The Workflow Management Coalition Specification

Workflow Management Coalition Workflow Standard

Workflow Process Definition Interface -- XML Process Definition Language

Document Number WFMC-TC-1025
Document Status – Draft 0.03a (Alpha Status)

May 22, 2001
Version 0.03 (Draft)

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

First printing May 2001

Table of Content

1. Change History

Version 0.01 – Editor: Mike Marin (mmarin@filenet.com)

- Initial Version.

Versions 0.02/0.03 – Editor: Mike Marin (mmarin@filenet.com)

- Changes based on review by working group 1 during the New York meeting May 3 and 4 of 2001. This version has significant input from Roberta Norin (AP Engines), Robert Shapiro (Cape Visions), and all the other participants of the working group during the New York meeting.

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). 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

The WfMC has identified five functional interfaces to a 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 DTD 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 workflow process definition interface defines a common interchange format, which supports the transfer of workflow 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 workflow run-time products.

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

To provide a common method to access and describe workflow definitions, a workflow 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 the 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 workflow process definition (the "Minimum Meta Model"), the 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 workflow definition from XPDL.
- Export a workflow definition from the vendor's internal representation to XPDL.

A vendor can use a XSL style sheet to comply with those two operations.

All keywords and terms used within this specification are based upon the WfMC Glossary.

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 claiming conformance must generate valid, syntactically correct XPDL, and must be able to read all valid XPDL.

4.2. 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)

Workflow 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)

5. Overview of Process Definition Interchange

A Process Definition is defined as:

> *The representation of a business process in a form that supports automated manipulation, such as modelling, or enactment by a workflow 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 subflow, 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, workflow 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

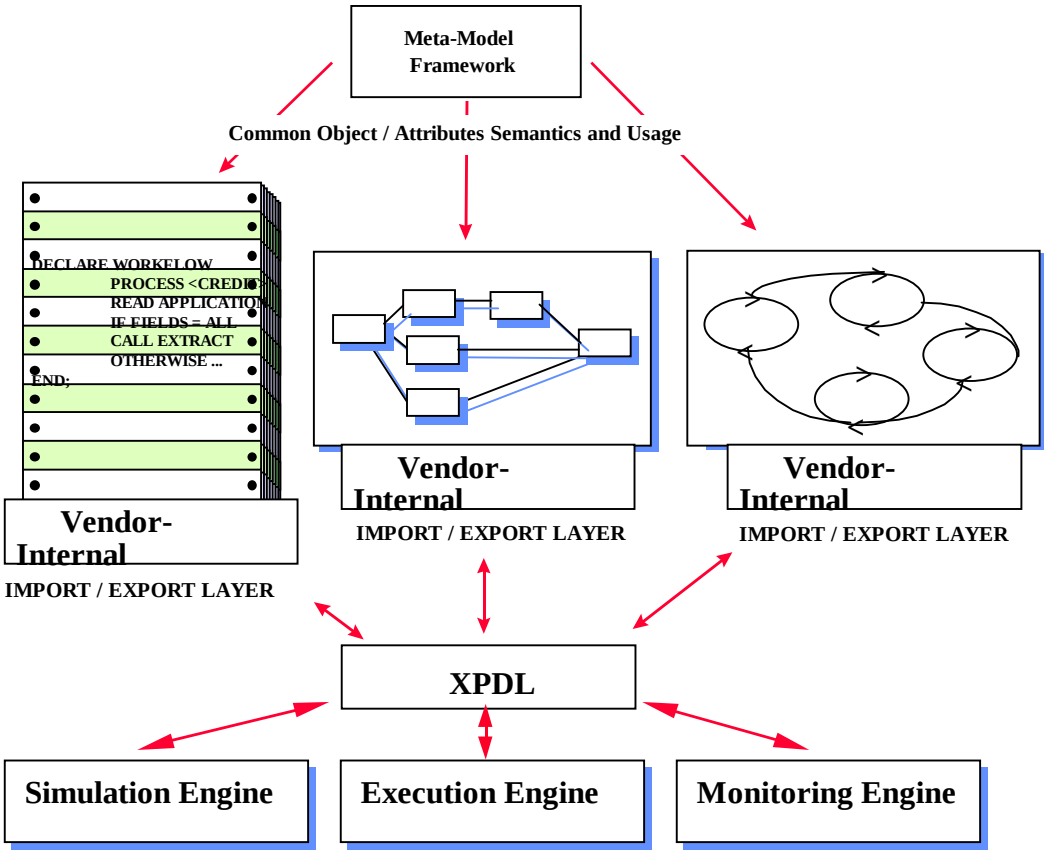

Figure 5-1: The Concept of the Process Definition Interchange

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.

The top-level entities are shown in the following figure:

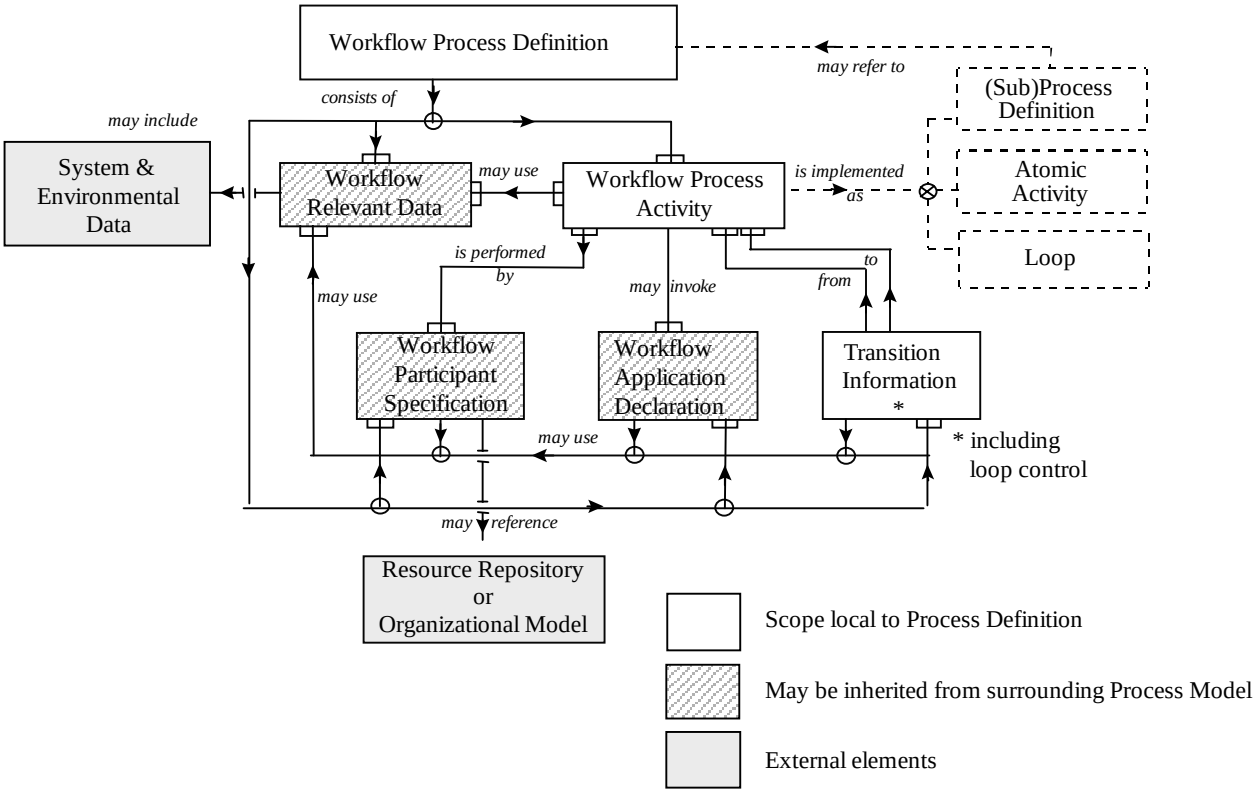

Figure 6-1: ***Meta-Model top-level entities***

For each of the above entities, there is an associated set of properties, which describe the characteristics of the entity. The following sections describe these entities and properties in more detail.

6.1. 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:

6.1.1. Workflow 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.).

6.1.2. Workflow 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 processed 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 a information on whether it is to be started / finished automatically by the workflow management system or its priority relative to other activities where contention for resource or system services occurs. Usage of specific workflow relevant data 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).

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 workflow service, or (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 workflow relevant data between calling and called process (and, where necessary, on return).

An activity may also be specified as a loop, which acts as controlling activity for repeated execution of a set of activities within the same process definition. In this case the set of looping activities is connected to the controlling (loop) activity by special loop begin/end transitions.

A number of activities may form an inline block, which is specified by particular transition constructs, and may be used, for example, to represent an explosion of a higher-level activity within a process definition as an alternative to a subflow.

Finally, a dummy activity is a skeletal 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.

6.1.3. 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 in 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 workflow 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 dummy 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 dummy 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.

6.1.4. Workflow 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 workflow participant declaration) which may be allocated to it. The workflow 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 workflow participant declaration.

6.1.5. 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.

6.1.6. Workflow Application Declaration

This provides descriptions of the IT applications or interfaces which may be invoked by the workflow 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 workflow management system. The workflow application definition reflects the interface between the workflow engine and the application or interface, including any parameters to be passed.

6.1.7. Workflow Relevant Data

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 workflow 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. Workflow relevant data 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 workflow process relevant data.

6.1.8. System & Environmental Data

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

6.1.9. 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 workflow relevant data, system or environmental data or participant data. Expressions may be formed using such data types to support conditional evaluations.

6.2. Process Definitions, Package & Process Repository

As indicated in the diagram above, the process model includes various entities whose scope may be wider than a single process definition. In particular the definitions of participants, applications and workflow relevant data may be referenced

from a number of process definitions. The meta-model assumes the use of a common process definition repository, associated with the workflow management system, 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.

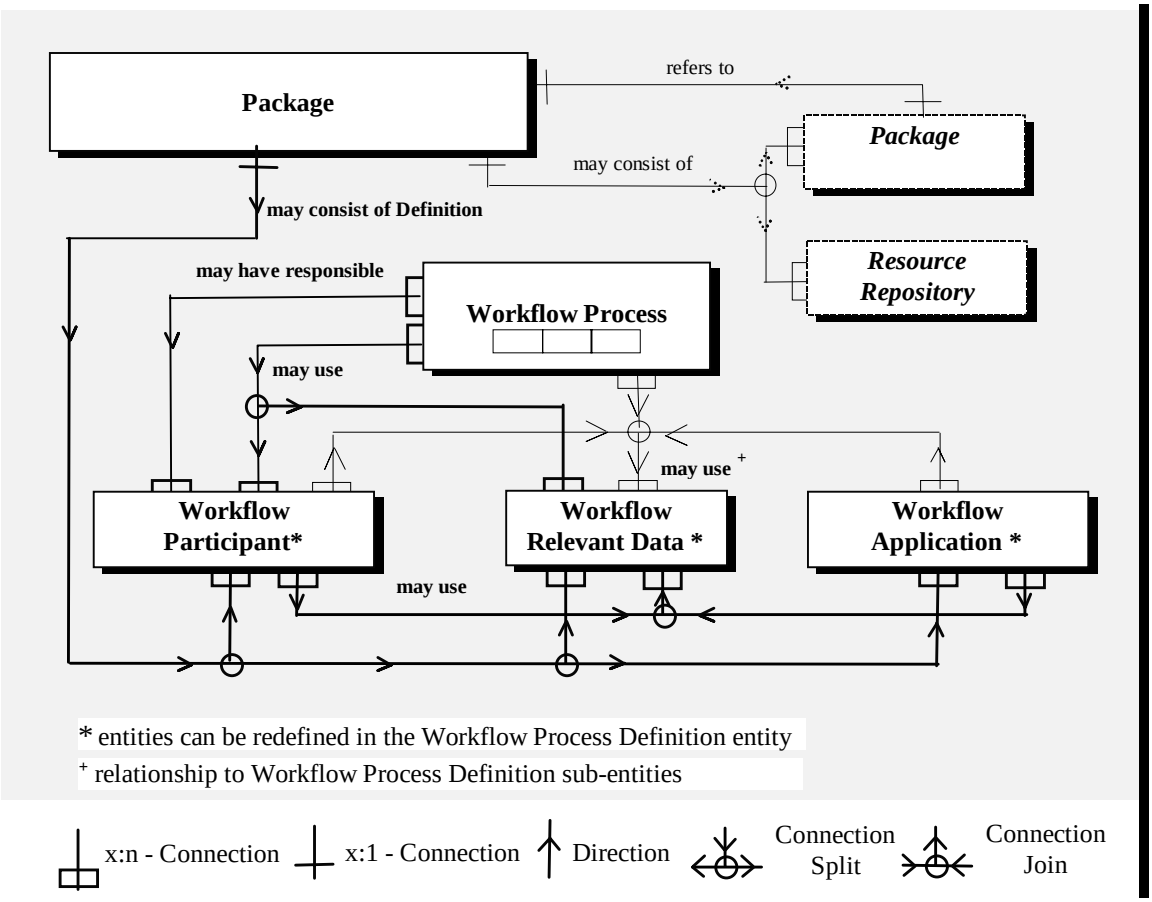

Figure 3.2 Workflow Process Model Entities

The package provides a container to hold a number of common attributes from the workflow 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

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

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

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
- Workflow participant specifications

- Workflow application 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

Workflow 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).

6.3. Elements Overview

The following table gives an overview of major elements defined within XPDL.

- The first row contains attributes and elements common to all major elements. All major elements have the attributes *id* and *name and may contain a Description and Extended Attributes.*
- The second row contains specific properties of the respective major element.
- The third group consists of elements that may contain references to other elements.
- Documentation and Icon elements contain presentation information to be used by the executing engine.
- The fifth group contains information relevant for simulation and process optimisation (BPR-relevant information).

Further elements and predefined attributes may be added to the model to create future conformance levels. A short description and the semantics of all elements are given in the subsequent chapters.

Package	Workflow Process	Activity	Transition	Application	Data Field (Workflow Relevant Data)	Participant
- Id - Name - Description - Extended Attributes	- Id - Name - Description - Extended Attributes	- Id - Name - Description - Extended Attributes	- Id - Name - Description - Extended Attributes	- Id - Name - Description - Extended Attributes	- Id - Name - Description - Extended Attributes	- Id - Name - Description - Extended Attributes
- XPDL Version - Source Vendor ID - Creation Date - Version - Author - Codepage - Country Key - Publication Status	- Creation Date - Version - Author - Codepage - Country Key - Publication Status - Priority	- Automation Mode - Split - Join - Loop - Inline Block - Priority - Limit			- Data Type	- Participant Type

- Conformance Class - Priority Unit	- Limit - Valid From Date - Valid To Date	- Start Mode - Finish Mode				
- Responsible - External Package	- Parameters - Responsible	- Performer - Tool - Subflow - Actual Parameter	- Condition - From - To	- Parameters	- Initial Value	
- Documentation - Icon	- Documentation - Icon	-Documentation- -Icon				
- Cost Unit	-Duration Unit - Duration - Waiting Time - Working Time	-Cost - Duration - Waiting Time - Working Time				

Table 6-1: Overview of Elements

6.3.1. 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; the mechanisms described below to support extension provide a standardized means of expressing the extension for interchange purposes but may require localized system adaptation to provide any associated runtime support during process enactment.

6.3.1.1. Extended 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 which need to be exchanged between systems. Any run-time semantics associated with the use of the extended attribute during process enactment are separately specified and require bilateral agreement between the exporter and the importing workflow service.

6.3.1.2. 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.

6.4. Elements Common for Multiple Entities

6.4.1. 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.

```
<!ELEMENT ExtendedAttribute ANY>
<!ATTLIST ExtendedAttribute
            Name NMTOKEN #REQUIRED
            Value CDATA #IMPLIED
>
```

	Description
Name	Used to identify the Extended Attribute
Value	Value required for a particular product.

Table 6-2: Extended Attributes

6.4.2. Formal Parameters

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

```
<!ELEMENT FormalParameters (FormalParameter*)>
<!ELEMENT FormalParameter (DataType, Description?)>
<!ATTLIST FormalParameter
            Id NMTOKEN #REQUIRED
            Index NMTOKEN #IMPLIED
            Mode (IN | OUT | INOUT) "IN"
>
```

	Description
Id	Identifier for the parameter
Index	Index of the parameter
Mode	IN - The Input Parameters OUT - The Output Parameters INOUT – Parameters used as input and output.

Table 6-3: *Formal Parameters*

6.4.2.1. 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 workflow relevant data 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 workflow relevant data 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 workflow relevant data entity). This is zero-restore semantics.

6.4.2.2. Concurrency semantics

Copying and restoring of parameters are treated as atomic operations; to avoid access conflicts from concurrent operations on workflow relevant data 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 workflow relevant data item) at the time of the return call.

6.4.2.3. 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. Type compatibility is required within the definitions and may be enforced by the run-time workflow system. The effects of violation are locally defined and do not form part of this specification

6.5. 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:

- Workflow Process Activity
- Transition Information
- Workflow Participant Specification
- Workflow Application Declaration
- Workflow Relevant Data

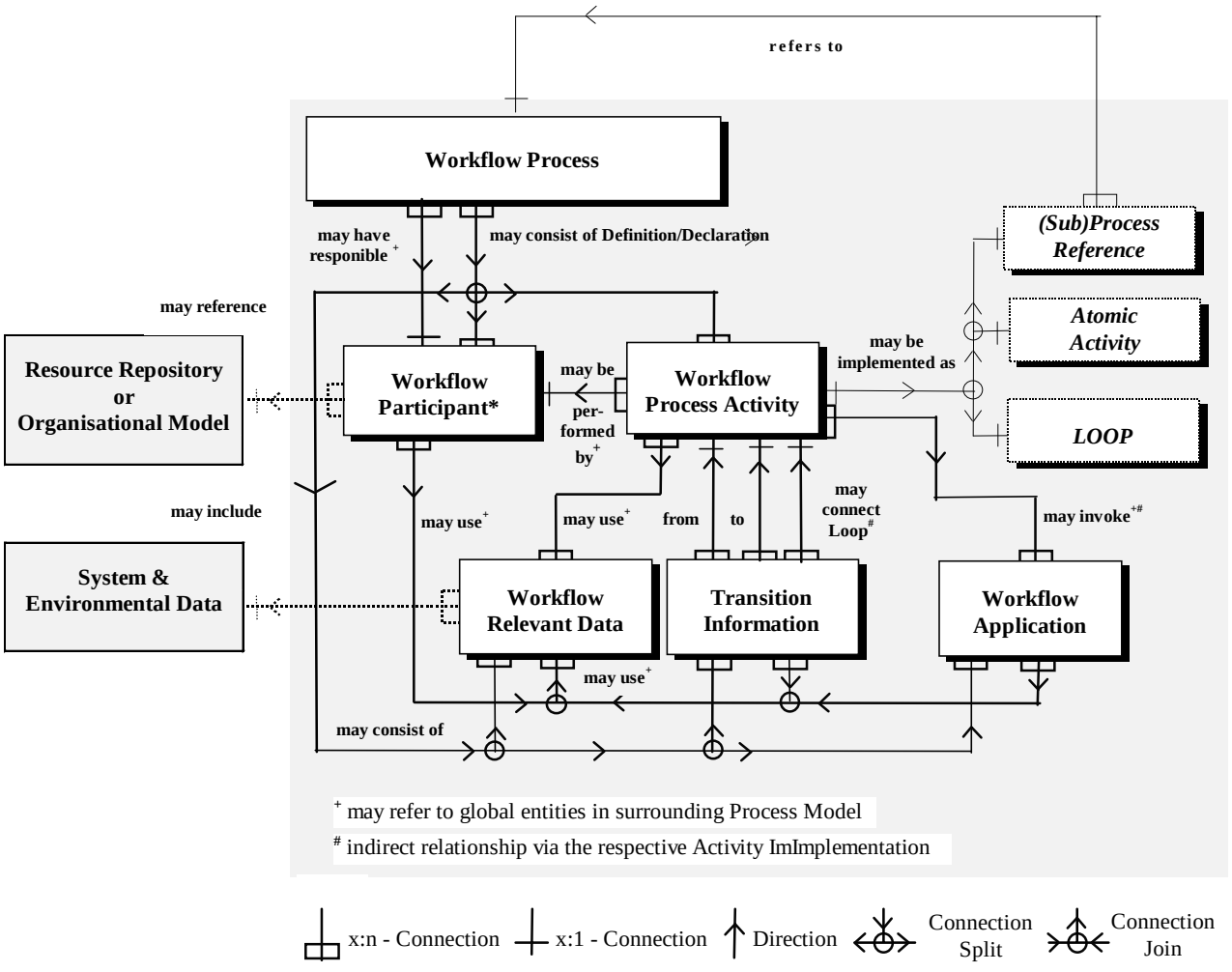

Figure 6-2: Workflow Process Definition Meta Model

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

6.5.1. Workflow Process Definition

The Workflow Process Definition defines the elements that make up a workflow. 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 Workflow 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 workflow 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 workflow relevant data 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 workflow relevant data 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 .

```
<!ELEMENT WorkflowProcesses (WorkflowProcess*)>
<!ELEMENT WorkflowProcess
            (ProcessHeader,
             RedefinableHeader?,
             FormalParameters?,
             (%Type;)*,
             DataFields?,
             Participants?,
             Applications?,
             Activities?,
             Transitions?,
             ExtendedAttributes?)
>
<!ATTLIST WorkflowProcess
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
<!ELEMENT ProcessHeader
            (Created?,
             Description?,
             Priority?,
             Limit?,
             ValidFrom?,
             ValidTo?,
             TimeEstimation?)
>
<!ATTLIST ProcessHeader
            DurationUnit (Y | M | D | h | m | s) #IMPLIED
>
<!ELEMENT Priority (#PCDATA)>
<!ELEMENT Limit (#PCDATA)>
<!ELEMENT TimeEstimation
            (WaitingTime?,
             WorkingTime?,
```



```
            Duration?)
>
<!ELEMENT WaitingTime (#PCDATA)>
<!ELEMENT WorkingTime (#PCDATA)>
<!ELEMENT Duration (#PCDATA)>
<!ELEMENT ValidFrom (#PCDATA)>
<!ELEMENT ValidTo (#PCDATA)>
<!ELEMENT Vendor (#PCDATA)>
<!ELEMENT Created (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Documentation (#PCDATA)>
<!ELEMENT PriorityUnit (#PCDATA)>
<!ELEMENT CostUnit (#PCDATA)>
<!ELEMENT RedefinableHeader
           (Author?,
            Version?,
            Codepage?,
            Countrykey?,
            Responsibles?)
>
<!ATTLIST RedefinableHeader
        PublicationStatus
                 (UNDER_REVISION
                 | RELEASED
                 | UNDER_TEST) #IMPLIED
>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Version (#PCDATA)>
<!ELEMENT Codepage (#PCDATA)>
<!ELEMENT Countrykey (#PCDATA)>
<!ELEMENT Responsible (#PCDATA)>
```

	Description
Id	Used to identify the workflow process.
Name	Text Used to identify the workflow process.
Description	Short textual description of the process.
Duration Unit	Duration unit.
Created	Creation date of workflow process definition.
Author	Name of the author of this workflow process definition. (The one, who put it into the repository)
Version	Version of this workflow process definition.
Codepage	The codepage used for the text parts. Default: Inherited from Model Definition.
Country key	A country key. Default: Inherited from Model Definition.
Responsible	Workflow participant, who is responsible for this workflow 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.

	Description
Publication Status	Status of the Workflow Process Definition. Default: Inherited from Model Definition.
Valid From	The date that the workflow 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.
Priority	The priority of the process type. Default: Inherited from Model Definition.
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.
Documentation	Operating System specific path- and filename of help file/description file.
Simulation Data	Estimations for simulation of a process
Duration	Expected duration time to perform a task in units of DurationUnit.
Cost	Average cost for cost management purposes (used within analysis environment).
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.
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.
Formal Parameters	Parameters, which are interchanged with the process (for use as Subflow).
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

Table 6-4: Attributes of Entity Workflow Process

6.5.2. Workflow Process Activity

The Workflow Activity Definition is used to define each elementary activity that makes up a workflow 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 workflow 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 workflow 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 workflow relevant data 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				
	None	Application	Subflow	Loop	Dummy (Route)
Transition Restriction	Normal	Normal	Normal, plus subflow call / return within activity	Normal, plus single loop control within activity	Normal; any additional controls implemented within Route activity
Participant Assignment	Normal	Normal	N/A	N/A	N/A
Application Assignment	None	Yes	N/A	N/A	N/A
Use of workflow Relevant Data	Normal	Normal	may be used in parameter passing	may be used in loop control conditions	may be used in routing control conditions

Table 6-5: Entity type relationships for different Activity types

Notes on usage:

Transition restrictions, subflow, loop 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, loop, 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.

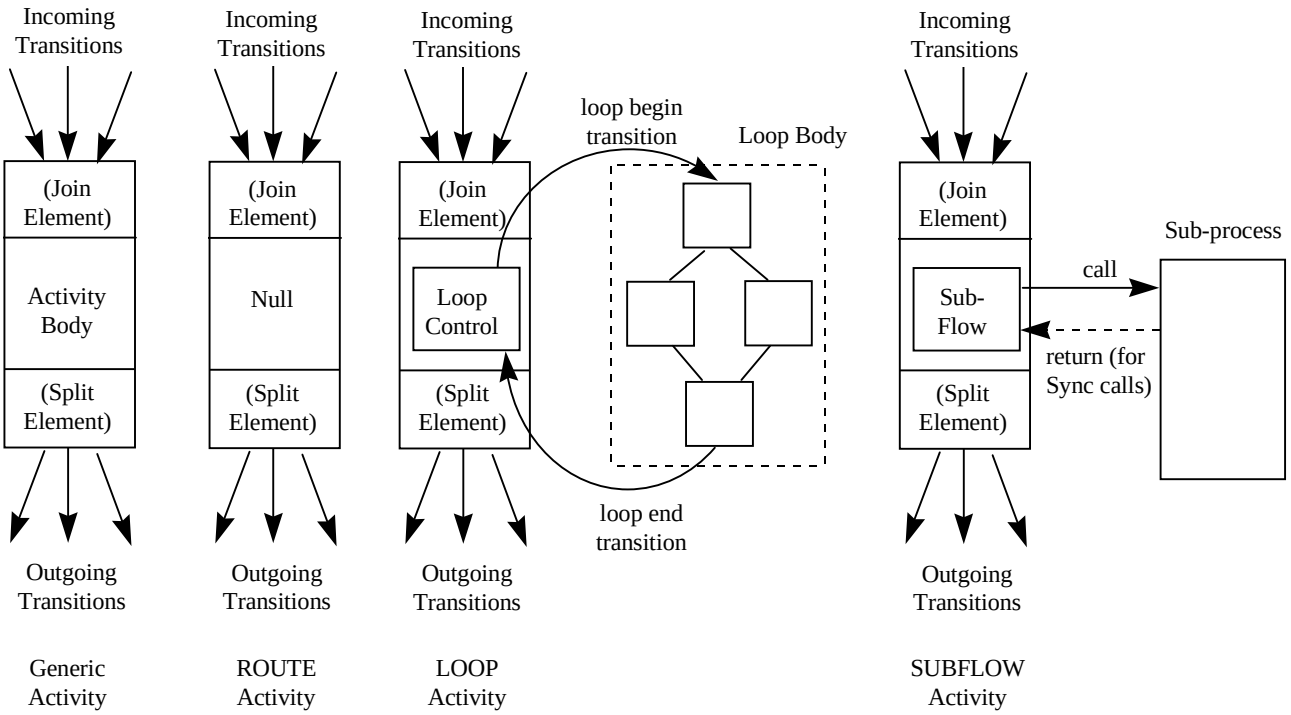

Figure 6-3: ***Activity Structures & Transition Conditions***

Where the implementation type is NONE, the workflow activity is manually controlled and its completion must be explicitly signaled to the workflow 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.

Workflow relevant data 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 workflow relevant data 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 or loop controlling activities do not manipulate workflow relevant data directly, but may refer to such data within conditional expressions within the join/split/loop control logic.

```
<!ELEMENT Activities (Activity*)>
<!ELEMENT Activity
           (Description?,
            Limit?,
            (Route | Implementation),
            Performer?,
            StartMode?,
            FinishMode?,
            Priority?,
            SimulationInformation?,
            Icon?,
            Documentation?,
            TransitionRestrictions?,
            ExtendedAttributes?)
>
<!ATTLIST Activity
           Id NMTOKEN #REQUIRED
           Name CDATA #IMPLIED
>
<!ELEMENT Route EMPTY>
```

```
<!ELEMENT Implementation
            (No | Tool+ | SubFlow | Loop)
>
<!ELEMENT No EMPTY>
<!ELEMENT Tool
            (ActualParameters?,
             Description?,
             ExtendedAttributes?)
>
<!ATTLIST Tool
            Id NMTOKEN #REQUIRED
            Type (APPLICATION | PROCEDURE) #IMPLIED
>
<!ELEMENT SubFlow (ActualParameters?)>
<!ATTLIST SubFlow
        Id CDATA #REQUIRED
            Execution (ASYNCHR | SYNCHR) #IMPLIED
>
<!ELEMENT Loop (Condition?)>
<!ATTLIST Loop
            Kind (WHILE | REPEAT_UNTIL) #IMPLIED
>
<!ELEMENT ActualParameter (#PCDATA)>
<!ELEMENT Performer (#PCDATA)>
<!ELEMENT StartMode (%Mode;)>
<!ELEMENT FinishMode (%Mode;)>
<!ELEMENT Automatic EMPTY>
<!ELEMENT Manual EMPTY>
<!ELEMENT Icon (#PCDATA)>
<!ELEMENT TransitionRestriction
            (InlineBlock?,
             Join?,
             Split?)
>
<!ELEMENT InlineBlock
            (BlockName?,
             Description?,
             Icon?,
             Documentation?,
             ExtendedAttributes?)
>
<!ATTLIST InlineBlock
            Begin NMTOKEN #REQUIRED
            End NMTOKEN #REQUIRED
>
<!ELEMENT BlockName (#PCDATA)>
<!ELEMENT Join EMPTY>
<!ATTLIST Join
            Type (AND | XOR) #IMPLIED
>
<!ELEMENT Split (TransitionRefs?)>
<!ATTLIST Split
            Type (AND | XOR) #IMPLIED
>
<!ELEMENT TransitionRef EMPTY>
<!ATTLIST TransitionRef
            Id NMTOKEN #REQUIRED
>
<!ELEMENT SimulationInformation
            (Cost,
             TimeEstimation)
>
```



```
<!ATTLIST SimulationInformation
            Instantiation (ONCE | MULTIPLE) #IMPLIED
>
<!ELEMENT Cost (#PCDATA)>
```

	Description
Id	Used to identify the workflow process activity.
Name	Text Used to identify the workflow process activity.
Description	Short textual description of the activity.
Route	A "dummy" Activity
Implementation	A "regular" Activity (details and further Attributes see below)
Transition Restrictions	Provides further restrictions and context-related semantics description of Transitions

Table 6-6: Attributes of Entity Workflow Process Activity

6.5.2.1. Route Activity

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 and its execution has no effect on workflow relevant data or application data.

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.

An activity that is not a route is a "regular" activity and has an implementation and further attributes.

	Description
Implementation	Mandatory if not a Route. Alternative implementations are “no”, “subflow” or “loop”
Performer	Link to entity workflow participant. May be an expression. Default: Any Participant.
Mode	Execution control attribute: Description of the degree of automation of triggering and terminating an Activity.
Start	Describes how the execution of an Activity is triggered.
Finish	Describes how the system operates at the end of the Activity.

	Description
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, ...).
Documentation	The address (e.g. path- and filename) for a help file or a description file of the activity.
Icon	Address (path- and filename) for an icon to represent the activity.
Simulation Information	Estimations for simulation of an Activity. No default.
Instantiation	Defines the capability of an activity to be activated: once or many times (multiple)
Time Estimation	Expected duration (summary of working time and waiting time) in units of DurationUnit.
Cost	Average cost.
Working Time	Average working time in units of DurationUnit.
Waiting Time	Average waiting time in units of DurationUnit.

Table 6-7: Implementation Attributes of Entity Workflow Process Activity

6.5.2.2. Execution Control Attributes

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

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 workflow engine, i.e. the engine proceeds with execution of the activity within the workflow 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.

	Description
Start Mode	Describes how the execution of an activity is triggered.
Automatic	Triggered implicitly by the system. Default.
Manual	Triggered explicitly by the end user.
Finish Mode	Describes how the system operates at the end of the activity.
Automatic	Implies an automatic return when the invoked application finishes control. Default.
Manual	The end user has to terminate the activity explicitly.

Table 6-8: Entity Workflow Process Activity - Automation Mode Attributes

6.5.2.3. Implementation Alternatives

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 workflow)
Application	Implementation is supported by (one or more) application(s)
Subflow	Implementation by another process
Loop	Implementation is by a loop of other activities, connected by specific loop transitions.

Table 6-9: Implementation Alternatives of Entity Workflow Process Activity

It is assumed that the execution of the Activity is atomic with respect to the data under control of the Workflow engine. That implies that in the case of a system crash, an abort, or a cancellation of the Activity, the Workflow Relevant Data and the workflow 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 workflow execution, subflows or applications under execution.

6.5.2.3.1.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 Workflow 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 Workflow, 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.)

6.5.2.3.2.Application

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

	Description
Tool	A tool identifier
Procedure	A procedure identifier

Table 6-10: Entity Workflow Process Activity - Implementation as Application

6.5.2.3.3.Subflow

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.

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.

	Description
ASYNCHR	Executed asynchronously.
|SYNCHR	Executed synchronously.

Table 6-11: Entity Workflow Process Activity - Implementation as Workflow

6.5.2.3.4.Loop

The Activity is refined as a loop (Loop Control Activity) and controls the execution of the Loop repetition (Loop body). The Loop body is connected with the Loop Control Activity by the corresponding Loop connecting Transitions.

A Loop body represents a "bracket" for parts of a Workflow definition that are connected and have only connection to the rest of the definition via the corresponding Loop connecting Transitions.

A LOOP allows expression of repetition ("cycles") in the network in a restricted way. The programming-language like control constructs "WHILE ... DO ..." and "REPEAT ... UNTIL" are supported.

	Description
While	Restricted to a WHILE Loop
Repeat Until	Restricted to a REPEAT - UNTIL Loop

Table 6-12: Loop Kinds of Entity Workflow Process Activity

The Implementation of a Loop is executed, possibly zero times or repeatedly, until the loop condition is evaluated to FALSE (WHILE Loop) or to TRUE (REPEAT_UNTIL Loop), respectively. For the WHILE Loop the test of the condition is performed before (repeatedly) executing the Loop implementation, for the REPEAT_UNTIL Loop afterwards.

6.5.2.4. 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 Workflow Participant. Default: Any Participant.

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

The question whether the expression evaluation results in an empty set of performers or a non unique performer is to be handled by the 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 to 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 workflow 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 workflow engine and the resource repository or organizational model in handling these situations is not defined.

6.5.2.5. Simulation Attribute Instantiation

The Instantiation Attribute 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).

	Description
Once	The Activity can only be instantiated once. Default.
Multiple	The Activity can be instantiated multiple times.

Table 6-13: Entity Workflow Process Activity - Instantiation Attribute

6.5.2.6. Transition Restrictions

	Description
Inline Block	The Activity is first or last Activity of an Inline Block. (details see below).
Inline Block Begin	The first Activity of an Inline Block. The identifier is that of the block
Inline Block End	The last Activity of an Inline Block.
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 6-14: Transition Restriction Attributes of Entity Workflow Process Activity

6.5.2.6.1.Inline Block

An Inline Block represents a "bracket" for parts of a Workflow definition that are connected and connect to the rest of the definition via the activities having `begin` and the corresponding `end` attributes with the same block `id`. This provides a "light-weight" alternative implementation to the use of a subflow (in which the block activities would be declared as a separate process definition). An inline block identifier is referenced in the same way as an activity identifier and may have similar join (prior) and split (post) processing.

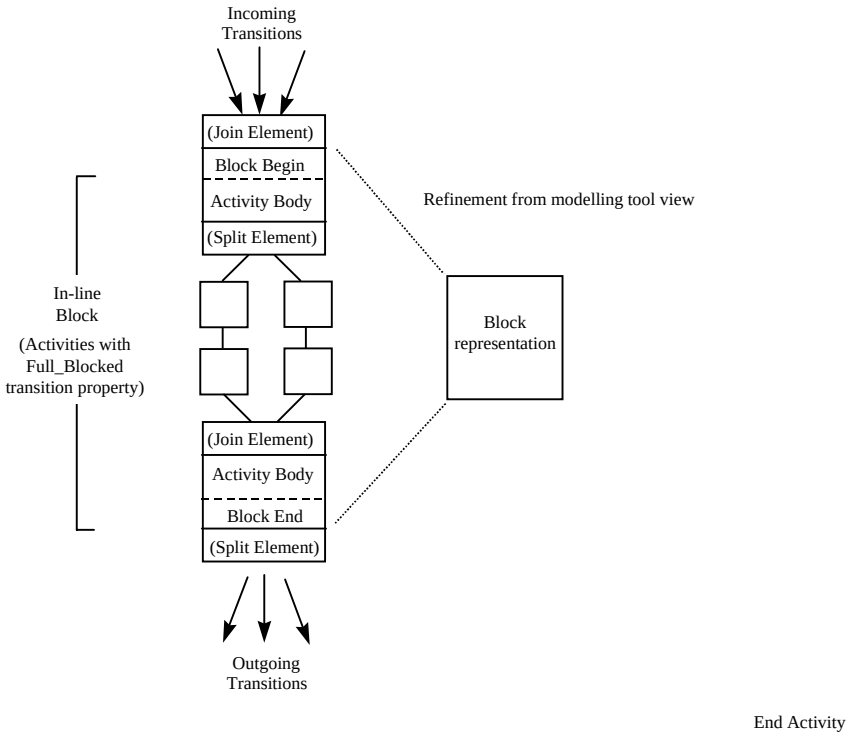

*Figure 6-4: **Inline Block Structure***

Activities within the inline block must follow the restrictions on transition usage as specified below (essentially not to include transitions to activities outside the block). An inline block operates with the same entity instances as the surrounding process definition and therefore does not include any workflow data copy/restore semantics or subflow audit data as would be associated with subflow invocation.

	Description
Block Name	Text Used to identify the block.
Description	Short textual description of the block.
Icon	Address (path- and filename) for an icon to represent the block. Allows graphical shrinking of the block body to an Icon.
Documentation	Any arbitrary textual description.

Table 6-15: Inline Block Attributes of Entity Workflow Process Activity

6.5.2.6.2.Join

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

	Description
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).
Xor	Join for alternative threads: No synchronisation is required..

Table 6-16: *JOIN alternatives of Entity Activity*

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.

6.5.2.6.3.Split

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

	Description
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.
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. If an unconditional Transition is evaluated or transition with condition OTHERWISE this ends the list evaluation.

Table 6-17: SPLIT alternatives of Entity Activity

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 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 same procedure is performed for the Transition with OTHERWISE (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).

6.5.2.7. 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 (excluding the Transitions connecting a Loop Activity) form an acyclic graph (or set of disjunct acyclic graphs). For cycles only a Loop Implementation of an Activity may be used.
- FULL-BLOCKED
  - For each join (or respectively split) there is exactly one corresponding split (or respectively join) of the same kind, and the Activity of the split and the corresponding join are also the pairing begin and end activities of an InlineBlock. In an And split no conditions are permitted; in an 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).

6.5.3. Transition Information

The Transition Information describes possible transitions between activities and the conditions that enable or disable them (the transitions) during workflow execution.

A process definition is seen as a network of edges between the Activity nodes (i.e. as a workflow 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).

For the identifiers and names defined in the Transition information the scope is the surrounding Workflow Process Definition (chapter 6.5.1).

Two Transition are distinguished, "regular" and Loop-connecting Transitions.

- For "regular" Transitions (without using the keyword loop) it is possible to define or synchronize multiple (concurrent or alternative) control threads (split, join) and sequences of Transitions between Activities (cascading Transitions/conditions) and blocking restrictions.
- Loop-connecting Transitions (using the keyword loop) allow the expression of cycles in the transition network. They connect the body of a Loop with the Loop Activity that is implemented by this body (see chapter 6.5.2.3). For all Transitions a from part and a to part are mandatory. Loop conditions are expressed in the loop Activity, not as Transition conditions.

```
<!ELEMENT Transition
            (Condition?,
             Description?,
             ExtendedAttributes?)
>
<!ATTLIST Transition
```

```
            Id NMTOKEN #REQUIRED
            From NMTOKEN #REQUIRED
            To NMTOKEN #REQUIRED
            Loop (NOLOOP | FROMLOOP | TOLOOP) #IMPLIED
            Name CDATA #IMPLIED
>
<!ELEMENT Condition (#PCDATA | XPression)*>
<!ATTLIST Condition
            Type (CONDITION | OTHERWISE) #IMPLIED
>
<!ELEMENT Xpression ANY>
```

	Description
Id	Used to identify the Transition.
Name	Text used to identify the Transition.
Description	Short textual description of the Transition.
No Loop	A "regular" Transition
From	Determines the FROM source of a Transition. (Activity Identifier)
To	Determines the TO target of a Transition (Activity Identifier)
Condition	A Transition condition expression based on workflow relevant data. (E.g. 'Contract' = 'SMALL' OR 'Contract' <$20,000). Default: TRUE
From Loop	Determines the FROM source of a Loop Connection Begin Transition. (Activity Identifier)
To	Determines the TO target of a Loop Connection Begin Transition (Activity Identifier)
From	Determines the FROM source of a Loop Connection End Transition. (Activity Identifier)
To Loop	Determines the TO target of a Loop Connection End Transition (Activity Identifier)

Table 6-18: Attributes of Entity Transition

6.5.4. Workflow Application Declaration

Workflow application declaration is a list of all applications or tools required and invoked by the workflow 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).

```
<!ELEMENT Applications (Application*)>
<!ELEMENT Application
            (Description?,
             FormalParameters?,
             ExtendedAttributes?)
>
<!ATTLIST Application
            Id NMTOKEN #REQUIRED
```



```
                Name CDATA #IMPLIED
        >
```

	Description
Id	Used to identify the workflow application definition
Name	Text used to identify an application (may be interpreted as a generic name of the tool).
Description	Short textual description of the application.
Formal Parameters	Parameters that are interchanged with the application via the invocation interface.

Table 6-19: Attributes of Entity Workflow Application

6.5.4.1. Invocation Parameters

A Workflow Application declaration may have parameter definitions for the (invocation) parameters as described in chapter 6.4.2. and also used within other entities.

The **parameter passing semantics** for invocation is described in chapter 6.4.2.

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.

6.5.5. Workflow Relevant Data

Workflow relevant data represent the variables of a workflow 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 workflow application data, which is data managed or accessed wholly by the invoked applications and which is not accessible to the workflow management system. The workflow relevant data list defines all data objects, which are required by the workflow process. The attribute `DataType` explicitly specifies all information needed for a workflow management system to define an appropriate data object for storing data, which is to be handled by an active instance of the workflow process.

Workflow relevant data can be defined in a workflow 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.

```
<!ELEMENT DataFields (DataField*)>
<!ELEMENT DataField
            (DataType,
             InitialValue?,
             Length?,
             Description?,
             ExtendedAttributes?)
>
<!ATTLIST DataField
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
            IsArray (TRUE | FALSE) "FALSE"
>
<!ELEMENT DataTypes (DataType*)>
<!ELEMENT DataType (%Type;)>
```



```
<!ELEMENT InitialValue (#PCDATA)>
<!ELEMENT Length (#PCDATA)>
```

	Description
Id	Used to identify the workflow relevant data.
Data Type	Datatype.
Name	Text used to identify the workflow relevant data
Length	The length of the data
Description	Short textual description of the data defined.
Initial Value	Pre-assignment of data for run time.

Table 6-20: Attributes of Entity Workflow Relevant Data

6.5.6. Workflow Participant Specification

The Workflow 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. 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 workflow participant entity declaration in a minimal resource repository or organizational model is the surrounding entity (Workflow 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.

```
<!ELEMENT Participants (Participant*)>
<!ELEMENT Participant (ParticipantType, Description?, ExtendedAttributes?)>
<!ATTLIST Participant
           Id NMTOKEN #REQUIRED
           Name CDATA #IMPLIED
>
<!ELEMENT ParticipantType EMPTY>
<!ATTLIST ParticipantType
           Type
                (RESOURCE_SET
                | RESOURCE
                | ROLE
                | ORGANIZATIONAL_UNIT
                | HUMAN,
                | SYSTEM) #REQUIRED
>
```

	Description
Id	Used to identify the workflow participant definition.
Name	Text used to identify a performer
Description	Short textual description of a workflow participant.
Participant Type	Definition of the type of workflow participant entity.

Table 6-21: Attributes of Entity Workflow Participant Declaration

6.5.6.1. 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.

	Description
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 6-22: Types of Workflow Participants

6.6. Package

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). The Package meta-model contains the following entity types:

- Workflow Process Definition
- Workflow Participant Specification
- Workflow Application Declaration
- Workflow Relevant Data

as described below.

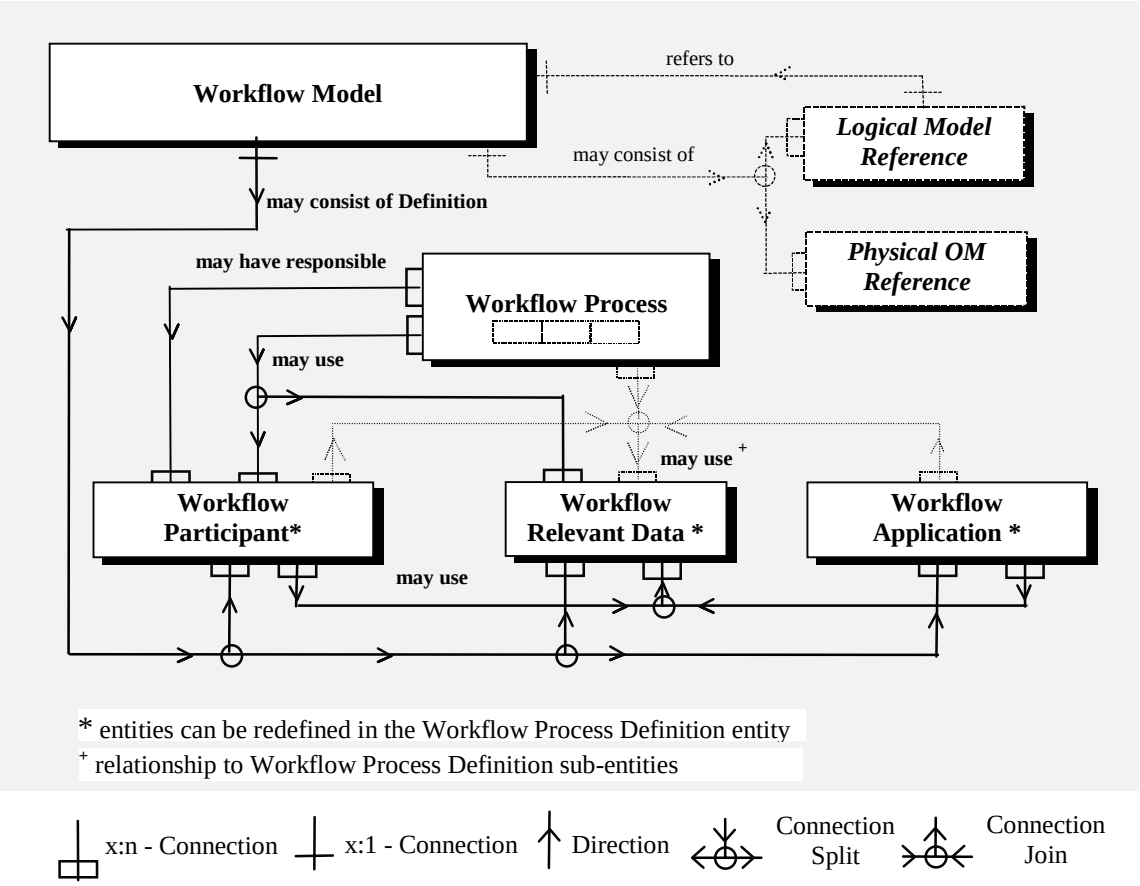

Figure 6-5: Package Definition Meta Model

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 workflow relevant data, 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.

6.6.1. Process Repository

The process definition import/export interface is assumed to operate to/from a workflow definition repository of some form associated with the 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 workflow 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).

```
<!ELEMENT Package
            (PackageHeader,
             RedefinableHeader?,
             ConformanceClass?,
             ExternalPackages?,
             TypeDeclarations?,
             Participants?,
             Applications?,
             DataFields?,
             WorkflowProcesses?,
             ExtendedAttributes?)
>
<!ATTLIST Package
            xmlns:xpdl CDATA #FIXED "http://www.wfmc.org/standards/docs/xpdl"
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
<!ELEMENT PackageHeader
            (XPDLVersion,
             Vendor,
             Created,
             Description?,
             Documentation?,
             PriorityUnit?,
             CostUnit?)
>
<!ELEMENT ExternalPackages (ExternalPackage*)>
<!ELEMENT ConformanceClass EMPTY>
<!ATTLIST ConformanceClass
        GraphConformance
                 (FULL_BLOCKED
                 | LOOP_BLOCKED
                 | NON_BLOCKED) #IMPLIED
>
<!ELEMENT ExternalPackage
            (ExternalPackage,
             ExtendedAttributes?)
>
<!ELEMENT Package EMPTY>
<!ATTLIST ExternalPackage
            href CDATA #IMPLIED
>
```

	Description
xmlns:xpdl	Namespace for XPDL.
Id	Used to identify the package.
Name	Text. Used to identify the package.
Package Description	Short textual description of the Package.
XPDL Version	Version of this specification. The current value, for this specification, is “0.02”.

	Description
Vendor	Defines the origin of this model definition and contains vendor's name, vendor's product name and product's release number.
Created	Creation date of Package Definition.
Version	Version of this Package Definition.
Author	Name of the author of this package definition.
Code page	The codepage used for the text parts
Country key	nnn as country number
Responsible	Workflow participant, who is responsible for this workflow process; the supervisor during run time Link to entity workflow participant. Workflow 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.
Documentation	Operating System specific path- and filename of help file/description file.
Cost Units	Units used in Simulation Data (Usually expressed in terms of a currency)
Conformance Class	Describes the Conformance Class to which the definitions in this model are restricted.
External. Package	List of references to external packages

Table 6-23: Attributes of Entity Package

6.6.2. Conformance Class

The following conformance classes are supported: The specified class applies to all the contained process definitions, unless it is re-defined locally at the process definition level.

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

Table 6-24: Conformance Classes of Entity Package

Further details are described in chapter 6.5.2.7.

6.6.3. External Package

External package allows reference to definitions contained within other packages.

	Description
Id	A Model Identifier. Logical reference to a Model

Table 6-25: External Model Reference Attributes of Entity Package

6.6.3.1. 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) Workflow relevant Data

Workflow process relevant data 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) Workflow participants and applications

- Workflow participants and applications have a scope and visibility equivalent to extended attributes. All referenced workflow relevant data 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 workflow 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.

7. XPDL Grammar

7.1. 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 workflow processes as well as all the associated tools and workflow participants, although it is not required. Also it is possible to define just parts of one process definition or common parts of several processes within one package (e.g. a workflow participant list or a workflow application list).

```
<!ELEMENT Package
            (PackageHeader,
             RedefinableHeader?,
             ConformanceClass?,
             ExternalPackages?,
             TypeDeclarations?,
             Participants?,
             Applications?,
             DataFields?,
             WorkflowProcesses?,
             ExtendedAttributes?)
>

<!ATTLIST Package
            xmlns:xpdl CDATA #FIXED "http://www.wfmc.org/standards/docs/xpdl"
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
```

7.1.1. Package definition Header

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

```
<!ELEMENT PackageHeader
                      (XPDLVersion,
                       Vendor,
                       Created,
                       Description?,
                       Documentation?,
                       PriorityUnit?,
                       CostUnit?)
>

<!ELEMENT XPDLVersion (#PCDATA)>

<!ELEMENT Vendor (#PCDATA)>

<!ELEMENT Created (#PCDATA)>

<!ELEMENT Description (#PCDATA)>

<!ELEMENT Documentation (#PCDATA)>
```



```
<!ELEMENT PriorityUnit (#PCDATA)>

<!ELEMENT CostUnit (#PCDATA)>
```

7.1.2. Redefinable Header

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

```
<!ELEMENT RedefinableHeader
                      (Author?,
                       Version?,
                       Codepage?,
                       Countrykey?,
                       Responsibles?)
>

<!ATTLIST RedefinableHeader
        PublicationStatus
                            (UNDER_REVISION
                            | RELEASED
                            | UNDER_TEST) #IMPLIED
>

<!ELEMENT Author (#PCDATA)>

<!ELEMENT Version (#PCDATA)>

<!ELEMENT Codepage (#PCDATA)>

<!ELEMENT Countrykey (#PCDATA)>

<!ELEMENT Responsibles (Responsible*)>

<!ELEMENT Responsible (#PCDATA)>
```

7.1.3. Conformance Class Declaration

The conformance class declaration allows description of the conformance class to which the definitions in this model definition are restricted.

```
<!ELEMENT ConformanceClass EMPTY>

<!ATTLIST ConformanceClass
        GraphConformance
                (FULL_BLOCKED
                | LOOP_BLOCKED
                | NON_BLOCKED) #IMPLIED
>
```

7.1.4. External Package Reference

External package reference allows referencing definitions in another Package definition or in other systems providing an

Interface to the Workflow Management system (e.g. a legacy Organisation Description Management Tool).

```
<!ELEMENT ExternalPackages (ExternalPackage*)>
<!ELEMENT ExternalPackage
            (ExternalPackage,
             ExtendedAttributes?)
>
<!ATTLIST ExternalPackage
            href CDATA #IMPLIED
>
```

7.2. Workflow Process Definition

The workflow process definition defines the elements that make up a workflow.

```
<!ELEMENT WorkflowProcesses (WorkflowProcess*)>
<!ELEMENT WorkflowProcess
            (ProcessHeader,
             RedefinableHeader?,
             FormalParameters?,
             (%Type;)*,
             DataFields?,
             Participants?,
             Applications?,
             Activities?,
             Transitions?,
             ExtendedAttributes?)
>
<!ATTLIST WorkflowProcess
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
```

7.2.1. Workflow Process Definition Header

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

```
<!ELEMENT ProcessHeader
            (Created?,
             Description?,
             Priority?,
             Limit?,
             ValidFrom?,
             ValidTo?,
             TimeEstimation?)
>
<!ATTLIST ProcessHeader
            DurationUnit (Y | M | D | h | m | s) #IMPLIED
>
<!ELEMENT Priority (#PCDATA)>
<!ELEMENT Limit (#PCDATA)>
<!ELEMENT TimeEstimation
            (WaitingTime?,
             WorkingTime?,
```



```
            Duration?)
>
<!ELEMENT WaitingTime (#PCDATA)>
<!ELEMENT WorkingTime (#PCDATA)>
<!ELEMENT Duration (#PCDATA)>
<!ELEMENT ValidFrom (#PCDATA)>
<!ELEMENT ValidTo (#PCDATA)>
```

7.3. Workflow Process Activity

The following rule will be used to describe all necessary activities. In addition it allows expression of further condition evaluation and structure restrictions of Transitions.

```
<!ELEMENT Activities (Activity*)>
<!ELEMENT Activity
           (Description?,
            Limit?,
            (Route | Implementation),
            Performer?,
            StartMode?,
            FinishMode?,
            Priority?,
            SimulationInformation?,
            Icon?,
            Documentation?,
            TransitionRestrictions?,
            ExtendedAttributes?)
>
<!ATTLIST Activity
           Id NMTOKEN #REQUIRED
           Name CDATA #IMPLIED
>
<!ELEMENT Route EMPTY>
<!ELEMENT Implementation
           (No | Tool+ | SubFlow | Loop)
>
<!ELEMENT No EMPTY>
<!ELEMENT Tool
           (ActualParameters?,
            Description?,
            ExtendedAttributes?)
>
<!ATTLIST Tool
           Id NMTOKEN #REQUIRED
           Type (APPLICATION | PROCEDURE) #IMPLIED
>
<!ELEMENT SubFlow (ActualParameters?)>
<!ATTLIST SubFlow
        Id CDATA #REQUIRED
           Execution (ASYNCHR | SYNCHR) #IMPLIED
>
<!ELEMENT Loop (Condition?)>
<!ATTLIST Loop
           Kind (WHILE | REPEAT_UNTIL) #IMPLIED
>

<!ELEMENT Performer (#PCDATA)>
```



```
<!ELEMENT StartMode (%Mode;)>
<!ELEMENT FinishMode (%Mode;)>
<!ENTITY % Mode
             "Automatic
            | Manual"
>
<!ELEMENT Automatic EMPTY>
<!ELEMENT Manual EMPTY>
<!ELEMENT Icon (#PCDATA)>
<!ELEMENT TransitionRestrictions (TransitionRestriction*)>
<!ELEMENT TransitionRestriction
            (InlineBlock?,
             Join?,
             Split?)
>
<!ELEMENT InlineBlock
            (BlockName?,
             Description?,
             Icon?,
             Documentation?,
             ExtendedAttributes?)
>
<!ATTLIST InlineBlock
            Begin NMTOKEN #REQUIRED
            End NMTOKEN #REQUIRED
>
<!ELEMENT BlockName (#PCDATA)>
<!ELEMENT Join EMPTY>
<!ATTLIST Join
            Type (AND | XOR) #IMPLIED
>
<!ELEMENT Split (TransitionRefs?)>
<!ATTLIST Split
            Type (AND | XOR) #IMPLIED
>

<!ELEMENT TransitionRefs (TransitionRef*)>
<!ELEMENT TransitionRef EMPTY>
<!ATTLIST TransitionRef
            Id NMTOKEN #REQUIRED
>
<!ELEMENT SimulationInformation
            (Cost,
             TimeEstimation)
>
<!ATTLIST SimulationInformation
            Instantiation (ONCE | MULTIPLE) #IMPLIED
>
<!ELEMENT Cost (#PCDATA)>
```

7.3.1. Parameters

A parameter in the context of the XPDL is defined by workflow process relevant data; a parameter list is the aggregation of parameters in a list. Parameters are used to be passed along between process and subflow, between (sub)process and application etc..

7.3.1.1. Generic formal parameter definition

Formal parameters are part of the attribute sequence in the XPDL definition of Workflow Process Definition and Workflow

Application: We distinguish call input parameters and call output parameters.

The generic form of the part in the attribute sequence is:

```
<!ELEMENT FormalParameters (FormalParameter*)>
<!ELEMENT FormalParameter (DataType, Description?)>
<!ATTLIST FormalParameter
            Id NMTOKEN #REQUIRED
            Index NMTOKEN #IMPLIED
            Mode (IN | OUT | INOUT) "IN"
>
```

7.3.1.2. Formal-actual parameter mapping

The mapping of actual to formal parameters during invocation of metamodel entities is defined by a parameter map list:

```
<!ELEMENT ActualParameters (ActualParameter*)>
<!ELEMENT ActualParameter (#PCDATA)>
```

The actual parameter list maps the actual to the formal parameter in sequence, i.e. the first actual maps to the first formal, the second actual maps to the second formal etc. The semantics is defined for formal and actual parameter lists holding the same number of parameters. Alternatively an extended parameter mapping semantics may be specified vendor specific (e.g. setting to zero for missing parameters, ignoring surplus operators etc.). The mechanism for that extension is not provided here.

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

7.4. Transition Information

The Transition Information describes the possible transitions between the activities and the conditions under which they are taken into account. Further control and structure restrictions may be expressed in the Activity definition.

```
<!ELEMENT Transitions (Transition*)>
<!ELEMENT Transition
            (Condition?,
             Description?,
             ExtendedAttributes?)
>
<!ATTLIST Transition
            Id NMTOKEN #REQUIRED
            From NMTOKEN #REQUIRED
            To NMTOKEN #REQUIRED
            Loop (NOLOOP | FROMLOOP | TOLOOP) #IMPLIED
            Name CDATA #IMPLIED
>
<!ELEMENT Condition (#PCDATA | XPression)*>
<!ATTLIST Condition
            Type (CONDITION | OTHERWISE) #IMPLIED
>
<!ELEMENT Xpression ANY>
```


7.5. Workflow Application Declaration

Workflow application declaration is a list of all applications or tools required and invoked by the workflow processes defined within the XPDL file. A workflow application declaration may have parameter definitions used for the invocation parameters and also used within other entities.

```
<!ELEMENT Applications (Application*)>
<!ELEMENT Application
            (Description?,
             FormalParameters?,
             ExtendedAttributes?)
>
<!ATTLIST Application
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
```

7.6. Workflow Relevant Data

Workflow relevant data represent the variables of a workflow process or package definition.

```
<!ELEMENT DataFields (DataField*)>
<!ELEMENT DataField
            (DataType,
             InitialValue?,
             Length?,
             Description?,
             ExtendedAttributes?)
>
<!ATTLIST DataField
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
            IsArray (TRUE | FALSE) "FALSE"
>
<!ELEMENT DataTypes (DataType*)>
<!ELEMENT DataType (%Type;)>
<!ELEMENT InitialValue (#PCDATA)>
<!ELEMENT Length (#PCDATA)>
```

7.7. Data Types

```
<!ELEMENT TypeDeclarations (TypeDeclaration*)>
<!ELEMENT TypeDeclaration
            ((%Type;),
             Description?,
             ExtendedAttributes?)
>
<!ATTLIST TypeDeclaration
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
```



```
>
<!ENTITY % Type
              "%ComplexType;
            | BasicType
            | PlainType
            | DeclaredType"
>
```

7.7.1. Basic Data Types

```
<!ELEMENT BasicType EMPTY>
<!ATTLIST BasicType
        Type
                (STRING
                | FLOAT
                | INTEGER
                | REFERENCE
                | DATETIME) #REQUIRED
>
```

7.7.2. Plain Data Types

Plain data are basics, Boolean and performers:

```
<!ELEMENT PlainType EMPTY>
<!ATTLIST PlainType
            Type
                 (BOOLEAN
                 | UNIT
                 | PERFORMER) #REQUIRED
>
```

A data instance of a Boolean type is one having one of the values TRUE or FALSE. Although the internal representation of these values is not defined in the XPDL (usually 1 and 0), they match with the constant representations TRUE and FALSE, respectively.

A data instance of a performer type is one having a value of a declared workflow participant.

7.7.3. Complex Data Types

Complex data permits definition of arrays, records, enumerations, lists or a superset of them within extended attributes or workflow process relevant data, and provides means to access the data defined.

```
<!ENTITY % ComplexType
              "RecordType
            | UnionType
            | EnumerationType
            | ArrayType
            | ListType"
>
<!ELEMENT RecordType (Member+)>
<!ELEMENT UnionType (Member+)>
<!ELEMENT EnumerationType (EnumerationValue+)>
<!ELEMENT EnumerationValue EMPTY>
<!ATTLIST EnumerationValue
            Name NMTOKEN #REQUIRED
>
```



```
<!ELEMENT Member (%Type;)>
<!ELEMENT ArrayType (%Type;)>
<!ATTLIST ArrayType
            LowerIndex NMTOKEN #REQUIRED
            UpperIndex NMTOKEN #REQUIRED
>
<!ELEMENT ListType (%Type;)>
```

7.7.4. Declared Data Types

```
<!ELEMENT DeclaredType EMPTY>
<!ATTLIST DeclaredType
            Id IDREF #REQUIRED
>
```

7.8. Extended Attributes

In addition to the predefined entities and attributes, an extended attribute element is included to allow every vendor of a workflow process definition tool to specify their own set of attributes for the main objects of the meta-model (assuming it is not covered by those already defined).

```
<!ELEMENT ExtendedAttributes (ExtendedAttribute*)>
<!ELEMENT ExtendedAttribute ANY>
<!ATTLIST ExtendedAttribute
        Name NMTOKEN #REQUIRED
            Value CDATA #IMPLIED
>
```

7.9. Workflow Participants

The Workflow Participants are those elements of a resource repository or organisational model that are either acting parties in a workflow process or responsible for it. This definition is an abstraction level between the real performer and the activity, which has to be performed. It may refer to an external organisational model or resource repository. Actors may be defined by a membership in an organisational unit, by a function, role or competence, or by relations to actors of already performed activities, etc.

```
<!ELEMENT Participants (Participant*)>
<!ELEMENT Participant (ParticipantType, Description?, ExtendedAttributes?)>
<!ATTLIST Participant
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
<!ELEMENT ParticipantType EMPTY>
<!ATTLIST ParticipantType
            Type
                  (RESOURCE_SET
                  | RESOURCE
                  | ROLE
                  | ORGANIZATIONAL_UNIT
                  | HUMAN
                  | SYSTEM) #REQUIRED
>
```


8. XPDL DTD

This section presents the full DTD for XPDL.

```
<?xml version = '1.0' encoding='us-ascii'?>
<!ENTITY % ComplexType
              "RecordType
            | UnionType
            | EnumerationType
            | ArrayType
            | ListType"
>

<!ENTITY % Type
              "%ComplexType;
            | BasicType
            | PlainType
            | DeclaredType"
>
<!ENTITY % Mode
              "Automatic
            | Manual"
>
<!ELEMENT Package
            (PackageHeader,
             RedefinableHeader?,
             ConformanceClass?,
             ExternalPackages?,
             TypeDeclarations?,
             Participants?,
             Applications?,
             DataFields?,
             WorkflowProcesses?,
             ExtendedAttributes?)
>
<!ATTLIST Package
            xmlns:xpdl CDATA #FIXED "http://www.wfmc.org/standards/docs/xpdl"
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
<!ELEMENT PackageHeader
            (XPDLVersion,
             Vendor,
             Created,
             Description?,
             Documentation?,
             PriorityUnit?,
             CostUnit?)
>
<!ELEMENT ExternalPackages (ExternalPackage*)>
<!ELEMENT TypeDeclarations (TypeDeclaration*)>
<!ELEMENT Participants (Participant*)>
<!ELEMENT Applications (Application*)>
<!ELEMENT DataFields (DataField*)>
<!ELEMENT WorkflowProcesses (WorkflowProcess*)>
<!ELEMENT ExtendedAttributes (ExtendedAttribute*)>
<!ELEMENT Responsibles (Responsible*)>
<!ELEMENT FormalParameters (FormalParameter*)>
```



```
<!ELEMENT Activities (Activity*)>
<!ELEMENT Transitions (Transition*)>
<!ELEMENT TransitionRestrictions (TransitionRestriction*)>
<!ELEMENT ActualParameters (ActualParameter*)>
<!ELEMENT TransitionRefs (TransitionRef*)>
<!ELEMENT Participant (ParticipantType, Description?, ExtendedAttributes?)>
<!ATTLIST Participant
           Id NMTOKEN #REQUIRED
           Name CDATA #IMPLIED
>
<!ELEMENT ParticipantType EMPTY>
<!ATTLIST ParticipantType
           Type
                (RESOURCE_SET
                | RESOURCE
                | ROLE
                | ORGANIZATIONAL_UNIT
                | HUMAN
                | SYSTEM ) #REQUIRED
>
<!ELEMENT XPDLVersion (#PCDATA)>
<!ELEMENT Vendor (#PCDATA)>
<!ELEMENT Created (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT Documentation (#PCDATA)>
<!ELEMENT PriorityUnit (#PCDATA)>
<!ELEMENT CostUnit (#PCDATA)>
<!ELEMENT ExtendedAttribute ANY>
<!ATTLIST ExtendedAttribute
        Name NMTOKEN #REQUIRED
           Value CDATA #IMPLIED
>
<!ELEMENT RedefinableHeader
           (Author?,
            Version?,
            Codepage?,
            Countrykey?,
            Responsibles?)
>
<!ATTLIST RedefinableHeader
        PublicationStatus
                (UNDER_REVISION
                | RELEASED
                | UNDER_TEST) #IMPLIED
>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Version (#PCDATA)>
<!ELEMENT Codepage (#PCDATA)>
<!ELEMENT Countrykey (#PCDATA)>
<!ELEMENT Responsible (#PCDATA)>
<!ELEMENT ConformanceClass EMPTY>
<!ATTLIST ConformanceClass
        GraphConformance
                (FULL_BLOCKED
                | LOOP_BLOCKED
                | NON_BLOCKED) #IMPLIED
>
<!ELEMENT ExternalPackage
           (ExternalPackage,
            ExtendedAttributes?)
>
<!ATTLIST ExternalPackage
```

```
            href CDATA #IMPLIED
>
<!ELEMENT TypeDeclaration
            ((%Type;),
             Description?,
             ExtendedAttributes?)
>
<!ATTLIST TypeDeclaration
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
<!ELEMENT PlainType EMPTY>
<!ATTLIST PlainType
            Type
                 (BOOLEAN
                 | UNIT
                 | PERFORMER) #REQUIRED
>
<!ELEMENT BasicType EMPTY>
<!ATTLIST BasicType
        Type
                 (STRING
                 | FLOAT
                 | INTEGER
                 | REFERENCE
                 | DATETIME) #REQUIRED
>
<!ELEMENT RecordType (Member+)>
<!ELEMENT UnionType (Member+)>
<!ELEMENT EnumerationType (EnumerationValue+)>
<!ELEMENT EnumerationValue EMPTY>
<!ATTLIST EnumerationValue
            Name NMTOKEN #REQUIRED
>
<!ELEMENT Member (%Type;)>
<!ELEMENT ArrayType (%Type;)>
<!ATTLIST ArrayType
            LowerIndex NMTOKEN #REQUIRED
            UpperIndex NMTOKEN #REQUIRED
>
<!ELEMENT ListType (%Type;)>
<!ELEMENT DeclaredType EMPTY>
<!ATTLIST DeclaredType
            Id IDREF #REQUIRED
>
<!ELEMENT WorkflowProcess
            (ProcessHeader,
             RedefinableHeader?,
             FormalParameters?,
             (%Type;)*,
             DataFields?,
             Participants?,
             Applications?,
             Activities?,
             Transitions?,
             ExtendedAttributes?)
>
<!ATTLIST WorkflowProcess
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
<!ELEMENT ProcessHeader
```



```
              (Created?,
               Description?,
               Priority?,
               Limit?,
               ValidFrom?,
               ValidTo?,
               TimeEstimation?)
>
<!ATTLIST ProcessHeader
            DurationUnit (Y | M | D | h | m | s) #IMPLIED
>
<!ELEMENT Priority (#PCDATA)>
<!ELEMENT Limit (#PCDATA)>
<!ELEMENT TimeEstimation
            (WaitingTime?,
             WorkingTime?,
             Duration?)
>
<!ELEMENT WaitingTime (#PCDATA)>
<!ELEMENT WorkingTime (#PCDATA)>
<!ELEMENT Duration (#PCDATA)>
<!ELEMENT ValidFrom (#PCDATA)>
<!ELEMENT ValidTo (#PCDATA)>
<!ELEMENT DataField
            (DataType,
             InitialValue?,
             Length?,
             Description?,
             ExtendedAttributes?)
>
<!ATTLIST DataField
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
            IsArray (TRUE | FALSE) "FALSE"
>
<!ELEMENT DataTypes (DataType*)>
<!ELEMENT DataType (%Type;)>
<!ELEMENT InitialValue (#PCDATA)>
<!ELEMENT Length (#PCDATA)>
<!ELEMENT Application
            (Description?,
             FormalParameters?,
             ExtendedAttributes?)
>
<!ATTLIST Application
            Id NMTOKEN #REQUIRED
            Name CDATA #IMPLIED
>
<!ELEMENT Activity
            (Description?,
             Limit?,
             (Route | Implementation),
             Performer?,
             StartMode?,
             FinishMode?,
             Priority?,
             SimulationInformation?,
             Icon?,
             Documentation?,
             TransitionRestrictions?,
             ExtendedAttributes?)
>
```



```
<!ATTLIST Activity
           Id NMTOKEN #REQUIRED
           Name CDATA #IMPLIED
>
<!ELEMENT Route EMPTY>
<!ELEMENT Implementation
           (No | Tool+ | SubFlow | Loop)
>
<!ELEMENT No EMPTY>
<!ELEMENT Tool
           (ActualParameters?,
            Description?,
            ExtendedAttributes?)
>
<!ATTLIST Tool
           Id NMTOKEN #REQUIRED
           Type (APPLICATION | PROCEDURE) #IMPLIED
>
<!ELEMENT SubFlow (ActualParameters?)>
<!ATTLIST SubFlow
        Id CDATA #REQUIRED
           Execution (ASYNCHR | SYNCHR) #IMPLIED
>
<!ELEMENT Loop (Condition?)>
<!ATTLIST Loop
           Kind (WHILE | REPEAT_UNTIL) #IMPLIED
>
<!ELEMENT ActualParameter (#PCDATA)>
<!ELEMENT Performer (#PCDATA)>
<!ELEMENT StartMode (%Mode;)>
<!ELEMENT FinishMode (%Mode;)>
<!ELEMENT Automatic EMPTY>
<!ELEMENT Manual EMPTY>
<!ELEMENT Icon (#PCDATA)>
<!ELEMENT TransitionRestriction
           (InlineBlock?,
            Join?,
            Split?)
>
<!ELEMENT InlineBlock
           (BlockName?,
            Description?,
            Icon?,
            Documentation?,
            ExtendedAttributes?)
>
<!ATTLIST InlineBlock
           Begin NMTOKEN #REQUIRED
           End NMTOKEN #REQUIRED
>
<!ELEMENT BlockName (#PCDATA)>
<!ELEMENT Join EMPTY>
<!ATTLIST Join
           Type (AND | XOR) #IMPLIED
>
<!ELEMENT Split (TransitionRefs?)>
<!ATTLIST Split
           Type (AND | XOR) #IMPLIED
>
<!ELEMENT TransitionRef EMPTY>
<!ATTLIST TransitionRef
           Id NMTOKEN #REQUIRED
```



```
>
<!ELEMENT SimulationInformation
            (Cost,
             TimeEstimation)
>
<!ATTLIST SimulationInformation
            Instantiation (ONCE | MULTIPLE) #IMPLIED
>
<!ELEMENT Cost (#PCDATA)>
<!ELEMENT Transition
            (Condition?,
             Description?,
             ExtendedAttributes?)
>
<!ATTLIST Transition
            Id NMTOKEN #REQUIRED
            From NMTOKEN #REQUIRED
            To NMTOKEN #REQUIRED
            Loop (NOLOOP | FROMLOOP | TOLOOP) #IMPLIED
            Name CDATA #IMPLIED
>
<!ELEMENT Condition (#PCDATA | XPression)*>
<!ATTLIST Condition
            Type (CONDITION | OTHERWISE) #IMPLIED
>
<!ELEMENT Xpression ANY>
<!ELEMENT FormalParameter (DataType, Description?)>
<!ATTLIST FormalParameter
            Id NMTOKEN #REQUIRED
            Index NMTOKEN #IMPLIED
            Mode (IN | OUT | INOUT) "IN"
>
```


9. To Do List (Non-Normative)

This section includes comments and suggestions to improve XPDL. Some of the suggestions were received via email, and in those cases the email is reproduced in here.

AP Engines (Norin, Roberta [mailto:rnorin@APEngines.com]):

1. Standard XML syntax for expressions to be transition conditions
2. An activity element that contains data assignments expressed via an XML standard, similar to condition expressions.
3. Use of XML schema to define datatypes.
4. We have some very complex data types that we want to refer to but not necessarily define within the XPDL file. For example, we could just refer to a java class path or a URL. I am using the REFERENCE basic type with an extension but it seems like there should be a standard way of referring to something.
5. We have a number of activity types that we can specify using extended attributes but I wonder of some of them should be in the standard. The ones we will be using are:
    a. Timer
    b. Exception generator
    c. Exception branch -- a conditional branch activity that has one branch to take under normal circumstances and another to take if an exception is caught.
    d. Checkpoint
    e. E-Mail

Cape Vision:

All known Cape Vision requirements had been meet.

FileNET:

1. FileNET will like to see the ability to describe sub-maps and the ability to call sub-maps. Sub-maps are different than subflowes in that sub-maps are like blocks of activities within the same process.

Handysoft:

Handysoft presented their proposal for changes in the Australia meeting (Wollongong, 23 January 2001), they provided a slide presentation and a document describing their proposal.

1. Include the Organizational model in XPDL.
2. Change the way loops are defined. Add a loop attribute to transitions, and remove the loop attribute from the activity. This will provide a simpler more powerful loop construct.
3. New keywords including exceptions.

Silverglobe (Suhaina [mailto:suhainab@SILVERGLOBE.COM]):

1. one keyword i.e. 'Attachment Reference' which will identify if there are attachments attached to the process (for integration with imaging system).

Toshiba:

Toshiba have not presented any requirements, however they have done some prototyping and have a web page (http://www3.toshiba.co.jp/ccc/page/wpdlxml.html) with information about their approach.

Other suggestions for version one (New York meeting):

1. Provide a schema instead or in addition to the DTD
2. Simplify redundant explanation of the entities (sections 1, 6.2-6, 7, and 8)
3. Provide a single meta-model expressed in UML.
4. Examine context data and make a clear distinction between workflow relevant data and application data.
5. Work on participant types and its relationship with external resource repository and/or external organizational model.
6. The description should start with the workflow model/package and follow with the workflow process.
7. Review the usage of country key and codepage.

