The Workflow Management Coalition Specification

Workflow Management Coalition Workflow Standard - Interoperability Abstract Specification

Document Number WFMC-TC-1012

20 October 1996
Version 1.0

This Specification has been authored by Workflow Management Coalition members.

Workflow Management Coalition
Avenue Marcel Thiry 204
1200 Brussels
Belgium
Tel: +32 2 774 96 33
Fax: +32 2 774 96 90
Email: 100113.1555@compuserve.com
or WfMC@eyam.be
www:http://www.aiai.ed.ac.uk/wfmc
or http://www.arms.ohio-state.edu/wfmc

1. Purpose

This document is a Workflow Management Coalition Standard providing an abstract specification which defines the functionality required to support interoperability between different workflow engines. It is intended that this document provide a basis for early experimentation by workflow product vendors to provide informed feedback, leading to the production of a ratified standard in the fullness of time. Workflow product vendors should use this document to understand the principles of how interoperability between workflow engines is effected using the WfMC Standards. They should then refer to specific transport binding specifications for details of how conformant implementations must work.

2. Audience

The intended audience for this document is the Workflow Management Coalition, as well as others that are interested in the efforts of the Coalition.

3. Scope

3.1. Scope of The Interoperability Specification

The Workflow Management Coalition Standard - 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.

3.2. Models of Interoperability

In its earlier work on the topic of interoperability (see [ICL95] and [WfMC006]), the Workflow Management Coalition has identified a number of different models of interoperability and a number of different levels against which vendors of workflow products may 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.

3.2.1. 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 3.1 below)

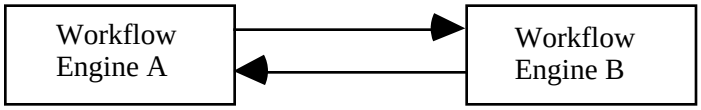

Figure 3.1 Direct interaction between workflow engines

two or more workflow engines operating within the same workflow enactment service (see figure 3.2 below)

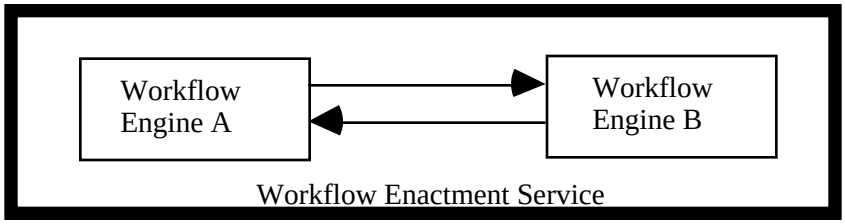

Figure 3.2. Interaction between workflow engines within an enactment service

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 3.3 below)

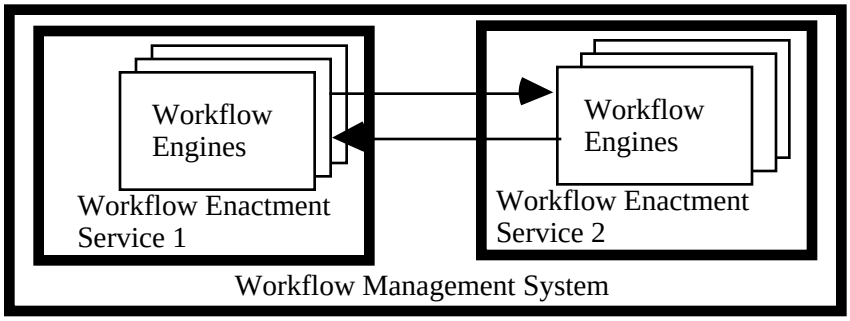

Figure 3.3 Interoperating workflow engines in different enactment services

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 workflow enactment services).

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 which 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 as a result of some exchange of verification tokens[1] in a business context.

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

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.

3.2.2. Effecting Interoperability

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 3.4 below)

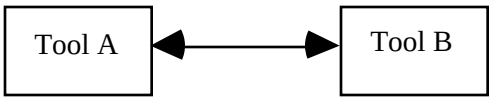

Figure 3.4 Direct interaction between software tools

2. message passing (see figure 3.5 below)

Figure 3.5 Software tools interacting by passing messages

3. bridging (using some form of encapsulation, translation or gateway mechanism as shown in figure 3.6 below)

Figure 3.6 Software tools interacting via a gateway that performs protocol transliteration

4. use of a shared data store (common repository - see figure 3.7 below).

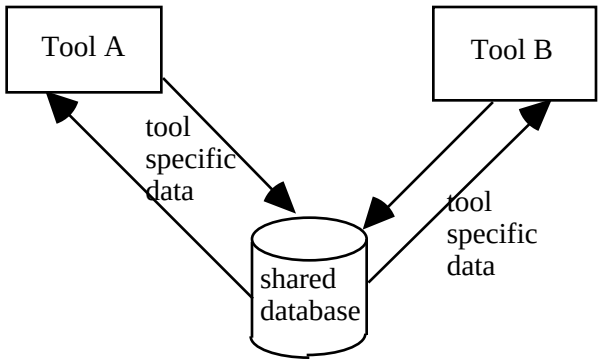

Figure 3.7 Software tools integrated via use of a common repository

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.

3.2.3. 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, 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 WAPI interfaces alongside workflow products that do not have WAPI interface capability.

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.

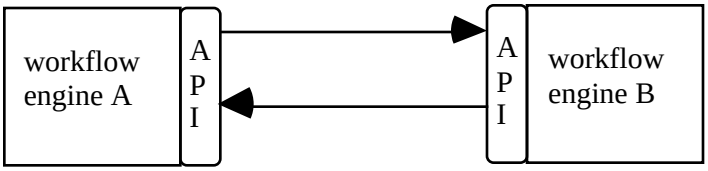

Figure 3.8 Workflow engines interoperating via API calls

or

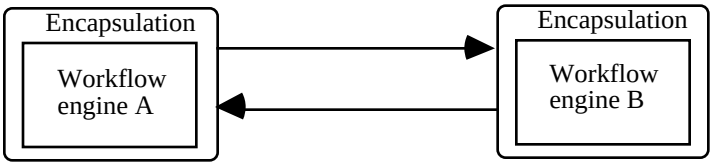

Figure 3.9 Encapsulated interoperating workflow engines

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 is in the process of defining 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:

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.

[WfMC0020]

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.

3.2.4. Models of Interoperability

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

1. Chained processes

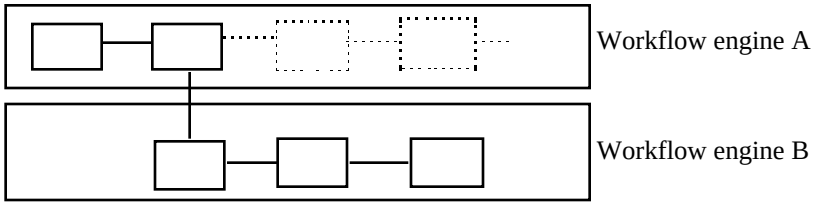

Figure 3.10 The chained model of interoperability

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.

2. Nested sub-process

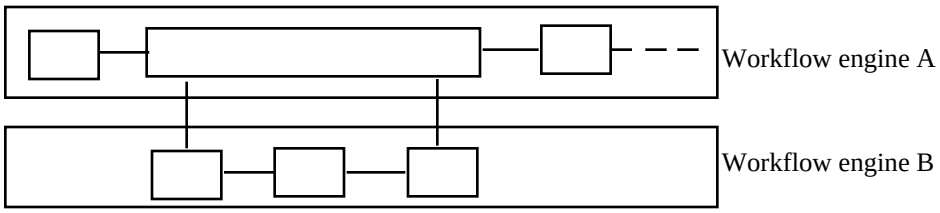

Figure 3.12. The nested sub-process model of interoperability

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 and then waits for its termination before carrying on with its own enactment.

3. Parallel synchronized

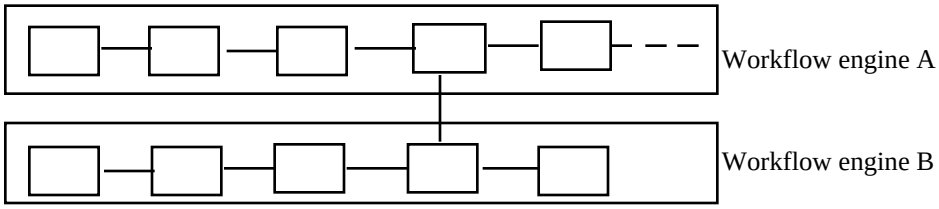

figure 3.13. The parallel synchronized model of interoperability

The parallel synchronized model of interoperability assumes that two workflow engines simultaneously enact process instances and that at some point in the definition of each of the process instances, a

rendezvous has been specified. Having achieved the rendezvous point, the first workflow engine (which engine is first is not specified) waits for the other to achieve its rendezvous point. Once the enactment of both process instances has achieved the respective rendezvous points, there is some (unspecified) interchange between the workflow engines, before both continue with the enactment of their respective process instances.

The parallel synchronized model of interoperability is outside the scope of what this standard is currently trying to achieve.

4. Overview

4.1 Design Assumptions

The basic design 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].

4.2 Design Objectives

The design 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.

4.2.1 Starting a chained sub-process

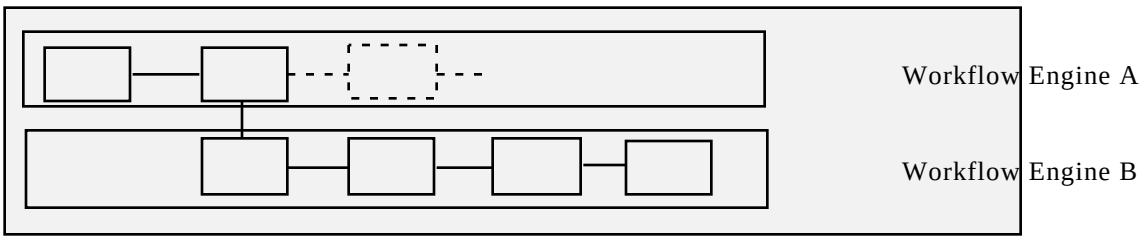

Figure 4.1. main process starts a chained sub-process on another workflow engine

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.

[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.

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.

In terms of the operations defined in section 5 below, the operations listed in table 4.1 would be used to effect the dialogue between two workflow engines required to support the interoperability shown in figure 4.1.

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
A	Start Process Enactment
B	Response to Start Process Enactment
A	Relinquish Process Instance
B	Response to Relinquish Process Instance

table 4.1: 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 Workflow 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 *Start Process Enactment* operation. 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
Start Process Enactment
Relinquish Process Instance

Workflow Engine B

Response to Set Process Instance Attributes
Response to Start Process Enactment
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.

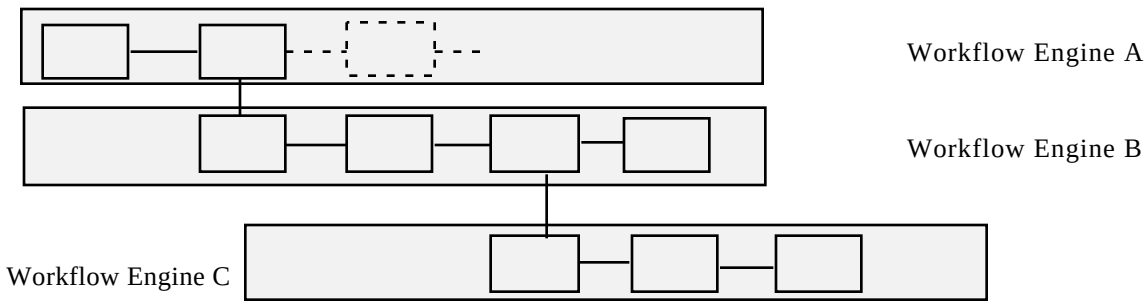

Figure 4.2. main process starts a chained sub-process on another workflow engine which in turn creates a chained sub-process on a third workflow engine

4.2.2 **Starting a sub-process that completes a process step**

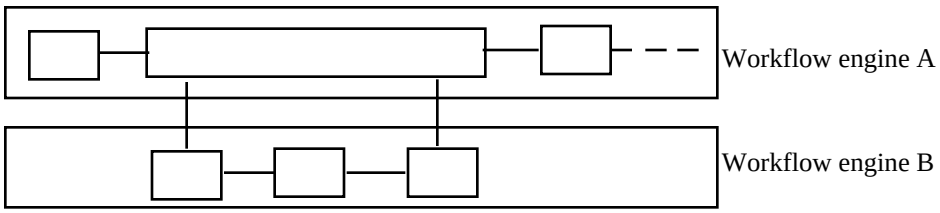

figure 4.3. a process instance starts a sub-process on another workflow engine and waits for completion

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 B	Create Process Instance Response to Create Process Instance
A B	Set Process Instance Attributes Response to Set Process Instance Attributes
B A	Get Process Instance Attributes Response to Get Process Instance Attributes
A B	Start Process Enactment Response to Start Process Enactment
B A	Notify Process Instance Attribute Changed Response to Notify Process Instance Changed
A B	Get Process Instance Attributes Response to Get Process Instance Attributes
A B	Relinquish Process Instance Response to Relinquish Process Instance

table 4.2: 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 Workflow 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 *Start Process Enactment* operation. Once enactment has started, Workflow Engine B will notify 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 *Notify Process Instance Completed* 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 Workflow 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
- Start Process Enactment

Workflow Engine B
- Response to Set Process Instance Attributes
- Response to Start Process Enactment

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].

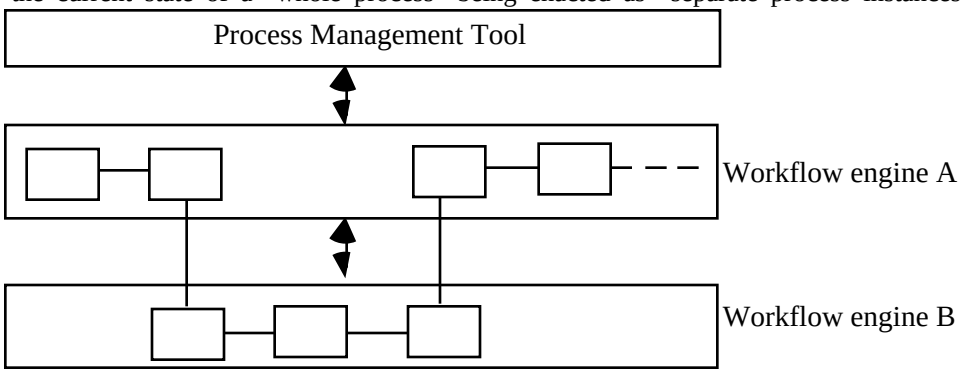

figure 4.4: use of a process management tool in a multiple workflow engine environment

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 4.3: additional operations required to support use of a process management tool in a multiple workflow engine environment

None of the above scenarios require support for rendezvous (required for parallel synchronized models of interoperability - see [WfMC006]), which is outside the current scope of this work.

4.3 Defined Terms and Abbreviations

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

4.4 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.

4.5 Naming Conventions

The operations are specified using the Object Management Group's Interface Definition Language (IDL) as given in [OMG93]. 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].

5. Specification of Operators for Effecting Interoperation

In the following text, the IDL specifications are intended as 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 possible that for workflow engines interoperating synchronously (say via an Object Request Broker) the IDL specifications will provide a basis from which implementation could be attempted. 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.

5.1. Connection Operations

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

5.2. Process Control Interactions

5.2.1 Create Process Instance

Specification

```
WMAReturnCode WMRequestCreateProcessInstance (
                    in engine_identifier        WMAEngineID,
                    in process_definition_id    WMAObjectID,
                    in return_flag              WMABool,
                    in parent_pid               WMAObjectID,
                    in activity_id              WMA_Activity_id,
                    out sub_process_id          WMA_Process_instance_id,
                    out user_id                 WMAResourceID,
                    out role_id                 WMAResourceID
                                );
```

Description This operation identifies a process definition which the receiving workflow engine will be required to enact.

Parameters

engine_identifier	identifies the target workflow engine and domain
process_definition_id[3]	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[4]
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

[3]The initial assumption for early implementations of this specification is that the value of the process_definition_id parameter has been provided by a human agent. It may be that later on more sophisticated ways of ascertianing the value of this parameter might be attempted, but the emphasis at this time is to keep things simple.

[4] 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.

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)

Return Values Success | Failure | Operation not performed

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 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 enacting workflow engine will not notify the initiating workflow engine of any state changes.

Request Message Format
Note that message routing information is deemed to be implementation/binding specific and outside the scope of the abstract specification.

Field	**Value**
CreateProcessInstance	
ProcessDefinitionID	Identifier of the process definition that the target workflow engine is required to use to create the process instance
ReturnFlag	True => nested sub-process False => chained
ParentPID	Identifier of initial (root) process on requesting workflow engine that requires the new process instance to be created
ProcessID	Identifier of process on requesting workflow engine that requires the new process instance to be created
ActivityID	Identifier of activity on requesting workflow engine that requires the new process instance to be created
Timestamp	Timestamp for when request was sent by requesting workflow engine
DomainID	ID of domain within which the workflow engines are interoperating
SourceNodeID	Identifier for requesting workflow engine
SourceUserID	User ID responsible for request (may be null)
SourceRoleID	Role ID responsible for request (may be null)

TargetUserID	User ID responsible for process instance on target workflow engine (may be null)[5]
TargetRoleID	Role ID responsible for process instance on target workflow engine (may be null)
SourceBusinessDefinitionProcessName	Business Definition Process Name for requesting process instance (may be null)
MessageRoutingInformation	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to CreateProcessInstance	
Return code	Success | Failure | No_operation
Timestamp	Timestamp for when requested process instance was created
ProcessID	ID of newly created process instance
UserID	User ID of primary user (may be null)[6]
RoleID	Role assumed by primary user (may be null)
TargetProcessBusinessDefinitionName	may be null
TargetState	State of new process instance
DomainID	Identifier for workflow domain in which the workflow engines are interoperating
TargetNodeID	Node of target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

Audit Data:

Request

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

Operation

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

---

[5] May not be known by source workflow engine

[6] Where interoperability is inter-organisational, it may not be appropriate to pass user ids across organiastion boundaries.

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

5.2.2 Get Process Instance State

Specification

```
WMAReturnCode WMRequestGetProcessInstanceState  (
                                in engine_identifier     WMAEngineID,
                                in process               WMAObjectID,
                                out state                WMAObjectState
                                        );
```

Description Get the current status of a given process instance which is being enacted by another workflow engine.

Parameters engine_identifier identifies the target workflow engine

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

state — the returned state[7] - a string prefixed by one of:
- open.not-running[8]
- open.running[9]
- closed.completed
- closed.terminated
- closed.aborted

Return Values Success | Failure | Operation_not_performed | Operation_not_implemented

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.

Request Message Format

Field	Value
GetProcessInstanceState	
ProcessID	Identifier of process on target workflow engine for which the state is being requested
DomainID	Identifier for workflow domain within which the workflow engines are interoperating
SourceNodeID	Identifier for source workflow engine
MessageRoutingInformation	Key values used to route the message to its correct destination

---

[7] 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.

[8] Created but not yet started

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

Response Message Format

Field	Value
ResponsetoGetProcessInstanceState	
ReturnCode	Success | Failure | Operation_not_performed | Operation_not_implemented
ProcessID	Identifier of process on target workflow engine for which the state is being requested
State	State of process instance
DomainID	Identifier for workflow domain within which the workflow engines are interoperating
TargetNodeID	Identifier for target workflow engine
MessageRoutingInformation	Key values used to route the message to its correct destination

Audit Data: Not Specified

5.2.3 **Set Process Instance Attributes**

Specification

```
WMAReturnCode WMRequestSetProcessInstance Attributes (
                    in engine_identifier    WMAEngineID,
                    in root_pid             WMAObjectID,
                    in activity_id          WMAObjectID,
                    in process_id           WMAObjectID,
                    out attributes          WMAAttributeList
                                );
```

Description Sets the value(s) of process instance attributes (process relevant data) in a selected process definition. 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 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 length of the attribute the value to which the attribute is to be set and returning: the name of each attribute set

Return Values Success | Failure | Operation_not_performed | Operation_not_implemented

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

Request Message Format

Field	Value
SetProcessInstanceAttributes	
RootPID	PID of root workflow process instance
ProcessID	The PID of the process instance on the target workflow engine for which changes to attribute values are being requested.
ActivityID	Activity ID identifying the process step within the process instance that is requesting the setting of the specified attributes
Number	The number of Attributes for which value changes are required.
Name[10]	attribute identifier
Type	attribute type
Length	new attribute length
Value	new attribute value
...	...
DomainID	Identifier for the workflow domain within which the workflow engines are interoperating
SourceNodeID	Identifier for source workflow engine
MessageRoutingInformation	Key values used to route the message to its correct destination

Response Message Format

Field	Value
ResponsetoSetProcessInstanceAttributes	Message identifier assigned by target workflow engine or by the transport mechanism
ReturnCode	Success | Failure | Operation_not_performed | Operation_not_implemented
ProcessID	The PID of the process instance on the target workflow engine for which changes to attribute values were requested.
Number	The number of Attributes for which value changes have succeeded (should be equal to the number of attributes for which changes were requested).
Name[11]	The name of an attribute that has had its value changed
Timestamp	Time when this attribute value was changed
...	...

[10] Information repeated for each attribute value to be set.
[11] Information repeated for each attribute value set.

DomainID	Identifier for the workflow domain within which the workflow engines are interoperating
TargetNodeID	Identifier for target workflow engine
MessageRoutingInformation	Key values used to route the message to its correct destination

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

5.2.4 Get Process Instance Attributes

Specification

```
WMAReturnCode WMRequestGetProcessInstanceAttributes (
                    in engine_identifier   WMAEngineID,
                    in process_id          WMAObjectID,
                    in root_pid            WMAObjectID,
                    in activity_id         WMAObjectID,
                    out attributes         WMAAttributeList
                                                    );
```

Description Returns the value(s)[12] of the requested process instance attributes (process relevant data).

Parameters 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 length of the attribute the value to which the attribute is to be set

Return Values Success | Failure | Operation not performed

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.

Request Message Format

Field	Value
GetProcessInstanceAttributes	
RootPID	Identifier of initial (root) process on source workflow engine
ProcessID	The PID of the process instance on the target workflow engine from which attribute values are being requested.
ActivityID	Identifier of activity on source workflow engine
Timestamp	Timestamp for when request was sent by source workflow engine

[12] A working assumption is that for efficiency reasons it should be possible to obtain sets of workflow relevant data in a single interaction.

SourceUserID	User ID responsible for request (maybe null)
SourceRoleID	Role ID responsible for request (may be null)
TargetUserID	User ID responsible for process instance on target workflow engine (may be null)
TargetRoleID	Role ID responsible for process instance on target workflow engine (may be null)
SourceBusinessDefinitionProcessName	Business Definition Process Name for source process instance (may be null)
Number	The number of Attributes for which values are being requested.
Name[13]	attribute identifier
...	...
DomainID	ID of domain within which the workflow engines are interoperating
SourceNodeID	Identifier for source workflow engine
MessageRoutingInformation	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response toGetProcessInstanceAttributes	
Return Code	Success | Failure | No_operation
ProcessID	The PID of the process instance on the source workflow engine for which attribute values were requested.
Number	The number of Attributes for which values have been supplied
Name[14]	attribute identifier
Type	attribute type
Length	new attribute length
Value	new attribute value
...	...
DomainID	Identifier for the workflow domain within which the two workflow engines are interoperating
TargetNodeID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

[13] Information repeated for each attribute value set.
[14] Information repeated for each attribute value to be set.

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

5.2.5 **Start Process Instance**

Specification

```
WMAReturnCode WMRequestStartProcessInstance (
                    in engine_identifier   WMAEngineID,
                    in process_id          WMAObjectID
                    );
```

Description Start the enactment of the identified workflow process instance on another workflow engine. Both workflow engines create appropriate audit records as specified in [WfMC015] to record that enactment of the process instance has started.

Parameters engine_identifier identifies the target workflow engine

process_id — the id of the process instance for which enactment is to be started

Return Values Success | Failure | Operation_not_performed | Operation_not_implemented

Rationale It is an operational requirement of users of interoperating workflow systems that it be possible to initiate the enactment of sub-process workflows.

Request Message Format

Field	**Value**
StartProcessInstance	
ProcessID	The PID of the Process Instance that the target workflow engine is being requested to enact
DomainID	Domain ID for set of workflow engines interoperating within the current context
SourceNodeID	Identifier for source workflow engine
SourceUserID	User ID responsible for request (maybe null)
SourceRoleID	Role ID responsible for request (may be null)
TargetUserID	User ID responsible for process instance on target workflow engine (may be null)
TargetRoleID	Role ID responsible for process instance on target workflow engine (may be null)
MessageRoutingInformation	Key values used to route the message to its correct destination

Response Message Format

Field	Value
Response to StartProcessInstance	
ReturnCode	Success | Failure | No_operation
TargetUserID	User ID of primary user (may be null)
TargetRoleID	Role assumed by primary user (may be null)
ProcessID	The PID of the Process Instance that the target workflow engine was requested to enact
ActivityID	The ID of the first activity in the enacted process instance (may be null)
Timestamp	The time that enactment commenced.
DomainID	Domain ID for set of workflow engines interoperating within the current context
TargetNodeID	Identifier for target workflow engine
MessageRoutingInformation	Key values used to route the message to its correct destination

Audit Data

The following audit data records would be created as a result of the target workflow engine successfully commencing enactment of the designated process instance at the behest of the requesting workflow engine.

Request

Requesting Workflow Engine: Link to Remote Subprocess Audit Data
Event Code: WMSentRequestStartProcessInstance

Target Workflow Engine: Link from Remote Subprocess Audit Data
Event Code: WMReceivedRequestStartProcessInstance

Operation

Requesting Workflow Engine: Remote Process Operations Audit Data
Event Code: WMStartedProcessInstance

Target Workflow Engine: Create/Start Process/Subprocess Instance
Event Code: WMStartedProcessInstance

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

5.2.6 **Process Instance Started**

Specification

```
WMAReturnCode WMNotifyProcessInstanceStarted (
                                in engine_identifier    WMAEngineID,
                                 in process_id          WMAObjectID
                                );
```

Description Notifies the invoking workflow engine that the enactment of the given process instance has started on the target workflow engine. This operation is necessary for situations where the process instance has been created on the host at the behest of another workflow engine, but not started. It may be that enactment of such a process instance would be started once certain pre-conditions have been satisfied. In such a circumstance, it would be appropriate for the host workflow engine to notify the other workflow engine when enactment of the process instance had started.

Parameters engine_identifier identifies the invoking workflow engine

process_id the id of the process instance that has started

Return Values Success | Failure

Rationale In certain circumstances the invoking workflow engine may need to know when enactment of the given sub-process has begun on the target workflow engine.

Notification Message Format

Field	**Value**
ProcessInstanceStarted	
ProcessID	The PID of the Process Instance on the source workflow engine that has started
RemotePID	The PID of the parent process instance on the target workflow engine
RemoteAID	The ID of the activity instance within the process instance on the remote workflow engine that is to be notified that the sub-process instance has started.
SourceUserID	User ID responsible for the process instance that caused the notification request (may be null)
SourceRoleID	Role ID responsible for the process instance that caused the notification request (may be null)
TargetUserID	User ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be

	null)
TargetRoleID	Role ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
Timestamp	Time enactment of the process instance began
DomainID	ID of domain within which the workflow engines are interoperating
SourceNodeID	Identifier for source workflow engine
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to ProcessInstanceStarted	
DomainID	Domain ID for set of workflow engines interoperating within the current context
TargetNodeID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

Audit Data

The following audit data records would be created as a result of the notifying workflow engine starting enactment of a sub-process on behalf of the target workflow engine.

Actual Event

Notifying Workflow Engine:	Create/Start Process/Subprocess Instance Event Code: WMStartedProcessInstance

Notification

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

5.2.7 **Abort Process Instance**

Specification

```
WMAReturnCode WMRequestAbortProcessInstance (
                                    in engine_identifier    WMAEngineID,
                                    in process_id           WMAObjectID,
                                                        );
```

Description Abort the enactment of the identified workflow process instance on another workflow engine. Both workflow engines create appropriate audit records as specified in [WfMC015] to record that the process instance was aborted.

Parameters engine_identifier identifies the target workflow engine

process_id the id of the process instance that is to be aborted

Return Values Success | Failure | Operation not performed

Rationale It is an operational requirement of users of interoperating workflow systems that it be possible to abort the enactment of sub-process workflows when this becomes necessary.

Request Message Format

Field	**Value**
AbortProcessInstance	
ProcessID	The PID of the Process Instance that the target workflow engine is being requested to abort
SourcePID	The ID of the process instance that has caused the request to abort the sub-process instance (may be null)
SourceAID	The ID of the activity in the current process instance that has caused the request to abort the sub-process instance (may be null)
DomainID	Domain ID for set of workflow engines interoperating within the current context
SourceNodeID	Identifier for source workflow engine
SourceUserID	User ID responsible for request (maybe null)
SourceRoleID	Role ID responsible for request (may be null)
TargetUserID	User ID responsible for process instance on target workflow engine (may be null)
TargetRoleID	Role ID responsible for process instance on target

	workflow engine (may be null)
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to AbortProcessInstance	
ReturnCode	Success | Failure | No_operation
TargetUserID	User ID of primary user (may be null)
TargetRoleID	Role assumed by primary user (may be null)
ProcessID	The PID of the Process Instance that the target workflow engine was requested to abort
ActivityID	The ID of the current activity in the enacted process instance that has been aborted (may be null)
Timestamp	The time that enactment was aborted.
DomainID	Domain ID for set of workflow engines interoperating within the current context
TargetNodeID	Identifier for target workflow engine
MessageRoutingInformation	Key values used to route the message to its correct destination

Audit Data

The following audit data records would be created as a result of the target workflow engine successfully aborting enactment of the designated process instance at the behest of the requesting workflow engine[15].

Request

Requesting Workflow Engine:	Link to Remote Process Audit Data Event Code: WMSentRequestAbortProcessInstance
Target Workflow Engine:	Link from Remote Subprocess Audit Data Event Code: WMReceivedRequestAbortProcessInstance

Operation

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

Response

Target Workflow Engine:	Link to Remote Subprocess Audit Data Event Code: WMSentAbortedProcessInstance
Requesting Workflow Engine:	Link from Remote Subprocess Audit Data Event Code: WMReceivedAbortedProcessInstance

[15] Do we want to track failed attempts to abort a process instance in the audit data?

5.2.8 **Process Instance Aborted**

Specification

```
WMAReturnCode WMNotifyProcessInstanceAborted (
                                    in engine_identifier    WMAEngineID,
                                     in process_id          WMAObjectID
                                        );
```

Description Notifies the invoking workflow engine that the enactment of the given process instance has been aborted locally.

Parameters

engine identifier	identifies the invoking workflow engine
process_id	the id of the process instance

Return Values Success | Failure

Rationale In certain circumstances the invoking workflow engine may need to know if the enactment of a sub-process has been aborted on the target workflow engine.

Notification Message Format

Field	**Value**
ProcessInstanceAborted	
ProcessID	The PID of the Process Instance on the source workflow engine that has aborted
ActivityID	The AID of the current activity instance in the process instance at the time it aborted
RemotePID	The PID of the parent process instance on the target workflow engine
RemoteAID	The ID of the activity instance within the process instance on the remote workflow engine that is to be notified that the sub-process instance has aborted.
Timestamp	Time enactment of the process instance was aborted
DomainID	Domain ID for set of workflow engines interoperating within the current context
Source Node ID	Identifier for source workflow engine
SourceUserID	User ID responsible for the process instance that caused the notification request (may be null)

SourceRoleID	Role ID responsible for the process instance that caused the notification request (may be null)
TargetUserID	User ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
TargetRoleID	Role ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to ProcessInstanceAborted	
DomainID	Domain ID for set of workflow engines interoperating within the current context
TargetNodeID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

Audit Data

The following audit data records would be created as a result of the notifying workflow engine aborting enactment of a sub-process created at the behest of the target workflow engine.

Actual Event

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

Notification

Notifying Workflow Engine:	Link to Remote Process Operations Audit Data Event Code: WMSentAbortedProcessInstance
Target Workflow Engine:	Link from Remote Process Operations Audit Data Event Code: WMReceivedAbortedProcessInstance

5.2.9 **Terminate Process Instance**

Specification

```
WMAReturnCode WMRequestTerminateProcessInstance (
                    in engine_identifier   WMAEngineID,
                    in process_id          WMAObjectID
                                                );
```

Description Terminate the enactment of a workflow process instance on another workflow engine. Both workflow engines create appropriate audit records as specified in [WfMC015] to record that enactment of the process instance has been terminated.

Parameters

engine identifier	identifies the target workflow engine
process_id	the id of the process instance that is to be terminated

Return Values Success | Failure | Operation not performed

Rationale It is an operational requirement of users of interoperating workflow systems that it be possible to gracefully terminate the enactment of sub-process workflows prior to their natural completion when this becomes necessary.

Request Message Format

Field	**Value**
TerminateProcessInstance	
ProcessID	The PID of the Process Instance that the target workflow engine is being requested to terminate
SourceAID	The ID of the activity in the current process instance that has caused the request to terminate the sub-process instance (may be null)
DomainID	Domain ID for set of workflow engines interoperating within the current context
Source Node ID	Identifier for source workflow engine
SourceUserID	User ID responsible for request (maybe null)
SourceRoleID	Role ID responsible for request (may be null)
TargetUserID	User ID responsible for process instance on target workflow engine (may be null)
TargetRoleID	Role ID responsible for process instance on target workflow engine (may be null)
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to TerminateProcessInstance	
TargetUserID	User ID of primary user (may be null)
TargetRoleID	Role assumed by primary user (may be null)
ProcessID	The PID of the Process Instance that the target workflow engine was requested to terminate
ActivityID	The ID of the current activity in the enacted process instance that has been terminated (may be null)
Timestamp	The time that enactment was terminated.
DomainID	Domain ID for set of workflow engines interoperating within the current context
Target Node ID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

Audit Data

The following audit data records would be created as a result of the target workflow engine successfully terminating enactment of the designated process instance at the behest of the requesting workflow engine.

Request

Requesting Workflow Engine:	Link to Remote Process Audit Data Event Code: WMSentRequestTerminateProcessInstance
Target Workflow Engine:	Link from Remote Subprocess Audit Data Event Code: WMReceivedRequestTerminateProcessInstance

Operation

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

Response

Target Workflow Engine:	Link to Remote Subprocess Audit Data Event Code: WMSentTerminatedProcessInstance
Requesting Workflow Engine:	Link from Remote Subprocess Audit Data Event Code: WMReceivedTerminatedProcessInstance

5.2.10 Process Instance Terminated

Specification

```
WMAReturnCode WMNotifyProcessInstanceTerminated (
                    in engine_identifier    WMAEngineID,
                    in process_id           WMAObjectID
                                        );
```

Description Notifies the invoking workflow engine that the enactment of the given process instance has been prematurely terminated locally.[16]

Parameters engine_identifier identifies the invoking workflow engine

process_id the id of the process instance

Return Values Success | Failure

Rationale In certain circumstances the invoking workflow engine may need to know if the enactment of the sub-process has been terminated prematurely.

Notification Message Format

Field	Value
Process Instance Terminated	
ProcessID	The PID of the Process Instance on the source workflow engine that has terminated
ActivityID	The AID of the current activity instance in the process instance at the time it was terminated
Remote Process Instance ID	The PID of the parent process instance on the target workflow engine
RemoteAID	The ID of the activity instance within the process instance on the remote workflow engine that is to be notified that the sub-process instance has been terminated.
Timestamp	Time enactment of the process instance was terminated
DomainID	Domain ID for set of workflow engines interoperating within the current context
Source Node ID	Identifier for source workflow engine
SourceUserID	User ID responsible for the process instance that caused the notification request (may be null)

[16] There is a question about whether we need this operation/state. It can be argued that the states process competed and process aborted are sufficient. Comments please.

SourceRoleID	Role ID responsible for the process instance that caused the notification request (may be null)
TargetUserID	User ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
TargetRoleID	Role ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to ProcessInstanceTerminated	
DomainID	Domain ID for set of workflow engines interoperating within the current context
TargetNodeID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

Audit Data

The following audit data records would be created as a result of the termination 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: WMTerminatedProcessInstance

Notification

Notifying Workflow Engine:	Link to Remote Process Operations Audit Data Event Code: WMSentTerminatedProcessInstance
Target Workflow Engine:	Link from Remote Process Operations Audit Data Event Code: WMReceivedTerminatedProcessInstance

5.2.11 **Change Process Instance State**

Specification

```
WMAReturnCode WMRequestChangeProcessInstanceState (
                    in engine_identifier   WMAEngineID,
                    in process_id          WMAObjectID,
                    in state               WMAObjectState
                    );
```

Description Change the state of a designated process instance which is enacted on another workflow engine from for example, suspended to resumed or vice versa. Both workflow engines create appropriate audit records as specified in [WfMC015] to record that enactment of the suspended process instance was suspended or resumed.

Parameters engine_identifier identifies the target workflow engine

process_id	the id of the process instance that is to be suspended or resumed
state	the new state to which the process instance is to be switched

Return Values Success | Failure | Operation not performed

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

Request Message Format

Field	**Value**
Change Process Instance State	
ProcessID	The PID of the Process Instance on the target workflow engine for which the state change is being requested
NewState	The state to which the designated process instance is to be changed
SourcePID	The ID of the process instance that has caused the request to change the state of the sub-process instance (may be null)
SourceAID	The ID of the activity in the current process instance that has caused the request to change the state of the sub-process instance (may be null)
SourceUserID	User ID responsible for request (maybe null)
SourceRoleID	Role ID responsible for request (may be null)

DomainID	Domain ID for set of workflow engines interoperating within the current context
Source Node ID	Identifier for source workflow engine
SourceUserID	User ID responsible for request (maybe null)
SourceRoleID	Role ID responsible for request (may be null)
TargetUserID	User ID responsible for process instance on target workflow engine (may be null)
TargetRoleID	Role ID responsible for process instance on target workflow engine (may be null)
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to Change Process Instance State	
TargetUserID	User ID of primary user (may be null)
TargetRoleID	Role assumed by primary user (may be null)
ProcessID	The PID of the Process Instance that the target workflow engine for which the state change was requested
ActivityID	The ID of the current activity in the enacted process instance at the time the state change occurred (may be null)
Timestamp	The time that the state of the process instance was changed.
DomainID	Domain ID for set of workflow engines interoperating within the current context
Target Node ID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

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

5.2.12 **Process Instance Completed**

Specification

```
WMAReturnCode WMNotifyProcessInstanceCompleted (
                                in engine_identifier    WMAEngineID,
                                in process_id           WMAObjectID
                                    );
```

Description Notify the invoking workflow engine that the enactment of the given process instance has completed normally.

Parameters engine_identifier identifies the invoking workflow engine

process_id the id of the process instance that has completed

Return Values Success | Failure

Rationale In circumstances where the invoking process instance hangs, waiting for the enacted sub-process to complete, the workflow engine enacting the sub-process must have a means of communicating the completion to the invoking engine.

Notification Message Format

Field	**Value**
ProcessInstanceCompleted	
ProcessID	The PID of the Process Instance on the source workflow engine that has completed
ActivityID	The AID of the current activity instance in the process instance at the time it completed
RemotePID	The PID of the parent process instance on the target workflow engine
RemoteAID	The ID of the activity instance within the process instance on the remote workflow engine that is to be notified that the sub-process instance completed.
Timestamp	Time enactment of the process instance completed
DomainID	Domain ID for set of workflow engines interoperating within the current context
Source Node ID	Identifier for source workflow engine
SourceUserID	User ID responsible for the process instance that caused the notification request (may be null)
SourceRoleID	Role ID responsible for the process instance that caused the notification request (may be null)

TargetUserID	User ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
TargetRoleID	Role ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to Process Instance Completed	
DomainID	Domain ID for set of workflow engines interoperating within the current context
Target Node ID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

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: WMCompletedProcessInstance

Notification

Notifying Workflow Engine:	Link to Remote Process Operations Audit Data Event Code: WMSentCompletedProcessInstance
Target Workflow Engine:	Link from Remote Subprocess Audit Data Event Code: WMReceivedCompletedProcessInstance

5.2.13 **List Process Instances**

Specification

```
WMAObjectIDList WMRequestListProcessInstances (
                    in engine_identifier    WMAEngineID,
                    in filter               WMAFilter
                                          );
```

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

Parameters engine_identifier identifies the invoking workflow engine

filter	arbitrary criteria for the selection of process instances (e.g. PID, user, definition id, ...)

Return Values A list (which may be an empty list) of Process instance IDs

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.

Request Message Format

Field	**Value**
List Process Instances	
Filter	Filter identifying process instances to be reported on

Response Message Format

Field	**Value**
Message Number	Message identifier assigned by the target workflow engine or by the transport mechanism
Response to Get Process Instances	
Number of instances	The number of process instances being reported on
Process instance ID[17]	ID of a process instance satisfying filter criteria

Audit Data: Not specified

[17] Repeated as required by Number of Process Instances field

5.2.14 **Process State Changed**

Specification

```
WMAReturnCode WMNotifyProcessInstanceStateChange (
                    in engine_identifier    WMAEngineID,
                     in process_id           WMAObjectID,
                    in new_state            WMAObjectState
                                                    );
```

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

Parameters

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

Return Values Success | Failure

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.

Notification Message Format

Field	**Value**
ProcessInstanceStateChange	
ProcessID	The PID of the Process Instance on the source workflow engine
ActivityID	The AID of the current activity instance in the process instance at the time the state change occurred
RemotePID	The PID of the parent Process Instance on the target workflow engine
RemoteAID	The ID of the activity instance within the process instance on the remote workflow engine that is to be notified that the sub-process instance state has changed.
Timestamp	Time process instance underwent a state change
State	New state of process instance

DomainID	Domain ID for set of workflow engines interoperating within the current context
Source Node ID	Identifier for source workflow engine
SourceUserID	User ID responsible for request (maybe null)
SourceRoleID	Role ID responsible for request (may be null)
TargetUserID	User ID responsible for process instance on target workflow engine (may be null)
TargetRoleID	Role ID responsible for process instance on target workflow engine (may be null)
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to ProcessInstanceStateChanged	
DomainID	Domain ID for set of workflow engines interoperating within the current context
Target Node ID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

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

5.2.15 **Process Attributes Changed**

Specification

```
WMAReturnCode WMNotifyProcessAttributesChanged (
                        in engine_identifier    WMAEngineID,
                        in process_id           WMAObjectID,
                        in attributes           WMAAttributeList
                                                );
```

Description Sets the value(s) of process instance attributes (process relevant data) in a selected process definition.

Parameters 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 length of the attribute the value to which the attribute has been set

Return Values Success | Failure

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.

Notification Message Format

Field	**Value**
ProcessInstanceAttributesChanged	
ProcessID	The PID of the process instance on the source workflow engine for which the attribute value has changed
ActivityID	The AID of the current activity instance in the process instance at the time the attribute value changed
RemotePID	The PID of the parent Process Instance on the target workflow engine
RemoteAID	The ID of the activity instance within the process instance on the remote workflow engine that is to be notified that the sub-process instance completed.
Timestamp	The time when the attribute value changed

DomainID	Domain ID for set of workflow engines interoperating within the current context
Source Node ID	Identifier for source workflow engine
Name[18]	attribute identifier
Type	attribute type
Length	new attribute length
Value	new attribute value
...	...
SourceUserID	User ID responsible for the process instance that caused the notification request (may be null)
SourceRoleID	Role ID responsible for the process instance that caused the notification request (may be null)
TargetUserID	User ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
TargetRoleID	Role ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to ProcessInstanceAttributesChanged	
DomainID	Domain ID for set of workflow engines interoperating within the current context
Target Node ID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

[18] Information repeated for each attribute value changed

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

5.2.16 **Relinquish Process Instance**

Specification

```
WMAReturnCode WMRelinquishProcessInstance (
                                in engine_identifier    WMAEngineID,
                                in process_id           WMAObjectID
                                                                );
```

Description **N**otify another workflow engine that as far as this workflow engine is concerned, it may now release all memory containing data structures pertaining to the given process instance and/or not to send notification messages concerning the enactment of that process instance.

Parameters

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

Return Values Success | Failure

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.

Notification Message Format

Field	**Value**
Relinquish Process Instance	
ProcessID	The PID of the Process Instance on the target workflow engine
DomainID	Identifier for the workflow domain within which the two workflow engines are interoperating
SourceNodeID	Identifier for source workflow engine
SourceUserID	User ID responsible for the process instance that caused the notification request (may be null)
SourceRoleID	Role ID responsible for the process instance that caused the notification request (may be null)
TargetUserID	User ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
TargetRoleID	Role ID responsible for the process instance on the target workflow engine which caused the newly started process instance to be created (may be null)
Message Routing Information	Key values used to route the message to its correct destination

Response Message Format

Field	**Value**
Response to RelinquishProcessInstance	
Return Code	Success | Failure
DomainID	Identifier for the workflow domain within which the two workflow engines are interoperating
Target Node ID	Identifier for target workflow engine
Message Routing Information	Key values used to route the message to its correct destination

Audit Data: Not specified.

6. Implementation Issues

6.1 Session Management and Message Handling

For certain transport mechanisms e.g. MAPI, 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. The following text is a proposal for such a mechanism.

It is assumed that in some way messages carrying requests for operations to be performed and responses to those requests are passed between two or more interoperating workflow engines. The base assumption is that for every message there is a response. Messages are passed between interoperating workflow engines during a session. There are two styles of interoperability that it is possible to effect. The first of these uses *atomic transmission*, where a dialogue between two workflow engines is achieved by swapping messages one for one i.e.

A makes a request of B

B replies telling A whether the request was successful or not

A makes a request of B

B replies telling A whether the request was successful or not

B makes a request of A

A replies telling B whether the request was successful or not
....

This style of dialogue is characteristic of interoperability between two workflow engines communicating synchronously. It may, where appropriate, also be employed between two workflow engines communicating asynchronously. The main constraint here is the requirement for an adequate response time.

An alternative approach that may be employed for conducting dialogues between workflow engines is best characterized as *batched* transmission. Workflow engines that communicate asynchronously (e.g. via electronic mail) may batch up groups of request messages and send them as a *session*[19] to the target workflow engine for processing. The target workflow engine would process each message in turn and return either a series of response messages, one for each message sent, or return a batch of response messages - again one for each message sent.

It is important to be clear about what happens if the receiving workflow engine fails to successfully carry out a requested operation. Because we cannot assume use of transactions and the ability to roll back a batch, the best that can be achieved is:

- processing of the batch of requests stops

[19]There are thus two kinds of session - those bracketing batches of requests and batches of responses and those deliniating dialogues where the two workflow engines are "logged on" to one another.

- a message reporting the failed request is constructed and appended to the list of response messages for successful operation requests
- a message indicating operation not performed is constructed and added to the list of response messages for each request not yet performed
- a stop session message is appended to the list of response messages which is then returned to the initiating workflow engine.

It is desirable that the workflow designer manages the number of operation requests in a batch so that in such circumstances, the complexity of the task of repairing the state of the enacted workflow is not unduly complicated. This aside, the situation is not substantially different to when an operation request fails in an interoperability dialogue that uses atomic transmission.

Each session has a two part session identifier. The initiating workflow engine assigns a "source session id" and the target workflow engine (the one that the initiating workflow engine wishes to start a session with) assigns a "target session id". If the source engine alone sets the session id, then there is the possibility (albeit remote) that a target workflow engine working to multiple source engines could receive the same session id from more than one source. Similarly, if the target engine alone sets the session id, then a source workflow engine, working to multiple targets, could receive back the same session id from more than one engine. To overcome this problem, a truly unique session id can be achieved by having both engines provide a session id component that is to each, unique. The combined source/target pairing of session ids is then guaranteed to be unique for both engines. Session ids would be handled in some concrete data structure represented in the specifications given in Section 5 above by the abstract data type *WMAEngineID*.

Messages within a session are uniquely numbered. One way of doing this is to treat the enactment of a sub-process as the sole subject of a session and to increment message numbers through the life of the sub-process instance. The rationale behind this is that it is necessary to protect against messages being lost or delivered out of sequence. The receiving workflow engine must be able to identify:

- messages it has received and acted on
- messages received but because they are out of sequence, they are not yet eligible for processing
- messages that have not been received (e.g. if only messages 1, 2 and 5 have been received it is possible to deduce that messages 3 and 4 are missing).

The semantic of the message number is either *"Message n I have sent to engine X during session S"* or *"Message n I have received from engine X during session S"*. To allow each engine to distinguish messages it has sent and messages it has received, both engines will number messages they send to the other engine in increments of 2, starting from zero. Responses to request/notification messages are numbered by adding 1 to the request/notification message number.

To avoid the possibility that within a session two request/notification messages could have the same number[20], each engine is required to establish a session with the other (one for incoming requests and one for outgoing requests).

[20] Only possible for atomic messaging scenarios - so the following text does not apply to workflow engines interoperating in batch mode.

Using the above concepts it is possible to guarantee:

1. that in an environment where there may be many workflow engines conducting interoperability sessions with each other, every session is uniquely identifiable

2. that within such an environment every message is uniquely identifiable.

6.2 Security Considerations

It is assumed that the implementation of security policies for the administration of interoperating workflow engines is outside the scope of this standard. Rather, this is seen as a matter for the workflow designer and the organization(s) which owns/operates the workflow domains.

6.3 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..

7. Evaluation Criteria

7.1 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)

7.2. 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**
WMRequestCreateProcessInstance	Yes	Yes
WMRequestGetProcessInstanceState	Yes	Yes
WMRequestSetProcessInstanceAttributes	Yes	Yes
WMRequestGetProcessInstanceAttributes	No	No
WMRequestStartProcessInstance	Yes	Yes
WMRequestAbortProcessInstance	Yes	Yes
WMRequestTerminateProcessInstance	Yes	No
WMRequestChangeProcessInstanceState[21]	Yes	Yes

[21] It will be necessary to declare any implementation constraints on the set of state changes that are supported

WMRequestListProcessInstances	No	No
WMNotifyProcessInstanceStarted	Yes	No
WMNotifyProcessInstanceAborted	Yes	No
WMNotifyProcessInstanceCompleted	Yes	No
WMNotifyProcessInstanceTerminated	No	No
WMNotifyProcessInstanceAttributeValueChanged	No	No
WMNotifyProcessInstanceStateChange	No	No
WMRelinquishProcessInstance	No	No

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 that include:

	Workflow Engine A		**Workflow Engine B**	
Operation	**Initiate**	**Respond**	**Initiate**	**Respond**
WMRequestCreateProcessInstance	Yes			Yes
WMRequestSetProcessInstanceAttributes	Yes			Yes
WMRequestGetProcessInstanceAttributes		Yes	Yes	
WMRequestStartProcessInstance	Yes			Yes
WMRelinquishProcessInstance	Yes			Yes

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

	Workflow Engine A		**Workflow Engine B**	
Operation	**Initiate**	**Respond**	**Initiate**	**Respond**
WMRequestCreateProcessInstance	Yes			Yes
WMRequestSetProcessInstanceAttributes	Yes			Yes
WMRequestGetProcessInstanceAttributes	Yes	Yes	Yes	Yes
WMRequestStartProcessInstance	Yes			Yes
WMNotifyProcessInstanceAttributeValueChanged		Yes	Yes	
WMRelinquishProcessInstance	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.

8. 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
[Tucker 95]		Tucker M., Baldwin C., Chase H. and MacDougall L., WfMC Compliance Paper, National Life of Vermont, May 1995.

