The Workflow Management Coalition Specification

Workflow Management Coalition Workflow Standard - Interoperability Abstract Specification

Document Number WFMC-TC-1012

30 November 1999

Version 2.0b (Draft)

Workflow Management Coalition
2 Crown Walk
Winchester
SO22 5XE
United Kingdom
Tel: +44 1962 873401
Fax: +44 1962 868111
Email: wfmc@wfmc.org
web: http://www.wfmc.org

First printing September 1998

Table of Content

1 Change History

Version 2.0 – Editor: Mike Marin (mmarin@filenet.com)

- Further abstracted the specification to a higher level to cover distinct concrete bindings.

Version 1.1 – Editor: Mike Anderson (mja@process.icl.co.uk)

- Draft of version 2.0

Version 1.0 – Editor: Mike Anderson (mja@process.icl.co.uk)

- Initial Version

2 Purpose

This document is an abstract specification, which defines the functionality required to support interoperability between different workflow engines. Workflow product vendors should use this document to understand the principles of how interoperability between workflow engines are effected using the WfMC Standards. They should then refer to specific transport binding specifications for details of how conformant implementations must work.

This document does not present a standard that can be implemented, instead it presents a series of principles that must be followed to submit interoperability standards to the Workflow Management Coalition. Interoperability implementations cannot claim compliance with this specification, they can only claim compliance with a specific binding of this specification.

3 Audience

The intended audience is primarily vendor organizations who seek to prepare and submit binding specifications of the Interoperability Standard to the Workflow Management Coalition.

4 Scope

4.1 Scope of The Interoperability Specification

The Workflow Management Coalition Standard for Interoperability defines the mechanisms that workflow product vendors are required to implement in order that one workflow engine may make requests of another workflow engine to effect the:

- selection
- instantiation
- enactment

of known process definitions by that other engine. The requesting workflow engine should (optionally) also be able to receive back status information and the results of the enactment of the process definition. As far as possible, this is to be done in a way that is “transparent to the user”. This interface is intended for the use of vendor organizations not users. As a side effect of facilitating the above communications between workflow engines, there is a stated requirement that audit data be produced.

4.2 Workflow Engine Interoperability

In its earlier work on the topic of interoperability (see [ICL95] and [WfMC006]), the Workflow Management Coalition identified a number of different models of interoperability, and a number of different

levels, against which vendors of workflow products might measure their offerings. A synopsis of these is presented here to assist the understanding of readers who come to this document without first having had access to earlier documents.

4.2.1 Interoperability

The following terminology is taken from the Workflow Management Coalition Glossary [WfMC000].

Workflow Interoperability is described as:

> " the ability of two or more workflow engines to communicate and interoperate in order to coordinate and execute workflow process instances across those engines."

A *Workflow Engine* is described as:

> " A software service or "engine" that provides the run time execution environment for a workflow instance."

A *Workflow Process Instance* is defined to be:

> " ...an instance of a workflow process definition which includes the automated aspects of a process instance... created and managed by a Workflow Management System"

A *Workflow Management System* is defined to be:

> " A system that completely defines, manages and executes workflow processes through the execution of software whose order of execution is driven by a computer representation of the workflow process logic."
>
> " A Workflow Management System consists of one or more Workflow Enactment Services".
>
> " A Workflow Enactment Service consists of one or more Workflow Process Engines."

Hence, we can conclude that interoperability can occur between:

two or more workflow engines (see Figure 4-1)

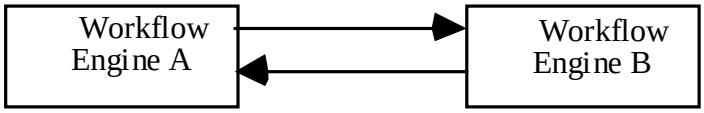

Figure 4-1 Direct interaction between workflow engines

Two or more workflow engines operating within the same workflow enactment service (see Figure 4-2 below)

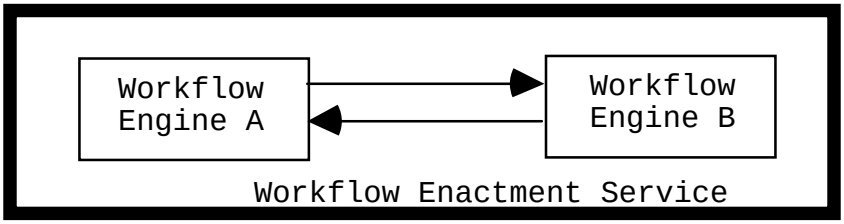

Figure 4-2 Interaction between workflow engines within an enactment service

Two or more workflow enactment services (i.e. two or more workflow engines operating from within two or more workflow enactment services) within the bounds of a Workflow Management System. (See **Figure 4-3** below)

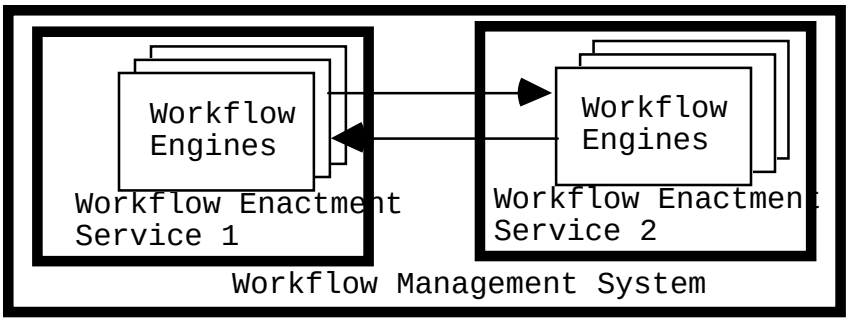

Figure 4-3 Interoperating workflow engines in different enactment services

The Workflow Management Coalition Reference Model [WfMC003] expands on the definition of a Workflow Enactment Service to explain that it:

> " ... provides the run-time environment in which process instantiation and activation occurs, utilizing one or more workflow management engines, responsible for interpreting and activating part, or all, of the process definition and interacting with the external resources necessary to process the various activities."

From the above, we can infer that two process engines having different run-time environments can be taken to be different workflow enactment services.

"External resources" can be taken to be:

(i) Human agents (via workflow client applications);

(ii) Software tools invoked to perform particular tasks;

(iii) Other workflow engines (which may individually or collectively constitute a workflow enactment service).

The Reference Model describes Workflow Domains, which may contain one or more workflow engines. Workflow Domains can be thought of as being defined by some form of business agreement to allow two or more workflow engines to interoperate. Two interoperating workflow engines will share the same workflow domain that identifies the context within which the interoperation takes place. There are two kinds of workflow domain:

Open workflow domains - which describe the set of workflow engines that a workflow engine ends up interoperating with each other, as a result of some exchange of verification tokens[1] in a business context.

Closed workflow domains - which describe the trusted set of workflow engines within which interoperability is to be allowed within a given business context.

Workflow domains, as used in the operation specifications given in section 5 of this document, describe just such logical groupings of workflow engines. It is a matter for the implementors of workflow solutions as to the exact basis on which these workflow domains are defined. Whether a particular workflow domain can be classified as being open or closed is, similarly, a matter for the implementors and owners of a given workflow solution. The precise terms and conditions under which two workflow engines are allowed to interoperate can be modelled/formalised as an *interoperability contract.* For closed workflow domains, this is likely to be a formal specification or trading agreement. For open workflow domains it may just be a set of instructions on how to effect interoperability with "*our process X*".

4.2.2 Effecting Interoperability

[1] What these tokens are and the nature of the exchange is not defined in this document.

Interoperability between software tools is normally taken to mean the ability to share data and/or functionality by two or more tools. A software tool can be any piece of software that performs a specific (set of) functions, such as a text editor, a mail tool, a corporate database server or a workflow management system. Interoperability is normally achieved using one of the following strategies:

1. Direct interaction between the tools (see Figure 4-4 below)

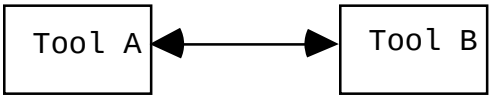

Figure 4-4 Direct interaction between software tools

2. Message passing (see Figure 4-5 below)

Figure 4-5 Software tools interacting by passing messages

3. Bridging (using some form of encapsulation, translation or gateway mechanism as shown in Figure 4-6 below)

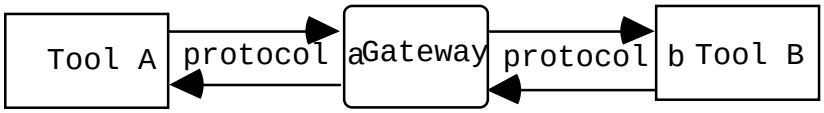

Figure 4-6 Software tools interacting via a gateway that performs protocol transliteration

4. Use of a shared data store (common repository - see Figure 4-7 below).

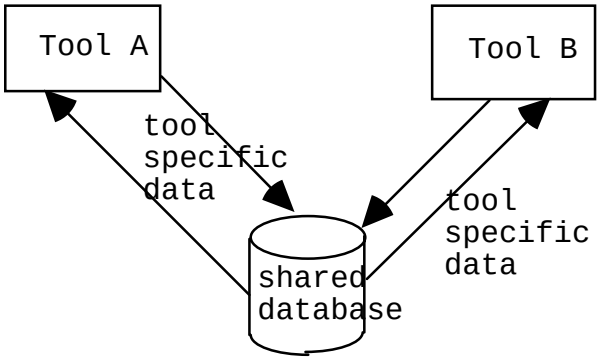

Figure 4-7 Software tools integrated via use of a common repository

This last approach is not specifically addressed by the reference model, but given that there are workflow products which move work from one activity to another via an internal database, using a common (shared) database to move work packages between workflow products is allowable as an alternative way of effecting interoperability between those products. At an abstract level this approach can be viewed as being just another form of store-forward mechanism.

4.2.3 Levels of Interoperability

The Workflow Management Coalition's Interoperability White Paper [WfMC006] identifies eight levels of interoperability. The levels are distinguished by the architectural and consequent operational characteristics of implementations of workflow engines.

Level 1 - No interoperability

This level is characterized by products that have no way of communicating with each other and hence no potential for interoperability.

Level 2 - Coexistence

There is no standard approach to the interoperability of workflow products at this level. Rather there is exploitation of industry, national and international standards by vendors of workflow products to improve the availability of their products on multiple platforms. The effect is that increasing numbers of workflow products become available and can coexist on the same platform(s).

Thus, this level is characterized by workflow products sharing the same run-time environment (hardware, operating system, and network). This level does not imply any direct interaction between different workflow products, but does enable organizations to implement different parts of a "whole process" using different workflow products as appropriate to their needs and the availability of suitable products.

The means of interfacing between different workflow products is through the active participation of human agents (this process has finished, so I now start that one).

This level will also characterize situations where there exist workflow products interoperating with each other using WfMC Standards alongside workflow products that have not implemented the WfMC Standards.

Gateways

A gateway is a mechanism that allows specific workflow products to move work between each other (see figure 3.6 above). A gateway may be part of (one of) the products that use it or may be a separate product. Gateways may or may not exist on the same platform and are primarily concerned with the transfer of workflow control data and, where necessary, application data between different process instances. In certain circumstances an application may act as a gateway between two workflow systems. Gateways use protocol converters to map data and command formats from one domain to another. Gateway implementations may vary in the options for translation that they provide. Gateways may also be required to transfer control of data objects from one workflow management system to another. Where more than two workflow instances are involved, the gateway will also have to perform routing operations. The Interoperability White Paper defines two levels of gateway.

Level 3 - Unique Gateways

This level is characterized by workflow products working together using some bridging mechanism that performs:

Routing of operations between workflow engines and instances

Translation and delivery of workflow relevant data

Translation and delivery of workflow application data

Level 3a - Common Gateway API

This level is characterized by workflow products working together using gateways that share a common (standard) API. This level carries the implication that the operations supported by different gateway mechanisms have been normalized to produce a common subset that can be supported by a standard, but does not exclude the possibility of supersets.

Level 4 - Limited Common API Subset

This level is characterized by workflow products that share a common (standard) API that allows them to interact (interoperate) with each other directly in order to move and manage work between them.

To implement this level of interoperability requires that a core set of API function calls are defined in a published standard and that most/all workflow engines can implement that API. The implementation models for this level are actually quite simple and are based on the use of APIs or encapsulations, i.e.

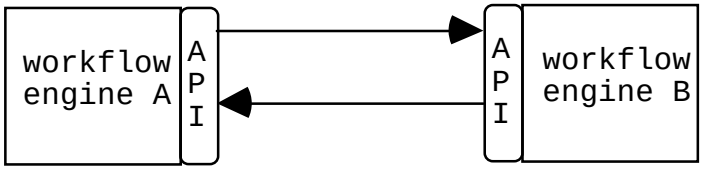

Figure 4-8 Workflow engines interoperating via API calls

or

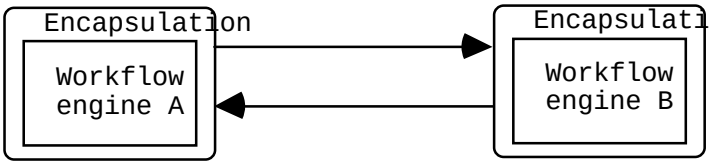

Figure 4-9 Encapsulated interoperating workflow engines

The implementation of the encapsulations or APIs will need to handle any necessary data transformations. In order to avoid the need to implement multiple APIs for a given workflow product to support interoperation with different workflow products, it may be necessary to define neutral information formats to handle the transport of workflow relevant and workflow application data. Each implemented API would then be required to convert to/from the neutral information format.

Level 5 - Complete workflow API

This level is characterized by all workflow products sharing a single standard API that gives access to the full range of possible operations by any workflow management system. This does exclude any domain specific functionality that might be offered by workflow products developed to address the needs of particular market segments.

To define a complete workflow API requires detailed study of the operational command sets of all workflow products on the market in order to deduce the intersecting set of operations that can be supported (in some way) by all products. The wide range of types of workflow products on the market will necessarily impose constraints on what is realistically achievable at this level and it may well be that all that can be done in reaching this level will be achieved by continuous evolution at level 4. The key requirement is that a set of common operations can be defined, probably at an abstract level. These operations must be mapped to operations for each workflow product supporting this level of interoperability by the vendor of that product. The implementation of this mapping mechanism will require the same implementation models as for level 4.

Level 6 - Shared Definition Formats

This level is characterized by different workflow products having a shared format for process definitions that covers routing decisions; user access rights and the maintenance of workflow system resources. The consequence of this is that an organization can produce a single definition for each process that is to be supported on a workflow system, and can guarantee the behavior of the process whatever the workflow engine used to enact it. Constraints on this approach will naturally arise from the different forms and characteristics of present and future workflow products.

A simplistic view would be to state that all workflow products will support all possible operations taken from a defined set (otherwise they cannot call themselves workflow products). A more realistic approach is to recognize that different workflow products designed to solve problems in particular application domains will have specialized functionality that allows them to meet the needs of those domains. They will also have generic functionalities that are common to all or most workflow products and it is these that offer the best

hope for achieving this level of interoperability. Functionality outside of this generic set must be treated as part of a superset that can only be dealt with by certain classes of workflow product (i.e. those with the appropriate operational profile). In this view of the world, there are two steps to achieving a standard that will support this level of interoperability:

1. the definition of the generic set of functionality
2. the definition of operational profiles for different classes of workflow product in order to identify those that can be used to provide specific functionalities.

The Workflow Management Coalition has defined a process definition language (WPDL) [WfMC0020] in order to take the first step. The additional functionalities could be offered using an extended language definition in the same way as computer programmers extend the functionality of a programming language by including types, functions and constants from header files and compile and run time libraries.

The potential offered by this approach is not only that a single process definition can be enacted upon a variety of workflow engines, but also that parts of a suitably modular process definition can be enacted on different workflow engines (as resources become available). The ability to switch work between different work groups that use different workflow products offers a degree of flexibility that promises increases in levels of efficiency and responsiveness of the organization as a whole.

The WPDL as defined is intended to support the definition of workflow processes in a product independent formalism that can be mapped or translated to the specific process definition formalisms that are understood by individual workflow engines.

Two interfaces are identified as being necessary for workflow engines to be able to work with WPDL [WfMC0020]:

1. **Import** a process definition from a character stream of definitions according to the common process definition language into the vendor's internal representation.
2. **Export** a process definition from the vendor's internal representation to a character stream according to the common process definition language.

Level 7 - Protocol Compatibility

This level assumes that all API client/server communication including the transmission of definitions, workflow transactions and recovery is standardized. To achieve this level of interoperability, vendors may be required to support a number of different mechanisms through which such interoperation can be effected.

Level 8 - Common Look and Feel Utilities

This level assumes that in addition to the preceding levels, all workflow products present the user with the same standard user interface or at least “look and feel”. For commercial and practical reasons, this level may never actually be attained.

4.2.4 Models of Interoperability

Previous work on the subject of interoperability by the Workflow Management Coalition has identified the following models of interoperability.

4.2.4.1 Chained processes

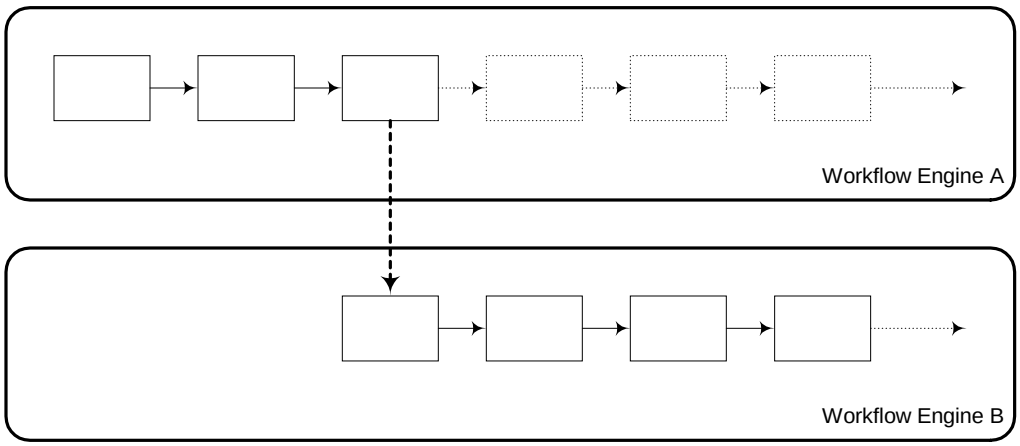

Figure 4-10 The chained model of interoperability

This model of interoperability assumes that the process instance being enacted on Workflow Engine A triggers the creation and enactment of a sub-process instance on Workflow Engine B. Once enactment of the sub-process instance has begun on Workflow Engine B, Workflow Engine A may terminate or may continue with the enactment of its own process instance. It takes no further interest in the newly created sub-process instance.

4.2.4.2 Nested synchronous sub-process

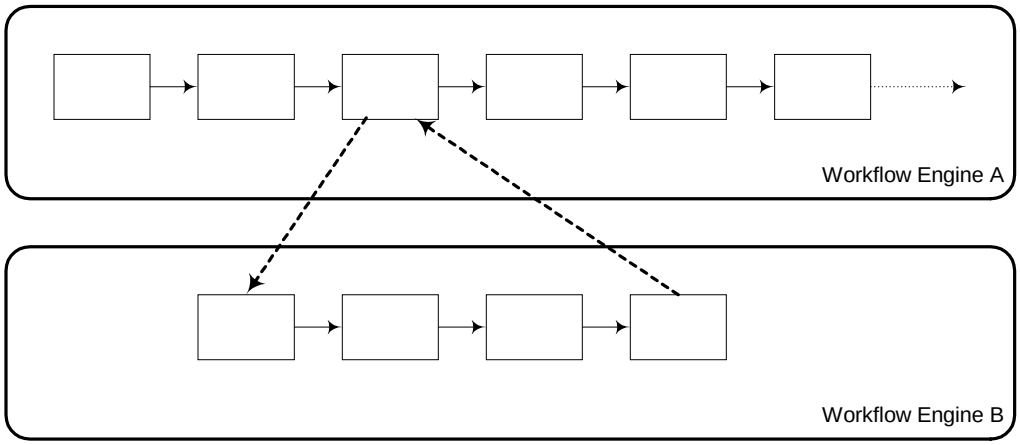

Figure 4-11 The nested sub-process model of interoperability

The nested sub-process model of interoperability assumes that a process instance enacted on a workflow engine causes the creation and enactment of a sub-process instance on a second engine. The activity on the invoking workflow engine remains active until the sub-process reaches some form of termination at which time it completes and allows forward enactment of the thread of activity within the process instance. Synchronization is achieved by notification of changes in the values of designated process instance attributes or in the state of the sub-process instance.

4.2.4.3 Event synchronized sub-process

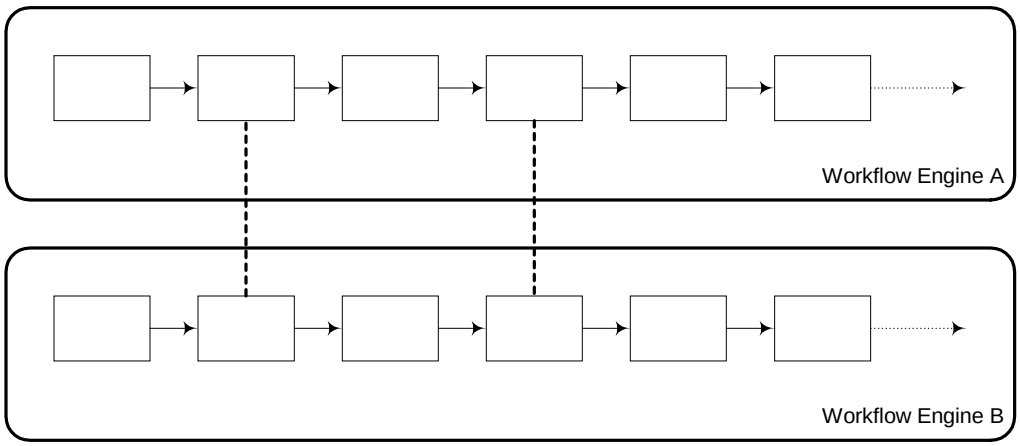

Figure 4-12 Event synchronization by triggering activities

Besides the two methods of process synchronization outlined above, there is an additional need to be able to trigger activities in a process being enacted on a different workflow engine. This triggering of events may arise due to a sub-process being aborted by its enacting workflow or as part of a defined check-pointing logic between two process instances being enacted on separate workflow engines.

4.2.4.4 Nested sub-process (Polling/Deferred Synchronous)

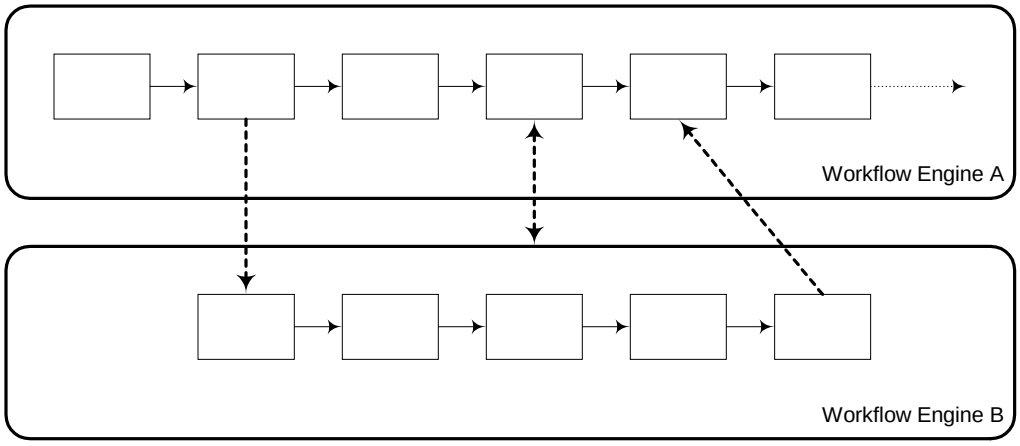

Figure 4-13 The nested sub-process (polling/deferred synchronous) model of interoperability

The nested sub-process (polling) model of interoperability allows one workflow engine to create a process instance on another according to a known process definition; to instantiate the process instance and to cause enactment of the instantiated process instance. The invoking workflow engine carries on with the enactment of the process instance that invoked the sub-process until it reaches a point where it needs to effect a rendezvous with its child sub-process. At this stage, it polls the enacting workflow engine to determine when the sub-process has reached completion. It is also possible that the sub-process on the enacting engine is prematurely terminated before the parent process reaches its rendezvous point. The invoking workflow engine will receive notification of how the sub-process achieves termination. Rendezvous between the two process instances is managed by the enacting workflow engines as follows:

- if the sub-process instance reaches termination before the invoking process is ready to deal with the event, the workflow engine queues the termination event until it is required

- if the invoking process instance requests the outcome of the enacted sub-process before it has achieved termination, the workflow engine enacting the sub-process queues the request until it can be satisfied.

4.3 Process Administration

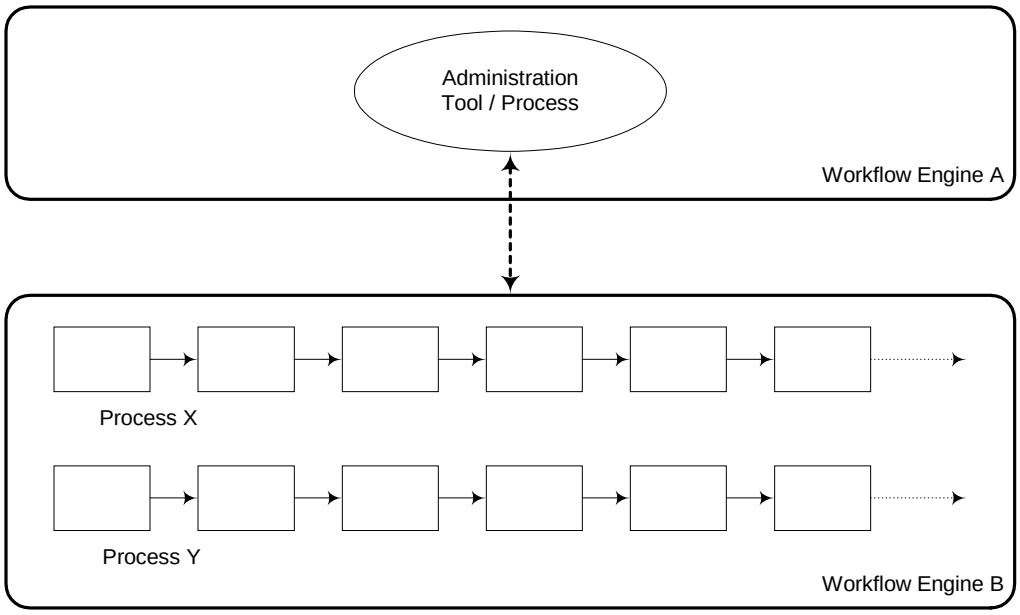

Figure 4-14 Process administration supported by interoperability functionality

The process administration interoperability model is concerned with workflow administration tools or activities defined within workflows enacted on one workflow engine to conduct the following process administration activities on process instances enacted on another WfMC conformant workflow engine:

- list process instances currently being enacted on behalf of the querying workflow engine
- ascertain the current state of a given process instance being enacted on behalf of the querying workflow engine
- start and stop enactment of sub-process instances
- monitor the progress of enacted sub-processes
- get and set values of process relevant data

5 Overview

5.1 Assumptions

The basic assumption underpinning this work is that two or more workflow engines exist (they may be two or more instances of the same workflow product or instances of different workflow products) which can communicate with each other in order to effect the:

- Selection
- Instantiation
- Enactment

of known process definitions and (optionally) the return of the results of the performance of a nested process definition to the invoking workflow engine. No assumptions are made about how such

communication is effected, only that it is effected. Similarly, no assumptions are made about the architecture or operating characteristics of the workflow products. A necessary distinction is made between the operational characteristics of workflow engines that communicate with each other synchronously and workflow engines that communicate with each other asynchronously. The principle of transparency across the interface assumes that process definitions are specified using a common definition protocol such as WPDL as described in [WfMC0020].

5.2 Objectives

The objective is to define a standard capable of supporting the implementation of nested sub-processes across multiple workflow engines. In the following descriptions of possible interoperability scenarios, the term Workflow Engine A is used to denote the workflow engine enacting the (parent) process instance that causes some other workflow engine, termed Workflow Engine B, to initiate enactment of a (child) sub-process instance.

5.2.1 Starting a chained sub-process

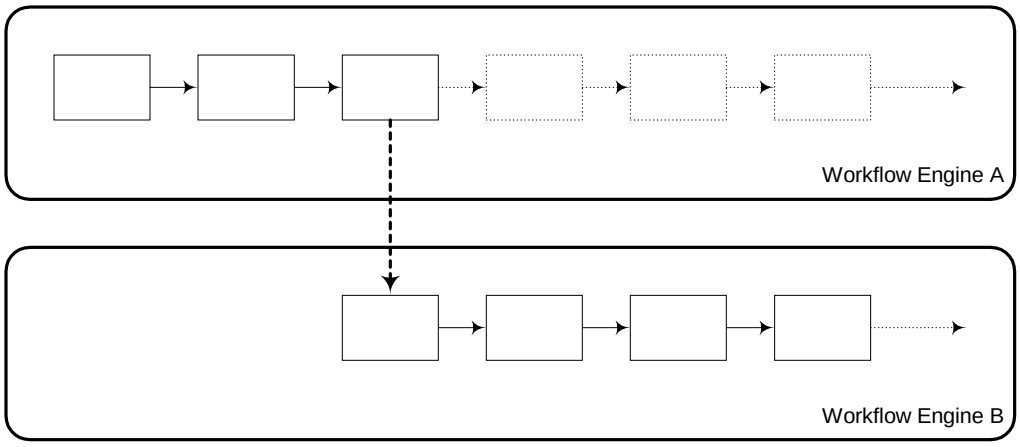

Figure 5-1 main process starts a chained sub-process on another workflow engine

This scenario involves the invoking workflow engine (which is running the parent workflow process instance) creating/starting a new (child) process instance as a sub-process to be enacted by some other workflow engine. In this first scenario, the invoking workflow engine does not wait for the sub-process to complete, but either terminates or carries on with the next step in the parent process model.

To see how this works in practice, assume the existence of two workflow engines A and B. To meet the objectives set out above, it must be possible for Workflow Engine A to:

1. Select a known process definition managed by or accessible to Workflow Engine B
2. Pass Workflow Engine B process relevant data (which may include the location of application data) in order to instantiate the selected definition
3. Request Workflow Engine B to enact the selected process definition

The transfer of application data between interoperating workflow engines is deemed to be outside the scope of this specification and of the WAPI in general. The notification of the location of application data to be processed by tools invoked during the enactment of workflow activities is treated as the passing of process relevant data.

There are a number of possibilities regarding the selection of the process definition to be enacted by Workflow Engine B.

1. The definition is owned (managed) by Workflow Engine A and is passed to Workflow Engine B when it is required.

2. The definition is managed by Workflow Engine B and requested by Workflow Engine A when required (this implies that Workflow Engine A has knowledge of the existence and location of the definition).

3. All definitions are stored in some shared space (say a repository) and can be retrieved as required by any of the workflow engines involved.

For the purposes of this standard, these possibilities can be reduced to two options;

1. The definition of the sub-process to be enacted is passed from one workflow engine to another;

2. The location of the definition of the sub-process to be enacted is known and accessible to the workflow engine that is to enact it.

This standard assumes that if process definitions were to be passed between workflow engines, this would occur at the request of Workflow Engine B on instruction from Workflow Engine A and would be effected by the workflow engines using some other exchange mechanism, possibly one that involves Process Definition Interchange as defined in [WfMC0020]. Thus it is only necessary for this standard to address option 2.

A key issue for the specification of concrete bindings is whether the interoperation between the two workflow engines is to be effected using some model of synchronous communication, such as would be required by direct connection via TCP/IP, or asynchronous communication which could be effected using some form of store-forward mechanism such as electronic mail. There are thus two distinct cases that have to be considered[2]:

- Synchronous interoperation and
- Asynchronous interoperation.

The main difference between the two modes of working lies in the requirement for both workflow engines to be "on-line" at the same time in order to effect an interoperability dialogue. Such differences are addressed in specific bindings written to correspond to this abstract specification and in the text of binding specific Interoperability Proving Frameworks.

In terms of the operations defined in section 5 below, the operations listed in Table 5-1 would be used to effect the dialogue between two workflow engines required to support the interoperability shown in Figure 5-1 above.

Workflow Engine	Operation
A	Create Process Instance
B	Response to Create Process Instance
A	Set Process Instance Attributes
B	Response to Set Process Instance Attributes
B	Get Process Instance Attributes
A	Response to Get Instance Attributes

[2] The working assumption here is that interoperability via an object request broker can be either synchronous or asynchronous and thus there are only two cases. Object request brokers do support a third mode of working called deferred synchronous where the invoking application fires off its message, carries on with its own work and claims the reply from the ORB some time later. This third mode of operation is outside the scope of the current standard.

A	Change Process Instance State
B	Response to Change Process Instance State
A	Relinquish Process Instance
B	Response to Relinquish Process Instance

Table 5-1: operations required to start a chained sub-process.

Note that the dialogue between the two workflow engines is based on the notion of request/response message pairings, where the response returns a status indicating the success, failure or other outcome of the requested operation. Such responses are distinct from notifications made by Workflow Engine B of state changes or changes to the values of process instance attributes that may occur during the life of the enacted sub-process. Notifications are not sent where such change occurs in response to some received instruction for which a response message already conveys the necessary information.

Example

Let us assume that Workflow Engine A requires to initiate the enactment of a defined sub-process on Workflow Engine B.

Workflow Engine A would connect to Workflow Engine B and pass it an instruction to create a new process instance based on a known process definition using the *Create Process Instance* operation. Workflow Engine B responds by notifying Workflow Engine A of the PID of the created process instance.

Workflow Engine A may set values of workflow relevant data items in the definition using the *Set Process Instance Attributes* operation. Workflow Engine B responds by notifying Workflow Engine A that the operation has succeeded/failed.

Where necessary Workflow Engine B may ask Workflow Engine A to assign values to workflow relevant data items using the *Get Process Instance Attributes* operation. Workflow Engine A responds by providing Workflow Engine B with the requested values.

Workflow Engine A will request/instruct Workflow Engine B to start enactment of the process instance using the *ChangeProcessInstanceState* operation to switch the process instance state to *open.running*. Workflow Engine B will notify Workflow Engine A when this has occurred.

If response times are not adequate to support/sustain atomic transmission, Workflow Engine A may batch request messages for transmission to Workflow Engine B. Workflow Engine B will return a batch of response messages, one for each request message in the batch sent by Workflow Engine A.

Assuming batched transmission, requests and responses might be batched as follows:

Workflow Engine A

Create Process Instance

Workflow Engine B

Response to Create Process Instance

Workflow Engine A

Set Process Instance Attributes

Change Process Instance State

Relinquish Process Instance

Workflow Engine B

Response to Set Process Instance Attributes

Response to Change Process Instance State

Response to Relinquish Process Instance

Note that should a request message fail in the middle of a batch of requests, workflow engine B will return a batch of responses in which:

- those operations which succeeded prior to the failure return a success status
- the operation that failed returns a failed status
- the operations requested following the operation which failed return a status of operation not performed.

Process chains may be constructed involving creation of many process instances, e.g.

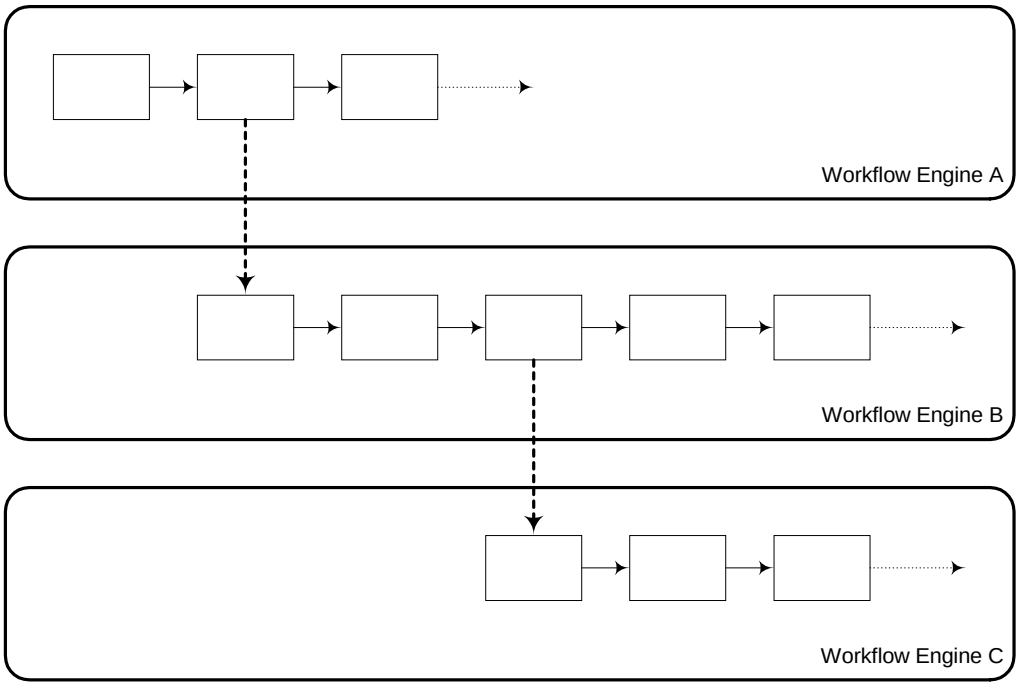

Figure 5-2 main process starts a chained sub-process on another workflow engine that in turn creates a chained sub-process on a third workflow engine

5.2.2 Starting a sub-process that completes a process step

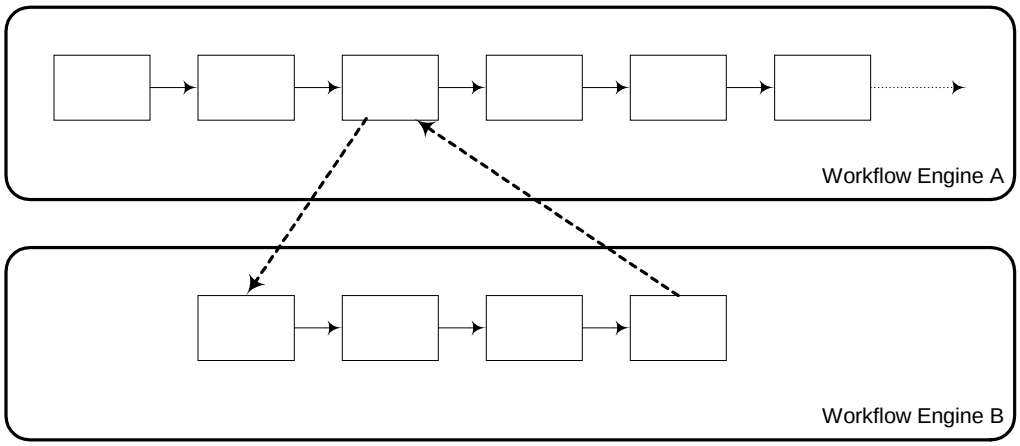

Figure 5-3 a process instance starts a sub-process on another workflow engine and waits for completion

In this scenario, the parent process instance waits for the child to complete, possibly taking back process relevant or application data before performing the next step in the process. To support scenarios where the conclusion of the enacted sub-process alone is the requirement for the continued enactment of the parent process, it is necessary to provide a mechanism for notification of the end of enactment of a sub-process.

Workflow Engine	**Operation**
A	Create Process Instance
B	Response to Create Process Instance
A	Set Process Instance Attributes
B	Response to Set Process Instance Attributes
B	Get Process Instance Attributes
A	Response to Get Process Instance Attributes
A	Change Process Instance State
B	Response to Change Process Instance State
B	Process Instance Attribute Changed
A	Response to Process Instance Changed
A	Get Process Instance Attributes
B	Response to Get Process Instance Attributes
A	Relinquish Process Instance
B	Response to Relinquish Process Instance

Table 5-2: operations required for a sub-process that completes a process step

Example

Let us assume that Workflow Engine A requires to initiate the enactment of a defined sub-process on Workflow Engine B, needing the results of the enactment to be able to continue the enactment of its own process definition.

Workflow Engine A would connect to Workflow Engine B and pass it an instruction to create a new process instance based on a known process definition using the *Create Process Instance* operation. Workflow Engine B responds by notifying Workflow Engine A of the PID of the created process instance.

Workflow Engine A may set values of workflow relevant data items in the definition using the *Set Process Instance Attributes* operation. Workflow Engine B responds by notifying Workflow Engine A that the operation has succeeded/failed.

Where necessary Workflow Engine B may ask Workflow Engine A to assign values to workflow relevant data items using the *Get Process Instance Attributes* operation. Workflow Engine A responds by providing Workflow Engine B with the requested values.

Workflow Engine A will request/instruct Workflow Engine B to start enactment of the process instance using the *ChangeProcessInstanceState* operation to change the state of the sub-process to *open.running*. Once enactment has started, Workflow Engine B will respond to the request, thus notifying Workflow Engine A that this has occurred.

When the enactment of the sub-process is finished, Workflow Engine B is required to communicate the product of the sub-process to Workflow Engine A (or at least notify it that it is now available). This can be achieved by Workflow Engine B using the *ProcessInstanceStateChanged* operation to tell Workflow Engine A that the sub-process enacted on Workflow Engine B has completed. Workflow Engine A may then ask Workflow Engine B for values for these workflow relevant data items using the *Get Process Instance Attributes* operation. Workflow Engine B responds by providing Workflow Engine A with the requested values.

Once it has retrieved all of the values it requires, Workflow Engine A might then tell Workflow Engine B that it is safe to release all pertinent memory structures relating to the enactment of the process instance. This could be achieved using the *Relinquish Process Instance* operation.

Assuming batched transmission, requests and responses might be batched as follows:

Workflow Engine A

Create Process Instance

Workflow Engine B

Response to Create Process Instance

Workflow Engine A

Set Process Instance Attributes

Change Process Instance State

Workflow Engine B

Response to Set Process Instance Attributes

Response to Change Process Instance State

Workflow Engine B

Notify Process Instance Attribute Changed

Workflow Engine A

Response to Notify Process Instance Attribute Changed

Workflow Engine A

Get Process Instance Attributes

Workflow Engine B

Response to Get Process Instance Attributes

Workflow Engine A

Relinquish Process Instance

Workflow Engine B

Response to Relinquish Process Instance

This scenario implies an ongoing "interest" in the progress of the enacted sub-process on the part of the invoking process. Thus, an additional operation to check the current state of a process instance can be envisaged, returning status information regarding the enacted sub-process to the invoking workflow engine. Such an operation would be necessary in order to effect queries across multiple workflow engines to ascertain the current state of a "whole process" being enacted as separate process instances [ICL95].

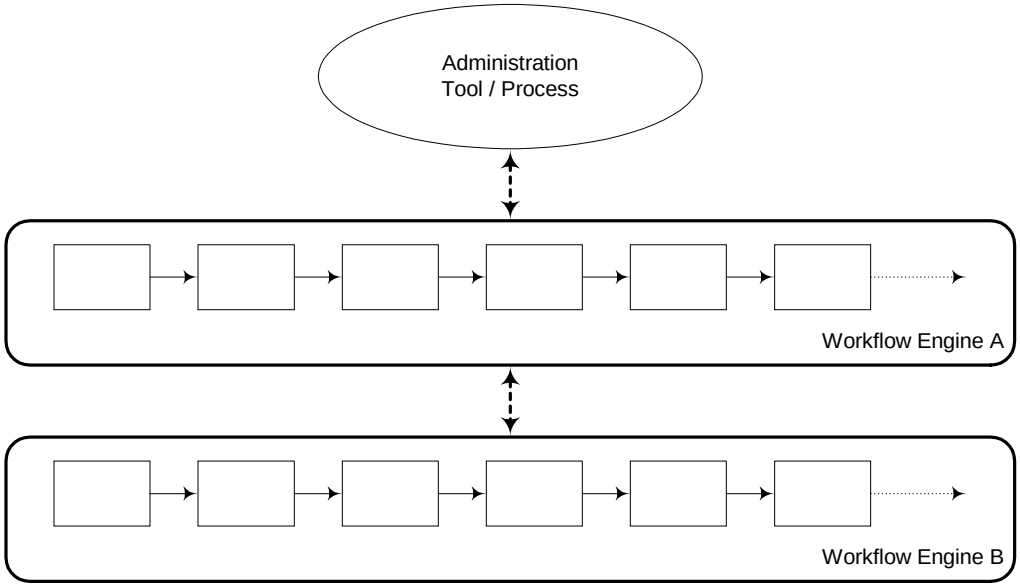

Figure 5-4use of a process management tool in a multiple workflow engine environment

In the example shown in figure 4.4, the arrows connecting Workflow Engine A with the Process Management Tool are effected using the facilities of WAPI interface 2 [WfMC1009] and/or WAPI interface 3 [WfMC0013]. The arrows connecting Workflow Engine A to Workflow Engine B are effected using the facilities of WAPI interface 4.

Workflow Engine	Operation
A	List Process Instances
A	Get Process Instance State

Table 5-3 additional operations required to support use of a process management tool in a multiple workflow engine environment

The following list describes the WfMC taxonomy of possible process instance states:

State	Description
Open	The Process Instance is enacted
open.running	The Process Instance is executing
open.notRunning	The Process Instance is temporarily not executing
open.notRunning.notStarted	The Process Instance has been created, but was not started yet
open.notRunning.suspended	Execution of the Process Instance was temporarily suspended
closed	Enactment of the Process Instance has been finished
closed.aborted	Enactment of the Process Instance has been aborted. It is an abnormal termination with no attempt to terminate sub-processes. It is used in catastrophic circumstances where nothing except clearing the process away can be done.
closed.terminate	Enactment of the Process Instance has been terminated. It is an abnormal but graceful termination, in which an attempt to terminate all running

	activities and sub-processes is attempted.
closed.completed	Enactment of the Process Instance has completed normally.

A workflow product vendor might decide to support refinement of states to a given level only or to omit certain states; valid sets of states include for example:

- *open.notRunning*, open.*running* and *closed*
- *open.notStarted*, open.*running*, closed.*completed* and closed.*terminated*
- ...

The following diagram illustrates the above states and potential state transitions.

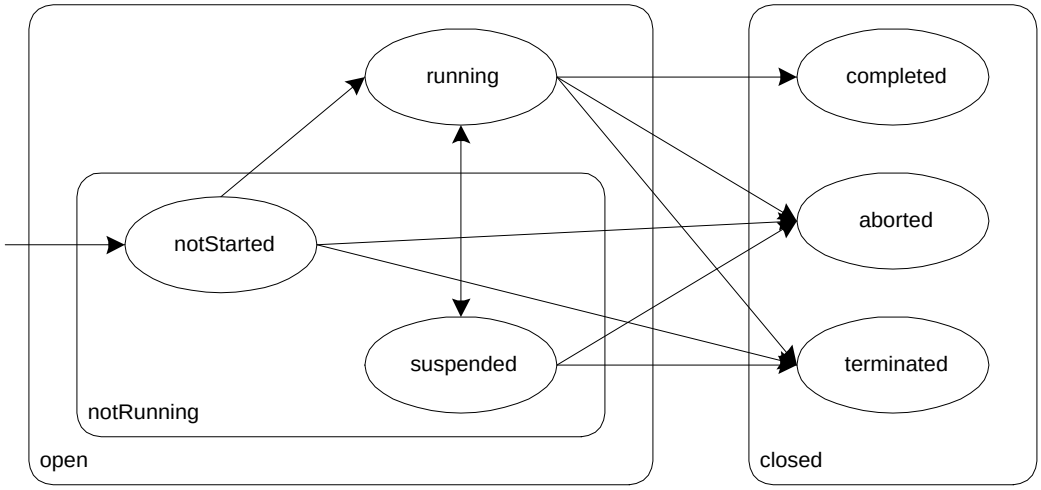

Figure 5-5 State transition model for a process instance

5.3 Defined Terms and Abbreviations

The terms used in this document are defined in the Workflow Management Coalition Glossary [WfMC000].

5.4 Conformance and Correspondence

This document is an abstract specification, and as such it is not possible for vendors of workflow products to claim conformance to it. The specification contained in this document is realized through specific binding specifications which have been adopted by the Workflow Management Coalition as demonstrating *correspondence* to the abstract specification. Vendors of workflow products and other interested parties are directed to these binding specifications for guidance in development of implementations and for associated conformance requirements. Users and vendors should refer to the appropriate Interoperability Proving Framework documents which describe how the WfMC assesses conformance for a given binding.

5.5 Naming Conventions

The data types used in this document are abstract data types which when the time comes to reify down to concrete interfaces could be mapped onto those defined in [WfMC1013] to produce C language bindings. Other language bindings must provide their own type definitions. States and return values used in this document are derived from [WfMC015].

6 Specification of Operators for Effecting Interoperation

The following text presents an abstract representation of the operations required to effect interoperability between two (or more) workflow engines. The message specifications are intended as abstract representations of the information that needs to be passed between two workflow engines in order to effect the operations described. It is expected that the specifications of message formats will apply to implementations that work either synchronously or asynchronously.

Three distinct classes of message are described below -

- request messages
- response messages
- notification messages

A request message is used when one workflow engine needs another workflow engine to perform some action on its behalf. Every request message is answered by a response message which tells the requesting engine the result of its request, i.e. the action(s) requested have been carried out or it was not possible to perform the actions requested because ...

During a protracted interoperation (the time line for such workflow interoperations can be a matter of hours, days or even weeks) there may be defined event points in the enactment of a sub-process when it is required that the parent process is made aware that a given milestone has been achieved. Alternatively it may be material to the ability of the parent process to progress to its conclusion if the enacted sub-process is prematurely terminated or fails in some way. To provide the capability for interacting workflow engines to handle these circumstances, notification messages are provided so that Workflow Engine B, enacting a sub-process on behalf of Workflow Engine A, may inform Workflow Engine A of significant events that occur during the enactment of the sub-process. Every notification message is answered by a response message which tells the notifying workflow engine that it has been received and understood.

In the specification of messages given below, the field “Message Routing Information” is provided for completeness. The requirement for its usage is dependent on the particular transport binding that you are implemented and it may not be needed at all for some bindings.

6.1 Connection Operations

Connection operations are assumed to be binding specific and outside the scope of the abstract specification.

6.2 Process Control Interactions

6.2.1 Change Process Instance State

Description Instruction to change the state of a designated process instance which is being enacted on another workflow engine from its current state to a designated new state. Both workflow engines create appropriate audit records as specified in [WfMC015] to record the old and new states for the process instance.

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *Change Process Instance State* operation

Engine identifier	identifies the target workflow engine
Process id	the id of the process that is to undergo the state change
State	the new state to which the process instance is to be switched

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Rationale It is an operational requirement of users of interoperating workflow systems that it be possible to start, terminate/abort or suspend enactment of sub-process workflows and resume enactment of suspended sub-process workflows when this becomes necessary.

Audit Data

The following audit data records would be created as a result of changing the state of a process instance being enacted on the target workflow engine on behalf of the requesting workflow engine.

Request

Requesting Workflow Engine:	Link to Remote Process Audit Data
	Event Code: WMSentRequestChangeProcessInstanceState
Target Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedRequestChangeProcessInstanceState

Operation

Requesting Workflow Engine:	Remote Process Operations Audit Data
	Event Code: WMChangedProcessInstanceState
Target Workflow Engine:	Change Process/Subprocess Instance State
	Event Code: WMChangedProcessInstanceState

Response

Target Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentChangedProcessInstanceState
Requesting Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedChangedProcessInstanceState

6.2.2 Create Process Instance

Description Instruction to a receiving workflow engine to create a new process instance based on a designated process definition..

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *Create Process Instance* operation

Engine identifier	identifies the target workflow engine and domain
Process definition id	the id of the process definition that is to be enacted
Return flag	indicates whether the target workflow engine is required to communicate any return values (nested sub-process)
Parent pid	the initial process instance unique id (the id of the process instance operating on the invoking (parent) workflow engine[3]
Activity id	the id of the activity in the parent process instance which is causing this request to create a sub-process
Sub-process id	the process instance unique id of the process instance that has been created by the selection of the given process definition
User id	the id of the primary user of the process instance to be created (may be null)
Role id	the id of the role assumed by the primary user (may be null)

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Rationale As explained in 4.2 above, it is necessary that a workflow engine be able to communicate the identity of a process definition to another workflow engine in order for the latter to enact it.

It is also necessary that the workflow engine creating a new process instance should know whether or not to communicate status information to the workflow engine that initiated the request to do so. If the return flag is set to TRUE, then the Request Message *Profile* field will be set to *nest* and as a result, the initiating workflow engine will be notified of all status change information (started, suspended, resumed, completed...) until such time as the process instance reaches termination of one form or another or the initiating workflow engine issues an instruction that it wishes to relinquish its interest in the new process instance. If the return flag is set to FALSE, then the Request Message *Profile* field will be set to *chain* and as a result, the enacting workflow engine will not notify the initiating workflow engine of any state changes.

Audit Data:

Request

Requesting Workflow Engine:	Link to Remote Sub-process Audit Data
	Event Code: WMSentRequestStartProcessInstance
Target Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedRequestStartProcessInstance

Operation

[3] Note that this is the id of the invoking process. If the invoking process is itself an invoked sub-process, the id used is that of the invoking sub-process, not that of its parent.

Requesting Workflow Engine:	Remote Process Operations Audit Data
	Event Code: WMCreatedProcessInstance
Target Workflow Engine:	Create/Start Process/Subprocess Instance
	Event Code: WMCreatedProcessInstance

Response

Target Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentStartedProcessInstance
Requesting Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedStartedProcessInstance

6.2.3 Get Process Instance Attributes

Description Instruction to return the value(s)[4] of the requested process instance attributes (process relevant data).

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *Get Process Instance Attributes* operation

Engine identifier	identifies the target workflow engine
Process id	the id of the process instance that owns the workflow relevant data being requested
Root pid	the initial process instance unique ID (the ID of the process instance operating on the invoking (parent) workflow engine
Activity id	the ID of the activity in the parent process instance which is causing this request to retrieve process instance attributes
Attributes	a list of attribute specifications giving for each attribute to be set: the name of the attribute the type of the attribute

the value to which the attribute is set

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Rationale Checking values of process relevant data is one way in which a workflow engine can check on the progress of a workflow being enacted on another workflow engine.

Audit Data:

The following audit data records would be created as a result of the target workflow engine successfully providing each requested attribute value at the behest of the requesting workflow engine.

Request

Requesting Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentRequestGetProcessInstanceAttribute
Target Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedRequestGetProcessInstanceAttribute

Response

Target Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentRetrievedProcessInstanceAttribute
Requesting Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedRetrievedProcessInstanceAttribute

[4] For efficiency reasons it should be possible to obtain sets of workflow relevant data in a single interaction.

6.2.4 Get Process Instance State

Description Instruction to another workflow engine to return the current status of a given process instance which is it is enacting.

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *Get Process Instance State* operation

Engine identifier — identifies the target workflow engine

Process — the id of the process instance that is being enacted

State — the returned state[5] - a string prefixed by one of:

open.not-running[6]

open.running[7]

closed.completed

closed.terminated

closed.aborted

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Rationale For long term sub-processes with a life beyond that of the session during which they were created, it is important that the invoking workflow engine be able to check as necessary that the invoked sub-process is alive and well or has completed.

Audit Data:

The following audit data records would be created as a result of the target workflow engine successfully providing the process state at the behest of the requesting workflow engine.

Request

Requesting Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentRequestGetProcessInstanceState[8]
Target Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedRequestGetProcessInstanceState

Response

Target Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentRetrievedProcessInstanceState

---

[5] The return values here are taken from [WfMC015]. There is a difference here with the *WMFetchProcessInstanceStatesList* operation defined in [WfMC1009] which makes no assumptions about what states a process instance can have.

[6] Created but not yet started

[7] Once started, processes can be in the active or suspended state until one way or another they are terminated.

[8] New event codes to be confirmed with WG5 – alternatively we could use those given for requesting arrtibute values (see 5.2.3 above).

Requesting Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedRetrievedProcessInstanceState

6.2.5 **Process Instance Attributes Changed**

Description Instruction to another workflow engine to sets the value(s) of designated process instance attributes (process relevant data) for a given process instance.

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *Process Instance Attributes Changed* operation

Engine identifier	identifies the target workflow engine
Process id	the id of the process instance that owns the process relevant data being requested
Attributes	a list of attribute data structures giving for each attribute: the name of the attribute the type of the attribute the value to which the attribute has been set

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Rationale This operation is provided so that a workflow engine enacting a sub-process can notify the workflow engine enacting the parent process instance that the values of particular elements of workflow relevant data has been changed. This facility allows for tracking of milestones in the management of workflows enacted in a multi-engined domain.

Audit Data

The following audit data records would be created as a result of the notifying workflow engine having changed the value of a notifiable attribute.

Actual Event

Notifying Workflow Engine:	Change Process Instance Attributes Audit Data
	Event Code: WMAssignedProcessInstanceAttribute

Notification

Notifying Workflow Engine:	Link to Remote Process Operations Audit Data
	Event Code: WMSentChangedProcessInstanceAttribute
Target Workflow Engine:	Link from Remote Process Operations Audit Data
	Event Code: WMReceivedChangedProcessInstanceAttribute

6.2.6 Process Instance State Changed

Description Notification to another workflow engine of a state change in a (sub) process in which it has a registered interest.

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *Process Instance State Changed* operation

Engine identifier	identifies the invoking workflow engine
Process id	PID of the process instance that has undergone a state change
New State	State the process instance has changed to

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Rationale In circumstances where the invoking process instance needs to be made aware of protracted inactivity of a sub-process instance enacted by another workflow engine, the workflow engine enacting the sub-process must have a means of communicating state changes (e.g. suspend or resume) to the invoking engine.

Audit Data

The following audit data records would be created as a result of the completion of a process instance being enacted on the notifying workflow engine on behalf of the target workflow engine.

Actual Event

Notifying Workflow Engine:	Change Process/Subprocess Instance State Audit Data
	Event Code: WMChangedProcessInstanceState

Notification

Notifying Workflow Engine:	Link to Remote Process Operations Audit Data
	Event Code: WMSentChangedProcessInstanceState
Target Workflow Engine:	Link from Remote Process Operations Audit Data
	Event Code: WMReceivedChangedProcessInstanceState

6.2.7 Set Process Instance Attributes

Description Instruction to another workflow engine to set the value(s) of process instance attributes (process relevant data) for a given process instance that it is enacting. The attribute list sent to target workflow engine will contain value specifications for one or more process instance attributes to be set. The target workflow engine will attempt to set attribute values in the order in which they occur in the list. It returns a response message containing a list of those attributes for which the set operation was successful. In the event that, part way through actioning a list of attribute values some error occurs, then the attribute for which it was unable to set a value will not be contained in the response message and the return code value will indicate that a failure occurred. The target workflow engine will not action a list of attribute values beyond the point at which a failure occurs.

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *Set Process Instance Attributes* operation

Engine identifier	identifies the target workflow engine
Root pid	the initial process instance unique ID (the ID of the process instance operating on the invoking (parent) workflow engine
Activity id	the ID of the activity in the parent process instance which is causing this request to set process instance attributes
Process id	the id of the process instance that owns the process relevant data being requested
Attributes	a list of attribute specifications giving for each attribute to be set: the name of the attribute the type of the attribute the value to which the attribute is to be set

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Rationale Process definitions are only partial and must be fully (or sufficiently) instantiated before enactment may commence.

Audit Data:

The following audit data records would be created as a result of the target workflow engine successfully changing each attribute value at the behest of the requesting workflow engine.

Request

Requesting Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentRequestChangeProcessInstanceAttribute
Target Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedRequestChangeProcessInstanceAttribute

Operation

Requesting Workflow Engine:	Remote Process Operations Audit Data
	Event Code: WMAssignedProcessInstanceAttribute
Target Workflow Engine:	Change Process Instance Attributes

Event Code: WMAssignedProcessInstanceAttribute

Response

Target Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentChangedProcessInstanceAttribute
Requesting Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedChangedProcessInstanceAttribute

6.2.8 Trigger Activity

Description Instruction to another workflow engine to trigger a designated activity defined for a process instance which it is enacting.

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *Trigger Activity* operation

Engine identifier	identifies the target workflow engine
Root pid	the initial process instance unique ID (the ID of the process instance operating on the invoking (parent) workflow engine
Activity id	the ID of the activity in the target process instance which is to be triggered
Process id	the id of the process instance that owns the process activity to be triggered

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Audit Data

Request

Requesting Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentChangeActivityInstanceState[9]
Target Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedChangeActivitiyInstanceState

Operation

Requesting Workflow Engine:	Remote Process Operations Audit Data
	Event Code: WMEventOccurred
Target Workflow Engine:	Change Activitiy Instance State Audit Data
	Event Code: WMChangedActivityInstanceState

Response

Target Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentChangedActivityInstanceState
Requesting Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedChangedActivityInstanceState

[9] New event codes for Request & Response to be agreed with WG5

6.2.9 List Process Instances

Description Return a list of the PIDs of process instances selected by the criteria given in the filter parameter.

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *List Process Instances* operation

Engine identifier	identifies the invoking workflow engine
Filter	arbitrary criteria for the selection of process instances (e.g. PID, user, definition id, ...)

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Rationale This operation is necessary to support the use of process management tools which operate in a multiple workflow engine environment. Such queries will be passed from one workflow engine to another until either the query can be answered or the closure of the set of known or possible workflow engines is completed. The rules for defining the set of possible workflow engines are not defined.

Audit Data:

Request

Requesting Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentRequestListProcessInstances[10]
Target Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedRequestListProcessInstances

Response

Target Workflow Engine:	Link to Remote Subprocess Audit Data
	Event Code: WMSentRetrievedProcessInstanceData
Requesting Workflow Engine:	Link from Remote Subprocess Audit Data
	Event Code: WMReceivedRetrievedProcessInstanceData

[10] New event codes to be confirmed with WG5.

6.2.10 Relinquish Process Instance

Description **N**otification to another workflow engine that it may now release all memory containing data structures pertaining to the given process instance and/or not bother to send notification messages concerning the enactment of that process instance.

Parameters The following parameters might reasonably be expected to be provided to a function that would construct the messages used to effect the *Relinquish Process Instance* operation

Engine identifier	identifies the target workflow engine
Process id	PID of the sub-process instance in which this workflow engine is no longer interested

The workflow engine might reasonably be expected to be able to compute the other values necessary to compute the request message shown below.

Rationale For certain dialogue structures it will be necessary that one workflow engine tells another that it is now safe to release all data structures it holds in memory relating to a given process instance and/or that it is no longer interested in receiving notification messages for that process instance.

There are two circumstances in which a WMRelinquishProcessInstance message is intended to be used. Workflow Engine B enacting a sub-process at the behest of Workflow Engine A will:

- send notification messages to the other workflow engine at appropriate points, e.g. on completion of the enactment of the enacted sub-process
- maintain the value of process instance attributes until it is told it may safely release them

The WMRelinquishProcessInstance operation is provided so that in the case of a chained model of interoperability in which Workflow Engine A initiates the enactment of a sub-process on Workflow Engine B but then takes no further interest in it, Workflow Engine B can be told not to send notification messages to Workflow Engine A and will then assume on completion of the enactment that it may safely release all associated data structures in memory.

The alternative use of the WMRelinquishProcessInstance operation is for nested sub-process models of interoperability in which Workflow Engine A initiates the enactment of a sub-process on Workflow Engine B and then waits for its completion in order to retrieve the value(s) of some process instance attribute(s). On receipt of notification that either the value of the designated process instance attribute(s) has changed or that the sub-process has reached completion, Workflow Engine A will

1. retrieve the value(s) of the attribute(s) from Workflow Engine A using WMRequestGetProcessInstanceAttributes
2. use WMRelinquishProcessInstance to tell Workflow Engine B it has no further interest in the sub-process.

Audit Data:

Audit Data

Request

Requesting Workflow Engine:	Link to Remote Subprocess Audit Data

Event Code: WMSentRelinquishProcessInstance[11]

Target Workflow Engine: Link from Remote Subprocess Audit Data

Event Code: WMReceivedRelinquishProcessInstance

Response

Target Workflow Engine: Link to Remote Subprocess Audit Data

Event Code: WMSentRelinquishedProcessInstance

Requesting Workflow Engine: Link from Remote Subprocess Audit Data

Event Code: WMReceivedReRelinquishedProcessInstance

[11] New event codes to be confirmed with WG5

7 Implementation Issues

7.1 Managing Interoperability

Interoperability Contracts

Organisations allowing interoperation within or between workflow domains will do so within the context of an interoperability contract.

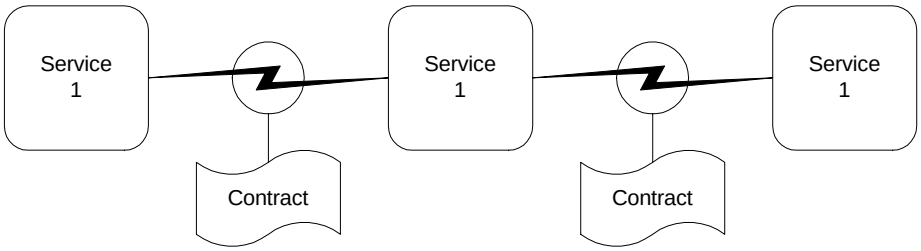

Figure 7-1 The role of interoperability contracts in a value chain

Interoperability contracts denote trading agreements across service boundaries which operate between workflow applications enacted on different workflow engines. These co-operating workflow engines may be within the same organisation or within separate organisations that collectively operate a value chain. An interoperability contract governing workflow engine interoperability across a service boundary will describe elements from the list presented below according to the nature and requirements of the business process being supported across the service boundary and the requirements of the organisations operating that process.

1. which workflow engines within one service domain are visible to/capable of interoperating with workflow engines in the other service domain
2. which workflow definitions can be enacted within one service domain at the behest of workflow engines in the other service domain
3. the transport binding supported (e.g. MIME, jFlow, Wf-XML,…)
4. for each workflow definition identified in the contract:
    - the conformance profile required to effect interoperability
    - input/instantiation requirements
    - for each traded (shared) element of workflow relevant data
        - access rights (readable/writable)
        - value constraints (minimum/maximum values, number of permitted updates/accesses)
    - outcomes/outputs/returned elements of workflow relevant data
    - audit data policy
        - operations to be audited
        - attributes to be audited
    - change control policy
5. security policy and implementation
    - authentication
        - workflow engines

- roles/user identities
- support for/ policy on non-repudiation
- shared key cryptography & key management
- handling security breaches

6. exception handling/recovery protocols & transactional behaviour
7. service level agreements (SLAs), metrication/escalation and performance penalties

The following is an example of how an interoperability contract might be constructed. Note that a semi-colon in column 1 denotes that the line contains a comment.

```
; Contract definition for proving Chained Interoperability
; capability

ContractName=BindingValidation

; Protocol being used
; One of I4MIME | CORBA | Wf-XML
Protocol=I4MIME

; List of workflow engines that are allowed to use this
; contract as a basis for effecting interoperability
; Note that a "*" indicates an open contract i.e. any engine can
; be traded with

ValidAddress=WorkflowEngineB

; Response timer - used for retransmission protocols for
; asynchronous messaging

Timer=24:00:00

; Attachment support indicator
; 0 - attachments are not supported
; 1 - part number referencing
; 2 - filename referencing
Attachments=0

; Specify the number of lost message retries
Retries=3

; Other configuration information
; …

; List of process definitions operated under this contract
; For each process that may be invoked, list process definition
; attributes whose values may be exchanged in order to support
; the process interoperability and list operations to be audited.
; For each attribute define:
;     1. Name
;     2. Type
;     3. Read/Write permission as R | W
;     4. Mandatory (prior to enactment) or optional as M | O
;     5. Audited as Y | N[12]
```

[12] New requirement for establishing audit policy

```
;     6. Default value (to be used if no value supplied)
; delimit each setting by a colon ":"


ValidProcessDefinitionID=ChainedOrderWidgets

ValidProcessDefinitionID=ChainedProcessOrder
Field=CustomerName:WMTText:W:M:Y:None
Field=CustomerAddress:WMTText:W:M:N:Unknown Address
Field=OrderDate:WMTDateTime:W:M:Y:
Field=ProductCode:WMTText:W:M:Y:None
Field=Quantity:WMTInt16:W:M:Y:0
;List operations for which audit data is required[13]
Audit=CreateProcessInstance
Audit=SetProcessInstanceAttributes
Audit=ProcessInstanceAttributeChanged
Audit=ProcessInstanceStateChanged
```

Individual interoperability contracts will have a unique `contract_id` identifier, determined by the organisations trading across the service boundary, which is used to support authentication mechanisms.

7.2 Session Management and Message Handling

For certain transport mechanisms e.g. CORBA, session and message handling is dealt with by the transport layer. For other transport mechanisms, e.g. basic Unix mail, it will be necessary to have some scheme which guarantees the integrity of the dialogue between two co-operating workflow engines. Where session management is a necessary part of the protocol described in a given binding, the binding document will give a clear account of how this is to be effected

7.3 Security Considerations

Implementation of security policy is seen as being primarily a matter for the workflow designer and the organization(s) which owns/operates the workflow domains. This Standards and its associated bingdings do provide building blocks that can be used to implement effective security policies. This section should be read in conjunction with the above text on the subject of Interoperability Contracts and in the light of the discussion presented in the Coalition's security white paper [WMC1019].

Authentication

Authentication is supported through the use of session management (where available) and through the use of the:

Workflow Engine Identifier

Process Instance Identifier

Activity Instance Identifier

Process Definition Identifier

User Identifier (optional)

Role Identifier (optional)

which can be used to enforce identification and access control as described in the prevailing interoperability contract.

[13] New requirement for implementing audit data policy – discuss with WG5.

Authorization & Access Control

The workflow designer must use the facilities offered by the deployed workflow product to validate the above information and police the external access that is exposed via this interface.

Audit

A workflow designer making product selection choices must choose products which implement the audit data requirements outlined in this document if they want to be able to have access to the level of audit data that the Workflow Management Coalition would recommend. Audit data is an optional part of this specification and some vendors may elect not to implement these facilities fully.

Data Privacy

There is currently no support for data privacy. This is seen as an area where use of secure LANs and corporate encryption algorithms should be considered. This area is problematical for definers of a Standards that is intended to be deployed internationally as there are different legal constraints in different countries. The Workflow Management Coalition is contemplating production of an S-MIME Interoperability Binding to this document. Whatever mechanisms are used to support data privacy, they must be compatible at both ends for interoperability to occur.

Data Integrity

Beyond the requirement for data privacy, data integrity is to be supported by mechanisms such as check-sum validation described in particular binding specification documents.

Non Repudiation

Support for non-repudiation will depend on effective implementation of Authentication and Audit Data policies as described above.

7.4 Bindings

Vendors will need to implement bindings to their workflow engines that correspond to this specification. Specifications of concrete bindings will be published separately from this standard and corresponding implementations must conform to these bindings.

8 Evaluation Criteria

8.1 Conformance Statements

In order that a user of workflow products may evaluate which workflow products are capable of interoperation with which other workflow products (workflow engine to workflow engine) and that they may have a basis for assessing the level of interoperability achievable between two particular workflow products, the following scheme is presented.

To enable a purchaser to match compatible workflow products from different vendors, each vendor should publish the interoperability capabilities of their product giving clear indication of:

1. the transport mechanism(s) it uses to effect interoperability with other workflow engines
2. the style(s) of interoperability dialogue it can support (batched, atomic or both)
3. the mode(s) of interoperability dialogue it employs (half-duplex or full duplex)

8.2 Capabilities

The main factor affecting the ability of two workflow engines to interoperate will be the capability enshrined in each of them to respond to messages they receive and to initiate requests and pass data as part of an interoperability dialogue. The objective is to be able to distinguish between workflow engines that:

- are entirely passive partners in an interoperability, only capable of receiving instructions to create and initiate the enactment of new process instances and acting on them
- are capable of passing/receiving workflow relevant data as well as receiving instructions to create and initiate the enactment of new process instances and acting on them
- are capable of asking for workflow relevant data as well as receiving instructions to create and initiate the enactment of new process instances and acting on them

The vendor of a workflow product should produce a capability matrix that, for each operation defined in section 5 of this document, shows whether their workflow engine can initiate the message associated with that operation and whether it can respond to it. e.g.

Operation	**Initiate**	**Respond**
Change Process Instance State	Yes	Yes
Create Process Instance	Yes	Yes
Get Process Instance Attributes	No	No
Get Process Instance State	Yes	Yes
Trigger Activity	Yes	Yes
List Process Instances	No	No
Process Instance Attribute Changed	No	No
Process Instance State Change	Yes	No
Relinquish Process Instance	No	No
Set Process Instance Attributes	Yes	Yes

Specific examples of capability matrices can be found in the Interoperability Proving Framework specification document [WMC1021]

A number of dialogue structures, based on the style of capability matrix defined above will be defined by the Workflow Management Coalition for evaluating the conformance of individual workflow engines to particular bindings. These *capability profiles* can be used to determine how two workflow engines that use the same transport can be used together.

To assess whether two workflow engines are capable of interoperating users should compare their capability matrices. To be capable of effecting the interoperability dialogue given in table 4.1 above, the two workflow engines would need to have capability matrices. For example:

	Workflow Engine A		**Workflow Engine B**	
Operation	**Initiate**	**Respond**	**Initiate**	**Respond**
Create Process Instance	Yes			Yes
Set Process Instance Attributes	Yes			Yes
Get Process Instance Attributes		Yes	Yes	
Process Instance Attribute Changed	Yes			Yes
Relinquish Process Instance	Yes			Yes

and to effect the interoperability dialogue given in 4.2 above, the two workflow engines would need to have capability matrices, for example:

	Workflow Engine A	**Workflow Engine B**

Operation	Initiate	Respond	Initiate	Respond
Create Process Instance	Yes			Yes
Set Process Instance Attributes	Yes			Yes
Get Process Instance Attributes	Yes	Yes	Yes	Yes
Process Instance Attribute Changed		Yes	Yes	
Relinquish Process Instance	Yes			Yes

These tables are examples of the proposed capability profiles that are the subject of ongoing work by the Workflow Management Coalition and will be published in due course.

9 References

[ICL95]	ICL/LOGICON/DL-003	Workflow Standards Interoperability Approaches (2.0)
[OMG93]	OMG 93.12.43	The Common Object Request Broker: Architecture and Specification (1.2)
[WfMC000]		Workflow Management Coalition Glossary
[WfMC1003]	WFMC TC00 - 1003	The Workflow Reference Model
[WfMC006]	WFMC TC01 - 1006	Interoperability White Paper
[WfMC0020]	WFMC TC-0020	Workflow Management Coalition Interface 1: Process Definition Interchange
[WfMC1009]	WFMC-TC-1009	Workflow Management Coalition Interface 2 Application Programming Interface (WAPI) Specification
[WfMC1013]	WFMC-TC-1013	Workflow Application Programmer's (WAPI) Interface 2 Naming Conventions
[WfMC0013]	WFMC-TC10-0013	Workflow Management Application Programming Interface (WAPI) Interface 3
[WfMC015]	WFMC TC-1015	Workflow Management Coalition Interface 5 Audit Data Specification
[WMC1019]	WFMC-TC-1019	Workflow Management Coalition Workflow Security Considerations – White Paper
[WMC1012]	WFMC-TC-1021	Workflow Management Coalition Interoperability Proving Framework, June 1999
[Tucker 95]		Tucker M., Baldwin C., Chase H. and MacDougall L., WfMC Compliance Paper, National Life of Vermont, May 1995.

