         <xsd:element name="MessageIn" type="xpdl:MessageType"/>
         <xsd:element name="MessageOut" type="xpdl:MessageType"/>
         <xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
           minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Implementation" use="optional" default="WebService">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="WebService"/>
               <xsd:enumeration value="Other"/>
               <xsd:enumeration value="Unspecified"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Implementation	WebService | Other | Unspecified
MessageIn	A Message for the InMessage attribute MUST be entered. This indicates that the Message will be sent at the start of the Task, after the availability of any defined InputSets. A corresponding outgoing Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required.See Section 7.9.2

	Description
MessageOut	A Message for the OutMessage attribute MUST be entered. The arrival of this message marks the completion of the Task, which may cause the production of an OutputSet. A corresponding incoming Message Flow MAY be shown on the diagram. However, the display of the Message Flow is not required. See Section 7.9.2
Performers	One or more Performers MAY be entered. The Performers attribute defines the human resource that will be performing the Task. The Performers entry could be in the form of a specific individual, a group, or an organization.
WebServiceOperation	Describes the web services operation to be used by this task. (See section 7.9.4)

Table 56: Task User

7.6.5.3.8. TaskApplication (Tool)

The Activity is implemented by (one or more) tools. A tool may be an application program (link to entity Application); which may be invoked via IF3 - see the Workflow Client Application API (WAPI - Interface 2).

```
<xsd:element name="TaskApplication">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:choice minOccurs="0">
            <xsd:element ref="xpdl:ActualParameters"/>
            <xsd:element ref="xpdl:DataMappings"/>
         </xsd:choice>
         <xsd:element ref="xpdl:Description" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="PackageRef" type="xsd:NMTOKEN" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Actual Parameters	A list of parameters to be passed to the subflow. See section 7.1.3.3.
DataMappings	Alternative approach to passing values between process and application. See section 7.6.5.4.1
Description	Textual description
Extended Attributes	Optional extensions to meet individual implementation needs
Id	Identifier used to identify the application or procedure, depending on the Type.
Name	Name used to identify the application or procedure.
PackageRef	Used if the application is not in this package.

Table 57: Tool

7.6.5.4. SubFlow/ProcessRef

The Activity is refined as a subflow. The subflow may be executed synchronously or asynchronously. The subflow identifiers used are inherited from the surrounding Package declaration.

SubFlow calls are transactional if the parent Activity has been defined that way. See Table 32.

In the case of ***asynchronous execution*** the execution of the Activity is continued after a process instance of the referenced Process Definition is initiated (in this case execution proceeds to any post activity split logic after subflow initiation. No return parameters are supported from such called processes. Synchronization with the initiated subflow, if required, has to be done by other means such as events, not described in this document. This style of subflow is characterized as chained (or forked) subflow operation.

In the case of ***synchronous execution*** the execution of the Activity is suspended after a process instance of the referenced Process Definition is initiated. After execution termination of this process instance the Activity is resumed. Return parameters may be used between the called and calling processes on completion of the subflow. This style of subflow is characterized as hierarchic subflow operation.

```
<xsd:element name="SubFlow">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:choice minOccurs="0">
            <xsd:element ref="xpdl:ActualParameters"/>
            <xsd:element ref="xpdl:DataMappings"/>
         </xsd:choice>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:string" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="Execution" use="optional" default="SYNCHR">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="ASYNCHR"/>
               <xsd:enumeration value="SYNCHR"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="PackageRef" type="xsd:NMTOKEN" use="optional"/>
      <xsd:attribute name="InstanceDataField" type="xsd:string" use="optional"/>
      <xsd:attribute name="StartActivitySetId" type="xsd:NMTOKEN"
         use="optional"/>
      <xsd:attribute name="StartActivityId" type="xsd:NMTOKEN"
         use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Actual Parameters	A list of parameters to be passed to the subflow. See section 7.1.3.3.
DataMapping	Alternative approach to passing values between caller and called process. See section 7.6.5.4.1
PackageRef	Used if the subflow/process is not this package.
Execution	ASYNCHR Executed asynchronously. SYNCHR Executed synchronously. Is default.
ExtendedAttributes	See 6.4.14
Id	Used to identify the process that is invoked.
InstanceDataField	The name of the DataField in which to store the subflow instance id for subsequent use such as messaging or correlation. Typically used with ASYNCHR execution.
Name	The name of the SubFlow/Process.
StartActivityId	If present, StartActivityId must be the id of a start activity: • In the referenced Activity Set if that's present • In the top level activities of the referenced process otherwise
StartActivitySetId	If present, StartActivitySetId must match the id of an activity set in the referenced process.

Table 58: SubFlow

7.6.5.4.1. DataMapping

```
<xsd:element name="DataMapping">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Actual" type="xpdl:ExpressionType"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Formal" type="xsd:string" use="required"/>
      <xsd:attribute name="Direction" default="IN">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="IN"/>
               <xsd:enumeration value="OUT"/>
               <xsd:enumeration value="INOUT"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="DataMappings">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:DataMapping" 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:element>
```

	Description
Actual	Name of DataField whose value should be passed . If Direction is 'IN' can be an expression.
Direction	IN | OUT | INOUT
Formal	Name of DataField in receiving subprocess or application

Table 59: DataMapping

7.6.5.5. Reference

There may be times where a modeler may want to reference another Sub-Process that has been defined. If the two SubProcesses share the exact same behavior and properties, then by one referencing the other, the attributes that define the behavior only have to be created once and maintained in only one location.

There may be times where a modeler may want to reference another activity that has been defined. If the two (or more) activities share the exact same behavior, then by one referencing the other, the attributes that define the behavior only have to be created once and maintained in only one location.

```
<xsd:element name="Reference">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="ActivityId" type="xsd:NMTOKEN" use="required"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Activity Id	The Id of the activity that defines the desired behaviour.

Table 60: Reference

7.6.6. Performer Relationship

The relationship of the Activity to a (potential) performer is given by the Participant Assignment attribute. It provides a link to the entity Participant. Default: Any Participant.

The Participant identifiers used in the Performer attribute have either to be declared in the surrounding Process definition or are inherited from the surrounding Package declaration or coming from external packages.

The question whether the expression evaluation results in an empty set of performers or a non unique performer is to be handled by the process or workflow management system at run time or, where defined, by the external resource repository or organizational model. The runtime resolution of both cases is outside the scope of this specification

- In the first case (empty set) the engine may e.g. retry at a later time, or it may signal this to the supervisor of the process. The approach used is local to the WFMS and does not form part of this specification.
- The second case (non-unique) may arise where the performer definition is by function/skill type (defined as "Role") and/or is an organization unit, which is itself a container for a set of participants. In these situations the approach adopted for participant assignment is local to the WFMS and does not form part of this specification. Common scenarios are:
    - Where an activity includes multiple work items that may be implemented in parallel, separate work items may be presented to a number of performers.
    - In other situations the activity may be assigned according to a local load-balancing algorithm or presented to multiple potential performers in their work lists and assigned to the first accepting participant. (It is the responsibility of the engine to provide the appropriate behavior.)
    - The assignment of an activity to an organizational unit (e.g. a department) may result in the activity being offered to all members of the organizational unit and assigned to the first accepting participant or allow the manager of the unit to redirect the activity to a designated departmental member.

In all cases the participant assignments defined within the meta-model and expressed in XPDL only relate Activities to defined Participants (including the use of expressions and defined Functions) and do not differentiate between cases where the defined Participant is atomic (e.g. a person) or not (e.g. a team). The local behavior of the engine and the resource repository or organizational model in handling these situations is not defined.

7.6.7. Deadline

```
<xsd:element name="Deadline">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="DeadlineDuration" type="xpdl:ExpressionType"
            minOccurs="0"/>
         <xsd:element name="ExceptionName" minOccurs="0">
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                     <xsd:anyAttribute namespace="##other" processContents="lax"/>
                  </xsd:extension>
               </xsd:simpleContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Execution">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
```

```
                    <xsd:enumeration value="ASYNCHR"/>
                    <xsd:enumeration value="SYNCHR"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:complexType>
</xsd:element>
```

	Description	
Execution	Define the system behaviour on raising the arrival of the deadline	
	ASYNCHR	The deadline is to be raised asynchronously. This is an implicit AND SPLIT operation, where the activity continues and another thread is started following the named exception transition. Another deadline may occur on the same activity, because it continues running.
	SYNCHR	The activity is completed abnormally and flow continues on the named exception transition.
DeadlineDuration	An expression indicating the time of the deadline. This expression is implementation dependent and may include at least: Times relative to the beginning of the activity. (2 days) Fixed times (January 1) or (January 1, 2002) Times computed using relevant data field (varName days)	
ExceptionName	The name of the exception to be raised on arrival of the deadline.	

Table 61: Deadline

Deadlines are used to raise an exception upon the expiration of a specific period of time.

Upon the arrival of a deadline, an exception condition is raised and the appropriate exception transitions are followed. If the deadline is synchronous, then the activity is terminated before flow continues on the exception path. If the deadline is asynchronous, then an implicit AND SPLIT is performed, and a new thread of processing is started on the appropriate exception transition. Asynchronous exceptions can cause side effects, and should be used carefully. Some of these side effects are discussed later in this section.

A sample deadline is below. In the sample, an asynchronous “notifyException” will be raised after 3 days. The activity will continue normally at this point. If the activity is still executing after 5 days it will be terminated and a “timeoutException” will be raised.

Sample Deadline

```
<Deadline Execution="ASYNCHR">
      <DeadlineDuration>3 days</ DeadlineDuration>
      <ExceptionName>notifyException</ExceptionName>
</Deadline>
<Deadline Execution="SYNCHR">
      <DeadlineDuration>5 days</DeadlineDuration>
      <ExceptionName>timeoutException</ExceptionName>
</Deadline>
```

The syntax of the deadline duration is implementation dependent. The duation may be relative or absolute and may use relevant data field.

If a synchronous deadline occurs on a block activity or a subflow, stopping the activity includes stopping all the threads in the block, or the subflow and all its threads and nested subflows as well. From a modeling perspective, this can be treated as “immediate termination.” If an engine chooses to deviate from this, such as allowing an in-process manual activity to complete, it should document this behavior.

An asynchronous exception can be a powerful tool, allowing intermediate notification and graceful process termination by altering relevant data field. But an asynchronous exception can also create race conditions and possible side effects. For instance, the running activity could complete while the asynchronous exception is being processed. In addition, because an implicit split is performed, flow control can be complicated if the asynchronous exception processing joins back up with the deadlined processing thread. Care must be taken by the designer to properly handle race conditions and avoid unwanted side effects.

7.6.8. Simulation Information

```
<xsd:element name="SimulationInformation">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Cost"/>
         <xsd:element ref="xpdl:TimeEstimation"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Instantiation">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="ONCE"/>
               <xsd:enumeration value="MULTIPLE"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="TimeEstimation">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:WaitingTime" minOccurs="0"/>
         <xsd:element ref="xpdl:WorkingTime" minOccurs="0"/>
         <xsd:element ref="xpdl:Duration" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="WaitingTime">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="WorkingTime">
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Duration">
```

```
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="Cost">
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>
```

	Description	
Cost	Average cost.	
Duration	Expected duration time to perform a task in units of DurationUnit.	
Instantiation	Defines the capability of an activity to be activated. defines how many times an Activity can be activated for higher throughput (e.g. how many individuals can capture a role). This can be once or many times (multiple).	
	ONCE	The Activity can only be instantiated once. Default.
	MULTIPLE	The Activity can be instantiated multiple times.
Time Estimation	Expected duration (summary of working time, waiting time, and duration) in units of DurationUnit.	
Waiting Time	Average waiting time in units of DurationUnit.	
Working Time	Average working time in units of DurationUnit.	

Table 62: Simulation Information

7.6.9. Transition Restriction

```
      <xsd:element name="TransitionRestriction">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element ref="xpdl:Join" minOccurs="0"/>
               <xsd:element ref="xpdl:Split" minOccurs="0"/>
               <xsd:any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
         </xsd:complexType>
      </xsd:element>
```

	Description
Join	Specifies that the incoming Transitions of the Activity are JOIN-ed
Split	Specifies that the outgoing Transitions of the Activity are SPLIT-ed

Table 63: Transition Restrictions

7.6.9.1. Join

A join describes the semantics of an activity with multiple incoming Transitions.

```
<xsd:element name="Join">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Type">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="AND"/>
               <xsd:enumeration value="XOR"/>
               <xsd:enumeration value="XOREVENT"/>
               <xsd:enumeration value="OR"/>
               <xsd:enumeration value="COMPLEX"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="IncomingCondtion" type="xsd:string"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description	
Type	AND	Join of (all) concurrent threads within the process instance with incoming transitions to the activity: Synchronization is required. The number of threads to be synchronized might be dependent on the result of the conditions of previous AND split(s). Equivalent to BPMN 'OR' gateway merge logic. BPMN 'AND' gateway merge logic requires all incoming transition theads to be synchronized, regardless of previous splits.
	OR	See above.
	XOR	Join for alternative threads: No synchronisation is required.
	XOREVENT	Identical to XOR for merges.
	COMPLEX	This makes use of the attribute **IncomingCondition**. It determines which of the incoming transitions/Sequence Flow are required for the Process to continue. The expression may refer to process data and the status of the incoming Sequence Flow. For example, an expression may specify that any 3 out of 5 incoming Tokens will continue the Process. Another example would be an expression that specifies that a Token is required from Sequence Flow "a" and that a Token from either Sequence Flow "b" or "c" is acceptable. However, the expression should be designed so that the Process is not stalled at that location.

Table 64: *Join*

The AND join can be seen as a " rendezvous precondition" of the Activity; the activity is not initiated until the transition conditions on all incoming routes evaluate true.

The XOR join initiates the Activity when the transition conditions of any (one) of the incoming transitions evaluates true.

7.6.9.2. Split

A split describes the semantics where multiple outgoing Transitions for an Activity exist.

```
<xsd:element name="Split">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:TransitionRefs" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Type">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="AND"/>
               <xsd:enumeration value="XOR"/>
               <xsd:enumeration value="XOREVENT"/>
               <xsd:enumeration value="OR"/>
               <xsd:enumeration value="COMPLEX"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="OutgoingCondition" type="xsd:string"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="TransitionRef">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="TransitionRefs">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:TransitionRef"
            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:element>
```

	Description
Transition Refs	A list of outgoing transitions from the Activity. Each transition is identified by its Id.

	Description	
Type	AND	Defines a number of possible concurrent threads represented by the outgoing Transitions of this Activity. If the Transitions have conditions the actual number of executed parallel threads is dependent on the conditions associated with each transition, which are evaluated concurrently. Note that the BPMN 'AND' gateway is a special case where none of the transitions (sequence flow gates) have conditions.
	OR	Another BPMN variant of AND. Here the transitions may have conditions and multiple paths out may be chosen. The difference is that a default path may also be specified so that, in case none of the other paths are chosen, the default will be selected
	XOR	List of Identifiers of outgoing Transitions of this Activity, representing. alternatively executed transitions. The decision as to which single transition route is selected is dependent on the conditions of each individual transition as they are evaluated in the sequence specified in the list. The first TRUE condition (or no condition) ends the list evaluation.An OTHERWISE transition should be chosen if no prior transition is chosen. Equivalent to BPMN XOR data based Gateway.
	XOREVENT	Similar to XOR except that none of the transitions have conditions and there can be no OTHERWISE transition. The target activities must be Tasks with the TaskType attribute set to Receive or Intermediate Events with the Trigger attribute set to Message, Timer, Rule, or Link. If one transition/Gate target is a Task, then an Intermediate Event with a Trigger Message MUST NOT be used as a target for another transition/Gate. That is, messages MUST be received by only Receive Tasks or only Message Events, but not a mixture of both for a given Gateway.
	COMPLEX	This makes use of the attribute **OutgoingCondition**. It determines which of the outgoing transitions/Sequence Flow will be chosen for the Process to continue. The expression may refer to process data and the status of the incoming Sequence Flow. For example, an expression may evaluate Process data and then select different sets of outgoing Sequence Flow, based on the results of the evaluation. However, The expression should be designed so that at least one of the outgoing Sequence Flow will be chosen.

Table 65: Split

An AND split with transitions having conditions may be referred to as "conditional AND", "multiple-choice OR", or "nonexclusive OR", respectively. The number of actual concurrent threads is determined at execution time when evaluating the conditions. Following such an AND split the process instance (or thread of the process instance) is forked into a number of separate execution threads which result from the transitions condition evaluation. (Note that no list of identifiers is required since all outgoing transitions from the activity are evaluated and no sequence is necessary.)

If within the AND_SPLIT or XOR_SPLIT there is a transition having condition OTHERWISE, then a two-step evaluation is performed. In the first step evaluation is made of all the Transitions except that within the OTHERWISE condition. If none of them (including those having no condition) evaluate to TRUE, then in the second step the OTHERWISE transition is evaluated (only one transition with an OTHERWISE clause is permitted in the list of outgoing transitions from an activity).

An OTHERWISE alternative can be used to guarantee that there is no undefined status from the Process execution (i.e. at least one outgoing transition from an activity will always occur).

7.6.10. Conformance Classes

There are Conformance Classes restricting the Activity-Transition Net.

The following Conformance Classes are defined in the package:

- NON-BLOCKED
    - There is no restriction for this class.
- LOOP-BLOCKED
    - The Activities and Transitions of a Process Definition form an acyclic graph (or set of disjoint acyclic graphs).
- FULL-BLOCKED
    - For each join (or respectively split) there is exactly one corresponding split (or respectively join) of the same kind. In an AND split no conditions are permitted; in a XOR split an unconditional or OTHERWISE Transition is required if there is a Transition with a condition (i.e. an undefined result of transition evaluation is not permitted).

```
<xsd:element name="ConformanceClass">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="GraphConformance" use="optional" default="NON_BLOCKED">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="FULL_BLOCKED"/>
               <xsd:enumeration value="LOOP_BLOCKED"/>
               <xsd:enumeration value="NON_BLOCKED"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Graph Conformance	FULL, LOOP or NON as discussed above.

Table 66: Conformance Class

7.6.11. InputSets

The InputSets attribute defines the data requirements for input to the activity. Zero or more InputSets MAY be defined. Each InputSet is sufficient to allow the activity to be performed (if it has first been instantiated by the appropriate signal arriving from an incoming Sequence Flow).

One or more Inputs MUST be defined for each InputSet. An Input is an Artifact, usually a Document Object. Note that the Artifacts MAY also be displayed on the diagram and MAY be connected to the activity through an Association--however, it is not required for them to be displayed.

```
<xsd:element name="Input">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="ArtifactId" type="xsd:NMTOKEN" use="required"/>
```

```
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="InputSet">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xpdl:Input" maxOccurs="unbounded"/>
                <xsd:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="InputSets">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xpdl:InputSet" maxOccurs="unbounded"/>
                <xsd:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
```

	Description
Artifact	See section 6.4.7. and 7.1.7.
Input	A list of Artifacts.
InputSet	A list of Inputs.

Table 67: Input

7.6.12. OutputSets

The OutputSets attribute defines the data requirements for output from the activity. Zero or more OutputSets MAY be defined. At the completion of the activity, only one of the OutputSets may be produced--It is up to the implementation of the activity to determine which set will be produced. However, the IORules attribute MAY indicate a relationship between an OutputSet and an InputSet that started the activity.

One or more Outputs MUST be defined for each OutputSet. An Output is an Artifact, usually a Document Object. Note that the Artifacts MAY also be displayed on the diagram and MAY be connected to the activity through an Association--however, it is not required for them to be displayed.

```
    <xsd:element name="Output">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:any namespace="##other" processContents="lax"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="ArtifactId" type="xsd:NMTOKEN" use="required"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="OutputSet">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xpdl:Output" maxOccurs="unbounded"/>
                <xsd:any namespace="##other" processContents="lax"
```

```
               minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="OutputSets">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="xpdl:OutputSet" maxOccurs="unbounded"/>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>
```

	Description
Artifact	See section 6.4.7. and 7.1.7.
Output	A list of Artifacts.
OutSet	A list of outputs

Table 68: Output

7.6.13. Transaction

A Sub-Process activity, whether it is independent (implemented by subflow) or embedded (block activity) , can be set as being a Transaction, which will have a special behavior that is controlled through a transaction protocol (such as BTP or WSTransaction).

There are three basic outcomes of a Transaction:
• Successful completion: this will be shown as a normal Sequence Flow that leaves the Sub-Process.
• Failed completion (Cancel):
When a Transaction is cancelled, then the activities inside the Transaction will be subjected to the cancellation actions, which could include rolling back the process and compensation for specific activities. Note that other mechanisms for interrupting a Sub-Process will not cause Compensation (e.g., Error, Timer, and anything for a non-Transaction activity). A Cancel Intermediate Event, attached to the boundary of the activity, will direct the flow after the Transaction has been rolled back and all compensation has been completed. The Cancel Intermediate Event can only be used when attached to the boundary of a Transaction activity. It cannot be used in any Normal Flow and cannot be attached to a non-Transaction activity. There are two mechanisms that can signal the cancellation of a Transaction:
• A Cancel End Event is reached within the Transaction Sub-Process. A Cancel End Event can only be used within a Sub-Process that is set to a Transaction.
• A Cancel Message can be received via the Transaction Protocol that is supporting the execution of the Sub-Process.
• Hazard: This means that something went terribly wrong and that a normal success or cancel is not possible. We are using an Error to show Hazards. When a Hazard happens, the activity is interrupted (without Compensation) and the flow will continue from the Error Intermediate Event.

The behavior at the end of a successful Transaction Sub-Process is slightly different than that of a normal Sub-Process. When each path of the Transaction Sub-Process reaches a non-Cancel End Event(s), the flow does not immediately move back up to the higher-level Parent Process, as does a normal Sub-Process. First, the transaction protocol must verify that all the participants have successfully completed their end of the Transaction. Most of the time this will be true and the flow will then move up to the higher-level Process. But it is possible that one of the participants may end up with a problem that causes a Cancel or a Hazard. In this case, the flow will then move to the appropriate Intermediate Event, even though it had apparently finished successfully.
Note: The exact behavior and notation for defining Transactions is still an open issue.

Refer to the section entitled “Open Issues” on page 243 of the BPMN 1.0 specification for a complete list of the issues open for BPMN.

```
<xsd:element name="Transaction">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="TransactionId" type="xsd:string" use="required"/>
      <xsd:attribute name="TransactionProtocol" type="xsd:string" use="required"/>
      <xsd:attribute name="TransactionMethod" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="Compensate"/>
               <xsd:enumeration value="Store"/>
               <xsd:enumeration value="Image"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
TransactionId	The TransactionId attribute provides an identifier for the Transactions used within a package/diagram.
TransactionMethod	TransactionMethod is an attribute that defines the technique that will be used to undo a Transaction that has been cancelled. The default is Compensate, but the attribute MAY be set to Store or Image.
TransactionProtocol	This identifies the Protocol (e.g., WS-Transaction or BTP) that will be used to control the transactional behavior of the Sub-Process.

Table 69: Transaction

7.6.14. Loop

The attributes of Tasks and Sub-Processes will determine if they are repeated or performed once. There are two types of loops: Standard and Multi-Instance.

A Standard Loop activity will have a boolean expression that is evaluated after each cycle of the loop. If the expression is still True, then the loop will continue. There are two variations of the loop, which reflect the programming constructs of while and until. That is, a while loop will evaluate the expression before the activity is performed, which means that the activity may not actually be performed. The until loop will evaluate the expression after the activity has been performed, which means that the activity will be performed at least once.

Multi-Instance loops reflect the programming construct foreach. The loop expression for a Multi-Instance loop is a numeric expression evaluated only once before the activity is performed. The result of the expression evaluation will be an integer that will specify the number of times that the activity will be repeated. There are also two variations of the Multi-Instance loop where the instances are either performed sequentially or in parallel.

```
<xsd:element name="Loop">
   <xsd:complexType>
      <xsd:choice>
         <xsd:element ref="xpdl:LoopStandard"/>
         <xsd:element ref="xpdl:LoopMultiInstance"/>
      </xsd:choice>
      <xsd:attribute name="LoopType" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="Standard"/>
               <xsd:enumeration value="MultiInstance"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="LoopMultiInstance">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="MI_Condition" type="xsd:string" use="required"/>
      <xsd:attribute name="LoopCounter" type="xsd:integer"/>
      <xsd:attribute name="MI_Ordering" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="Sequential"/>
               <xsd:enumeration value="Parallel"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="MI_FlowCondition" use="optional" default="All">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="None"/>
               <xsd:enumeration value="One"/>
               <xsd:enumeration value="All"/>
               <xsd:enumeration value="Complex"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="ComplexMI_FlowCondition" type="xsd:string"
         use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="LoopStandard">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="LoopCondition" type="xsd:string" use="required"/>
      <xsd:attribute name="LoopCounter" type="xsd:integer"/>
      <xsd:attribute name="LoopMaximum" type="xsd:integer" use="optional"/>
      <xsd:attribute name="TestTime" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="Before"/>
               <xsd:enumeration value="After"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
LoopType	Standard or MultiInstance.
Standard	
LoopCondition	Standard Loops MUST have a boolean Expression to be evaluated, plus the timing when the expression SHALL be evaluated.

	Description
LoopCounter	This is updated at run time to count the number of executions of the loop and is available as a property to be used in expressions.
LoopMaximum	The Maximum an optional attribute that provides is a simple way to add a cap to the number of loops. This SHALL be added to the Expression defined in the LoopCondition.
TestTime	The expressions that are evaluated Before the activity begins are equivalent to a programming while function. The expression that are evaluated After the activity finishes are equivalent to a programming until function.
MultiInstance	
LoopCounter	The LoopCounter attribute is only applied for Sequential MultiInstance Loops and for processes that are being executed by a process engine. The attribute is updated at runtime by a process engine to count the number of loops as they occur. The LoopCounter attribute MUST be incremented at the start of a loop. Unlike a Standard loop, the modeler does not use this attribute in the MI_Condition Expression, but it can be used for tracking the status of a loop.
MI_Condition	MultiInstance Loops MUST have a numeric Expression to be evaluated--the Expression MUST resolve to an integer.
MI_Ordering	Sequential | Parallel This applies to only MultiInstance Loops. The MI_Ordering attribute defines whether the loop instances will be performed sequentially or in parallel. Sequential MI_Ordering is a more traditional loop. Parallel MI_Ordering is equivalent to multi-instance specifications that other notations, such as UML Activity Diagrams use.
MI_FlowCondition	None | One | All | Complex This attribute is equivalent to using a Gateway to control the flow past a set of parallel paths. An MI_FlowCondition of "None" is the same as uncontrolled flow (no Gateway) and means that all activity instances SHALL generate a token that will continue when that instance is completed. An MI_FlowCondition of "One" is the same as an Exclusive Gateway and means that the Token SHALL continue past the activity after only one of the activity instances has completed. The activity will continue its other instances, but additional Tokens MUST NOT be passed from the activity. An MI_FlowCondition of "All" is the same as a Parallel Gateway and means that the Token SHALL continue past the activity after all of the activity instances have completed. An MI_FlowCondition of "Complex" is the same as a Complex Gateway. The ComplexMI_FlowCondition attribute will determine the Token flow.
ComplexMI_FlowCondition	If the MI_FlowCondition attribute is set to "Complex," then an Expression Must be entered. This Expression that MAY reference Process data. The expression SHALL determine when and how many Tokens will continue past the activity.

Table 70: Loop

7.7. Transition Information

The Transition Information describes possible transitions between activities and the conditions that enable or disable them (the transitions) during execution. Further control and structure restrictions may be expressed in the Activity definition.

A process definition is seen as a network of edges between the Activity nodes (i.e. as a process diagram). All edges are directed and given by a pair of Activities:

(From node, to node).

The edges of the Activity net may be labelled by ***Transition conditions***. A Transition condition for a specific edge enables that transition if the condition evaluates to TRUE. If no routing condition is specified the Transition behaves as if a condition with value TRUE is present.

If there are multiple incoming or outgoing ("regular", see below) Transitions of an Activity, then further options to express control flow restrictions and condition evaluation semantics are provided in the Activity entity definition (AND/XOR variants of SPLIT/JOIN).

A loop may be represented via a transition that returns to an Activity that was on a path that led to the transition. Typically, at least one of the activities in the loop will have multiple outgoing transitions, one or more of which will contain an exit condition from the loop.

For the identifiers and names defined in the Transition information the scope is the surrounding Process Definition.

It is possible to define or synchronize multiple (concurrent or alternative) control threads (split, join) and sequences of Transitions between Activities (cascading Transitions/conditions).

```
<xsd:element name="Transition">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Condition" minOccurs="0"/>
         <xsd:element ref="xpdl:Description" minOccurs="0"/>
         <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
         <xsd:element ref="xpdl:Assignments" minOccurs="0"/>
         <xsd:element ref="xpdl:Object" minOccurs="0"/>
         <xsd:element ref="xpdl:ConnectorGraphicsInfos"
            minOccurs="0" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="From" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="To" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="Quantity" type="xsd:int" use="optional" default="1"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Assignments	See section 7.1.5
Condition	A Transition condition expression based on relevant data field. (E.g. 'Contract' = 'SMALL' OR 'Contract' <$20,000). Default: TRUE
ConnectorGraphicsInfos	See section 7.1.1.2.
Description	Short textual description of the Transition.
Extended Attributes	Optional extensions to meet individual implementation needs
From	Determines the FROM source of a Transition. (Activity Identifier)
Id	Used to identify the Transition.
Name	Text used to identify the Transition.
Object	See section 7.1.7.1
Quantity	The default value is 1. The value MUST NOT be less than 1. This attribute defines the number of Tokens that will be generated down the Sequence Flow.
To	Determines the TO target of a Transition (Activity Identifier)

Table 71: Transition Information

7.7.1. Condition

```
<xsd:element name="Condition">
   <xsd:complexType mixed="true">
      <xsd:choice minOccurs="0">
         <xsd:element ref="deprecated:Xpression" minOccurs="0"/>
```

```
            <xsd:element name="Expression" type="xpdl:ExpressionType" minOccurs="0"/>
         </xsd:choice>
         <xsd:attribute name="Type">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="CONDITION"/>
                  <xsd:enumeration value="OTHERWISE"/>
                  <xsd:enumeration value="EXCEPTION"/>
                  <xsd:enumeration value="DEFAULTEXCEPTION"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:attribute>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>
```

	Description	
Type	Define the type of transition condition, valid values are	
	CONDITION	Indicates that the transition is to be executed if its condition is satisfied,
	OTHERWISE	Indicates that the transition is the default transition that is executed if no conditions are met.
	EXCEPTION	Indicates that the transition is to be executed if there is an exception and its condition is satisfied.
	DEFAULTEXCEPTION	Indicates that the transition is the default transition that is executed if there is an exception and no exception conditions are met.
Xpression	A condition expression represented via XML markup.	

Table 72: Condition

7.7.1.1. Exception Conditions

The Exception and DEFAULTEXCEPTION types allow you to specify branches that are taken only when an Exception is raised. The EXCEPTION type is equivalent to the CONDITION type and the DEFAULTEXCEPTION matches the OTHERWISE type. The Condition may contain either the name of an Exception or a more complex expression. Except for the deadlines, exceptions are raised in an engine-specific manner. Like regular transitions, exception transitions are traversed only after the "From" activity has completed. An exception usually indicates abnormal completion.

The following example illustrates a set of transitions from an activity that includes exceptions: branches 1 and 2 are processed under normal conditions; branches 3 and 4 are processed if there is an exception.

```
<Transitions>
   <Transition Id="branch1" From="CheckBalance" To="ProcessRequest">
      <Condition Type="CONDITION">Balance > 1000</Condition>
   </Transition>
   <Transition Id="branch2" From="CheckBalance" To="InsufFunds">
      <Condition Type="OTHERWISE"/>
   </Transition>
   <Transition Id="branch3" From="CheckBalance" To="MessageDisplay">
      <Condition Type="EXCEPTION">ATMDownException</Condition>
   </Transition>
   <Transition Id="branch4" From="CheckBalance" To="SendAlarm">
      <Condition Type="DEFAULTEXCEPTION"/>
   </Transition>
</Transitions>
```

7.8. Partner Links

Partner links are used to define a communication link between two parties, each assuming a role in the communication. From a modeling perspective the roles are normally, but not always, given by the name of a pool or lane. If used this way, the message flow between two pool will correspond to a partener link with each role corresponding to the name of the corresponding pool.

Partner links are optional and normally used to model the communication at an abstract level using WSDL and port types. A WebServiceOperation (see 7.9.4) cau use a partner link for abstract modeling or service when a concrete web service is used and so port name is used instead of port type.

Partner links are defined in two levels. At the Package level, the parter link type defines a partner link name and one or two roles. The basic information about the partner link is defined at this level. At the process level, the partner link itself is defined using the partner link type. This allow partner link types to be reusable at the package level.

7.8.1. Partner Link Type

Partner link type defines the general information about a partner link at package scope. This allow multiple processes to use the same partner link type.

```
<xsd:element name="PartnerLinkType">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Role" maxOccurs="2">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:any namespace="##other" processContents="lax"
                     minOccurs="0" maxOccurs="unbounded"/>
               </xsd:sequence>
               <xsd:attribute name="portType" type="xsd:string" use="required"/>
               <xsd:attribute name="Name" type="xsd:string" use="required"/>
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
         </xsd:element>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="PartnerLinkTypes">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:PartnerLinkType" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Role	A partner link may have one or two roles. In most cases they will correspond to the name of the two pools (or lanes) that are interchanging messages.
Role Name	The name of the role.
PortType	The port type implemented by the role. Correspond to the WSDL port type.
Name	Name of the partner link type.

	Description
Id	Id of the partner link

Table 73: PartnerLinkType

7.8.2. Partner Link

Partner link is used in a process and refers to a partner link type. It defines the role of the process will play and the role the partner will use. MyRole element define the role the process is playing and the PartnerRole element define the role the partner is playing.

```
<xsd:element name="PartnerLink">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="MyRole" minOccurs="0">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:any namespace="##other" processContents="lax"
                     minOccurs="0" maxOccurs="unbounded"/>
               </xsd:sequence>
               <xsd:attribute name="RoleName" type="xsd:string" use="required"/>
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="PartnerRole" minOccurs="0">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element ref="xpdl:EndPoint"/>
                  <xsd:any namespace="##other" processContents="lax"
                     minOccurs="0" maxOccurs="unbounded"/>
               </xsd:sequence>
               <xsd:attribute name="RoleName" type="xsd:string" use="required"/>
               <xsd:attribute name="ServiceName" type="xsd:string" use="optional"/>
               <xsd:attribute name="PortName" type="xsd:string" use="optional"/>
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
         </xsd:element>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="PartnerLinkTypeId" type="xsd:NMTOKEN" use="required"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="PartnerLinks">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:PartnerLink" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
MyRole	Thefines the role the process is playing in the interaction with the partner.
MyRole RoleName	Must match one of the two roles defined in the partner link type.

	Description
PartnerRole	Thefines the role the partner is playing in the interaction with this process.
EndPoint	The end point for the partner. It may be the WSDL end point or the partner service listener end point.
PartnerRole RoleName	Must match one of the two roles defined in the partner link type.
ServiceName	The service name implemented by the partner and defined in the WSDL.
PortName	The port name implemented by the partner and defined in the WSDL.
Name	Name of this partner link. It may correspond to the name of the partner link type, but it is not required to be the same.
Id	Id of the partner link.

Table 74: PartnerLink

7.9. Messaging

Messages in XPDL are based on the WSDL model, and so, are not restricted to web services. When modeling and defining messaging they can be defined abstract in which case partner links should be used, or concrete in which case service should be used (see 7.9.4).

7.9.1. Message Flow

A Message Flow is used to show the flow of messages between two entities that are prepared to send and receive them. In BPMN, two separate Pools in the Diagram will represent the two entities. Thus, Message Flow MUST connect two Pools, either to the Pools themselves or to Flow Objects within the Pools. They cannot connect two objects within the same Pool.

```
<xsd:element name="MessageFlow">
   <xsd:complexType>
      <xsd:sequence minOccurs="0">
         <xsd:element name="Message" type="xpdl:MessageType" minOccurs="0"/>
         <xsd:element ref="xpdl:Object" minOccurs="0"/>
         <xsd:element ref="xpdl:ConnectorGraphicsInfos"
            minOccurs="0" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="Source" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Target" type="xsd:NMTOKEN" use="required"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="MessageFlows">
   <xsd:complexType>
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element ref="xpdl:MessageFlow"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description

	Description
Connector Graphics Info	See section 7.1.1.2.
Description	Short textual description of the Transition.
Extended Attributes	Optional extensions to meet individual implementation needs
Source	Determines the source of a MessageFlow (Activity or Pool).
Id	Used to identify the MessageFlow.
Message	Message is an optional attribute that identifies the Message that is being sent. See section 7.9.2.
Name	Text used to identify the MessageFlow.
Object	See section 7.1.7.1
Target	Determines the target of a MessageFlow (Activity or Pool).

Table 75: MessageFlow

7.9.2. Message Type

The Message type element is used in the definition of attributes for a Start Event, End Event, Intermediate Event, Task, Message Flow, etc.

```
<xsd:complexType name="MessageType">
   <xsd:sequence minOccurs="0">
      <xsd:choice minOccurs="0">
         <xsd:element ref="xpdl:ActualParameters"/>
         <xsd:element ref="xpdl:DataMappings"/>
      </xsd:choice>
      <xsd:any namespace="##other" processContents="lax"
         minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
   <xsd:attribute name="Name" type="xsd:string" use="optional"/>
   <xsd:attribute name="From" type="xsd:NMTOKEN" use="optional"/>
   <xsd:attribute name="To" type="xsd:NMTOKEN" use="optional"/>
   <xsd:attribute name="FaultName" type="xsd:NMTOKEN" use="optional"/>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
```

	Description
Id	Id of the mesage
Name	Text description of the Message.
From	Optional, but if present must be the name of a Participant/Process.
To	Optional, but if present must be the name of a Participant/Process.
Actual Parameters	A list of parameters that compose the message. See section 7.1.3.3.
DataMappings	Alternative approach to to build the message. See section 7.6.5.4.1
FaultName	When the message is an error message (for example an error response to a request), the FaultName correspond to the fault (exception). See WebServiceFaultCatch to handle the error in the reciving end.

Table 76: Message – attributes

7.9.3. End Point

The end point can be a service (the URL of the listener implementing the service), or a WSDL (the URL of the WSDL location).

```
<xsd:element name="EndPoint">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:ExternalReference"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="EndPointType" use="optional" default="WSDL">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="WSDL"/>
               <xsd:enumeration value="Service"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
ExternalReference	The URL of the end point.
EndPointType	Type of end point: WSDL — When the end point corresponds to the location of a WSDL file. Service — When the end point corresponds to the address of a listener implementing the service.

Table 77: End Point

7.9.4. Web ServiceOperation

A web services operation is defined by using a partner link or alternative describing the service.

```
<xsd:element name="WebServiceOperation">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:choice>
            <xsd:element name="Partner">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:any namespace="##other" processContents="lax"
                        minOccurs="0" maxOccurs="unbounded"/>
                  </xsd:sequence>
                  <xsd:attribute name="PartnerLinkId" type="xsd:NMTOKEN"
                     use="required"/>
                  <xsd:attribute name="RoleType" use="required">
                     <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                           <xsd:enumeration value="MyRole"/>
                           <xsd:enumeration value="PartnerRole"/>
                        </xsd:restriction>
                     </xsd:simpleType>
                  </xsd:attribute>
                  <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:complexType>
            </xsd:element>
```

```
                <xsd:element name="Service">
                   <xsd:complexType>
                      <xsd:sequence>
                         <xsd:element ref="xpdl:EndPoint"/>
                         <xsd:any namespace="##other" processContents="lax"
                            minOccurs="0" maxOccurs="unbounded"/>
                      </xsd:sequence>
                      <xsd:attribute name="ServiceName" type="xsd:string"
                         use="required"/>
                      <xsd:attribute name="PortName" type="xsd:string"
                         use="required"/>
                      <xsd:anyAttribute namespace="##other" processContents="lax"/>
                   </xsd:complexType>
                </xsd:element>
             </xsd:choice>
             <xsd:any namespace="##other" processContents="lax"
                minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
          <xsd:attribute name="OperationName" type="xsd:string" use="required"/>
          <xsd:anyAttribute namespace="##other" processContents="lax"/>
       </xsd:complexType>
    </xsd:element>
```

	Description
Partner	Included only when a partner link is used to define this operation.
PartnerLinkId	Partner link to be used.
RoleType	The role in the partner link that is being used MyRole Using the process role PartnerRole Using the partner role
OperationName	The name of the operation implemented by the service as it is defined in the WSDL.
Service	Included only when a concrete service operation is being defined.
ServiceName	The name of the service implementing the operation.
PortName	The port name in which the service is implementing the operation.
EndPoint	The end point implementing the service.

Table 78: Web Service Operation

7.9.5. Web Service Fault Catch

Used to catch web services faults and to either execute a block activity or a transition.

```
    <xsd:element name="WebServiceFaultCatch">
       <xsd:complexType>
          <xsd:sequence>
             <xsd:element name="Message" type="xpdl:MessageType" minOcurrs=”0”/>
             <xsd:choice>
                <xsd:element ref="xpdl:BlockActivity"/>
                <xsd:element ref="xpdl:TransitionRef"/>
             </xsd:choice>
             <xsd:any namespace="##other" processContents="lax"
                minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
          <xsd:attribute name="FaultName" type="xsd:NMTOKEN" use="optional"/>
          <xsd:anyAttribute namespace="##other" processContents="lax"/>
       </xsd:complexType>
    </xsd:element>
```

	Description

	Description
Message	Optional message that may contain information about the fault.
BlockActivity	Block activity that will be executed if the WebServiceFaultCatch is activated by a fault.
TransitionRef	TransitionRef that will be executed if the WebServiceFaultCatch is activated by a fault.
FaultName	Name of the fault that will be catch by this WebServiceFaultCatch. If the name is not provided, then the WebServiceFaultCatch will catch any fault generated by the web service.

Table 79: Web Service Fault Catch

7.10. Association

An Association is used to associate information and Artifacts with Flow Objects. Text and graphical non-Flow Objects can be associated with the Flow Objects and Flow. An Association is also used to show the activities used to compensate for an activity.

An Association is also used to associate Data Objects with other objects. A Data Object is used to show how documents are used throughout a Process. Refer to section 7.1.7.2 for more information on Data Objects.

```
<xsd:element name="Association">
   <xsd:complexType>
      <xsd:sequence minOccurs="0">
         <xsd:element ref="xpdl:Object"/>
         <xsd:element ref="xpdl:ConnectorGraphicsInfos"
            minOccurs="0" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Source" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Target" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="AssociationDirection" use="optional" default="None">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="None"/>
               <xsd:enumeration value="To"/>
               <xsd:enumeration value="From"/>
               <xsd:enumeration value="Both"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Associations">
   <xsd:complexType>
      <xsd:sequence maxOccurs="unbounded">
         <xsd:element ref="xpdl:Association"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Association Direction	None | To | Frm | Both
Connector Graphics Info	See section 7.1.1.2.
Description	Short textual description of the Association.
Source	Determines the source of an Association (any graphical object).
Id	Used to identify the Association.
Name	Text used to identify the Association.
Object	See section 7.1.7.1
Target	Determines the target of an Association (any graphical object).

Table 80: Association

7.11. Participants

The Participant is one of the following types: resource set, resource, organizational unit, role, human, or system. A role and a resource are used in the sense of abstract actors. . This definition is an abstraction level between the real performer and the activity, which has to be performed. During run time these abstract definitions are evaluated and assigned to concrete human(s) and/or program(s).

The scope of the identifier of a participant entity declaration in a minimal resource repository or organizational model is the surrounding entity (Process Definition or Process Model Definition) in which it is defined.

An external resource repository or organizational model may contain substantial additional information that complements the basic participant types presented in here.

```
<xsd:element name="Participant">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:ParticipantType"/>
         <xsd:element ref="xpdl:Description" minOccurs="0"/>
         <xsd:element ref="xpdl:ExternalReference" minOccurs="0"/>
         <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Participants">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Participant" 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:element>
```

	Description
Description	Short textual description of a participant.
ExternalReference	A reference to an external specification of a participant. See section 7.1.4

	Description
Extended Attributes	Optional extensions to meet individual implementation needs
Id	Used to identify the participant definition.
Name	Text used to identify a performer
Participant Type	Definition of the type of participant entity.

Table 81: Participant

7.11.1. Participant Entity Types

The Participant entity type attribute characterises the participant to be an individual, an organisational unit or an abstract resource such as a machine.

```
<xsd:element name="ParticipantType">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Type" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="RESOURCE_SET"/>
               <xsd:enumeration value="RESOURCE"/>
               <xsd:enumeration value="ROLE"/>
               <xsd:enumeration value="ORGANIZATIONAL_UNIT"/>
               <xsd:enumeration value="HUMAN"/>
               <xsd:enumeration value="SYSTEM"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description	
Type	RESOURCE_SET	A set of resources.
	RESOURCE	A specific resource agent.
	ROLE	This type allows performer addressing by a role or skill set. A role in this context is a function a human has within an organization. As a function isn't necessarily unique, a coordinator may be defined (for administrative purposes or in case of exception handling) and a list of humans the role is related to.
	ORGANIZATIONAL_UNIT	A department or any other unit within an organizational model.
	HUMAN	A human interacting with the system via an application presenting a user interface to the participant.
	SYSTEM	An automatic agent.

Table 82: Participant Entity Type

7.12. Relevant data field

Relevant data field represent the variables of a process or Package Definition. They are typically used to maintain decision data (used in conditions) or reference data values (parameters), which are passed between activities or subflow. This may be differentiated from application data, which is data managed or accessed wholly by the invoked applications and which is not accessible to the process or workflow management system. The relevant data field list defines all data objects, which are required by the process. The attribute `DataType` explicitly specifies all information needed for a process or workflow management system to define an appropriate data object for storing data, which is to be handled by an active instance of the process.

Relevant data field can be defined in a process and in a Package. The scopes differ in that the former may only be accessed by entities defined inside that process, while the latter may be used also e.g. to define the parameters of a process entity.

Where parameters are passed to a called subflow outside the current model definition (e.g. to support remote process invocation) it is the responsibility of the process designer(s) to ensure that data type compatibility exists across the parameter set.

```
<xsd:element name="DataField">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:DataType"/>
         <xsd:element name="InitialValue" type="xpdl:ExpressionType"
            minOccurs="0"/>
         <xsd:element ref="xpdl:Length" minOccurs="0"/>
         <xsd:element ref="xpdl:Description" minOccurs="0"/>
         <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:attribute name="IsArray" use="optional" default="FALSE">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="TRUE"/>
               <xsd:enumeration value="FALSE"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="Correlation" type="xsd:boolean" use="optional"
         default="false"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="DataFields">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:DataField" 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:element>
```

	Description
Data Type	Data type of the process variable. See Section 7.13
Description	Short textual description of the data defined.
Extended Attributes	Optional extensions to meet individual implementation needs

	Description
Id	Used to identify the relevant data field.
Initial Value	Pre-assignment of data for run time.
Is Array	Indicates if it is an array
Length	The length of the data
Name	Text used to identify the relevant data field
Correlation	Used in BPMN mapping to BPEL.

Table 83: Relevant data field

7.13. Data Types

Data types consist of a set of standard types that may be used as part of the data specification of relevant data field, formal parameters, and Processes. You can also declare a new data type within a TypeDeclaration and use it wherever the standard data types are used. A data type may be selected from the following set of types.

```
<xsd:element name="DataType">
   <xsd:complexType>
      <xsd:group ref="xpdl:DataTypes"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:group name="DataTypes">
   <xsd:choice>
      <xsd:element ref="xpdl:BasicType"/>
      <xsd:element ref="xpdl:DeclaredType"/>
      <xsd:element ref="xpdl:SchemaType"/>
      <xsd:element ref="xpdl:ExternalReference"/>
      <xsd:element ref="xpdl:RecordType"/>
      <xsd:element ref="xpdl:UnionType"/>
      <xsd:element ref="xpdl:EnumerationType"/>
      <xsd:element ref="xpdl:ArrayType"/>
      <xsd:element ref="xpdl:ListType"/>
   </xsd:choice>
</xsd:group>
```

	Description
Array Type	A fixed size set of data all of the same data type (deprecated).
Basic Type	A simple type: STRING, INTEGER, FLOAT, DATETIME, REFERENCE, BOOLEAN, or PERFORMER.
Declared Type	A reference to a data type declared in a TypeDeclaration element.
Enumeration Type	A set of legal values of a variable or parameter (deprecated).
ExternalReference	A reference to a type defined in an external document. See Section 7.1.4
List Type	An unbounded set of data all of the same data type (deprecated).
Record Type	A set of members that may be of different types (deprecated).
SchemaType	A data type defined using an XML schema.
Union Type	A set of members only one of which will be used for an instance of the data (deprecated).

Table 84: Standard Data Types

7.13.1. Basic Data Types

```
<xsd:element name="BasicType">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Length" minOccurs="0"/>
         <xsd:element ref="xpdl:Precision" minOccurs="0"/>
         <xsd:element ref="xpdl:Scale" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Type" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN">
               <xsd:enumeration value="STRING"/>
               <xsd:enumeration value="FLOAT"/>
               <xsd:enumeration value="INTEGER"/>
               <xsd:enumeration value="REFERENCE"/>
               <xsd:enumeration value="DATETIME"/>
               <xsd:enumeration value="BOOLEAN"/>
               <xsd:enumeration value="PERFORMER"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description	
Type	STRING	A finite-length sequence of characters
	FLOAT	A floating point or double precision number. The maximum size of the number is not specified in XPDL
	INTEGER	A number represented by an optional sign followed by a finite-length sequence of decimal digits. The maximum size of the integer is not specified in XPDL
	REFERENCE	A reference to an external data type – now deprecated. The ExternalReference is the recommended way to refer to an external data type.
	DATETIME	A specific instant of time. The date format is not specified within XPDL.
	BOOLEAN	A data instance of a Boolean type is one having one of the values TRUE or FALSE. The internal representation of these values is not defined in the XPDL
	PERFORMER	A data instance of a performer type is one having a value of a declared participant.

Table 85: Basic Data Types

7.13.2. Complex Data Types

XPDL permits the definition of complex data types such as arrays, records, unions, enumerations, and lists. Complex data types are defined using the SchemaType. The RecordType, UnionType, EnumerationType, ArrayType, and ListType, which were used in the past to define complex data, are now deprecated. They have been left in the xpdl schema for compatibility with previous versions.

7.13.2.1.Schema Type

The SchemaType allows users to define a data type using XML schema syntax. It may also be used to define an XML string that should conform to the schema.

```
<xsd:element name="SchemaType">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

The following, for example, could describe a C++ or Java class, a C structure, or an XML string:

```
<SchemaType>
   <schema xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="PO">
         <complexType>
            <sequence>
               <element name="CustomerName" type="string"/>
               <element name="Address" type="string"/>
               <element name="OrderNumber" type="string"/>
            </sequence>
         </complexType>
      </element>
   </schema>
</SchemaType>
```

7.13.2.2.Record Type

```
<xsd:element name="RecordType">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:Member" maxOccurs="unbounded"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<xsd:element name="Member">
   <xsd:complexType>
      <xsd:group ref="xpdl:DataTypes"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Member	A field in the record.
DataTypes	Data type of a member. See Table 84: Standard Data Types.

Table 86: Record Type

7.13.2.3.Union Type

```
<xsd:element name="UnionType">
   <xsd:complexType>
```

```
         <xsd:sequence>
            <xsd:element ref="xpdl:Member" maxOccurs="unbounded"/>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="Member">
      <xsd:complexType>
         <xsd:group ref="xpdl:DataTypes"/>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>
```

	Description
Member	A field in the union.
DataTypes	Data type of a member. See Table 84: Standard Data Types.

Table 87: Union Type

7.13.2.4.Enumeration Type

```
   <xsd:element name="EnumerationType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="xpdl:EnumerationValue" maxOccurs="unbounded"/>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="EnumerationValue">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="Name" type="xsd:NMTOKEN" use="required"/>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>
```

	Description
Enumeration Value	An element that represents one of the values in an enumeration.
Name	The name of the value.

Table 88: Enumeration Type

7.13.2.5.Array Type

```
   <xsd:element name="ArrayType">
      <xsd:complexType>
         <xsd:group ref="xpdl:DataTypes"/>
         <xsd:attribute name="LowerIndex" type="xsd:NMTOKEN" use="required"/>
         <xsd:attribute name="UpperIndex" type="xsd:NMTOKEN" use="required"/>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>
```

	Description
DataTypes	The data type of array entries. See Table 84: Standard Data Types.
Lower Index	The lower bound of an ArrayType.
Upper Index	The upper bound of an ArrayType.

Table 89: Array Type

7.13.2.6. List Type

```
   <xsd:element name="ListType">
      <xsd:complexType>
         <xsd:group ref="xpdl:DataTypes"/>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>
```

	Description
DataTypes	The data type of list entries. See Table 84: Standard Data Types.

Table 90: List Type

7.13.3. Declared Data Types

It is possible to reuse a complex data definition wherever you can use a standard XPDL type. Define the data type under a TypeDeclaration and then refer to it using the DeclaredType data type.

7.13.3.1. Type Declaration

```
   <xsd:element name="TypeDeclaration">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:group ref="xpdl:DataTypes"/>
            <xsd:element ref="xpdl:Description" minOccurs="0"/>
            <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="Id" type="xsd:ID" use="required"/>
         <xsd:attribute name="Name" type="xsd:string" use="optional"/>
         <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="TypeDeclarations">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="xpdl:TypeDeclaration"
               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:element>
```

	Description
DataTypes	The data type. See Table 84: Standard Data Types.
Description	An informal description of the data type.
ExtendedAttributes	Optional extensions to meet individual implementation needs.
Id	An identifier for the TypeDeclaration.
Name	The name of the TypeDeclaration.

Table 91: Type Declaration

Example to reuse a SchemaType to define a purchase order:

```
<TypeDeclarations>
   <TypeDeclaration Id=”POType” Name=”PurchaseOrder”>
      <SchemaType>
         <schema xmlns="http://www.w3.org/2000/10/XMLSchema">
            <element name="PO">
               <complexType>
                  <sequence>
                    <element name="CustomerName" type="string"/>
                    <element name="Address" type="string"/>
                    <element name="OrderNumber" type="string"/>
                  </sequence>
               </complexType>
            </element>
         </schema>
      </SchemaType>
   </TypeDeclaration>
</TypeDeclarations>
```

7.13.3.2.Declared Type

```
<xsd:element name="DeclaredType">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:IDREF" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>
```

	Description
Id	A reference to a data type declared in a TypeDeclaration.
Name	A name for the declared type

Table 92: Declared Data Type

Example of using the DeclaredType type to define a number of POType variables.

```
<DataFields>
   <DataField Id=”newPO”>
      <DataType>
         <DeclaredType Id=”POType”/>
      </DataType>
   </DataField>
   <DataField Id=”checkedPO”>
      <DataType>
```

```
               <DeclaredType Id=”POType”/>
            </DataType>
         </DataField>
      </DataFields>
```

8. Samples

8.1. Sample Process

This sample process was created to illustrate some of the features of XPDL and does not represent any real processes or necessarily the best way to accomplish the process. The XPDL in the example was generated by an Open Source Java Application that was created by Global 360/CapeVisions and edited to incorporate unsupported features and improve readability.

The process represents an order entry system in which an order enters the system as a formatted string. The Package is composed of a main process and two subprocesses. The following discussion includes a brief overview of these processes along with a discussion of some of the data types, extended attributes, and external references used in the sample. Finally the XPDL is displayed.

8.1.1. The Processes

8.1.1.1. The EOrder Main Process

The main process takes a formatted string as an input and returns a string that indicates whether the order was confirmed or rejected. It contains the following steps:

- The string is first converted to a complex data object. If an exception is caught (indicating that the string is incorrectly formatted), an alarm is raised and the order is rejected.
- The data is checked for accuracy.
- The process determines whether payment is via a purchase order or a credit card.
- Credit card orders are sent to a subprocess that authorizes the credit purchase.
- Purchase orders are validated by an application that checks the vendor's record and authorizes the purchase amount.
- The order is entered into the database and an order number is issued.
- The next three activities happen in parallel:

  An acceptance message is composed to return to the end user.

  A subprocess is invoked asynchronously to fill the order.

  An order confirmation email is sent to the end user. This activity is a special activity that is managed by the system. It uses ExtendedAttributes to specify the information the system needs for the email.
- If an order is rejected, either because it is inaccurate or cannot obtain authorization, a rejection message is composed, to return to the customer.

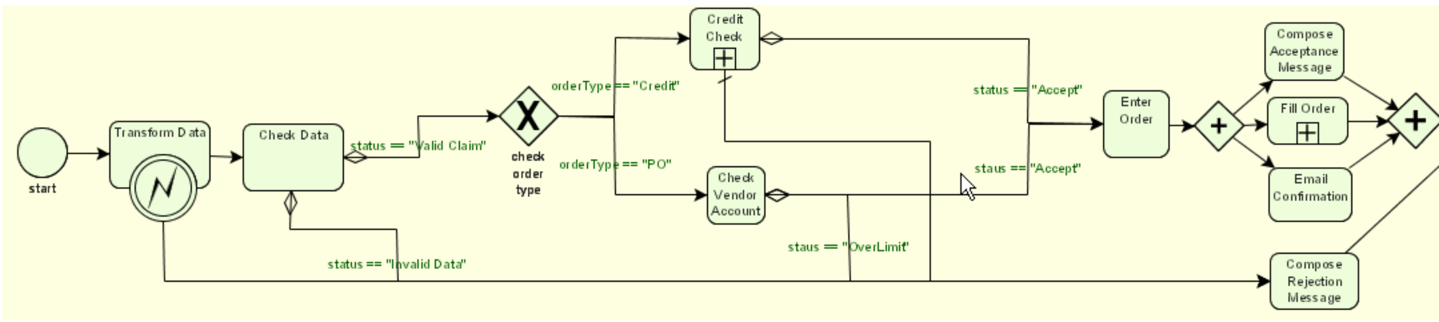


Figure 8-1: EOrder Main Process

8.1.1.2. The CreditCheck Subprocess

The CreditCheck subprocess sets up a CreditInfo object from the input parameters and then sends the information to a credit card web service for authorization. The web service returns a status string that is converted to an OrderStatus string and returned to the calling process.

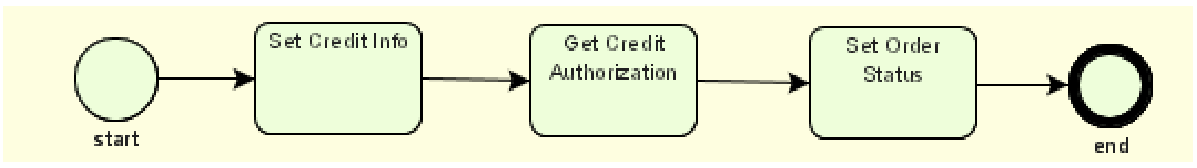


Figure 8-2: CreditCheck Subprocess

8.1.1.3. The FillOrder Subprocess

This subprocess handles the shipping and billing of the order. This process includes a participant called a "Shipper"

- The first activity displays the order information to a Shipper who ships the items in the order and records the status of the line items. The application returns the status of the order -- whether it is complete or backordered. This activity includes some deadlines. If the activity is not completed within 3 days, a notifyException is thrown an alarm is raised. If the activity is still not completed within 5 days, a timeoutException is thrown and the order is canceled.
- The process then determines if it is a PO or credit order.
- PO orders are sent to the billing system and then an electronic invoice is created and stored on a server.
- Credit card orders are sent to the credit card web service for charging and then an electronic receipt is created and stored on a server.
- The last step sends the invoice or receipt to the customer as an attachment to an email message. It uses ExtendedAttributes to specify the information needed for the email.

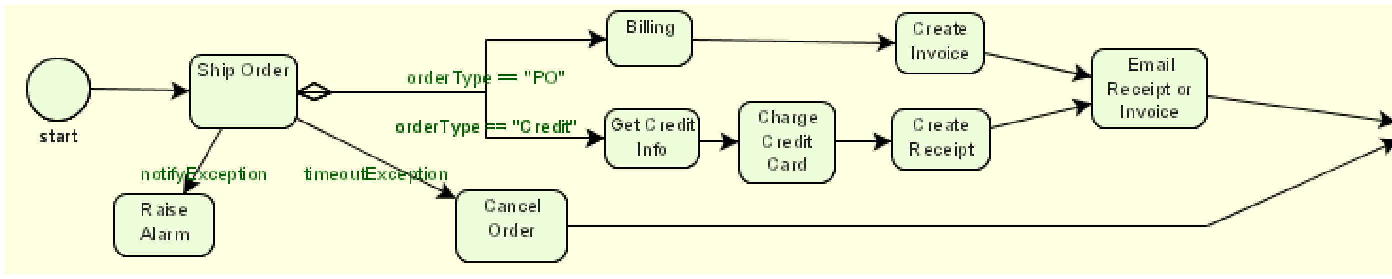


Figure 8-3: FillOrder Subprocess

8.1.2. Type Declarations

A number of data types are defined for the process.

- An Order is defined in a separate schema document and declared using an ExternalReference.

```
<TypeDeclaration Id="Order" Name="Order">
    <ExternalReference
location="http://wfmc.org/standards/docs/xpdl_sample/orderschema.xsd"/>
</TypeDeclaration>
```

The schema is:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:element name="Order">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Items">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="Item" maxOccurs="unbounded">
                       <xsd:complexType>
                           <xsd:attribute name="itemNumber"
                                   type="xsd:integer" use="required"/>
                           <xsd:attribute name="itemQty"
                                   type="xsd:integer" use="required"/>
                       </xsd:complexType>
                     </xsd:element>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
         <xsd:attribute name="accountNumber" type="xsd:integer" use="required"/>
         <xsd:attribute name="totalAmount" type="xsd:float" use="required"/>
         <xsd:attribute name="emailAddress" type="xsd:string" use="required"/>
         <xsd:attribute name="orderType" use="required">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="PO"/>
                  <xsd:enumeration value="Credit"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:attribute>
         <xsd:attribute name="cardType" use="required">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="MC-VISA"/>
                  <xsd:enumeration value="Discover"/>
                  <xsd:enumeration value="AMEX"/>
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:attribute>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

- An OrderStatus is directly defined using a SchemaType. It uses an XML schema to enumerate the valid strings that can represent the status.

```
<TypeDeclaration Id="OrderStatus" Name="OrderStatus">
   <SchemaType>
      <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified" attributeFormDefault="unqualified">
         <xsd:element name="Status">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                  <xsd:enumeration value="ValidData"/>
                     <xsd:enumeration value="InvalidData"/>
                     <xsd:enumeration value="Accept"/>
                     <xsd:enumeration value="BadCredit"/>
                     <xsd:enumeration value="OverLimit"/>
                     <xsd:enumeration value="BadDataFormat"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:element>
         </xsd:schema>
      </SchemaType>
   </TypeDeclaration>
```

- A CardType type uses an ExternalReference to the cardType attribute within the Order schema.

```
   <TypeDeclaration Id="CardType" Name="CardType">
      <ExternalReference
location="http://wfmc.org/standards/docs/xpdl_sample/orderschema.xsd"
xref="cardType" namespace="orderschema/Order"/>
   </TypeDeclaration>
```

- A CreditInfo type uses an ExternalReference to a data type defined within a WSDL document.

```
   <TypeDeclaration Id="CreditInfo" Name="CreditInfo">
      <ExternalReference
location="http://wfmc.org/standards/docs/xpdl_sample/creditService.wsdl"
xref="CreditInfo"/>
   </TypeDeclaration>
```

Within the WSDL document, the type is defined as follows:

```
   <types>
      <schema xmlns="http://www.w3.org/2001/XMLSchema">
         <element name="CreditInfo">
            <complexType>
               <sequence>
                  <element name="MerchantNumber"/>
                  <element name="AccountNumber"/>
                  <element name="Amount"/>
                  <element name="CardType"/>
               </sequence>
            </complexType>
         </element>
      </schema>
   </types>
```

8.1.3. ExtendedAttributes

The process vendor defines several ExtendedAttributes to extend XPDL. The namespace, xmlns:xyz="http://www.xyzeorder.com/workflow", is designated for the ExtendedAttribute XML.

- There is an ExtendedAttribute to mark an activity as a SystemActivity, or one implemented by the process or workflow system. Three activities in the sample are so marked: email activities, alarm activities, and web service activities.

```
   <ExtendedAttribute Name="SystemActivity" Value="WebService"/>
   <ExtendedAttribute Name="SystemActivity" Value="Email"/>
   <ExtendedAttribute Name="SystemActivity" Value="Alarm"/>
```

- There is an ExtendedAttribute to provide information for the email activity. It should be assumed that some of this information is supplied in the process modeling tool.

```
   <ExtendedAttribute Name="Email">
      <xyz:Email to="emailAddress" subject="orderStatus">
         <xyz:Attachments>
            <xyz:Attachment>%%docURI</xyz:Attachment>
         </xyz:Attachments>
         <xyz:MessageText>Order number %%orderNumber is %%orderStatus. Thank-you
for ordering from PQR Products, Inc.</xyz:MessageText>
      </xyz:Email>
   </ExtendedAttribute>
```

8.1.4. External References

The sample process makes an external reference to a WSDL document, creditService.wsdl, to define the applications used for processing a credit card purchase. (Section 8.1.2 illustrates the use of the ExternalReference element to import data types from external documents.)

```
<Application Id="getCreditAuthorization">
   <ExternalReference
location="http://wfmc.org/standards/docs/xpdl_sample/creditService.wsdl"
xref="GetCreditAuthorization"/>
</Application>
```

Within the WSDL document the applications are defined as operations:

```
<message name="creditInput">
   <part name="CreditInfo" element="tns:CreditInfo"/>
</message>
<message name="creditOutput">
   <part name="status" type="string"/>
</message>
<portType name="CreditPortType">
   <operation name="GetCreditAuthorization">
      <input message="tns:creditInput"/>
      <output message="tns:creditOutput"/>
   </operation>
   <operation name="ChargeCreditAccount">
         <input message="tns:creditInput"/>
      <output message="tns:creditOutput"/>
   </operation>
</portType>
```

8.1.5. Sample XPDL

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.0 U (http://www.xmlspy.com) by Robert M Shapiro (Cape Visions) -->
<Package xmlns:xyz="http://www.xyzeorder.com/workflow" Id="1" Name="sample process" xmlns:deprecated="http://www.wfmc.org/2002/XPDL1.0"
xmlns="http://www.wfmc.org/2004/XPDL2.0alpha" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wfmc.org/2004/XPDL2.0alpha
	C:\DOCUME~1\ROBERT~1\MYDOCU~1\capevisions\bpmn\schema\bpmnxpdl_20.xsd">
	<PackageHeader>
		<XPDLVersion>2.0</XPDLVersion>
		<Vendor>Global 360</Vendor>
		<Created>06/04/2005 14:50:58 PM</Created>
	</PackageHeader>
	<ConformanceClass GraphConformance="NON_BLOCKED"/>
	<Script Type="text/javascript"/>
	<TypeDeclarations>
		<TypeDeclaration Id="Order" Name="Order">
			<ExternalReference location="http://wfmc.org/standards/docs/xpdl_sample/orderschema.xsd"/>
		</TypeDeclaration>
		<TypeDeclaration Id="OrderStatus" Name="OrderStatus">
			<SchemaType>
				<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
					<xsd:element name="Status">
						<xsd:simpleType>
							<xsd:restriction base="xsd:NMTOKEN">
								<xsd:enumeration value="ValidData"/>
								<xsd:enumeration value="InvalidData"/>
								<xsd:enumeration value="Accept"/>
								<xsd:enumeration value="BadCredit"/>
								<xsd:enumeration value="OverLimit"/>
								<xsd:enumeration value="BadDataFormat"/>
							</xsd:restriction>
						</xsd:simpleType>
					</xsd:element>
				</xsd:schema>
			</SchemaType>
		</TypeDeclaration>
		<TypeDeclaration Id="CardType" Name="CardType">
			<ExternalReference location="http://wfmc.org/standards/docs/xpdl_sample/orderschema.xsd" xref="cardType" namespace="orderschema/Order"/>
		</TypeDeclaration>
		<TypeDeclaration Id="CreditInfo" Name="CreditInfo">
			<ExternalReference location="http://wfmc.org/standards/docs/xpdl_sample/creditService.wsdl" xref="CreditInfo"/>
		</TypeDeclaration>
	</TypeDeclarations>
```

```
<Participants>
    <Participant Id="DBConnection">
        <ParticipantType Type="SYSTEM"/>
        <Description>Reference to Database Resource</Description>
    </Participant>
</Participants>
<Pools>
    <Pool Process="1" Id="2" BoundaryVisible="false">
        <Lanes/>
        <NodeGraphicsInfos>
          <NodeGraphicsInfo Page="1"/>
        </NodeGraphicsInfos>
    </Pool>
    <Pool Process="2" Id="3" Name="" BoundaryVisible="true">
        <Lanes>
            <Lane Id="0" Name="Lane-0" ParentLane="3">
                <NodeGraphicsInfos>
                <NodeGraphicsInfo Page="1" Width="1176.0" Height="239.0" BorderColor="-16777216" FillColor="-32">
                    <Coordinates XCoordinate="22.0" YCoordinate="4.0"/>
                </NodeGraphicsInfo>
                </NodeGraphicsInfos>
            </Lane>
        </Lanes>
        <NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" Width="1200.0" Height="247.0" BorderColor="-16777216" FillColor="-32">
            <Coordinates XCoordinate="0.0" YCoordinate="0.0"/>
        </NodeGraphicsInfo>
        </NodeGraphicsInfos>
    </Pool>
    <Pool Process="3" Id="5" Name="" BoundaryVisible="true">
        <Lanes>
            <Lane Id="1" Name="Lane-1" ParentLane="5">
        <NodeGraphicsInfos>
                <NodeGraphicsInfo Page="1" Width="1176.0" Height="80.0" BorderColor="-16777216" FillColor="-32">
                    <Coordinates XCoordinate="22.0" YCoordinate="252.0"/>
                </NodeGraphicsInfo>
        </NodeGraphicsInfos>
            </Lane>
        </Lanes>
        <NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" Width="1200.0" Height="88.0" BorderColor="-16777216" FillColor="-32">
            <Coordinates XCoordinate="0.0" YCoordinate="248.0"/>
        </NodeGraphicsInfo>
        </NodeGraphicsInfos>
    </Pool>
    <Pool Process="4" Id="7" Name="" BoundaryVisible="true">
        <Lanes>
            <Lane Id="2" Name="Lane-2" ParentLane="7">
        <NodeGraphicsInfos>
                <NodeGraphicsInfo Page="1" Width="1176.0" Height="156.0" BorderColor="-16777216" FillColor="-32">
                    <Coordinates XCoordinate="22.0" YCoordinate="342.0"/>
                </NodeGraphicsInfo>
        </NodeGraphicsInfos>
            </Lane>
        </Lanes>
        <NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" Width="1200.0" Height="164.0" BorderColor="-16777216" FillColor="-32">
            <Coordinates XCoordinate="0.0" YCoordinate="338.0"/>
        </NodeGraphicsInfo>
        </NodeGraphicsInfos>
    </Pool>
</Pools>
<WorkflowProcesses>
    <WorkflowProcess Id="2" Name="EORDER">
        <ProcessHeader/>
        <FormalParameters>
            <FormalParameter Id="orderString" Mode="IN">
                <DataType>
                    <BasicType Type="STRING"/>
                </DataType>
            </FormalParameter>
            <FormalParameter Id="returnMessage" Mode="OUT">
                <DataType>
                    <BasicType Type="STRING"/>
                </DataType>
            </FormalParameter>
        </FormalParameters>
        <Applications>
            <Application Id="transformData">
                <FormalParameters>
                    <FormalParameter Id="orderStringIn" Mode="IN">
                        <DataType>
                            <BasicType Type="STRING"/>
                        </DataType>
                    </FormalParameter>
                    <FormalParameter Id="orderInfo" Mode="OUT">
                        <DataType>
                            <DeclaredType Id="Order"/>
                        </DataType>
                    </FormalParameter>
                </FormalParameters>
            </Application>
```

```
                <Application Id="checkData">
                        <FormalParameters>
                                <FormalParameter Id="orderInfo" Mode="IN">
                                        <DataType>
                                                <DeclaredType Id="Order"/>
                                        </DataType>
                                </FormalParameter>
                                <FormalParameter Id="statusOut" Mode="OUT">
                                        <DataType>
                                                <DeclaredType Id="OrderStatus"/>
                                        </DataType>
                                </FormalParameter>
                        </FormalParameters>
                </Application>
                <Application Id="checkVendor">
                        <FormalParameters>
                                <FormalParameter Id="accountNumberIn" Mode="IN">
                                        <DataType>
                                                <BasicType Type="INTEGER"/>
                                        </DataType>
                                </FormalParameter>
                                <FormalParameter Id="amountIn" Mode="IN">
                                        <DataType>
                                                <BasicType Type="FLOAT"/>
                                        </DataType>
                                </FormalParameter>
                                <FormalParameter Id="statusOut" Mode="OUT">
                                        <DataType>
                                                <DeclaredType Id="OrderStatus"/>
                                        </DataType>
                                </FormalParameter>
                        </FormalParameters>
                </Application>
                <Application Id="enterOrder">
                        <FormalParameters>
                                <FormalParameter Id="orderInfoIn" Mode="IN">
                                        <DataType>
                                                <DeclaredType Id="Order"/>
                                        </DataType>
                                </FormalParameter>
                                <FormalParameter Id="orderNumber" Mode="OUT">
                                        <DataType>
                                                <BasicType Type="INTEGER"/>
                                        </DataType>
                                </FormalParameter>
                        </FormalParameters>
                </Application>
                <Application Id="composeMessage">
                        <FormalParameters>
                                <FormalParameter Id="statusIn" Mode="IN">
                                        <DataType>
                                                <DeclaredType Id="OrderStatus"/>
                                        </DataType>
                                </FormalParameter>
                                <FormalParameter Id="orderNumber" Mode="IN">
                                        <DataType>
                                                <BasicType Type="INTEGER"/>
                                        </DataType>
                                </FormalParameter>
                        </FormalParameters>
                </Application>
        </Applications>
        <DataFields>
                <DataField Id="1" Name="orderNumber" IsArray="FALSE">
                        <DataType>
                                <BasicType Type="INTEGER"/>
                        </DataType>
                        <Length>0</Length>
                        <Description/>
                </DataField>
                <DataField Id="3" Name="status" IsArray="FALSE">
                        <DataType>
                                <BasicType Type="STRING"/>
                        </DataType>
                        <Length>0</Length>
                        <Description/>
                </DataField>
                <DataField Id="4" Name="orderInfo" IsArray="FALSE">
                        <DataType>
                                <BasicType Type="STRING"/>
                        </DataType>
                        <Length>0</Length>
                        <Description/>
                </DataField>
        </DataFields>
        <ActivitySets/>
        <Activities>
                <Activity Id="10" Name="Transform Data">
                        <Implementation>
                                <Task>
                                        <TaskApplication Id="transformData">
                                                <ActualParameters>
                                                        <ActualParameter>orderString</ActualParameter>
                                                        <ActualParameter>orderInfo</ActualParameter>
                                                </ActualParameters>
                                        </TaskApplication>
                                </Task>
                        </Implementation>
        <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="0" Width="75.0" Height="50.0" BorderColor="-16777216" FillColor="-1114150">
                                <Coordinates XCoordinate="128.0" YCoordinate="96.0"/>
                        </NodeGraphicsInfo>
        </NodeGraphicsInfos>
                </Activity>
                <Activity Id="12" Name="Check Data">
                        <Implementation>
                                <Task>
                                        <TaskApplication Id="checkData">
                                                <ActualParameters>
                                                        <ActualParameter>orderInfo</ActualParameter>
                                                        <ActualParameter>status</ActualParameter>
                                                </ActualParameters>
                                        </TaskApplication>
                                </Task>
                        </Implementation>
                        <TransitionRestrictions>
                                <TransitionRestriction>
                                        <Split Type="XOR">
                                                <TransitionRefs>
                                                        <TransitionRef Id="17"/>
                                                        <TransitionRef Id="23"/>
                                                </TransitionRefs>
                                        </Split>
                                </TransitionRestriction>
                        </TransitionRestrictions>
        <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="0" Width="75.0" Height="50.0" BorderColor="-16777216" FillColor="-1114150">
                                <Coordinates XCoordinate="228.0" YCoordinate="98.0"/>
                        </NodeGraphicsInfo>
        </NodeGraphicsInfos>
                </Activity>
                <Activity Id="13" Name="check order type">
                        <Route GatewayType="XOR" MarkerVisible="true"/>
                        <TransitionRestrictions>
                                <TransitionRestriction>
                                        <Split Type="XOR">
                                                <TransitionRefs>
                                                        <TransitionRef Id="18"/>
                                                        <TransitionRef Id="20"/>
                                                </TransitionRefs>
                                        </Split>
                                </TransitionRestriction>
                        </TransitionRestrictions>
        <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="0" Width="44.0" Height="44.0" BorderColor="-16777216" FillColor="-1114150">
                                <Coordinates XCoordinate="420.0" YCoordinate="70.0"/>
                        </NodeGraphicsInfo>
        </NodeGraphicsInfos>
                </Activity>
                <Activity Id="14" Name="Check Vendor Account">
                        <Implementation>
                                <Task>
                                        <TaskApplication Id="checkVendor">
                                                <ActualParameters>
                                                        <ActualParameter>orderInfo.AccountNumber</ActualParameter>
                                                        <ActualParameter>orderInfo.ToltalAmount</ActualParameter>
                                                        <ActualParameter>status</ActualParameter>
                                                </ActualParameters>
                                        </TaskApplication>
                                </Task>
                        </Implementation>
                        <TransitionRestrictions>
                                <TransitionRestriction>
                                        <Split Type="XOR">
                                                <TransitionRefs>
                                                        <TransitionRef Id="24"/>
                                                        <TransitionRef Id="28"/>
                                                </TransitionRefs>
                                        </Split>
                                </TransitionRestriction>
                        </TransitionRestrictions>
        <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="0" Width="43.0" Height="43.0" BorderColor="-16777216" FillColor="-1114150">
                                <Coordinates XCoordinate="576.0" YCoordinate="130.0"/>
                        </NodeGraphicsInfo>
        </NodeGraphicsInfos>
                </Activity>
                <Activity Id="19" Name="Credit Check">
                        <Implementation>
                                <SubFlow Id="3" Execution="SYNCHR">
                                        <ActualParameters>
                                                <ActualParameter>orderInfo.accountNumber</ActualParameter>
                                                <ActualParameter>orderInfo.cardType</ActualParameter>
                                                <ActualParameter>orderInfo.emailAddress</ActualParameter>
```

```
                    <ActualParameter>status</ActualParameter>
                </ActualParameters>
            </SubFlow>
        </Implementation>
        <TransitionRestrictions>
            <TransitionRestriction>
                <Split Type="XOR">
                    <TransitionRefs>
                        <TransitionRef Id="27"/>
                        <TransitionRef Id="25"/>
                    </TransitionRefs>
                </Split>
            </TransitionRestriction>
        </TransitionRestrictions>
<NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" LaneId="0" Width="52.0" Height="46.0" BorderColor="-16777216" FillColor="-1114150">
            <Coordinates XCoordinate="563.0" YCoordinate="11.0"/>
        </NodeGraphicsInfo>
</NodeGraphicsInfos>
    </Activity>
    <Activity Id="21" Name="Compose Rejection Message">
        <Implementation>
            <Task>
                <TaskApplication Id="composeMessage">
                    <ActualParameters>
                        <ActualParameter>status</ActualParameter>
                        <ActualParameter>orderNumber</ActualParameter>
                    </ActualParameters>
                </TaskApplication>
            </Task>
        </Implementation>
        <TransitionRestrictions>
            <TransitionRestriction>
                <Join Type="XOR"/>
            </TransitionRestriction>
        </TransitionRestrictions>
<NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" LaneId="0" Width="66.0" Height="42.0" BorderColor="-16777216" FillColor="-1114150">
            <Coordinates XCoordinate="998.0" YCoordinate="195.0"/>
        </NodeGraphicsInfo>
</NodeGraphicsInfos>
    </Activity>
    <Activity Id="26" Name="Enter Order">
        <Implementation>
            <Task>
                <TaskApplication Id="enterOrder">
                    <ActualParameters>
                        <ActualParameter>orderInfo</ActualParameter>
                        <ActualParameter>orderNumber</ActualParameter>
                    </ActualParameters>
                </TaskApplication>
            </Task>
        </Implementation>
        <Performers>
            <Performer>DBConnection</Performer>
        </Performers>
        <TransitionRestrictions>
            <TransitionRestriction>
                <Join Type="XOR"/>
            </TransitionRestriction>
        </TransitionRestrictions>
<NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" LaneId="0" Width="49.0" Height="50.0" BorderColor="-16777216" FillColor="-1114150">
            <Coordinates XCoordinate="873.0" YCoordinate="73.0"/>
        </NodeGraphicsInfo>
</NodeGraphicsInfos>
    </Activity>
    <Activity Id="29" Name="Fill Order">
        <Implementation>
            <SubFlow Id="4" Execution="SYNCHR">
                <ActualParameters>
                    <ActualParameter>orderNumber</ActualParameter>
                    <ActualParameter>orderInfo.orderType</ActualParameter>
                    <ActualParameter>orderInfo.emailAddress</ActualParameter>
                </ActualParameters>
            </SubFlow>
        </Implementation>
<NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" LaneId="0" Width="60.0" Height="35.0" BorderColor="-16777216" FillColor="-1114150">
            <Coordinates XCoordinate="996.0" YCoordinate="78.0"/>
        </NodeGraphicsInfo>
</NodeGraphicsInfos>
    </Activity>
    <Activity Id="30" Name="">
        <Route GatewayType="AND" MarkerVisible="true"/>
        <TransitionRestrictions>
            <TransitionRestriction>
                <Split Type="AND">
                    <TransitionRefs>
                        <TransitionRef Id="36"/>
                        <TransitionRef Id="39"/>
                        <TransitionRef Id="40"/>
```

```
                                    </TransitionRefs>
                              </Split>
                        </TransitionRestriction>
                  </TransitionRestrictions>
<NodeGraphicsInfos>
                  <NodeGraphicsInfo Page="1" LaneId="0" Width="37.0" Height="37.0" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="941.3999633789062" YCoordinate="81.20000457763672"/>
                  </NodeGraphicsInfo>
</NodeGraphicsInfos>
            </Activity>
            <Activity Id="31" Name="">
                  <Route GatewayType="AND" MarkerVisible="true"/>
                  <TransitionRestrictions>
                        <TransitionRestriction>
                              <Join Type="AND"/>
                        </TransitionRestriction>
                  </TransitionRestrictions>
<NodeGraphicsInfos>
                  <NodeGraphicsInfo Page="1" LaneId="0" Width="41.0" Height="41.0" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="1086.800048828125" YCoordinate="75.19999694824219"/>
                  </NodeGraphicsInfo>
</NodeGraphicsInfos>
            </Activity>
            <Activity Id="33" Name="Compose Acceptance Message">
                  <Implementation>
                        <Task>
                              <TaskApplication Id="composeMessage">
                                    <ActualParameters>
                                          <ActualParameter>status</ActualParameter>
                                          <ActualParameter>orderNumber</ActualParameter>
                                    </ActualParameters>
                              </TaskApplication>
                        </Task>
                  </Implementation>
<NodeGraphicsInfos>
                  <NodeGraphicsInfo Page="1" LaneId="0" Width="59.0" Height="43.0" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="994.0" YCoordinate="22.0"/>
                  </NodeGraphicsInfo>
</NodeGraphicsInfos>
            </Activity>
            <Activity Id="34" Name="Email Confirmation">
                  <Implementation>
                        <No/>
                  </Implementation>
                  <ExtendedAttributes>
                        <ExtendedAttribute Name="SystemActivity" Value="Email"/>
                        <ExtendedAttribute Name="Email">
                              <xyz:Email to="orderInfo.emailAddress" subject="Order orderNumber">
                                    <xyz:MessageText>Order number orderNumber is being processed.
                                          Thank-you for ordering from PQR Products, Inc</xyz:MessageText>
                              </xyz:Email>
                        </ExtendedAttribute>
                  </ExtendedAttributes>
<NodeGraphicsInfos>
                  <NodeGraphicsInfo Page="1" LaneId="0" Width="62.0" Height="40.0" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="997.0" YCoordinate="131.0"/>
                  </NodeGraphicsInfo>
</NodeGraphicsInfos>
            </Activity>
            <Activity Id="46" Name="">
                  <Event>
                        <IntermediateEvent Trigger="Error" Target="10">
                              <ResultError ErrorCode="1"/>
                        </IntermediateEvent>
                  </Event>
<NodeGraphicsInfos>
                  <NodeGraphicsInfo Page="1" LaneId="0" Width="50.0" Height="50.0" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="142.0" YCoordinate="121.0"/>
                  </NodeGraphicsInfo>
</NodeGraphicsInfos>
            </Activity>
            <Activity Id="48" Name="start">
                  <Event>
                        <IntermediateEvent Trigger="None"/>
                  </Event>
<NodeGraphicsInfos>
                  <NodeGraphicsInfo Page="1" LaneId="0" Width="39.0" Height="39.0" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="59.0" YCoordinate="103.0"/>
                  </NodeGraphicsInfo>
</NodeGraphicsInfos>
            </Activity>
            <Activity Id="50" Name="end">
                  <Event>
                        <IntermediateEvent Trigger="None"/>
                  </Event>
<NodeGraphicsInfos>
                  <NodeGraphicsInfo Page="1" LaneId="0" Width="33.0" Height="33.0" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="1150.0" YCoordinate="79.0"/>
                  </NodeGraphicsInfo>
</NodeGraphicsInfos>
            </Activity>
</Activities>
```

```
<Transitions>
    <Transition Id="16" Name="" From="10" To="12">
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="203.5" YCoordinate="122.82308197021484"/>
                <Coordinates XCoordinate="228.5" YCoordinate="123.2227783203125"/>
            </ConnectorGraphicsInfo>
        <ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="20" Name="" From="13" To="19">
        <Condition Type="CONDITION">orderType == "Credit"</Condition>
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="464.1615295410156" YCoordinate="92.83846282958984"/>
                <Coordinates XCoordinate="506.0" YCoordinate="93.0"/>
                <Coordinates XCoordinate="507.0" YCoordinate="33.0"/>
                <Coordinates XCoordinate="563.5" YCoordinate="32.13414764404297"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="18" Name="" From="13" To="14">
        <Condition Type="CONDITION">orderType == "PO"</Condition>
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="464.5" YCoordinate="92.5"/>
                <Coordinates XCoordinate="506.0" YCoordinate="92.0"/>
                <Coordinates XCoordinate="507.0" YCoordinate="151.0"/>
                <Coordinates XCoordinate="576.6181030273438" YCoordinate="151.88186645507812"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="28" Name="" From="14" To="26">
        <Condition Type="CONDITION">staus == "Accept"</Condition>
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="619.450927734375" YCoordinate="151.95091247558594"/>
                <Coordinates XCoordinate="816.0" YCoordinate="151.0"/>
                <Coordinates XCoordinate="817.0" YCoordinate="98.0"/>
                <Coordinates XCoordinate="853.5" YCoordinate="96.11983489990234"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="24" Name="" From="14" To="21">
        <Condition Type="CONDITION">staus == "OverLimit"</Condition>
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="619.3720092773438" YCoordinate="151.8720245361328"/>
                <Coordinates XCoordinate="681.0" YCoordinate="151.0"/>
                <Coordinates XCoordinate="683.0" YCoordinate="216.0"/>
                <Coordinates XCoordinate="993.5" YCoordinate="215.60791015625"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="25" Name="" From="19" To="21">
        <Condition Type="OTHERWISE"/>
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="589.5" YCoordinate="57.5"/>
                <Coordinates XCoordinate="589.0" YCoordinate="111.0"/>
                <Coordinates XCoordinate="743.0" YCoordinate="111.0"/>
                <Coordinates XCoordinate="743.0" YCoordinate="216.0"/>
                <Coordinates XCoordinate="993.5" YCoordinate="215.63043212890625"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="35" Name="" From="26" To="30">
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="922.5" YCoordinate="98.00992584228516"/>
                <Coordinates XCoordinate="939.65673828125" YCoordinate="97.15673828125"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="36" Name="" From="30" To="29">
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="974.5991821289062" YCoordinate="96.90082550048828"/>
                <Coordinates XCoordinate="996.5" YCoordinate="97.54743194580078"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="37" Name="" From="29" To="31">
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="1056.5" YCoordinate="96.90939331054688"/>
                <Coordinates XCoordinate="1078.59423828125" YCoordinate="97.09429168701172"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="39" Name="" From="30" To="33">
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
```

```
                        <Coordinates XCoordinate="970.4146728515625" YCoordinate="91.71468353271484"/>
                        <Coordinates XCoordinate="1000.2640380859375" YCoordinate="65.5"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="40" Name="" From="30" To="34">
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="970.9235229492188" YCoordinate="108.17644500732422"/>
                        <Coordinates XCoordinate="1001.8546752929688" YCoordinate="131.5"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="41" Name="" From="33" To="31">
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="1053.5" YCoordinate="62.4793586730957"/>
                        <Coordinates XCoordinate="1095.1297607421875" YCoordinate="88.3702392578125"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="42" Name="" From="34" To="31">
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="1057.8876953125" YCoordinate="131.5"/>
                        <Coordinates XCoordinate="1095.8941650390625" YCoordinate="104.79414367675781"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="49" Name="" From="48" To="10">
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="97.99977111816406" YCoordinate="122.40513610839844"/>
                        <Coordinates XCoordinate="128.5" YCoordinate="122.25675201416016"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="47" Name="" From="46" To="21">
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="166.2860107421875" YCoordinate="170.98980712890625"/>
                        <Coordinates XCoordinate="165.0" YCoordinate="216.0"/>
                        <Coordinates XCoordinate="998.5" YCoordinate="216.5"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="17" Name="" From="12" To="13">
                <Condition Type="CONDITION">status == "Valid Claim"</Condition>
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="303.5" YCoordinate="123.5"/>
                        <Coordinates XCoordinate="360.0" YCoordinate="123.0"/>
                        <Coordinates XCoordinate="359.0" YCoordinate="92.0"/>
                        <Coordinates XCoordinate="420.5" YCoordinate="92.5"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="23" Name="" From="12" To="21">
                <Condition Type="CONDITION">status == "Invalid Data"</Condition>
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="264.84259033203125" YCoordinate="148.5"/>
                        <Coordinates XCoordinate="263.0" YCoordinate="177.0"/>
                        <Coordinates XCoordinate="343.0" YCoordinate="177.0"/>
                        <Coordinates XCoordinate="344.0" YCoordinate="216.0"/>
                        <Coordinates XCoordinate="998.5" YCoordinate="216.5"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="100" Name="" From="31" To="50">
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="1128.2655029296875" YCoordinate="96.23451232910156"/>
                        <Coordinates XCoordinate="1150.002197265625" YCoordinate="95.77201080322266"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="101" Name="" From="21" To="50">
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="1053.5" YCoordinate="62.4793586730957"/>
                        <Coordinates XCoordinate="1095.1297607421875" YCoordinate="88.3702392578125"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="27" Name="" From="19" To="26">
                <Condition Type="CONDITION">status == "Accept"</Condition>
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="615.5" YCoordinate="34.5"/>
                        <Coordinates XCoordinate="817.0" YCoordinate="34.0"/>
                        <Coordinates XCoordinate="816.0" YCoordinate="97.0"/>
```

```
                                    <Coordinates XCoordinate="873.5" YCoordinate="98.19938659667969"/>
                              </ConnectorGraphicsInfo>
                        </ConnectorGraphicsInfos>
                  </Transition>
            </Transitions>
      </WorkflowProcess>
      <WorkflowProcess Id="3" Name="CreditCheck" AccessLevel="PRIVATE">
            <ProcessHeader/>
            <FormalParameters>
                  <FormalParameter Id="accountNumber" Mode="IN">
                        <DataType>
                              <BasicType Type="INTEGER"/>
                        </DataType>
                  </FormalParameter>
                  <FormalParameter Id="amount" Mode="IN">
                        <DataType>
                              <BasicType Type="FLOAT"/>
                        </DataType>
                  </FormalParameter>
                  <FormalParameter Id="cardType" Mode="IN">
                        <DataType>
                              <DeclaredType Id="CardType"/>
                        </DataType>
                  </FormalParameter>
                  <FormalParameter Id="status" Mode="OUT">
                        <DataType>
                              <DeclaredType Id="OrderStatus"/>
                        </DataType>
                  </FormalParameter>
            </FormalParameters>
            <Applications>
                  <Application Id="setCreditInfo">
                        <Description>Creates and initializes a CreditInfo object.</Description>
                        <FormalParameters>
                              <FormalParameter Id="accountNumber" Mode="IN">
                                    <DataType>
                                          <BasicType Type="INTEGER"/>
                                    </DataType>
                              </FormalParameter>
                              <FormalParameter Id="amount" Mode="IN">
                                    <DataType>
                                          <BasicType Type="FLOAT"/>
                                    </DataType>
                              </FormalParameter>
                              <FormalParameter Id="cardType" Mode="IN">
                                    <DataType>
                                          <DeclaredType Id="CardType"/>
                                    </DataType>
                              </FormalParameter>
                              <FormalParameter Id="creditInfo" Mode="OUT">
                                    <DataType>
                                          <DeclaredType Id="CreditInfo"/>
                                    </DataType>
                              </FormalParameter>
                        </FormalParameters>
                  </Application>
                  <Application Id="getCreditAuthorization">
                        <Description>Gets credit authorization from a charge card web service.</Description>
                        <ExternalReference location="http://wfmc.org/standards/docs/xpdl_sample/creditService.wsdl" xref="GetCreditAuthorization"/>
                  </Application>
                  <Application Id="setOrderStatus">
                        <Description>Converts status returned by credit check to OrderStatus.</Description>
                        <FormalParameters>
                              <FormalParameter Id="creditStatus" Mode="IN">
                                    <DataType>
                                          <BasicType Type="STRING"/>
                                    </DataType>
                              </FormalParameter>
                              <FormalParameter Id="orderStatus" Mode="OUT">
                                    <DataType>
                                          <DeclaredType Id="OrderStatus"/>
                                    </DataType>
                              </FormalParameter>
                        </FormalParameters>
                  </Application>
            </Applications>
            <DataFields>
                  <DataField Id="creditStatus" IsArray="FALSE">
                        <DataType>
                              <BasicType Type="STRING"/>
                        </DataType>
                        <Length>0</Length>
                  </DataField>
            </DataFields>
            <ActivitySets/>
            <Activities>
                  <Activity Id="52" Name="start">
                        <Event>
                              <StartEvent Trigger="None"/>
                        </Event>
            <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="1" Width="37.0" Height="37.0" BorderColor="-16777216" FillColor="-1114150">
```

```
                    <Coordinates XCoordinate="81.0" YCoordinate="272.0"/>
                </NodeGraphicsInfo>
        </NodeGraphicsInfos>
            </Activity>
            <Activity Id="53" Name="Set Credit Info">
                <Implementation>
                    <Task>
                        <TaskApplication Id="setCreditInfo">
                            <ActualParameters>
                                <ActualParameter>accountNumber</ActualParameter>
                                <ActualParameter>amount</ActualParameter>
                                <ActualParameter>cardType</ActualParameter>
                                <ActualParameter>creditInfo</ActualParameter>
                            </ActualParameters>
                        </TaskApplication>
                    </Task>
                </Implementation>
                <Performers>
                    <Performer>DBConnection</Performer>
                </Performers>
        <NodeGraphicsInfos>
                <NodeGraphicsInfo Page="1" LaneId="1" Width="75.0" Height="50.0" BorderColor="-16777216" FillColor="-1114150">
                    <Coordinates XCoordinate="162.0" YCoordinate="265.0"/>
                </NodeGraphicsInfo>
        </NodeGraphicsInfos>
            </Activity>
            <Activity Id="54" Name="Get Credit Authorization">
                <Implementation>
                    <Task>
                        <TaskApplication Id="getCreditAuthorization">
                            <ActualParameters>
                                <ActualParameter>creditInfo</ActualParameter>
                                <ActualParameter>creditStatus</ActualParameter>
                            </ActualParameters>
                        </TaskApplication>
                    </Task>
                </Implementation>
                <ExtendedAttributes>
                    <ExtendedAttribute Name="SystemActivity" Value="WebService"/>
                </ExtendedAttributes>
        <NodeGraphicsInfos>
                <NodeGraphicsInfo Page="1" LaneId="1" Width="75.0" Height="50.0" BorderColor="-16777216" FillColor="-1114150">
                    <Coordinates XCoordinate="286.0" YCoordinate="266.0"/>
                </NodeGraphicsInfo>
        </NodeGraphicsInfos>
            </Activity>
            <Activity Id="55" Name="Set Order Status">
                <Implementation>
                    <Task>
                        <TaskApplication Id="setOrderStatus">
                            <ActualParameters>
                                <ActualParameter>creditStatus</ActualParameter>
                                <ActualParameter>status</ActualParameter>
                            </ActualParameters>
                        </TaskApplication>
                    </Task>
                </Implementation>
        <NodeGraphicsInfos>
                <NodeGraphicsInfo Page="1" LaneId="1" Width="75.0" Height="50.0" BorderColor="-16777216" FillColor="-1114150">
                    <Coordinates XCoordinate="412.0" YCoordinate="267.0"/>
                </NodeGraphicsInfo>
        </NodeGraphicsInfos>
            </Activity>
            <Activity Id="56" Name="end">
                <Event>
                    <EndEvent Result="None"/>
                </Event>
        <NodeGraphicsInfos>
                <NodeGraphicsInfo Page="1" LaneId="1" Width="37.0" Height="37.0" BorderColor="-16777216" FillColor="-1114150">
                    <Coordinates XCoordinate="529.0" YCoordinate="275.0"/>
                </NodeGraphicsInfo>
        </NodeGraphicsInfos>
            </Activity>
        </Activities>
        <Transitions>
            <Transition Id="57" Name="" From="52" To="53">
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="117.99984741210938" YCoordinate="293.57501220703125"/>
                        <Coordinates XCoordinate="162.5" YCoordinate="293.7554016113281"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="58" Name="" From="53" To="54">
                <ConnectorGraphicsInfos>
                    <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="237.5" YCoordinate="294.14959716796875"/>
                        <Coordinates XCoordinate="286.5" YCoordinate="294.507080078125"/>
                    </ConnectorGraphicsInfo>
                </ConnectorGraphicsInfos>
            </Transition>
            <Transition Id="59" Name="" From="54" To="55">
```

```
                    <ConnectorGraphicsInfos>
                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                            <Coordinates XCoordinate="361.5" YCoordinate="295.1332092285156"/>
                            <Coordinates XCoordinate="412.5" YCoordinate="295.5028076171875"/>
                        </ConnectorGraphicsInfo>
                    </ConnectorGraphicsInfos>
                </Transition>
                <Transition Id="60" Name="" From="55" To="56">
                    <ConnectorGraphicsInfos>
                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                            <Coordinates XCoordinate="487.5" YCoordinate="294.5947265625"/>
                            <Coordinates XCoordinate="526.0115356445312" YCoordinate="293.1529846191406"/>
                        </ConnectorGraphicsInfo>
                    </ConnectorGraphicsInfos>
                </Transition>
            </Transitions>
        </WorkflowProcess>
        <WorkflowProcess Id="4" Name="Fill Order" AccessLevel="PRIVATE">
            <ProcessHeader/>
            <FormalParameters>
                <FormalParameter Id="orderNumber" Mode="IN">
                    <DataType>
                        <BasicType Type="INTEGER"/>
                    </DataType>
                    <Description>Order number assigned to the order.</Description>
                </FormalParameter>
                <FormalParameter Id="orderType" Mode="IN">
                    <DataType>
                        <ExternalReference location="http://wfmc.org/standards/docs/xpdl_sample/orderschema.xsd" xref="orderType"
namespace="orderschema/Order"/>
                    </DataType>
                </FormalParameter>
                <FormalParameter Id="emailAddress" Mode="IN">
                    <DataType>
                        <BasicType Type="STRING"/>
                    </DataType>
                </FormalParameter>
            </FormalParameters>
            <Participants>
                <Participant Id="Shipper">
                    <ParticipantType Type="ROLE"/>
                    <Description>Order shipper</Description>
                </Participant>
            </Participants>
            <Applications>
                <Application Id="shipOrder">
                    <Description>This application presents a screen that presents order information
                        for the order identified by shipOrder. The user may update the order with
                        any changes such as back order information. It returns a string indicating
                        whether the order is complete or on back order.</Description>
                    <FormalParameters>
                        <FormalParameter Id="OrderNumberParam" Mode="IN">
                            <DataType>
                                <BasicType Type="INTEGER"/>
                            </DataType>
                        </FormalParameter>
                        <FormalParameter Id="Status" Mode="OUT">
                            <DataType>
                                <BasicType Type="STRING"/>
                            </DataType>
                            <Description>The String that describes the status -- either "Complete"
                                or "Backorder"</Description>
                        </FormalParameter>
                    </FormalParameters>
                </Application>
                <Application Id="charge">
                    <Description>Charges the credit card and prepares a receipt for a credit order</Description>
                    <ExternalReference location="http://wfmc.org/standards/docs/xpdl_sample/creditService.wsdl" xref="ChargeCreditAccount"/>
                </Application>
                <Application Id="billAccount">
                    <Description>Bills the vendor account</Description>
                    <FormalParameters>
                        <FormalParameter Id="orderNumberParam" Mode="IN">
                            <DataType>
                                <BasicType Type="INTEGER"/>
                            </DataType>
                        </FormalParameter>
                    </FormalParameters>
                </Application>
                <Application Id="createInvoice">
                    <Description>Creates an invoice using the order information and stores it on a
                        server.</Description>
                    <FormalParameters>
                        <FormalParameter Id="orderNumber" Mode="IN">
                            <DataType>
                                <BasicType Type="INTEGER"/>
                            </DataType>
                        </FormalParameter>
                        <FormalParameter Id="docURI" Mode="OUT">
                            <DataType>
                                <BasicType Type="STRING"/>
                            </DataType>
                        </FormalParameter>
                    </FormalParameters>
                </Application>
                <Application Id="createReceipt">
                    <Description>Creates a receiptusing the order information and stores it on a
                        server.</Description>
                    <FormalParameters>
                        <FormalParameter Id="orderNumber" Mode="IN">
                            <DataType>
                                <BasicType Type="INTEGER"/>
                            </DataType>
                        </FormalParameter>
                        <FormalParameter Id="docURI" Mode="OUT">
                            <DataType>
                                <BasicType Type="STRING"/>
                            </DataType>
                        </FormalParameter>
                    </FormalParameters>
                </Application>
                <Application Id="cancelOrder">
                    <FormalParameters>
                        <FormalParameter Id="orderNumberIn" Mode="IN">
                            <DataType>
                                <BasicType Type="INTEGER"/>
                            </DataType>
                        </FormalParameter>
                    </FormalParameters>
                </Application>
            </Applications>
            <DataFields>
                <DataField Id="docURI" IsArray="FALSE">
                    <DataType>
                        <BasicType Type="STRING"/>
                    </DataType>
                    <Description>URI of receipt or invoice.</Description>
                </DataField>
                <DataField Id="orderStatus" IsArray="FALSE">
                    <DataType>
                        <BasicType Type="STRING"/>
                    </DataType>
                </DataField>
                <DataField Id="creditInfo" IsArray="FALSE">
                    <DataType>
                        <DeclaredType Id="CreditInfo"/>
                    </DataType>
                </DataField>
                <DataField Id="creditStatus" IsArray="FALSE">
                    <DataType>
                        <BasicType Type="STRING"/>
                    </DataType>
                </DataField>
            </DataFields>
            <ActivitySets/>
            <Activities>
                <Activity Id="61" Name="start">
                    <Event>
                        <IntermediateEvent Trigger="None"/>
                    </Event>
            <NodeGraphicsInfos>
                    <NodeGraphicsInfo Page="1" LaneId="2" Width="33.0" Height="33.0" BorderColor="-16777216" FillColor="-1114150">
                        <Coordinates XCoordinate="81.0" YCoordinate="375.0"/>
                    </NodeGraphicsInfo>
            </NodeGraphicsInfos>
                </Activity>
                <Activity Id="62" Name="Ship Order">
                    <Description>View order and enter fulfillment info</Description>
                    <Implementation>
                        <Task>
                            <TaskApplication Id="shipOrder">
                                <ActualParameters>
                                    <ActualParameter>orderNumber</ActualParameter>
                                    <ActualParameter>orderStatus</ActualParameter>
                                </ActualParameters>
                            </TaskApplication>
                        </Task>
                    </Implementation>
                    <Performers>
                        <Performer>DBConnection and Shipper</Performer>
                    </Performers>
                    <Deadline Execution="ASYNCHR">
                        <DeadlineDuration>3 days</DeadlineDuration>
                        <ExceptionName>notifyException</ExceptionName>
                    </Deadline>
                    <Deadline Execution="SYNCHR">
                        <DeadlineDuration>5 days</DeadlineDuration>
                        <ExceptionName>timeoutException</ExceptionName>
                    </Deadline>
                    <TransitionRestrictions>
                        <TransitionRestriction>
                            <Split Type="AND">
                                <TransitionRefs>
                                    <TransitionRef Id="73"/>
```

```
                                        <TransitionRef Id="74"/>
                                        <TransitionRef Id="81"/>
                                        <TransitionRef Id="82"/>
                                    </TransitionRefs>
                                </Split>
                            </TransitionRestriction>
                        </TransitionRestrictions>
                <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="2" Width="57.0" Height="39.0" BorderColor="-16777216" FillColor="-1114150">
                            <Coordinates XCoordinate="167.0" YCoordinate="373.0"/>
                        </NodeGraphicsInfo>
                </NodeGraphicsInfos>
                    </Activity>
                    <Activity Id="63" Name="Billing">
                        <Implementation>
                            <Task>
                                <TaskApplication Id="billAccount">
                                    <ActualParameters>
                                        <ActualParameter>orderNumber</ActualParameter>
                                    </ActualParameters>
                                </TaskApplication>
                            </Task>
                        </Implementation>
                        <Performers>
                            <Performer>DBConnection</Performer>
                        </Performers>
                <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="2" Width="45.0" Height="29.0" BorderColor="-16777216" FillColor="-1114150">
                            <Coordinates XCoordinate="388.0" YCoordinate="350.0"/>
                        </NodeGraphicsInfo>
                </NodeGraphicsInfos>
                    </Activity>
                    <Activity Id="64" Name="Create Invoice">
                        <Implementation>
                            <Task>
                                <TaskApplication Id="createInvoice">
                                    <ActualParameters>
                                        <ActualParameter>orderNumber</ActualParameter>
                                        <ActualParameter>docUri</ActualParameter>
                                    </ActualParameters>
                                </TaskApplication>
                            </Task>
                        </Implementation>
                        <Performers>
                            <Performer>DBConnection</Performer>
                        </Performers>
                <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="2" Width="47.0" Height="32.0" BorderColor="-16777216" FillColor="-1114150">
                            <Coordinates XCoordinate="541.0" YCoordinate="351.0"/>
                        </NodeGraphicsInfo>
                </NodeGraphicsInfos>
                    </Activity>
                    <Activity Id="65" Name="Email Receipt or Invoice">
                        <Implementation>
                            <No/>
                        </Implementation>
                <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="2" Width="61.0" Height="42.0" BorderColor="-16777216" FillColor="-1114150">
                            <Coordinates XCoordinate="645.0" YCoordinate="370.0"/>
                        </NodeGraphicsInfo>
                </NodeGraphicsInfos>
                    </Activity>
                    <Activity Id="66" Name="end">
                        <Event>
                            <IntermediateEvent Trigger="None"/>
                        </Event>
                <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="2" Width="33.0" Height="33.0" BorderColor="-16777216" FillColor="-1114150">
                            <Coordinates XCoordinate="807.0" YCoordinate="395.0"/>
                        </NodeGraphicsInfo>
                </NodeGraphicsInfos>
                    </Activity>
                    <Activity Id="67" Name="Create Receipt">
                        <Implementation>
                            <Task>
                                <TaskApplication Id="createReceipt">
                                    <ActualParameters>
                                        <ActualParameter>orderNumber</ActualParameter>
                                        <ActualParameter>docUri</ActualParameter>
                                    </ActualParameters>
                                </TaskApplication>
                            </Task>
                        </Implementation>
                        <Performers>
                            <Performer>DBConnection</Performer>
                        </Performers>
                <NodeGraphicsInfos>
                        <NodeGraphicsInfo Page="1" LaneId="2" Width="52.0" Height="32.0" BorderColor="-16777216" FillColor="-1114150">
                            <Coordinates XCoordinate="539.0" YCoordinate="402.0"/>
                        </NodeGraphicsInfo>
                </NodeGraphicsInfos>
                    </Activity>
```

```
    <Activity Id="68" Name="Get Credit Info">
        <Implementation>
            <Task>
                <TaskApplication Id="getCreditInfo">
                    <ActualParameters>
                        <ActualParameter>orderNumber</ActualParameter>
                        <ActualParameter>creditInfo</ActualParameter>
                    </ActualParameters>
                </TaskApplication>
            </Task>
        </Implementation>
        <Performers>
            <Performer>DBConnection</Performer>
        </Performers>
<NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" LaneId="2" Width="50.0" Height="31.0" BorderColor="-16777216" FillColor="-1114150">
            <Coordinates XCoordinate="387.0" YCoordinate="402.0"/>
        </NodeGraphicsInfo>
</NodeGraphicsInfos>
    </Activity>
    <Activity Id="69" Name="Charge Credit Card">
        <Implementation>
            <No/>
        </Implementation>
<NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" LaneId="2" Width="51.0" Height="43.0" BorderColor="-16777216" FillColor="-1114150">
            <Coordinates XCoordinate="458.0" YCoordinate="398.0"/>
        </NodeGraphicsInfo>
</NodeGraphicsInfos>
    </Activity>
    <Activity Id="70" Name="Raise Alarm">
        <Implementation>
            <No/>
        </Implementation>
        <ExtendedAttributes>
            <ExtendedAttribute Name="SystemActivity" Value="Alarm"/>
        </ExtendedAttributes>
<NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" LaneId="2" Width="53.0" Height="33.0" BorderColor="-16777216" FillColor="-1114150">
            <Coordinates XCoordinate="127.0" YCoordinate="447.0"/>
        </NodeGraphicsInfo>
<NodeGraphicsInfos>
    </Activity>
    <Activity Id="71" Name="Cancel Order">
        <Description>View order and enter fulfillment info</Description>
        <Implementation>
            <Task>
                <TaskApplication Id="cancelOrder">
                    <ActualParameters>
                        <ActualParameter>orderNumber</ActualParameter>
                    </ActualParameters>
                </TaskApplication>
            </Task>
        </Implementation>
        <Performers>
            <Performer>DBConnection</Performer>
        </Performers>
<NodeGraphicsInfos>
        <NodeGraphicsInfo Page="1" LaneId="2" Width="59.0" Height="38.0" BorderColor="-16777216" FillColor="-1114150">
            <Coordinates XCoordinate="308.0" YCoordinate="445.0"/>
        </NodeGraphicsInfo>
</NodeGraphicsInfos>
    </Activity>
</Activities>
<Transitions>
    <Transition Id="72" Name="" From="61" To="62">
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="113.98829650878906" YCoordinate="380.12127685546875"/>
                <Coordinates XCoordinate="168.5" YCoordinate="382.1752624511719"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="75" Name="" From="63" To="64">
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="433.5" YCoordinate="365.4674987792969"/>
                <Coordinates XCoordinate="541.5" YCoordinate="367.2218933105469"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="76" Name="" From="68" To="69">
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                <Coordinates XCoordinate="437.5" YCoordinate="419.1833190917969"/>
                <Coordinates XCoordinate="458.5" YCoordinate="419.70098876953125"/>
            </ConnectorGraphicsInfo>
        </ConnectorGraphicsInfos>
    </Transition>
    <Transition Id="77" Name="" From="69" To="67">
        <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
```

```
                                                <Coordinates XCoordinate="509.5" YCoordinate="419.9206237792969"/>
                                                <Coordinates XCoordinate="539.5" YCoordinate="419.32568359375"/>
                                        </ConnectorGraphicsInfo>
                                </ConnectorGraphicsInfos>
                        </Transition>
                        <Transition Id="78" Name="" From="64" To="65">
                                <ConnectorGraphicsInfos>
                                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                                                <Coordinates XCoordinate="588.5" YCoordinate="372.7461853027344"/>
                                                <Coordinates XCoordinate="645.5" YCoordinate="385.0826416015625"/>
                                        </ConnectorGraphicsInfo>
                                </ConnectorGraphicsInfos>
                        </Transition>
                        <Transition Id="79" Name="" From="67" To="65">
                                <ConnectorGraphicsInfos>
                                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                                                <Coordinates XCoordinate="591.5" YCoordinate="412.4445495605469"/>
                                                <Coordinates XCoordinate="645.5" YCoordinate="399.2513732910156"/>
                                        </ConnectorGraphicsInfo>
                                </ConnectorGraphicsInfos>
                        </Transition>
                        <Transition Id="80" Name="" From="65" To="66">
                                <ConnectorGraphicsInfos>
                                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                                                <Coordinates XCoordinate="706.5" YCoordinate="395.7386169433594"/>
                                                <Coordinates XCoordinate="807.14794921875" YCoordinate="409.29522705078125"/>
                                        </ConnectorGraphicsInfo>
                                </ConnectorGraphicsInfos>
                        </Transition>
                        <Transition Id="83" Name="" From="71" To="66">
                                <ConnectorGraphicsInfos>
                                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                                                <Coordinates XCoordinate="262.5" YCoordinate="464.5"/>
                                                <Coordinates XCoordinate="706.0" YCoordinate="464.0"/>
                                                <Coordinates XCoordinate="808.4353637695312" YCoordinate="418.23101806640625"/>
                                        </ConnectorGraphicsInfo>
                                </ConnectorGraphicsInfos>
                        </Transition>
                        <Transition Id="81" Name="notifyException" From="62" To="70">
                                <ConnectorGraphicsInfos>
                                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                                                <Coordinates XCoordinate="184.88658142089844" YCoordinate="412.5"/>
                                                <Coordinates XCoordinate="164.15447998046875" YCoordinate="447.5"/>
                                        </ConnectorGraphicsInfo>
                                </ConnectorGraphicsInfos>
                        </Transition>
                        <Transition Id="82" Name="timeoutException" From="62" To="71">
                                <ConnectorGraphicsInfos>
                                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                                                <Coordinates XCoordinate="206.23150634765625" YCoordinate="412.5"/>
                                                <Coordinates XCoordinate="223.3087158203125" YCoordinate="445.5"/>
                                        </ConnectorGraphicsInfo>
                                </ConnectorGraphicsInfos>
                        </Transition>
                        <Transition Id="73" Name="" From="62" To="63">
                                <Condition Type="CONDITION">orderType == "PO"</Condition>
                                <ConnectorGraphicsInfos>
                                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                                                <Coordinates XCoordinate="224.5" YCoordinate="393.11090087890625"/>
                                                <Coordinates XCoordinate="324.0" YCoordinate="393.0"/>
                                                <Coordinates XCoordinate="324.0" YCoordinate="365.0"/>
                                                <Coordinates XCoordinate="388.5" YCoordinate="365.13006591796875"/>
                                        </ConnectorGraphicsInfo>
                                </ConnectorGraphicsInfos>
                        </Transition>
                        <Transition Id="74" Name="" From="62" To="68">
                                <Condition Type="CONDITION">orderType == "Credit"</Condition>
                                <ConnectorGraphicsInfos>
                                        <ConnectorGraphicsInfo Page="1" BorderColor="-16777216" FillColor="-1114150">
                                                <Coordinates XCoordinate="224.5" YCoordinate="393.11090087890625"/>
                                                <Coordinates XCoordinate="324.0" YCoordinate="393.0"/>
                                                <Coordinates XCoordinate="324.0" YCoordinate="417.0"/>
                                                <Coordinates XCoordinate="387.5" YCoordinate="417.8579406738281"/>
                                        </ConnectorGraphicsInfo>
                                </ConnectorGraphicsInfos>
                        </Transition>
                </Transitions>
        </WorkflowProcess>
    </WorkflowProcesses>
    <ExtendedAttributes>
        <ExtendedAttribute Name="System" Value="CapeVisions"/>
        <ExtendedAttribute Name="Creator" Value="Sketchpad Prototype 2"/>
    </ExtendedAttributes>
</Package>
```

8.2. Extending XPDL Schema

As described in section 7.1.2.2 Namespace Qualified Extensions, the XPDL schema allow for extensions. The following example, show how a vendor can add elements and attributes. For the example, vendor Xyz will enhance the Participant element by adding an attribute called Organization, and an element called Address. The enhancement will be done in a way that vendor xyz, and any other party, can valiadate a XPDL file containing vendor Xyz extensions.

8.2.1. Xyz Schema

Xyz vendor creates the following schema file to extend the XPDL schema. Note that XyzExtensions element is only needed to disambiguate the schema in the case Address is optional (like in this particular case); otherwise, no separation element is required.

```
<!-- Must import the XPDL schema  -->
<xsd:import namespace="http://www.wfmc.org/2004/XPDL2.0alpha"
   schemaLocation="file:bpmnxpdl_23.xsd"/>

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xyz="http://www.xyz.com/2005/XYZ.XPDL2"
    xmlns:xpdl="http://www.wfmc.org/2004/XPDL2.0alpha"
    xmlns:deprecated="http://www.wfmc.org/2002/XPDL1.0"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.xyz.com/2005/XYZ.XPDL2"
    elementFormDefault="qualified" attributeFormDefault="unqualified">

<!-- Start Xyz's XPDL extensions -->
<!-- (only include XPDL elements that will be extended) -->
<xsd:element name="Participant">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="xpdl:ParticipantType"/>
         <xsd:element ref="xpdl:Description" minOccurs="0"/>
         <xsd:element ref="xpdl:ExternalReference" minOccurs="0"/>
         <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>

         <!-- Add Xyz new Participant Elements -->
         <xsd:element name="XyzExtensions"/>
         <xsd:element ref="xyz:Address" minOccurs="0"/>
         <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
      <xsd:attribute name="Name" type="xsd:string" use="optional"/>

      <!-- Add Xyz new Participant Attributes -->
      <xsd:attribute name="Organization" type="xsd:string"  form="qualified"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
</xsd:element>

<!-- End Xyz's XPDL extensions -->

 <!-- ========================= -->
<!-- Start Xyz Definitions -->
<xsd:element name="Address">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Street" type="xsd:string"/>
         <xsd:element name="City" type="xsd:string"/>
         <xsd:element name="State" type="xsd:string"/>
         <xsd:element name="Zip" type="xsd:int"/>
      </xsd:sequence>
   </xsd:complexType>
```

```
</xsd:element>

<!-- End Xyz Definitions -->

</xsd:schema>
```

8.2.2. Xyz Test XPDL Document

The following XPDL 2.0 compliant test file validates against the XPDL 2.0 schema, using the Xyz schema as an external schema. Note that vendor Xyz includes a vendor extension section to provide the location of the Xyz schema, and a document describing the extensions. The vendor extension information can be useful for other vendors that may want to interchange XPDL files with Xyz.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://www.wfmc.org/2004/XPDL2.0alpha"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xyz="http://www.xyz.com/2005/XYZ.XPDL2"
   xmlns:xpdl="http://www.wfmc.org/2004/XPDL2.0alpha"
   xsi:schemaLocation="http://www.wfmc.org/2004/XPDL2.0alpha
       file:bpmnxpdl_23.xsd" Id="testFile">

<PackageHeader>
   <XPDLVersion>2.0</XPDLVersion>
   <Vendor>Xyz</Vendor>
   <Created>9/5/2005</Created>
   <VendorExtensions>
      <VendorExtension ToolId="XyzTool"
         schemaLocation="http://www.xzy.sample/xyz.xsd"
         extensionDescription="http://www.xyz.sample/Description.html"/>
   </VendorExtensions>
</PackageHeader>

<Participants>
   <Participant Id="testParticipant" xyz:Organization="Test Organization">
      <ParticipantType Type="HUMAN">
         <xyz:XyZExtensions/>
         <xyz:Address>
            <xyz:Street>123 My House Street</xyz:Street>
            <xyz:City>Big City</xyz:City>
            <xyz:State>California</xyz:State>
            <xyz:Zip>92626</xyz:Zip>
         </xyz:Address>
      </ParticipantType>
   </Participant>
</Participants>
</Package>
```

9. XPDL Schema

This section presents the full Schema for XPDL.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Tim Stephenson, Wojciech Zurek (TIBCO Software Inc.) -->
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Marin, Mike (FileNet Corporation) -->
<!-- edited with XML Spy v4.0 U (http://www.xmlspy.com) by Robert M Shapiro (Cape Visions) -->
<xsd:schema targetNamespace="http://www.wfmc.org/2004/XPDL2.0alpha" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:deprecated="http://www.wfmc.org/2002/XPDL1.0" xmlns:xpdl="http://www.wfmc.org/2004/XPDL2.0alpha" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xsd:import namespace="http://www.wfmc.org/2002/XPDL1.0" schemaLocation="http://www.wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd"/>
	<xsd:element name="Activities">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Activity" 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:element>
	<xsd:element name="Activity">
		<xsd:annotation>
			<xsd:documentation>BPMN extension</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Description" minOccurs="0"/>
				<xsd:element ref="xpdl:Limit" minOccurs="0"/>
				<xsd:choice minOccurs="0">
					<xsd:element ref="xpdl:Route"/>
					<xsd:element ref="xpdl:Implementation"/>
					<xsd:choice minOccurs="0">
						<xsd:element ref="deprecated:BlockActivity"/>
						<xsd:element ref="xpdl:BlockActivity"/>
					</xsd:choice>
					<xsd:element ref="xpdl:Event"/>
				</xsd:choice>
				<xsd:element ref="xpdl:Transaction" minOccurs="0"/>
				<xsd:element ref="xpdl:Performer" minOccurs="0"/>
				<xsd:element ref="xpdl:Performers" minOccurs="0"/>
				<xsd:element ref="deprecated:StartMode" minOccurs="0"/>
				<xsd:element ref="deprecated:FinishMode" minOccurs="0"/>
				<xsd:element ref="xpdl:Priority" minOccurs="0"/>
				<xsd:choice minOccurs="0">
					<xsd:element ref="deprecated:Deadline" minOccurs="0" maxOccurs="unbounded"/>
					<xsd:element ref="xpdl:Deadline" minOccurs="0" maxOccurs="unbounded"/>
				</xsd:choice>
				<xsd:element ref="xpdl:SimulationInformation" minOccurs="0"/>
				<xsd:element ref="xpdl:Icon" minOccurs="0"/>
				<xsd:element ref="xpdl:Documentation" minOccurs="0"/>
				<xsd:element ref="xpdl:TransitionRestrictions" minOccurs="0"/>
				<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
				<xsd:element ref="xpdl:DataFields" minOccurs="0"/>
				<xsd:element ref="xpdl:InputSets" minOccurs="0"/>
				<xsd:element ref="xpdl:OutputSets" minOccurs="0"/>
				<xsd:element ref="xpdl:IORules" minOccurs="0"/>
				<xsd:element ref="xpdl:Loop" minOccurs="0"/>
				<xsd:element ref="xpdl:Assignments" minOccurs="0"/>
				<xsd:element ref="xpdl:Object" minOccurs="0"/>
				<xsd:element ref="xpdl:NodeGraphicsInfos" minOccurs="0"/>
				<xsd:choice minOccurs="0">
					<xsd:sequence>
						<xsd:element name="Extensions"/>
						<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
					</xsd:sequence>
				</xsd:choice>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:attribute name="StartActivity" type="xsd:boolean" use="optional">
				<xsd:annotation>
					<xsd:documentation> Designates the first activity to be executed when the process is instantiated. Used when there is no other way to
determine this Conflicts with BPMN StartEvent and no process definition should use both.</xsd:documentation>
				</xsd:annotation>
			</xsd:attribute>
			<xsd:attribute name="Status" use="optional" default="None">
				<xsd:annotation>
					<xsd:documentation> BPMN: Status values are assigned during execution. Status can be treated as a property and used in expressions local
to an Activity. It is unclear that status belongs in the XPDL document.</xsd:documentation>
				</xsd:annotation>
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="None"/>
						<xsd:enumeration value="Ready"/>
						<xsd:enumeration value="Active"/>
```

```
                        <xsd:enumeration value="Cancelled"/>
                        <xsd:enumeration value="Aborting"/>
                        <xsd:enumeration value="Aborted"/>
                        <xsd:enumeration value="Completing"/>
                        <xsd:enumeration value="Completed"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="StartMode">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:NMTOKEN">
                        <xsd:enumeration value="Automatic"/>
                        <xsd:enumeration value="Manual"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="FinishMode">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:NMTOKEN">
                        <xsd:enumeration value="Automatic"/>
                        <xsd:enumeration value="Manual"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="StartQuantity" type="xsd:integer" use="optional" default="1"/>
            <xsd:attribute name="IsATransaction" type="xsd:boolean" use="optional" default="false"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="ActivitySet">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xpdl:Activities" minOccurs="0"/>
                <xsd:element ref="xpdl:Transitions" minOccurs="0"/>
                <xsd:element ref="xpdl:Object" minOccurs="0"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="Name" type="xsd:string" use="optional">
                <xsd:annotation>
                    <xsd:documentation source="added to XPDL 2.0"/>
                </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="AdHoc" type="xsd:boolean" use="optional" default="false">
                <xsd:annotation>
                    <xsd:documentation>BPMN: for Embedded subprocess</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="AdHocOrdering" use="optional" default="Parallel">
                <xsd:annotation>
                    <xsd:documentation>BPMN: for Embedded subprocess</xsd:documentation>
                </xsd:annotation>
                <xsd:simpleType>
                    <xsd:restriction base="xsd:NMTOKEN">
                        <xsd:enumeration value="Sequential"/>
                        <xsd:enumeration value="Parallel"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="AdHocCompletionCondition" type="xsd:string" use="optional">
                <xsd:annotation>
                    <xsd:documentation>BPMN: for Embedded subprocess</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="DefaultStartActivityId" type="xsd:NMTOKEN" use="optional"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="ActivitySets">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xpdl:ActivitySet" 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:element>
    <xsd:element name="ActualParameters">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="ActualParameter" type="xpdl:ExpressionType" 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:element>
    <xsd:element name="Application">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xpdl:Description" minOccurs="0"/>
                <xsd:element name="Type" type="xpdl:ApplicationType" minOccurs="0"/>
                <xsd:choice>
```

```
					<xsd:element ref="xpdl:FormalParameters"/>
					<xsd:element ref="xpdl:ExternalReference" minOccurs="0"/>
				</xsd:choice>
				<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:complexType name="ApplicationType">
		<xsd:choice>
			<xsd:element name="Ejb">
				<xsd:annotation>
					<xsd:documentation> Call EJB component -- There can be max one formal parameter that is OUT, if it exists it has to be the last formal
parameter. no INOUT formal parameters</xsd:documentation>
				</xsd:annotation>
				<xsd:complexType>
					<xsd:sequence>
						<xsd:element name="JndiName">
							<xsd:complexType>
								<xsd:simpleContent>
									<xsd:extension base="xsd:string">
										<xsd:anyAttribute namespace="##other" processContents="lax"/>
									</xsd:extension>
								</xsd:simpleContent>
							</xsd:complexType>
						</xsd:element>
						<xsd:element name="HomeClass">
							<xsd:complexType>
								<xsd:simpleContent>
									<xsd:extension base="xsd:string">
										<xsd:anyAttribute namespace="##other" processContents="lax"/>
									</xsd:extension>
								</xsd:simpleContent>
							</xsd:complexType>
						</xsd:element>
						<xsd:element name="Method">
							<xsd:complexType>
								<xsd:simpleContent>
									<xsd:extension base="xsd:string">
										<xsd:anyAttribute namespace="##other" processContents="lax"/>
									</xsd:extension>
								</xsd:simpleContent>
							</xsd:complexType>
						</xsd:element>
						<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
					</xsd:sequence>
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:complexType>
			</xsd:element>
			<xsd:element name="Pojo">
				<xsd:annotation>
					<xsd:documentation> Call method on Java class -- There can be max one formal parameter that is OUT, if it exists it has to be the last formal
parameter. no INOUT formal parameters</xsd:documentation>
				</xsd:annotation>
				<xsd:complexType>
					<xsd:sequence>
						<xsd:element name="Class">
							<xsd:complexType>
								<xsd:simpleContent>
									<xsd:extension base="xsd:string">
										<xsd:anyAttribute namespace="##other" processContents="lax"/>
									</xsd:extension>
								</xsd:simpleContent>
							</xsd:complexType>
						</xsd:element>
						<xsd:element name="Method">
							<xsd:complexType>
								<xsd:simpleContent>
									<xsd:extension base="xsd:string">
										<xsd:anyAttribute namespace="##other" processContents="lax"/>
									</xsd:extension>
								</xsd:simpleContent>
							</xsd:complexType>
						</xsd:element>
						<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
					</xsd:sequence>
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:complexType>
			</xsd:element>
			<xsd:element name="Xslt">
				<xsd:annotation>
					<xsd:documentation> Execute Tranformation -- Formal Parameters restrictions: one IN and one OUT formal parameters or only one INOUT
formal parameter</xsd:documentation>
				</xsd:annotation>
				<xsd:complexType>
					<xsd:sequence>
						<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
					</xsd:sequence>
					<xsd:attribute name="location" type="xsd:anyURI"/>
```

```
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Script">
                <xsd:annotation>
                    <xsd:documentation> Execute Script -- No additional restrictions for formal parameters. The suggestion: every Formal Parameter should be
registered in the script scope as a global variable</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Expression" type="xpdl:ExpressionType" minOccurs="0"/>
                        <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="WebService">
                <xsd:annotation>
                    <xsd:documentation> For WSDL 1.2 -- Invoke WebService, all IN Fprmal Parameters will be mapped to input message, all OUT Formal
Parameters will be maped from output message</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element ref="xpdl:WebServiceOperation"/>
                        <xsd:element ref="xpdl:WebServiceFaultCatch" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:attribute name="InputMsgName" type="xsd:string" use="required">
                        <xsd:annotation>
                            <xsd:documentation>The name of inputMessage as defined in the WSDL which will help in uniquely identifying the operation to be
invoked</xsd:documentation>
                        </xsd:annotation>
                    </xsd:attribute>
                    <xsd:attribute name="OutputMsgName" type="xsd:string" use="optional">
                        <xsd:annotation>
                            <xsd:documentation>The name of inputMessage as defined in the WSDL which will help in uniquely identifying the operation to be
invoked</xsd:documentation>
                        </xsd:annotation>
                    </xsd:attribute>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="BusinessRule">
                <xsd:annotation>
                    <xsd:documentation>Invoke business rule</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="RuleName">
                            <xsd:complexType>
                                <xsd:simpleContent>
                                    <xsd:extension base="xsd:string">
                                        <xsd:anyAttribute namespace="##other" processContents="lax"/>
                                    </xsd:extension>
                                </xsd:simpleContent>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="Location">
                            <xsd:complexType>
                                <xsd:simpleContent>
                                    <xsd:extension base="xsd:anyURI">
                                        <xsd:anyAttribute namespace="##other" processContents="lax"/>
                                    </xsd:extension>
                                </xsd:simpleContent>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Form">
                <xsd:annotation>
                    <xsd:documentation>Placeholder for all form related additional information.</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="FormLayout" minOccurs="0">
                            <xsd:complexType>
                                <xsd:complexContent>
                                    <xsd:extension base="xsd:anyType">
                                        <xsd:anyAttribute namespace="##other" processContents="lax"/>
                                    </xsd:extension>
                                </xsd:complexContent>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:complexType>
            </xsd:element>
        </xsd:choice>
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:complexType>
    <xsd:element name="Applications">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xpdl:Application" 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:element>
    <xsd:element name="ArrayType">
        <xsd:complexType>
            <xsd:group ref="xpdl:DataTypes"/>
            <xsd:attribute name="LowerIndex" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="UpperIndex" type="xsd:NMTOKEN" use="required"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Artifact">
        <xsd:annotation>
            <xsd:documentation>BPMN: Not further defined here.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence minOccurs="0">
                <xsd:element ref="xpdl:Object" minOccurs="0"/>
                <xsd:element ref="xpdl:DataObject" minOccurs="0"/>
                <xsd:element ref="xpdl:NodeGraphicsInfos" minOccurs="0"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
            <xsd:attribute name="Name" type="xsd:string" use="optional"/>
            <xsd:attribute name="ArtifactType" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:NMTOKEN">
                        <xsd:enumeration value="DataObject"/>
                        <xsd:enumeration value="Group"/>
                        <xsd:enumeration value="Annotation"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="TextAnnotation" type="xsd:string" use="optional"/>
            <xsd:attribute name="Group" type="xsd:string" use="optional"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Artifacts">
        <xsd:annotation>
            <xsd:documentation>BPMN</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence maxOccurs="unbounded">
                <xsd:element ref="xpdl:Artifact"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Assignment">
        <xsd:annotation>
            <xsd:documentation>BPMN and XPDL</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Target" type="xpdl:ExpressionType">
                    <xsd:annotation>
                        <xsd:documentation> lvalue expression of the assignment, in XPDL may be the name of a DataField, in BPMN name of a Property, in
XPATH a reference</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="Expression" type="xpdl:ExpressionType">
                    <xsd:annotation>
                        <xsd:documentation>rvalue expression of the assignment</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="AssignTime" use="optional" default="Start">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:NMTOKEN">
                        <xsd:enumeration value="Start"/>
                        <xsd:enumeration value="End"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Assignments">
        <xsd:annotation>
            <xsd:documentation>BPMN and XPDL</xsd:documentation>
        </xsd:annotation>
```

```
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Assignment" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Association">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence minOccurs="0">
				<xsd:element ref="xpdl:Object"/>
				<xsd:element ref="xpdl:ConnectorGraphicsInfos" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Source" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Target" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:attribute name="AssociationDirection" use="optional" default="None">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="None"/>
						<xsd:enumeration value="To"/>
						<xsd:enumeration value="From"/>
						<xsd:enumeration value="Both"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Associations">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence maxOccurs="unbounded">
				<xsd:element ref="xpdl:Association"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Author">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="BasicType">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Length" minOccurs="0"/>
				<xsd:element ref="xpdl:Precision" minOccurs="0"/>
				<xsd:element ref="xpdl:Scale" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Type" use="required">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="STRING"/>
						<xsd:enumeration value="FLOAT"/>
						<xsd:enumeration value="INTEGER"/>
						<xsd:enumeration value="REFERENCE"/>
						<xsd:enumeration value="DATETIME"/>
						<xsd:enumeration value="BOOLEAN"/>
						<xsd:enumeration value="PERFORMER"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="BlockActivity">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="ActivitySetId" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="StartActivityId" type="xsd:NMTOKEN" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Category">
		<xsd:annotation>
```

```
			<xsd:documentation> BPMN (and XPDL??Allows arbitrary grouping of various types of elements, for reporting.)</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Categories">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Category" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Codepage">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Condition">
		<xsd:complexType mixed="true">
			<xsd:choice minOccurs="0">
				<xsd:element ref="deprecated:Xpression" minOccurs="0"/>
				<xsd:element name="Expression" type="xpdl:ExpressionType" minOccurs="0"/>
			</xsd:choice>
			<xsd:attribute name="Type">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="CONDITION"/>
						<xsd:enumeration value="OTHERWISE"/>
						<xsd:enumeration value="EXCEPTION"/>
						<xsd:enumeration value="DEFAULTEXCEPTION"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="ConformanceClass">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="GraphConformance" use="optional" default="NON_BLOCKED">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="FULL_BLOCKED"/>
						<xsd:enumeration value="LOOP_BLOCKED"/>
						<xsd:enumeration value="NON_BLOCKED"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="ConnectorGraphicsInfo">
		<xsd:annotation>
			<xsd:documentation>BPMN and XPDL</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence minOccurs="0">
				<xsd:element ref="xpdl:Coordinates" minOccurs="0" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="ToolId" type="xsd:NMTOKEN" use="optional"/>
			<xsd:attribute name="IsVisible" type="xsd:boolean" use="optional" default="true"/>
			<xsd:attribute name="Page" type="xsd:NMTOKEN" use="optional"/>
			<xsd:attribute name="Style" type="xsd:string" use="optional"/>
			<xsd:attribute name="BorderColor" type="xsd:string" use="optional"/>
			<xsd:attribute name="FillColor" type="xsd:string" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="ConnectorGraphicsInfos">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:ConnectorGraphicsInfo" 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:element>
```

```
<xsd:element name="Coordinates">
	<xsd:annotation>
		<xsd:documentation>BPMN and XPDL</xsd:documentation>
	</xsd:annotation>
	<xsd:complexType>
		<xsd:sequence>
			<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:attribute name="XCoordinate" type="xsd:double" use="optional"/>
		<xsd:attribute name="YCoordinate" type="xsd:double" use="optional"/>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
</xsd:element>
<xsd:element name="Cost">
	<xsd:complexType>
		<xsd:simpleContent>
			<xsd:extension base="xsd:string">
				<xsd:anyAttribute namespace="##other" processContents="lax"/>
			</xsd:extension>
		</xsd:simpleContent>
	</xsd:complexType>
</xsd:element>
<xsd:element name="CostUnit">
	<xsd:complexType>
		<xsd:simpleContent>
			<xsd:extension base="xsd:string">
				<xsd:anyAttribute namespace="##other" processContents="lax"/>
			</xsd:extension>
		</xsd:simpleContent>
	</xsd:complexType>
</xsd:element>
<xsd:element name="Countrykey">
	<xsd:complexType>
		<xsd:simpleContent>
			<xsd:extension base="xsd:string">
				<xsd:anyAttribute namespace="##other" processContents="lax"/>
			</xsd:extension>
		</xsd:simpleContent>
	</xsd:complexType>
</xsd:element>
<xsd:element name="Created">
	<xsd:complexType>
		<xsd:simpleContent>
			<xsd:extension base="xsd:string">
				<xsd:anyAttribute namespace="##other" processContents="lax"/>
			</xsd:extension>
		</xsd:simpleContent>
	</xsd:complexType>
</xsd:element>
<xsd:element name="DataField">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element ref="xpdl:DataType"/>
			<xsd:element name="InitialValue" type="xpdl:ExpressionType" minOccurs="0"/>
			<xsd:element ref="xpdl:Length" minOccurs="0"/>
			<xsd:element ref="xpdl:Description" minOccurs="0"/>
			<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
			<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
		<xsd:attribute name="Name" type="xsd:string" use="optional"/>
		<xsd:attribute name="IsArray" use="optional" default="FALSE">
			<xsd:simpleType>
				<xsd:restriction base="xsd:NMTOKEN">
					<xsd:enumeration value="TRUE"/>
					<xsd:enumeration value="FALSE"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
		<xsd:attribute name="Correlation" type="xsd:boolean" use="optional" default="false">
			<xsd:annotation>
				<xsd:documentation>Used in BPMN to support mapping to BPEL</xsd:documentation>
			</xsd:annotation>
		</xsd:attribute>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
</xsd:element>
<xsd:element name="DataFields">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element ref="xpdl:DataField" 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:element>
<xsd:element name="DataMappings">
	<xsd:annotation>
		<xsd:documentation>XPDL and BPMN:Maps fields or properties between calling and called processes or subprocesses</xsd:documentation>
	</xsd:annotation>
	<xsd:complexType>
		<xsd:sequence>
```

```
				<xsd:element ref="xpdl:DataMapping" 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:element>
	<xsd:element name="DataMapping">
		<xsd:annotation>
			<xsd:documentation>XPDL and BPMN:Maps fields between calling and called processes or subprocesses</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element name="Actual" type="xpdl:ExpressionType"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Formal" type="xsd:string" use="required"/>
			<xsd:attribute name="Direction" default="IN">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="IN"/>
						<xsd:enumeration value="OUT"/>
						<xsd:enumeration value="INOUT"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="DataObject">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:DataFields" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:attribute name="State" type="xsd:string" use="optional"/>
			<xsd:attribute name="RequiredForStart" type="xsd:boolean" use="required"/>
			<xsd:attribute name="ProducedAtCompletion" type="xsd:boolean" use="required"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="DataType">
		<xsd:complexType>
			<xsd:group ref="xpdl:DataTypes"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:group name="DataTypes">
		<xsd:choice>
			<xsd:element ref="xpdl:BasicType"/>
			<xsd:element ref="xpdl:DeclaredType"/>
			<xsd:element ref="xpdl:SchemaType"/>
			<xsd:element ref="xpdl:ExternalReference"/>
			<xsd:element ref="xpdl:RecordType"/>
			<xsd:element ref="xpdl:UnionType"/>
			<xsd:element ref="xpdl:EnumerationType"/>
			<xsd:element ref="xpdl:ArrayType"/>
			<xsd:element ref="xpdl:ListType"/>
		</xsd:choice>
	</xsd:group>
	<xsd:element name="Deadline">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element name="DeadlineDuration" type="xpdl:ExpressionType" minOccurs="0"/>
				<xsd:element name="ExceptionName" minOccurs="0">
					<xsd:annotation>
						<xsd:documentation>This name should match that specified in Transition/Condition/Expression</xsd:documentation>
					</xsd:annotation>
					<xsd:complexType>
						<xsd:simpleContent>
							<xsd:extension base="xsd:string">
								<xsd:anyAttribute namespace="##other" processContents="lax"/>
							</xsd:extension>
						</xsd:simpleContent>
					</xsd:complexType>
				</xsd:element>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Execution">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="ASYNCHR"/>
						<xsd:enumeration value="SYNCHR"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="DeclaredType">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:IDREF" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Description">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Documentation">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Duration">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="EndEvent">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:choice>
				<xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
				<xsd:element ref="xpdl:ResultError" minOccurs="0"/>
				<xsd:element ref="xpdl:ResultCompensation" minOccurs="0"/>
				<xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
				<xsd:element ref="xpdl:ResultMultiple" minOccurs="0"/>
			</xsd:choice>
			<xsd:attribute name="Result" use="optional" default="None">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="None"/>
						<xsd:enumeration value="Message"/>
						<xsd:enumeration value="Error"/>
						<xsd:enumeration value="Cancel"/>
						<xsd:enumeration value="Compensation"/>
						<xsd:enumeration value="Link"/>
						<xsd:enumeration value="Terminate"/>
						<xsd:enumeration value="Multiple"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:attribute name="Implementation" use="optional" default="WebService">
				<xsd:annotation>
					<xsd:documentation> Required if the Trigger or Result is Message</xsd:documentation>
				</xsd:annotation>
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="WebService"/>
						<xsd:enumeration value="Other"/>
						<xsd:enumeration value="Unspecified"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="EndPoint">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:ExternalReference"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="EndPointType" use="optional" default="WSDL">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="WSDL"/>
						<xsd:enumeration value="Service"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="EnumerationType">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:EnumerationValue" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="EnumerationValue">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Name" type="xsd:NMTOKEN" use="required"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Event">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:choice>
				<xsd:element ref="xpdl:StartEvent" minOccurs="0"/>
				<xsd:element ref="xpdl:IntermediateEvent" minOccurs="0"/>
				<xsd:element ref="xpdl:EndEvent" minOccurs="0"/>
			</xsd:choice>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:complexType name="ExpressionType" mixed="true">
		<xsd:choice minOccurs="0" maxOccurs="unbounded">
			<xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:choice>
		<xsd:attribute name="ScriptGrammar" type="xsd:anyURI" use="optional"/>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
	<xsd:element name="ExtendedAttribute">
		<xsd:complexType mixed="true">
			<xsd:choice minOccurs="0" maxOccurs="unbounded">
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:choice>
			<xsd:attribute name="Name" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Value" type="xsd:string"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="ExtendedAttributes">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:ExtendedAttribute" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="ExternalPackage">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="href" type="xsd:string"/>
			<xsd:attribute name="Id" type="xsd:NMTOKEN"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="ExternalPackages">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:ExternalPackage" 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:element>
	<xsd:element name="ExternalReference">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="xref" type="xsd:NMTOKEN" use="optional"/>
			<xsd:attribute name="location" type="xsd:anyURI" use="required"/>
			<xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="FormalParameter">
		<xsd:complexType>
```

```
			<xsd:sequence>
				<xsd:element ref="xpdl:DataType"/>
				<xsd:element ref="xpdl:Description" minOccurs="0"/>
				<xsd:element ref="xpdl:Length" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Mode" default="IN">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="IN"/>
						<xsd:enumeration value="OUT"/>
						<xsd:enumeration value="INOUT"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:attribute name="IsArray" type="xsd:boolean" use="optional" default="false"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="FormalParameters">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:choice minOccurs="0">
					<xsd:element ref="deprecated:FormalParameter" minOccurs="0" maxOccurs="unbounded"/>
					<xsd:element ref="xpdl:FormalParameter" minOccurs="0" maxOccurs="unbounded"/>
				</xsd:choice>
				<xsd:choice minOccurs="0">
					<xsd:sequence>
						<xsd:element name="Extensions"/>
						<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
					</xsd:sequence>
				</xsd:choice>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Icon">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:attribute name="XCOORD" type="xsd:integer" use="optional"/>
					<xsd:attribute name="YCOORD" type="xsd:integer" use="optional"/>
					<xsd:attribute name="WIDTH" type="xsd:integer" use="optional"/>
					<xsd:attribute name="HEIGHT" type="xsd:integer" use="optional"/>
					<xsd:attribute name="SHAPE" use="optional" default="RoundRectangle">
						<xsd:simpleType>
							<xsd:restriction base="xsd:NMTOKEN">
								<xsd:enumeration value="RoundRectangle"/>
								<xsd:enumeration value="Rectangle"/>
								<xsd:enumeration value="Ellipse"/>
								<xsd:enumeration value="Diamond"/>
								<xsd:enumeration value="Ellipse"/>
								<xsd:enumeration value="UpTriangle"/>
								<xsd:enumeration value="DownTriangle"/>
							</xsd:restriction>
						</xsd:simpleType>
					</xsd:attribute>
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Implementation">
		<xsd:complexType>
			<xsd:choice minOccurs="0">
				<xsd:element ref="xpdl:No" minOccurs="0"/>
				<xsd:element ref="deprecated:Tool" minOccurs="0" maxOccurs="unbounded"/>
				<xsd:element ref="xpdl:Task" minOccurs="0"/>
				<xsd:element ref="xpdl:SubFlow" minOccurs="0"/>
				<xsd:element ref="xpdl:Reference" minOccurs="0"/>
			</xsd:choice>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Input">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="ArtifactId" type="xsd:NMTOKEN" use="required"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="InputSet">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Input" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="InputSets">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:InputSet" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="IntermediateEvent">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:choice minOccurs="0">
				<xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
				<xsd:element ref="xpdl:TriggerTimer" minOccurs="0"/>
				<xsd:element ref="xpdl:ResultError" minOccurs="0"/>
				<xsd:element ref="xpdl:ResultCompensation" minOccurs="0"/>
				<xsd:element ref="xpdl:TriggerRule" minOccurs="0"/>
				<xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
				<xsd:element ref="xpdl:TriggerIntermediateMultiple" minOccurs="0"/>
			</xsd:choice>
			<xsd:attribute name="Trigger" use="required">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="None"/>
						<xsd:enumeration value="Message"/>
						<xsd:enumeration value="Timer"/>
						<xsd:enumeration value="Error"/>
						<xsd:enumeration value="Cancel"/>
						<xsd:enumeration value="Rule"/>
						<xsd:enumeration value="Link"/>
						<xsd:enumeration value="Compensation"/>
						<xsd:enumeration value="Multiple"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:attribute name="Implementation" use="optional" default="WebService">
				<xsd:annotation>
					<xsd:documentation>Required if the Trigger is Message</xsd:documentation>
				</xsd:annotation>
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="WebService"/>
						<xsd:enumeration value="Other"/>
						<xsd:enumeration value="Unspecified"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:attribute name="Target" type="xsd:NMTOKEN" use="optional">
				<xsd:annotation>
					<xsd:documentation> A Target MAY be included for the Intermediate Event. The Target MUST be an activity (Sub-Process or Task). This
means that the Intermediate Event is attached to the boundary of the activity and is used to signify an exception or compensation for that activity.</xsd:documentation>
				</xsd:annotation>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="IORules">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element name="Expression" type="xpdl:ExpressionType" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Join">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Type">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="AND"/>
						<xsd:enumeration value="XOR"/>
```

```
                                <xsd:enumeration value="XOREVENT"/>
                                <xsd:enumeration value="OR"/>
                                <xsd:enumeration value="COMPLEX"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:attribute>
                    <xsd:attribute name="IncomingCondtion" type="xsd:string"/>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Lane">
                <xsd:annotation>
                    <xsd:documentation>BPMN</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence minOccurs="0">
                        <xsd:element ref="xpdl:Object" minOccurs="0"/>
                        <xsd:element ref="xpdl:NodeGraphicsInfos" minOccurs="0"/>
                        <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
                    <xsd:attribute name="Name" type="xsd:string" use="optional"/>
                    <xsd:attribute name="ParentLane" type="xsd:NMTOKEN" use="optional"/>
                    <xsd:attribute name="ParentPool" type="xsd:NMTOKEN" use="optional"/>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Lanes">
                <xsd:annotation>
                    <xsd:documentation>BPMN</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element ref="xpdl:Lane" 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:element>
            <xsd:element name="Length">
                <xsd:complexType>
                    <xsd:simpleContent>
                        <xsd:extension base="xsd:string">
                            <xsd:anyAttribute namespace="##other" processContents="lax"/>
                        </xsd:extension>
                    </xsd:simpleContent>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Limit">
                <xsd:complexType>
                    <xsd:simpleContent>
                        <xsd:extension base="xsd:string">
                            <xsd:anyAttribute namespace="##other" processContents="lax"/>
                        </xsd:extension>
                    </xsd:simpleContent>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="ListType">
                <xsd:complexType>
                    <xsd:group ref="xpdl:DataTypes"/>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Loop">
                <xsd:annotation>
                    <xsd:documentation>BPMN (and possibly XPDL)</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:choice>
                        <xsd:element ref="xpdl:LoopStandard"/>
                        <xsd:element ref="xpdl:LoopMultiInstance"/>
                    </xsd:choice>
                    <xsd:attribute name="LoopType" use="required">
                        <xsd:simpleType>
                            <xsd:restriction base="xsd:NMTOKEN">
                                <xsd:enumeration value="Standard"/>
                                <xsd:enumeration value="MultiInstance"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:attribute>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="LoopMultiInstance">
                <xsd:annotation>
                    <xsd:documentation>BPMN</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:attribute name="MI_Condition" type="xsd:string" use="required"/>
                    <xsd:attribute name="LoopCounter" type="xsd:integer">
                         <xsd:annotation>
                              <xsd:documentation> This is updated at run time to count the number of executions of the loop and is available as a property to be used in
expressions. Does this belong in the XPDL?</xsd:documentation>
                         </xsd:annotation>
                    </xsd:attribute>
                    <xsd:attribute name="MI_Ordering" use="required">
                         <xsd:simpleType>
                              <xsd:restriction base="xsd:NMTOKEN">
                                   <xsd:enumeration value="Sequential"/>
                                   <xsd:enumeration value="Parallel"/>
                              </xsd:restriction>
                         </xsd:simpleType>
                    </xsd:attribute>
                    <xsd:attribute name="MI_FlowCondition" use="optional" default="All">
                         <xsd:simpleType>
                              <xsd:restriction base="xsd:NMTOKEN">
                                   <xsd:enumeration value="None"/>
                                   <xsd:enumeration value="One"/>
                                   <xsd:enumeration value="All"/>
                                   <xsd:enumeration value="Complex"/>
                              </xsd:restriction>
                         </xsd:simpleType>
                    </xsd:attribute>
                    <xsd:attribute name="ComplexMI_FlowCondition" type="xsd:string" use="optional"/>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:complexType>
          </xsd:element>
          <xsd:element name="LoopStandard">
               <xsd:annotation>
                    <xsd:documentation>BPMN</xsd:documentation>
               </xsd:annotation>
               <xsd:complexType>
                    <xsd:sequence>
                         <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                    <xsd:attribute name="LoopCondition" type="xsd:string" use="required"/>
                    <xsd:attribute name="LoopCounter" type="xsd:integer">
                         <xsd:annotation>
                              <xsd:documentation> This is updated at run time to count the number of executions of the loop and is available as a property to be used in
expressions. Does this belong in the XPDL?</xsd:documentation>
                         </xsd:annotation>
                    </xsd:attribute>
                    <xsd:attribute name="LoopMaximum" type="xsd:integer" use="optional"/>
                    <xsd:attribute name="TestTime" use="required">
                         <xsd:simpleType>
                              <xsd:restriction base="xsd:NMTOKEN">
                                   <xsd:enumeration value="Before"/>
                                   <xsd:enumeration value="After"/>
                              </xsd:restriction>
                         </xsd:simpleType>
                    </xsd:attribute>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:complexType>
          </xsd:element>
          <xsd:element name="Member">
               <xsd:complexType>
                    <xsd:group ref="xpdl:DataTypes"/>
                    <xsd:anyAttribute namespace="##other" processContents="lax"/>
               </xsd:complexType>
          </xsd:element>
          <xsd:complexType name="MessageType">
               <xsd:annotation>
                    <xsd:documentation>Formal Parameters defined by WSDL. Must constraint the parameters to either all in or all out, because Message is in a single
direction</xsd:documentation>
               </xsd:annotation>
               <xsd:sequence minOccurs="0">
                    <xsd:choice minOccurs="0">
                         <xsd:element ref="xpdl:ActualParameters"/>
                         <xsd:element ref="xpdl:DataMappings"/>
                    </xsd:choice>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
               </xsd:sequence>
               <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
               <xsd:attribute name="Name" type="xsd:string" use="optional"/>
               <xsd:attribute name="From" type="xsd:NMTOKEN" use="optional">
                    <xsd:annotation>
                         <xsd:documentation>This must be the name of a Participant</xsd:documentation>
                    </xsd:annotation>
               </xsd:attribute>
               <xsd:attribute name="To" type="xsd:NMTOKEN" use="optional">
                    <xsd:annotation>
                         <xsd:documentation>This must be the name of a participant</xsd:documentation>
                    </xsd:annotation>
               </xsd:attribute>
               <xsd:attribute name="FaultName" type="xsd:NMTOKEN" use="optional"/>
               <xsd:anyAttribute namespace="##other" processContents="lax"/>
          </xsd:complexType>
          <xsd:element name="MessageFlow">
               <xsd:annotation>
                    <xsd:documentation>:BPMN:</xsd:documentation>
               </xsd:annotation>
```

```
		<xsd:complexType>
			<xsd:sequence minOccurs="0">
				<xsd:element name="Message" type="xpdl:MessageType" minOccurs="0"/>
				<xsd:element ref="xpdl:Object" minOccurs="0"/>
				<xsd:element ref="xpdl:ConnectorGraphicsInfos" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:attribute name="Source" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Target" type="xsd:NMTOKEN" use="required"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="MessageFlows">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence minOccurs="0" maxOccurs="unbounded">
				<xsd:element ref="xpdl:MessageFlow"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="No">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="NodeGraphicsInfo">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Coordinates" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="ToolId" type="xsd:string" use="optional"/>
			<xsd:attribute name="IsVisible" type="xsd:boolean" use="optional" default="true"/>
			<xsd:attribute name="Page" type="xsd:NMTOKEN" use="optional"/>
			<xsd:attribute name="LaneId" type="xsd:NMTOKEN" use="optional"/>
			<xsd:attribute name="Height" type="xsd:double" use="optional"/>
			<xsd:attribute name="Width" type="xsd:double" use="optional"/>
			<xsd:attribute name="BorderColor" type="xsd:string" use="optional"/>
			<xsd:attribute name="FillColor" type="xsd:string" use="optional"/>
			<xsd:attribute name="Shape" type="xsd:string" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="NodeGraphicsInfos">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:NodeGraphicsInfo" 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:element>
	<xsd:element name="Object">
		<xsd:annotation>
			<xsd:documentation>BPMN: This is used to identify the Activity in an EndEvent Compensation???Also used to associate categories and ocumentation
with a variety of elements</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence minOccurs="0">
				<xsd:element ref="xpdl:Categories" minOccurs="0"/>
				<xsd:element ref="xpdl:Documentation" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required">
				<xsd:annotation>
					<xsd:documentation>This identifies any Object in the BPMN diagram.</xsd:documentation>
				</xsd:annotation>
			</xsd:attribute>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Output">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="ArtifactId" type="xsd:NMTOKEN" use="required"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="OutputSet">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Output" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="OutputSets">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:OutputSet" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Package" type="xpdl:PackageType"/>
	<xsd:complexType name="PackageType">
		<xsd:sequence>
			<xsd:element ref="xpdl:PackageHeader"/>
			<xsd:element ref="xpdl:RedefinableHeader" minOccurs="0"/>
			<xsd:element ref="xpdl:ConformanceClass" minOccurs="0"/>
			<xsd:element ref="xpdl:Script" minOccurs="0"/>
			<xsd:element ref="xpdl:ExternalPackages" minOccurs="0"/>
			<xsd:element ref="xpdl:TypeDeclarations" minOccurs="0"/>
			<xsd:element ref="xpdl:Participants" minOccurs="0"/>
			<xsd:element ref="xpdl:Applications" minOccurs="0"/>
			<xsd:element ref="xpdl:DataFields" minOccurs="0"/>
			<xsd:element ref="xpdl:PartnerLinkTypes" minOccurs="0"/>
			<xsd:element ref="xpdl:Pools" minOccurs="0"/>
			<xsd:element ref="xpdl:MessageFlows" minOccurs="0"/>
			<xsd:element ref="xpdl:Associations" minOccurs="0"/>
			<xsd:element ref="xpdl:Artifacts" minOccurs="0"/>
			<xsd:element ref="xpdl:WorkflowProcesses" minOccurs="0"/>
			<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
			<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
		<xsd:attribute name="Name" type="xsd:string" use="optional"/>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
	<xsd:element name="PackageHeader">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:XPDLVersion"/>
				<xsd:element ref="xpdl:Vendor"/>
				<xsd:element ref="xpdl:Created"/>
				<xsd:element ref="xpdl:Description" minOccurs="0"/>
				<xsd:element ref="xpdl:Documentation" minOccurs="0"/>
				<xsd:element ref="xpdl:PriorityUnit" minOccurs="0"/>
				<xsd:element ref="xpdl:CostUnit" minOccurs="0"/>
				<xsd:element ref="xpdl:VendorExtensions" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Participant">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:ParticipantType"/>
				<xsd:element ref="xpdl:Description" minOccurs="0"/>
				<xsd:element ref="xpdl:ExternalReference" minOccurs="0"/>
				<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="ParticipantType">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Type" use="required">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="RESOURCE_SET"/>
						<xsd:enumeration value="RESOURCE"/>
						<xsd:enumeration value="ROLE"/>
						<xsd:enumeration value="ORGANIZATIONAL_UNIT"/>
					<xsd:enumeration value="HUMAN"/>
					<xsd:enumeration value="SYSTEM"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
</xsd:element>
<xsd:element name="Participants">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element ref="xpdl:Participant" 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:element>
<xsd:element name="PartnerLink">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="MyRole" minOccurs="0">
				<xsd:complexType>
					<xsd:sequence>
						<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
					</xsd:sequence>
					<xsd:attribute name="RoleName" type="xsd:string" use="required"/>
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:complexType>
			</xsd:element>
			<xsd:element name="PartnerRole" minOccurs="0">
				<xsd:complexType>
					<xsd:sequence>
						<xsd:element ref="xpdl:EndPoint"/>
						<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
					</xsd:sequence>
					<xsd:attribute name="RoleName" type="xsd:string" use="required"/>
					<xsd:attribute name="ServiceName" type="xsd:string" use="optional"/>
					<xsd:attribute name="PortName" type="xsd:string" use="optional"/>
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:complexType>
			</xsd:element>
			<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:attribute name="name" type="xsd:string" use="required"/>
		<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
		<xsd:attribute name="PartnerLinkTypeId" type="xsd:NMTOKEN" use="required"/>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
</xsd:element>
<xsd:element name="PartnerLinks">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element ref="xpdl:PartnerLink" maxOccurs="unbounded"/>
			<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
</xsd:element>
<xsd:element name="PartnerLinkType">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="Role" maxOccurs="2">
				<xsd:complexType>
					<xsd:sequence>
						<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
					</xsd:sequence>
					<xsd:attribute name="portType" type="xsd:string" use="required"/>
					<xsd:attribute name="Name" type="xsd:string" use="required"/>
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:complexType>
			</xsd:element>
			<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
		<xsd:attribute name="name" type="xsd:string" use="required"/>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
</xsd:element>
<xsd:element name="PartnerLinkTypes">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element ref="xpdl:PartnerLinkType" maxOccurs="unbounded"/>
			<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:sequence>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
</xsd:element>
<xsd:element name="Performer">
	<xsd:annotation>
		<xsd:documentation>A String or Expression designating the Performer</xsd:documentation>
	</xsd:annotation>
	<xsd:complexType>
```

```
				<xsd:simpleContent>
					<xsd:extension base="xsd:string">
						<xsd:anyAttribute namespace="##other" processContents="lax"/>
					</xsd:extension>
				</xsd:simpleContent>
			</xsd:complexType>
		</xsd:element>
		<xsd:element name="Performers">
			<xsd:annotation>
				<xsd:documentation>BPMN and XPDL</xsd:documentation>
			</xsd:annotation>
			<xsd:complexType>
				<xsd:sequence>
					<xsd:element ref="xpdl:Performer" maxOccurs="unbounded"/>
					<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
				</xsd:sequence>
				<xsd:anyAttribute namespace="##other" processContents="lax"/>
			</xsd:complexType>
		</xsd:element>
		<xsd:element name="Pool">
			<xsd:annotation>
				<xsd:documentation>BPMN</xsd:documentation>
			</xsd:annotation>
			<xsd:complexType>
				<xsd:sequence>
					<xsd:element ref="xpdl:Lanes" minOccurs="0"/>
					<xsd:element ref="xpdl:Object" minOccurs="0"/>
					<xsd:element ref="xpdl:NodeGraphicsInfos" minOccurs="0"/>
					<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
				</xsd:sequence>
				<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
				<xsd:attribute name="Name" type="xsd:string" use="optional"/>
				<xsd:attribute name="Orientation" use="optional" default="HORIZONTAL">
					<xsd:simpleType>
						<xsd:restriction base="xsd:NMTOKEN">
							<xsd:enumeration value="HORIZONTAL"/>
							<xsd:enumeration value="VERTICAL"/>
						</xsd:restriction>
					</xsd:simpleType>
				</xsd:attribute>
				<xsd:attribute name="Process" type="xsd:NMTOKEN" use="optional"/>
				<xsd:attribute name="Participant" type="xsd:NMTOKEN" use="optional"/>
				<xsd:attribute name="BoundaryVisible" type="xsd:boolean" use="required"/>
				<xsd:anyAttribute namespace="##other" processContents="lax"/>
			</xsd:complexType>
		</xsd:element>
		<xsd:element name="Pools">
			<xsd:annotation>
				<xsd:documentation>BPMN</xsd:documentation>
			</xsd:annotation>
			<xsd:complexType>
				<xsd:sequence>
					<xsd:element ref="xpdl:Pool" 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:element>
		<xsd:element name="Precision">
			<xsd:complexType>
				<xsd:simpleContent>
					<xsd:extension base="xsd:short">
						<xsd:anyAttribute namespace="##other" processContents="lax"/>
					</xsd:extension>
				</xsd:simpleContent>
			</xsd:complexType>
		</xsd:element>
		<xsd:element name="Priority">
			<xsd:complexType>
				<xsd:simpleContent>
					<xsd:extension base="xsd:string">
						<xsd:anyAttribute namespace="##other" processContents="lax"/>
					</xsd:extension>
				</xsd:simpleContent>
			</xsd:complexType>
		</xsd:element>
		<xsd:element name="PriorityUnit">
			<xsd:complexType>
				<xsd:simpleContent>
					<xsd:extension base="xsd:string">
						<xsd:anyAttribute namespace="##other" processContents="lax"/>
					</xsd:extension>
				</xsd:simpleContent>
			</xsd:complexType>
		</xsd:element>
		<xsd:element name="ProcessHeader">
			<xsd:complexType>
				<xsd:sequence>
					<xsd:element ref="xpdl:Created" minOccurs="0"/>
					<xsd:element ref="xpdl:Description" minOccurs="0"/>
					<xsd:element ref="xpdl:Priority" minOccurs="0"/>
					<xsd:element ref="xpdl:Limit" minOccurs="0"/>
				<xsd:element ref="xpdl:ValidFrom" minOccurs="0"/>
				<xsd:element ref="xpdl:ValidTo" minOccurs="0"/>
				<xsd:element ref="xpdl:TimeEstimation" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="DurationUnit">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="Y"/>
						<xsd:enumeration value="M"/>
						<xsd:enumeration value="D"/>
						<xsd:enumeration value="h"/>
						<xsd:enumeration value="m"/>
						<xsd:enumeration value="s"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:complexType name="ProcessType">
		<xsd:sequence>
			<xsd:element ref="xpdl:ProcessHeader"/>
			<xsd:element ref="xpdl:RedefinableHeader" minOccurs="0"/>
			<xsd:element ref="xpdl:FormalParameters" minOccurs="0"/>
			<xsd:choice minOccurs="0">
				<xsd:annotation>
					<xsd:documentation>changes xpdl 1.0 order</xsd:documentation>
				</xsd:annotation>
				<xsd:sequence minOccurs="0">
					<xsd:element ref="xpdl:Participants" minOccurs="0"/>
					<xsd:element ref="xpdl:Applications" minOccurs="0"/>
					<xsd:element ref="xpdl:DataFields" minOccurs="0"/>
				</xsd:sequence>
				<xsd:sequence minOccurs="0">
					<xsd:element ref="deprecated:DataFields" minOccurs="0"/>
					<xsd:element ref="deprecated:Participants" minOccurs="0"/>
					<xsd:element ref="deprecated:Applications" minOccurs="0"/>
				</xsd:sequence>
			</xsd:choice>
			<xsd:element ref="xpdl:ActivitySets" minOccurs="0"/>
			<xsd:element ref="xpdl:Activities" minOccurs="0"/>
			<xsd:element ref="xpdl:Transitions" minOccurs="0"/>
			<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
			<xsd:element ref="xpdl:Assignments" minOccurs="0"/>
			<xsd:element ref="xpdl:PartnerLinks" minOccurs="0"/>
			<xsd:element ref="xpdl:Object" minOccurs="0"/>
			<xsd:choice minOccurs="0">
				<xsd:sequence>
					<xsd:element name="Extensions"/>
					<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
				</xsd:sequence>
			</xsd:choice>
		</xsd:sequence>
		<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
		<xsd:attribute name="Name" type="xsd:string" use="optional"/>
		<xsd:attribute name="AccessLevel" use="optional" default="PUBLIC">
			<xsd:simpleType>
				<xsd:restriction base="xsd:NMTOKEN">
					<xsd:enumeration value="PUBLIC"/>
					<xsd:enumeration value="PRIVATE"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
		<xsd:attribute name="ProcessType" use="optional" default="None">
			<xsd:annotation>
				<xsd:documentation>BPMN:</xsd:documentation>
			</xsd:annotation>
			<xsd:simpleType>
				<xsd:restriction base="xsd:NMTOKEN">
					<xsd:enumeration value="None"/>
					<xsd:enumeration value="Private"/>
					<xsd:enumeration value="Abstract"/>
					<xsd:enumeration value="Collaboration"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
		<xsd:attribute name="Status" use="optional" default="None">
			<xsd:annotation>
				<xsd:documentation> BPMN: Status values are assigned during execution. Status can be treated as a property and used in expressions local to a
Process. It is unclear that status belongs in the XPDL document.</xsd:documentation>
			</xsd:annotation>
			<xsd:simpleType>
				<xsd:restriction base="xsd:NMTOKEN">
					<xsd:enumeration value="None"/>
					<xsd:enumeration value="Ready"/>
					<xsd:enumeration value="Active"/>
					<xsd:enumeration value="Cancelled"/>
					<xsd:enumeration value="Aborting"/>
					<xsd:enumeration value="Aborted"/>
					<xsd:enumeration value="Completing"/>
					<xsd:enumeration value="Completed"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
		<xsd:attribute name="SuppressJoinFailure" type="xsd:boolean" use="optional" default="false"/>
		<xsd:attribute name="EnableInstanceCompensation" type="xsd:boolean" use="optional" default="false"/>
		<xsd:attribute name="AdHoc" type="xsd:boolean" use="optional" default="false">
			<xsd:annotation>
				<xsd:documentation>BPMN: for Embedded subprocess</xsd:documentation>
			</xsd:annotation>
		</xsd:attribute>
		<xsd:attribute name="AdHocOrdering" use="optional" default="Parallel">
			<xsd:annotation>
				<xsd:documentation>BPMN: for Embedded subprocess</xsd:documentation>
			</xsd:annotation>
			<xsd:simpleType>
				<xsd:restriction base="xsd:NMTOKEN">
					<xsd:enumeration value="Sequential"/>
					<xsd:enumeration value="Parallel"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
		<xsd:attribute name="AdHocCompletionCondition" type="xsd:string" use="optional">
			<xsd:annotation>
				<xsd:documentation>BPMN: for Embedded subprocess</xsd:documentation>
			</xsd:annotation>
		</xsd:attribute>
		<xsd:attribute name="DefaultStartActivitySetId" type="xsd:NMTOKEN" use="optional"/>
		<xsd:attribute name="DefaultStartActivityId" type="xsd:NMTOKEN" use="optional"/>
		<xsd:anyAttribute namespace="##other" processContents="lax"/>
	</xsd:complexType>
	<xsd:element name="RecordType">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Member" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="RedefinableHeader">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Author" minOccurs="0"/>
				<xsd:element ref="xpdl:Version" minOccurs="0"/>
				<xsd:element ref="xpdl:Codepage" minOccurs="0"/>
				<xsd:element ref="xpdl:Countrykey" minOccurs="0"/>
				<xsd:element ref="xpdl:Responsibles" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="PublicationStatus">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="UNDER_REVISION"/>
						<xsd:enumeration value="RELEASED"/>
						<xsd:enumeration value="UNDER_TEST"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Reference">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="ActivityId" type="xsd:NMTOKEN" use="required">
				<xsd:annotation>
					<xsd:documentation> Should be the Id of an activity which invokes a subflow (independent or embedded) or a task. In the BPMN speck this
atribute is called ProcessRef</xsd:documentation>
				</xsd:annotation>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Responsible">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Responsibles">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Responsible" 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:element>
        <xsd:element name="ResultCompensation">
            <xsd:annotation>
                <xsd:documentation>BPMN: Must be present if if Trigger or ResultType is Compensation.</xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="ActivityId" type="xsd:NMTOKEN" use="optional">
                    <xsd:annotation>
                        <xsd:documentation> This supplies the Id of the Activity to be Compensated. Used only for intermediate events or end events in the seuence
flow. Events attached to the boundary of an activity already know the Id.</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="ResultError">
            <xsd:annotation>
                <xsd:documentation>BPMN: Must be present if Trigger or ResultType is error.</xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="ErrorCode" type="xsd:string" use="required"/>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="ResultMultiple">
            <xsd:annotation>
                <xsd:documentation>BPMN: Must be present if ResultType is Multiple.</xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:annotation>
                        <xsd:documentation>at least two results must be present</xsd:documentation>
                    </xsd:annotation>
                    <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
                    <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
                    <xsd:element ref="xpdl:ResultCompensation" minOccurs="0"/>
                    <xsd:element ref="xpdl:ResultError" minOccurs="0"/>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="Route">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="GatewayType" use="optional" default="XOR">
                    <xsd:annotation>
                        <xsd:documentation> Used when needed for BPMN Gateways. Gate and sequence information is associated with the Transition
Element.</xsd:documentation>
                    </xsd:annotation>
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                            <xsd:enumeration value="XOR"/>
                            <xsd:enumeration value="OR"/>
                            <xsd:enumeration value="Complex"/>
                            <xsd:enumeration value="AND"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="Instantiate" type="xsd:boolean" use="optional" default="false"/>
                <xsd:attribute name="MarkerVisible" type="xsd:boolean" use="optional" default="false">
                    <xsd:annotation>
                        <xsd:documentation>Applicable only to XOR Gateways</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="Rule">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="Expression" type="xpdl:ExpressionType"/>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
                <xsd:attribute name="Name" type="xsd:string" use="optional"/>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="Scale">
            <xsd:complexType>
                <xsd:simpleContent>
                    <xsd:extension base="xsd:short">
                        <xsd:anyAttribute namespace="##other" processContents="lax"/>
                    </xsd:extension>
                </xsd:simpleContent>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="SchemaType">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="Script">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="Type" type="xsd:string" use="required"/>
                <xsd:attribute name="Version" type="xsd:string" use="optional"/>
                <xsd:attribute name="Grammar" type="xsd:anyURI" use="optional"/>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="SimulationInformation">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="xpdl:Cost"/>
                    <xsd:element ref="xpdl:TimeEstimation"/>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="Instantiation">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                            <xsd:enumeration value="ONCE"/>
                            <xsd:enumeration value="MULTIPLE"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="Split">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="xpdl:TransitionRefs" minOccurs="0"/>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="Type">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                            <xsd:enumeration value="AND"/>
                            <xsd:enumeration value="XOR"/>
                            <xsd:enumeration value="XOREVENT"/>
                            <xsd:enumeration value="OR"/>
                            <xsd:enumeration value="COMPLEX"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="OutgoingCondition" type="xsd:string"/>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="StartEvent">
            <xsd:annotation>
                <xsd:documentation>BPMN</xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
                <xsd:choice minOccurs="0">
                    <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
                    <xsd:element ref="xpdl:TriggerTimer" minOccurs="0"/>
                    <xsd:element ref="xpdl:TriggerRule" minOccurs="0"/>
                    <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
                    <xsd:element ref="xpdl:TriggerMultiple" minOccurs="0"/>
                </xsd:choice>
                <xsd:attribute name="Trigger" use="required">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                            <xsd:enumeration value="None"/>
                            <xsd:enumeration value="Message"/>
                            <xsd:enumeration value="Timer"/>
                            <xsd:enumeration value="Rule"/>
                            <xsd:enumeration value="Link"/>
                            <xsd:enumeration value="Multiple"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="Implementation" use="optional" default="WebService">
                    <xsd:annotation>
                        <xsd:documentation>Required if the Trigger is Message</xsd:documentation>
                    </xsd:annotation>
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                            <xsd:enumeration value="WebService"/>
                            <xsd:enumeration value="Other"/>
                            <xsd:enumeration value="Unspecified"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="SubFlow">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:choice minOccurs="0">
                        <xsd:element ref="xpdl:ActualParameters"/>
                        <xsd:element ref="xpdl:DataMappings"/>
                    </xsd:choice>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="Id" type="xsd:string" use="required">
                    <xsd:annotation>
                        <xsd:documentation>Used in XPDL and BPMN. In BPMN is equivalent to ProcessRef attribute.</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="Name" type="xsd:string" use="optional"/>
                <xsd:attribute name="Execution" use="optional" default="SYNCHR">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                            <xsd:enumeration value="ASYNCHR"/>
                            <xsd:enumeration value="SYNCHR"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="PackageRef" type="xsd:NMTOKEN" use="optional">
                    <xsd:annotation>
                        <xsd:documentation>BPMN: needed for independent subprocess</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="InstanceDataField" type="xsd:string" use="optional">
                    <xsd:annotation>
                        <xsd:documentation> Used to store the instance id of the subflow instantiated by the activity. This is then available later on (e.g. for
correlation, messaging etc.) especially in the case of asynchronous invocation.</xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="StartActivitySetId" type="xsd:NMTOKEN" use="optional"/>
                <xsd:attribute name="StartActivityId" type="xsd:NMTOKEN" use="optional"/>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="Task">
            <xsd:annotation>
                <xsd:documentation>BPMN</xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
                <xsd:choice>
                    <xsd:element ref="xpdl:TaskService"/>
                    <xsd:element ref="xpdl:TaskReceive"/>
                    <xsd:element ref="xpdl:TaskManual"/>
                    <xsd:element ref="xpdl:TaskReference"/>
                    <xsd:element ref="xpdl:TaskScript"/>
                    <xsd:element ref="xpdl:TaskSend"/>
                    <xsd:element ref="xpdl:TaskUser"/>
                    <xsd:element ref="xpdl:TaskApplication"/>
                </xsd:choice>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="TaskManual">
            <xsd:annotation>
                <xsd:documentation>BPMN</xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="xpdl:Performers"/>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:anyAttribute namespace="##other" processContents="lax"/>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="TaskReceive">
            <xsd:annotation>
                <xsd:documentation>BPMN</xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="Message" type="xpdl:MessageType"/>
                    <xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
                    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attribute name="Instantiate" type="xsd:boolean" use="required"/>
```

```
			<xsd:attribute name="Implementation" use="optional" default="WebService">
				<xsd:annotation>
					<xsd:documentation>Required if the Task is Receive</xsd:documentation>
				</xsd:annotation>
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="WebService"/>
						<xsd:enumeration value="Other"/>
						<xsd:enumeration value="Unspecified"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TaskReference">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="TaskRef" type="xsd:NMTOKEN" use="required"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TaskSend">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element name="Message" type="xpdl:MessageType"/>
				<xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
				<xsd:element ref="xpdl:WebServiceFaultCatch" minOccurs="0" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Implementation" use="optional" default="WebService">
				<xsd:annotation>
					<xsd:documentation>Required if the Task is Send</xsd:documentation>
				</xsd:annotation>
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="WebService"/>
						<xsd:enumeration value="Other"/>
						<xsd:enumeration value="Unspecified"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TaskService">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element name="MessageIn" type="xpdl:MessageType"/>
				<xsd:element name="MessageOut" type="xpdl:MessageType"/>
				<xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
				<xsd:element ref="xpdl:WebServiceFaultCatch" minOccurs="0" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Implementation" use="optional" default="WebService">
				<xsd:annotation>
					<xsd:documentation>Required if the Task is Service</xsd:documentation>
				</xsd:annotation>
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="WebService"/>
						<xsd:enumeration value="Other"/>
						<xsd:enumeration value="Unspecified"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TaskScript">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element name="Script" type="xpdl:ExpressionType">
					<xsd:annotation>
						<xsd:documentation>BPMN</xsd:documentation>
					</xsd:annotation>
				</xsd:element>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TaskUser">
		<xsd:annotation>
			<xsd:documentation>BPMN</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Performers"/>
				<xsd:element name="MessageIn" type="xpdl:MessageType"/>
				<xsd:element name="MessageOut" type="xpdl:MessageType"/>
				<xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Implementation" use="optional" default="WebService">
				<xsd:annotation>
					<xsd:documentation>Required if the Task is User</xsd:documentation>
				</xsd:annotation>
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="WebService"/>
						<xsd:enumeration value="Other"/>
						<xsd:enumeration value="Unspecified"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TimeEstimation">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:WaitingTime" minOccurs="0"/>
				<xsd:element ref="xpdl:WorkingTime" minOccurs="0"/>
				<xsd:element ref="xpdl:Duration" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TaskApplication">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:choice minOccurs="0">
					<xsd:element ref="xpdl:ActualParameters"/>
					<xsd:element ref="xpdl:DataMappings"/>
				</xsd:choice>
				<xsd:element ref="xpdl:Description" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:attribute name="PackageRef" type="xsd:NMTOKEN" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Transaction">
		<xsd:annotation>
			<xsd:documentation>BPMN: If SubProcess is a transaction then this is required.</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="TransactionId" type="xsd:string" use="required"/>
			<xsd:attribute name="TransactionProtocol" type="xsd:string" use="required"/>
			<xsd:attribute name="TransactionMethod" use="required">
				<xsd:simpleType>
					<xsd:restriction base="xsd:NMTOKEN">
						<xsd:enumeration value="Compensate"/>
						<xsd:enumeration value="Store"/>
						<xsd:enumeration value="Image"/>
					</xsd:restriction>
				</xsd:simpleType>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Transition">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Condition" minOccurs="0"/>
				<xsd:element ref="xpdl:Description" minOccurs="0"/>
				<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
				<xsd:element ref="xpdl:Assignments" minOccurs="0"/>
				<xsd:element ref="xpdl:Object" minOccurs="0"/>
				<xsd:element ref="xpdl:ConnectorGraphicsInfos" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="From" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="To" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:attribute name="Quantity" type="xsd:int" use="optional" default="1">
				<xsd:annotation>
					<xsd:documentation>Used only in BPMN. Specifies number of tokens on outgoing transition.</xsd:documentation>
				</xsd:annotation>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TransitionRef">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="Name" type="xsd:string" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TransitionRefs">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:TransitionRef" 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:element>
	<xsd:element name="TransitionRestriction">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Join" minOccurs="0"/>
				<xsd:element ref="xpdl:Split" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TransitionRestrictions">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:TransitionRestriction" 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:element>
	<xsd:element name="Transitions">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:Transition" 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:element>
	<xsd:element name="TriggerResultLink">
		<xsd:annotation>
			<xsd:documentation>BPMN: if the Trigger or Result Type is Link then this must be present.</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="LinkId" type="xsd:NMTOKEN" use="required"/>
			<xsd:attribute name="ProcessRef" type="xsd:NMTOKEN" use="required">
				<xsd:annotation>
					<xsd:documentation>This must identify a Process. Should be the Id of a process.</xsd:documentation>
				</xsd:annotation>
			</xsd:attribute>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TriggerResultMessage">
		<xsd:annotation>
			<xsd:documentation> BPMN: If the Trigger or Result Type is Message then this must be present</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element name="Message" type="xpdl:MessageType"/>
				<xsd:element ref="xpdl:WebServiceOperation" minOccurs="0"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="TriggerIntermediateMultiple">
		<xsd:annotation>
			<xsd:documentation>BPMN: if the TriggerType is Multiple then this must be present.</xsd:documentation>
		</xsd:annotation>
```

```
        <xsd:complexType>
            <xsd:sequence>
                <xsd:annotation>
                    <xsd:documentation>BPMN: For Multiple, at least two triggers must be present.</xsd:documentation>
                </xsd:annotation>
                <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
                <xsd:element ref="xpdl:TriggerTimer" minOccurs="0"/>
                <xsd:element ref="xpdl:ResultError" minOccurs="0"/>
                <xsd:element ref="xpdl:ResultCompensation" minOccurs="0"/>
                <xsd:element ref="xpdl:TriggerRule" minOccurs="0"/>
                <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="TriggerMultiple">
        <xsd:annotation>
            <xsd:documentation>BPMN: if the TriggerType is Multiple then this must be present.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:annotation>
                    <xsd:documentation>BPMN: For Multiple, at least two triggers must be present.</xsd:documentation>
                </xsd:annotation>
                <xsd:element ref="xpdl:TriggerResultMessage" minOccurs="0"/>
                <xsd:element ref="xpdl:TriggerTimer" minOccurs="0"/>
                <xsd:element ref="xpdl:TriggerRule" minOccurs="0"/>
                <xsd:element ref="xpdl:TriggerResultLink" minOccurs="0"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="TriggerRule">
        <xsd:annotation>
            <xsd:documentation>BPMN: if the TriggerType is Rule then this must be present.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="RuleName" type="xsd:string" use="required">
                <xsd:annotation>
                    <xsd:documentation>This is the nameof a Rule element.</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="TriggerTimer">
        <xsd:annotation>
            <xsd:documentation>BPMN: If the Trigger Type is Timer then this must be present</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="TimeDate" type="xsd:string" use="optional">
                <xsd:annotation>
                    <xsd:documentation>One of TimeDate or TimeCycle must be present</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="TimeCycle" type="xsd:string" use="optional"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="TypeDeclaration">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:group ref="xpdl:DataTypes"/>
                <xsd:element ref="xpdl:Description" minOccurs="0"/>
                <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
                <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="Id" type="xsd:ID" use="required"/>
            <xsd:attribute name="Name" type="xsd:string" use="optional"/>
            <xsd:anyAttribute namespace="##other" processContents="lax"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="TypeDeclarations">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xpdl:TypeDeclaration" 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:element>
    <xsd:element name="UnionType">
        <xsd:complexType>
            <xsd:sequence>
				<xsd:element ref="xpdl:Member" maxOccurs="unbounded"/>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="ValidFrom">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="ValidTo">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="Vendor">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="VendorExtension">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="ToolId" type="xsd:string" use="required"/>
			<xsd:attribute name="schemaLocation" type="xsd:anyURI" use="required"/>
			<xsd:attribute name="extensionDescription" type="xsd:anyURI" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="VendorExtensions">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:VendorExtension" 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:element>
	<xsd:element name="Version">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="WaitingTime">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="WebServiceFaultCatch">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element name="Message" type="xpdl:MessageType" minOccurs="0"/>
				<xsd:choice>
					<xsd:element ref="xpdl:BlockActivity"/>
					<xsd:element ref="xpdl:TransitionRef"/>
				</xsd:choice>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="FaultName" type="xsd:NMTOKEN" use="optional"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="WebServiceOperation">
		<xsd:annotation>
			<xsd:documentation>BPMN: If the Implementation is a WebService this is required.</xsd:documentation>
		</xsd:annotation>
		<xsd:complexType>
			<xsd:sequence>
				<xsd:choice>
```

```
					<xsd:element name="Partner">
						<xsd:complexType>
							<xsd:sequence>
								<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
							</xsd:sequence>
							<xsd:attribute name="PartnerLinkId" type="xsd:NMTOKEN" use="required"/>
							<xsd:attribute name="RoleType" use="required">
								<xsd:simpleType>
									<xsd:restriction base="xsd:NMTOKEN">
										<xsd:enumeration value="MyRole"/>
										<xsd:enumeration value="PartnerRole"/>
									</xsd:restriction>
								</xsd:simpleType>
							</xsd:attribute>
							<xsd:anyAttribute namespace="##other" processContents="lax"/>
						</xsd:complexType>
					</xsd:element>
					<xsd:element name="Service">
						<xsd:complexType>
							<xsd:sequence>
								<xsd:element ref="xpdl:EndPoint"/>
								<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
							</xsd:sequence>
							<xsd:attribute name="ServiceName" type="xsd:string" use="required"/>
							<xsd:attribute name="PortName" type="xsd:string" use="required"/>
							<xsd:anyAttribute namespace="##other" processContents="lax"/>
						</xsd:complexType>
					</xsd:element>
				</xsd:choice>
				<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
			</xsd:sequence>
			<xsd:attribute name="OperationName" type="xsd:string" use="required"/>
			<xsd:anyAttribute namespace="##other" processContents="lax"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="WorkflowProcess" type="xpdl:ProcessType"/>
	<xsd:element name="WorkflowProcesses">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref="xpdl:WorkflowProcess" 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:element>
	<xsd:element name="WorkingTime">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name="XPDLVersion">
		<xsd:complexType>
			<xsd:simpleContent>
				<xsd:extension base="xsd:string">
					<xsd:anyAttribute namespace="##other" processContents="lax"/>
				</xsd:extension>
			</xsd:simpleContent>
		</xsd:complexType>
	</xsd:element>
</xsd:schema>
```

10. Figures and Tables

10.1. Figures

Figure 5-1: The Concept of the Process Definition Interchange 9
Figure 6-1 Package Definition Meta Model 11
Figure 6-2: Process Definition Meta Model 13
Figure 7-1: Activity Structures & Transition Conditions 52
Figure 8-1: EOrder Main Process 109
Figure 8-2: CreditCheck Subprocess 110
Figure 8-3: FillOrder Subprocess 110

10.2. Tables

Table 1: Node Graphics Info 19
Table 2: ConnnectorGraphicsInfo 20
Table 3: Extended Attributes 21
Table 4: Formal Parameters 22
Table 5: External Reference 23
Table 6: Assignment 24
Table 7: Category 25
Table 8: Artifact 26
Table 9: Object 26
Table 10: Data Object 27
Table 11: Package Definition 29
Table 12: Package Definition Header – Attributes 30
Table 13: Vendor Extension -- Attributes 31
Table 14: Redefinable Header 33
Table 15: Conformance Class Declaration 33
Table 16: Script 34
Table 17: External Package Reference 35
Table 18: Application Declaration 36
Table 19: EJB Application Type 37
Table 20: POJO Application Type 37
Table 21: XSLT Application Type 38
Table 22: Script Application Type 38
Table 23: WebService Application Type 39
Table 24: BusinessRule Application Type 39

Table 25: Form Application Type 40
Table 26: Pools 41
Table 27: Lane 42
Table 28: Process Definition 45
Table 29: Process Definition Header 47
Table 30: Process Redefinable Header 49
Table 31: ActivitySet 50
Table 32: Entity type relationships for different Activity types 52
Table 33: Process Activity 56
Table 34: Route Activity 57
Table 35 BlockActivity 59
Table 36: Event 60
Table 37: Start Event Activity 61
Table 38: Intermediate Event Activity 63
Table 39: End Event Activity 64
Table 40: Event ResultCompensation 65
Table 41: Event ResultError 65
Table 42: Event ResultMultiple 66
Table 43: Event TriggerResultLink 66
Table 44: Event TriggerResultMessage 67
Table 45: Event TriggerIntermediat Multiple 67
Table 46: Event TriggerMultiple 68
Table 47: Event TriggerRule 68
Table 48: Event TriggerTimer 68
Table 49: Implementation Alternatives 69
Table 50: Task Manual 70
Table 51: Task Receive 71
Table 52: TaskReference 71
Table 53: TaskSend 72
Table 54: Task Service 73
Table 55: Task Script 73
Table 56: Task User 74
Table 57: Tool 74
Table 58: SubFlow 75
Table 59: DataMapping 76
Table 60: Reference 77
Table 61: Deadline 78
Table 62: Simulation Information 80
Table 63: Transition Restrictions 80

Table 64: Join 81
Table 65: Split 83
Table 66: Conformance Class 84
Table 67: Input 85
Table 68: Output 86
Table 69: Transaction 87
Table 70: Loop 89
Table 71: Transition Information 90
Table 72: Condition 91
Table 73: PartnerLinkType 93
Table 74: PartnerLink 94
Table 75: MessageFlow 95
Table 76: Message – attributes 95
Table 77: End Point 96
Table 78: Web Service Operation 97
Table 79: Web Service Fault Catch 98
Table 80: Association 99
Table 81: Participant 100
Table 82: Participant Entity Type 100
Table 83: Relevant data field 102
Table 84: Standard Data Types 102
Table 85: Basic Data Types 103
Table 86: Record Type 104
Table 87: Union Type 105
Table 88: Enumeration Type 105
Table 89: Array Type 106
Table 90: List Type 106
Table 91: Type Declaration 107
Table 92: Declared Data Type 107