The Workflow Management Coalition Specification

Workflow Management Coalition

White Paper - Events

Document Number

April 99

Send comments to wfmc@wfmc.org

Author: David Hollingsworth, ICL A&TC

Table of Contents

1. Introduction

1.1 Background

The Workflow Management Coalition is a non profit organisation with the objectives of advancing the opportunities for the exploitation of workflow technology through the development of common terminology and standards. It has been recognised that all work flow management products have some common characteristics, enabling them potentially to achieve a level of interoperability through the use of common standards for various functions.

The WFM Coalition has been established to identify these functional areas and develop appropriate specifications for implementation in workflow products. Such specifications will enable interoperability between heterogeneous workflow products and improved integration of workflow applications with other IT services such as electronic mail and document management, thereby improving the opportunities for the effective use of workflow technology within the IT market, to the benefit of both vendors and users of such technology.

1.2 Purpose

This document contains a proposed approach for classifying and handling the processing associated with Events. It includes proposals for extensions to I/Fs 1, 2/3 and 4 (and potentially audit data which may be associated with particular event occurrences). It builds upon the proposals originally made by Steve Dworkin (May 1998) and earlier draft notes from Klaus Dieter Kreplin, Dave Holllingsworth and Mike Anderson.

1.3 Scope

Its scope is the extension of the WfMC architecture and standards to incorporate events. It may be read in conjunction with the Workflow Reference Model, which describes the architecture used by the WfMC within its standardisation programme.

1.4 Cross References

WfMC-TC-1003	Workflow Reference Model
WfMC-TC-1009	Workflow Client Application APIs (WAPI)
WfMC-TC-1012	Workflow Interoperability Specifications
WfMC-TC-1015	Workflow Audit Data Specifications
WfMC-TC-1016	Workflow Process Definition Interchange

1.5 Revision History

This issue (0.5 - draft) is an initial version for comment and review..

2. Events - Basic Concepts

This section describes the basic concepts relating to events; it draws on material recently incorporated into the WfMC Glossary.

The purpose of events is to provide a mechanism which may be used to co-ordinate or synchronise different processing activities which are otherwise independent.

2.1 Definition

Event - the occurrence of a particular situation or condition which has (or may have) significance to one or more workflows. The significance may be:
to initiate or terminate a workflow process instance (or instances) or, more generally, to change its (their) state(s)
to enable a particular activity to be started, re-started or completed
to signal a condition to other process or sub-process instances, which is then used in subsequent processing by that (sub-) process instance or instances

An event has two elements:

A Trigger - or cause

Definition – the recognition of some predefined set of circumstances associated with the operation of the system which causes a particular action to be taken

An Action - or response

Definition – the pre-defined system response following a trigger condition.

The required response will often relate to some action in a different process or system to that causing the event, so that the system response may include the concept of ***notification*** of the event occurrence to an appropriate source (or sources), e.g. a process or system which has established an interest in being informed of such events. In particular, notification may need to span different workflow engines and hence require extensions to the interoperability protocol.

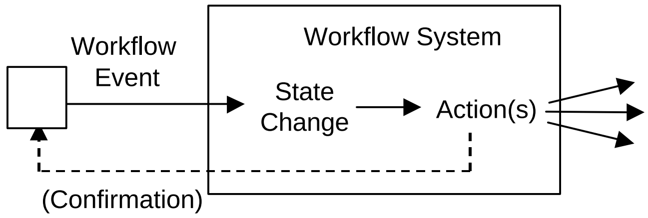
A further extension is that the initiating (triggering) process may optionally require ***confirmation*** when the response condition is actually actioned.

.

2.2 Workflow & Other Event Types

The term “Event” is widely used within the IT software engineering community and it is sensible to consider the characteristics of events in the context of a workflow system (i.e. a Workflow Event).

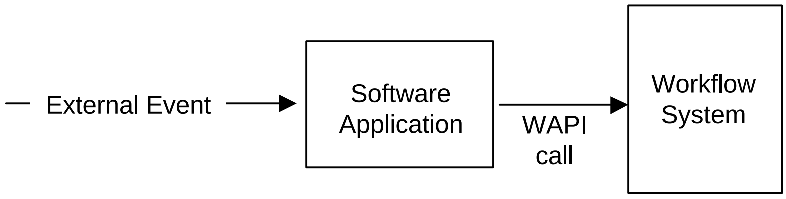
A workflow event is one in which the linkage between the trigger condition and the resultant system action involves the workflow management system.



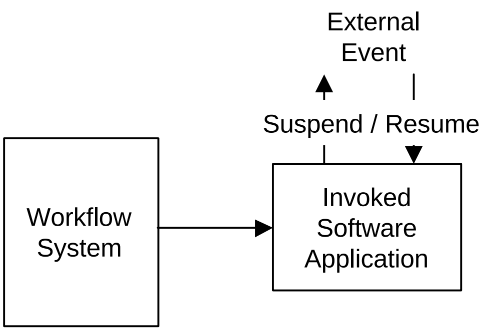
The workflow management system may implement the event handling mechanism itself, or may use the facilities provided within the underlying platform or networked environment. (For example, the OMG Common Object Services include an Event Service.)

Other types of event may be used internally within applications which participate within a workflow, for example:

a client application may monitor an external event source such as arriving email or fax and then initiate a new process instance or set an item of workflow relevant data using WAPI functions



an invoked application may suspend itself and wait for an particular type of event from some other source external to the workflow system before continuing).



Neither of these types of event is (or need be) visible to the workflow management system. In these cases it is the workflow application which handles any workflow related consequences of the event. Although such event types are common within workflow systems they do not need any specific extensions to the WfMC interfaces to handle them.

In contrast a workflow event has significance to the workflow system and causes a defined action via the workflow management software. For example this might result in the WFMS enabling a particular activity to be started or terminating a process instance. As such events are visible to the WFMS, the WfMC architecture needs to recognise them and develop specific interface definitions to permit the triggering of events and definition of the resultant actions.

2.3 Triggers & Actions

Workflow Event Triggers
Workflow events may be triggered by:
explicit activity action, for example by an application triggering the event by WAPI call
by a combination of circumstances defined within the process definition and detected by the workflow system (for example if no resource is available to assign a work item to, the WFMS might generate an event type "resource crisis").

One previously stated requirement is to enable a process instance to signal a co-ordination or synchronisation point with a different process instance. (For example, this might be required after completion of a particular activity and could be considered as a form of "And-Join" operating outside the context of a single process, where the threads to be "joined" are from different process instances.)

In general, to avoid over-complexity, it is assumed that such co-ordination between different process instances or threads will occur either because they share a common root PID (i.e. they were started as part of a common process instance) or because they share a key workflow relevant data attribute (for example,. although they were started as part of different process instances they relate to the same customer).

Workflow Event Actions
Workflow event actions may be:
(i) Implicit (as far as the WFMS is concerned) by notification of the event occurrence to a particular activity / process instance. This would typically cover the case where a process instance (or thread) has requested notification of interest from the WFMS of events of a particular type. The WFMS has no particular interest in the details of the action, since this is the responsibility of the application.

(ii) Explicit, for example defined within the process definition. Typical requirements might be for the workflow management system to undertake a specific workflow control action (terminate a process instance, enable a specific transition, etc.), or to set a workflow relevant data item (which is then evaluated at a subsequent time within a routing or resource assignment condition).

2.4 Notifications and Confirmations

Notifications
Notification may be required to a specific process instance or instances (via the WFMS) when the action is the responsibility of a workflow software application. Such notification may take several forms; an application may have suspended itself pending a particular event and be resumed by the WFMS when the event has occurred, or a particular API executed to query the WFMS if an event has occurred.

In other cases the WFMS may handle the response action itself (for example to enable a particular transition at a rendezvous point) and have no need to notify a software application (although notifications may need to be chained between workflow engines where remotely initiated process execution is occurring).

Workflow Interoperability Considerations
Particular requirements for notification arise when the trigger and required action occur on different workflow engines (or more strictly within different workflow domains). In these cases the linking of the trigger and action require specific event signalling (notification) across the interoperability interface. It is the responsibility of the initiating engine to identify the sink (target engine) for any event notification generated during the remote process execution. This is handled by the initiating engine transferring the "engine id" of the engine for event notification.

Event Confirmation
Where the triggering process requires confirmation delivery that a particular event has been notified and/or actioned a similar requirement arises to deliver a confirmation back to the initiating process. In some circumstances this may need to be notified asynchronously due to the nature of the underlying network connection and/or time delays prior to event notification and actioning.

2.5 Workflow Event Characteristics

Typing
A workflow event may have an associated event type such that a WFMS can differentiate events and the associated actions. Event types could encompass a complete class structure but for the purpose of this paper it is assumed that discrete event types plus a single composite class will suffice for initial WfMC work on the subject.

Context
A workflow event may also exhibit context, which defines the scope within which the event is of significance. The context may have (at least) two aspects - one based upon the process definition / process model paradigm and the other based upon the workflow domain. Thus an event may have local context within a particular process definition, such that it is not visible to process instances of other process definitions, or it may have a wider context across all process definitions within a workflow model (or conceptually even a wider context). An alternative view of events is that they have local significance within a workflow engine or a global significance across a co-operating set of engines (c.f. a workflow domain).

One important aspect of context is that it defines the scope within which workflow event type names must be unique. (This is a similar consideration to activity names or workflow relevant data names.) The id of the initiating process instance is not part of the context, although it may be required for event audit purposes.

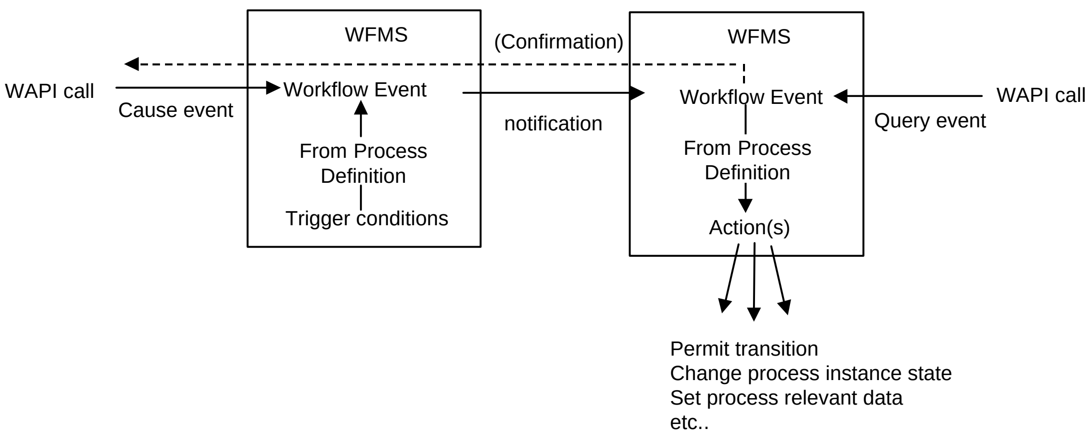
Persistence
Events under many circumstances will require persistent storage, due to the potential for time delay between the event trigger and the associated notification / action. There is a potential problem here, in that the strategies for event management by different products are likely to be different. Amongst the strategies which could be deployed are:

(i) Where no associated action / notification is already registered, ignore event or return appropriate confirmation status (e.g. unactioned)
(ii) If any currently initiated process could have an interest in the event, retain otherwise as (i)
(iii) retain for a defined time period (e.g. parameterised)
etc…
There is an associated issue which relates to the point at which cancelling / deleting the event occurs where more than one process may have an interest in the event (for example by way of a "query event" request). Again different product implementation characteristics are likely. Some further work is required to fully understand the best approach here.

Event Data
A workflow event may have associated data to enable the transfer of more information (between the triggering process and the actioned process) than can be derived from a simple event type. In a simple model of event handling this event data could be handled as workflow relevant data and / or using parameter passing semantics.

Event Audit Data
An event may have (optional) associated audit data requirements, for example the id of the triggering process instance (and possibly activity instance), the event type, date and time stamp, etc.

Summary of potential options



3. Requirements for Event Handling

This section considers potential implications on the process definition, on the WAPI command set and workflow interoperability.

3.1 Process Definition

Within the process definition there is a need to define:
event triggers and event actions

to identify how triggers are linked to events, and
to specify the rules by which event context is managed

Event Trigger
It is proposed that two broad classes of event trigger be defined.

The first may be thought of as transition related, for example “after activity X cause an event of type p”. This can be handled analogously to transitions so as to also include the context of a condition, if required, and of an event “transition” which is either synchronous (the triggering thread waits or terminates) or asynchronous where the triggering thread continues in parallel with the event trigger.

The second is related to WFMS detected conditions, typically relating to errors. Two particular conditions merit consideration - firstly when a resource assignment cannot be made and secondly when no transition can be made (hung activity). In both cases a construct of the form “on error X cause an event of type p” could be developed.

(I have deliberately avoided the construct of the WFMS causing an event when a conditional expression evaluates as true or false as it would places a requirement on the WFMS to continuously monitor such expressions and adds little value over what can be achieved by a workflow application.)

Event Actions
The most important required event action is to enable a process instance transition which is pending but waiting for the occurrence of the event. A proposed construct within the process definition would permit the joining of an incoming event of type p with other incoming transitions to an activity. This is, in effect, the converse of the event trigger action referred to above.

This style of event handing would complete what is required to handle the parallel synchronised workflow case. If data transfer is required between the threads of the process instances this could be accommodated by the use of event data

Other type of potentially required action include the creation (or termination) of process instances of a particular type following the occurrence of the particular event (… or more generally to define a change of state in response to the event).

Note that this use of events could provide a simple alternative approach to handling chained or hierarchical sub-processes by using events to trigger the initiation of the next sub-process or resumption of an existing sub-process. (However, this use of events is not recommended as a substitute to the existing sub-process approach as it could lead to further complexities in event handling to support parameter transfer.)

A further option is for the WFMS to set a workflow relevant data value in response to an incoming event of significance.

Linking Event Triggers and Actions
This is the responsibility of the WFMS and needs to embrace both local and global event types. It is not necessary to have anything additional within the process definition (although it may be sensible to define a conformance class, or classes, to indicate the possible behaviour of a WFMS in handling events). This subject is discussed later under interoperability and notification requirements.

3.2 WAPI Commands

It would be relatively straightforward to add a WAPI command to trigger an event; for example this could be of the type:
WMCauseEvent, where one of the input arguments identifies the event type. (Further arguments could be used, if required, to indicate the event context, for example local or global.)

It is not so easy to identify a simple and useful generalised procedure at API level to deal with notification or other event actions, since this requires either some form of callback facility or (polled) enquiry to query on particular events.

In conventional event management systems a process can suspend operation until the occurrence of a particular event type, when its processing is resumed. It would be possible to provide similar facilities via WAPI, allowing a workflow application to suspend a particular process (or activity) instance pending the occurrence of a specific workflow event type, either triggered by another local workflow application API call (Cause Event) or by an incoming event notification from a different engine.

A further extension of could be considered to provide a suspension against general events (multiple types) with another API call to enquire the event type following resumption. For example:
WMWaitforEvent - wait for event, either general or specific (filtered) providing synchronous operation,
WMQueryEvent - query on occurrence of a specific event type, or class of event types, or on the type of the last notified event

Event associated data – this could be treated as process relevant data.

No asynchronous notification across WAPI is proposed; the use of “process relevant” style event data, which can be “polled” by the application periodically, is an alternative approach.

3.3. Interoperability & Notifications

Event Interoperability concerns the notification of trigger conditions and/or actions between workflow engines (or potentially between any software component objects which have an interest)

To process the notification of a particular generated event type there are a number of considerations:
The action control data (or state data) associated with it which is passed

- id of triggering process, and the target engine id of that process
- filter (indicating the type of event, or a filter describing notification requirements)
- process relevant data (for transferring additional information associated with the event)

There are several different models which enabled the event context to be defined and hence control the notification process. The event could be:

- local to an engine
- local to threads of the common root process (i.e. following an And-Split)
- notified to the root process id on same / different engine
- notified to the calling process id
- globally broadcast among some predefined community
- notified to some external software process (not necessarily a workflow engine) which has registered interest (This of course requires a registration process for identifying interest.)
- etc…

In some ways the context of the event and associated data can be considered in the same way as the context of process relevant data. This is also an area where some further WfMC work would be useful.

[Note: This section requires further consideration for completion. There is considerable potential complexity and the next step for WfMC should be to define a simple workable approach for initial standardisation.]

3.4 Audit

Audit considerations require the provision of specific audit data associated with particular types of event occurrence and/or actions. This would be added as a further type of audit object (or perhaps separate objects relating to the trigger and the action), along with the data fields to be logged. The details remain to be defined.