SWAP: LEVERAGING THE WEB TO MANAGE WORKFLOW

Gregory Alan Bolcer • *Endeavors Technology, Inc., and University of California, Irvine* • *gbolcer@endeavors.org*
Gail Kaiser • *Columbia University* • *kaiser@cs.columbia.edu*

The Web's explosive growth has made it the platform of choice for Internet application developers. Workflow vendors are no exception to this trend. They are employing Web infrastructure and protocols in traditional workflow products to give customers worldwide connectivity for sharing information both inside and outside an organization.

Many organizations are beginning to discover what workflow vendors already know—namely, that the real value of the Web lies not just in its documents and resources, but also in the activities surrounding them. Collaborative work involves not only handoff and routing of data between humans, but the coordination of activities among them and with automated agents as well. Workflow engines typically ensure that the information ends up on the right desktop along with the tools to accomplish a slated task.

It is difficult to synchronize work and activity tracking within a technically diverse organization. Tools and formats typically differ among workgroups, as do skill levels and understanding among individual participants in a process. Browser-based user interfaces offer a mechanism to easily access distributed information and hand off documents and data over the Web, but at the expense of being able to effectively manage and track work activities. Web protocols provide no inherent support for automated change notification, handoff of control, or initiation of human- and computer-executed activities. In essence, there is no standard way for service requests to trigger a workflow process and monitor it across platforms and between organizations.

A Network-Based Protocol Solution

An Internet Engineering Task Force working group has been proposed to define a simple, lightweight, extensible protocol for communicating information about long-lived workflow activities over the Internet (see the sidebar, "Related Workflow Standards," for links to this group, its work, and other relevant Web sites). The ability to integrate work providers and work performers to support asynchronous services across an intranet, extranet, or the Internet is a powerful mechanism. It allows the use of software from multiple vendors to build scalable workflow automation solutions. To this end, more than 30 commercial vendors and research labs have contributed to the SWAP effort.

The work to date has been toward defining the Simple Workflow Access Protocol. SWAP is being designed to allow for interoperability between different workflow systems and the applications they support. Standardizing a protocol rather than defining a standard software library or API minimizes the implementation constraints on a vendor's workflow system. The workflow implementor must define only the structure of the calls and data placed on the wire. Interoperability overhead is minimized and grounded in a network-based protocol specification rather than the migration of process definitions from one system to another according to intersite agreements or common data format definitions.

To keep it simple, the scope of SWAP is limited to the interactions and requirements between a requester and a workflow service. This approach leverages existing Internet protocol designs and avoids the harder problem of prescribing a general interoperability workflow definition or interchange format.

For its purposes, the working group defines a *workflow process* as a series of steps that involves coordination of tasks, handoff or routing of information, and synchronization of activities. While a user can easily publish a Web page to disseminate information, it is very difficult to browse or publish access to a generic workflow service.

Current practice is through Common Gateway Interface scripting, whereby a server maps an HTTP method call to execution code resident on the receiving server. Because the code is usually programmed and supported in a way particular to the hosting organization, companies have hesitated to commit their core business processes outside their ability to control them. The code representing the

Related Workflow Standards

The Web site for the Simple Workflow Access Protocol working group is at http://www.ics.uci.edu/~ietfswap/. The site includes more information about SWAP as well as links to current IETF internet-draft documents at http://www.ietf.org/internet-drafts/. It also defines workflow as "the computerized facilitation or automation of a business process, in whole or part.

Other Workflow Standards

SWAP has overlapping concerns with many other standards efforts in the workflow and Web communities. Both the Workflow Management Coalition (WfMC) and the Object Management Group (OMG) both have proposed workflow standards[1,2] that are much more mature than SWAP. The Web sites for these organizations are http://www.aiim.org/wfmc/ and http://www.omg.org, respectively.

SWAP differs from these efforts in its focus on using HTTP as a substrate. This creates both an opportunity and a problem.

The opportunity, of course, is the prospect of capitalizing on HTTP's wide support, particularly on desktops and Internet-enabled devices that don't easily support traditional workflow installations. The problem is that HTTP doesn't currently offer the level of notification and transaction control required for cooperative work on the Web.[3]

The WfMC and OMG workflow standards require very sophisticated technologies to support distributed workflow execution and participation. Efforts to embrace and integrate the Web into these standards have proved difficult. The Web's minimalist infrastructure doesn't easily support flexibility, consistency, and assurances of a more robust traditional workflow solution.

Other Related Standards

What this specifically means to the SWAP working group is that protocol and data type definitions will be borrowed from other standards and working groups. SWAP intends to address these particular requirements by leveraging work done to date on HTTP and WebDAV. WebDAV has been a largely successful HTTP extension that provides a clean interface for assigning and retrieving properties on a resource. The home page for WebDAV is http://www.ietf.org/html.charters/webdav-charter.html.

The Internet printing protocol (IPP) shares several concerns regarding asynchronous job processing (see http://www.ietf.org/html.charters/ipp-charter.html).

The many notification protocols, such as the Generalized Event Notification Architecture, provide guidance to the issues and drawbacks of doing publish-subscribe and remote procedure notification on the Internet; and various XML standardization groups provide DTDs for certain types of domain-specific definitions and common business processes.

While all these issues are under consideration by the working group, the SWAP participants have determined that in the interest of time to market, external dependencies will be kept to a minimum. Formal incorporation of other standards work will be limited, but care will be taken to ensure future compatibility.

Why SWAP?

SWAP is related to several other protocol standardization efforts, including Remote Procedure Calls using XML encoding (XML/RPC) and HTTP and its extensions (HTTP/1.1, HTTP-NG). HTTP has been useful for many purposes because it doesn't require programmers to build different protocol engines to support highly related tasks. For protocols that aren't supported, the Protocol Extension Protocol allows extension of HTTP functionality.

Because SWAP involves generic services, several working group members have asked is workflow-specific about the protocol. The definition of the SWAP workflow protocol is useful in several ways. First, workflow has a unique requirement in that a generic workflow service, once invoked, can take anywhere from minutes to years to complete. How will the service change over this time? What will be the availability? How do updates and notifications take place? It is not yet clear whether existing notification and transaction protocols can be scaled to a problem that includes this type of asynchrony.

Second, workflow involves a mix of automated services and human-enacted activities. While protocols have focused on one or the other, only the workflow community has explored the issues for seamlessly intermixing of the two.

References

1. Workflow Management Coalition (WfMC), "The Workflow Management Coalition Reference Model," Tech. Report WFMC-TC-1003, v.1.1, Jan. 1995.
2. G. Taubes, "Taming Virtual Taskmasters," Networks series, *IBM Research*, Vol. 36, No. 3, 1998 pp. 7-9; also available online at http://www.research.ibm.com/resources/magazine/1998/issue_2/networks398.html.
3. Workflow Management Coalition (WfMC), "Workflow and Internet: Catalysts for Radical Change", WfMC white paper, June 1998; available online at http://www.aiim.org/wfmc/finalwp.pdf.

enactable workflow process tends to evolve, which can cause unexpected errors and exceptions and invocation mismatches. This makes maintaining interorganizational processes costly.

SWAP would solve these problems by using HTTP and WebDAV protocol extensions and transferring structured information encoded in XML.

Resource and Execution Model

SWAP is a work in progress, and definitions have yet to be formally reviewed and commented on.

However, after several birds-of-a-feather meetings, a consensus is emerging on some design approaches and definitions. The SWAP Internet draft,[1] which expires in February 1999, identifies the following operations as desirable goals:

- *Initiate*. Create, remotely set up, and invoke a workflow process. Set up the work with the appropriate values and permissions, or provide references to resources and data as needed.
- *Monitor*. Check the work's current status, get a history of the execution, or find out the current and possible states. The latter may include collectively reviewing a set of distributed activities.
- *Control*. Read and set the running state of remote, generic, asynchronous workflow services. Pause or resume an executing workflow after it is running, or terminate it when it is no longer needed.
- *Notify*. Send appropriate notifications of status changes to interested observers during normal execution or when exceptions occur.

A generic workflow service appears externally as a small number of different resources provided by a server. Resources are distinguished by their Web address in the form of a Uniform Resource Identifier, as opposed to a URL—first because a URL often confuses the name of a resource with its location. Additionally, an expectation of how named resources are generated is implicit in a URI scheme.

SWAP doesn't enforce the naming of the actual locator for a particular process resource; it allows the host company to determine this individually. The host company may then adopt a generative scheme that includes human-friendly identifiers where appropriate.

Implementation of these resources (services and/or objects) is outside the protocol's scope. SWAP is concerned only with ensuring that requests are made to valid URIs such that a proper and consistent result is returned.

SWAP uses the following interface definitions in forming a workflow model:

- *Process instance*. The actual performer of the workflow service. Every time a service is invoked, a new process instance is created, started, paused, resumed, or terminated.
- *Process definition*. The named resource from which process instances can be created. The process definition URI is the location of the generic service from which multiple instances are shared.
- *Observer*. A resource interested in the completion of a process instance. Given a list of observer URIs, a process instance can notify observers when the process has changed state or completed.
- *Activity observer*. A type of observer that allows a process instance to be notified upon completion of external work. A process instance may need to invoke a subprocess instance on a remote system. An activity observer simply provides an interface to browse parent-child process dependencies and to receive notification of a completed subprocess.
- *Work list/work item*. A process instance that represents a human-executed activity. Work items hold the references to activities for the people who are expected to perform them and who can, in turn, search and retrieve all work items assigned to them.

A workflow process instance is created when actual resources are committed to accomplishing a workflow service, usually when the service is invoked. Resources can support multiple interfaces. For example, if it is necessary to invoke a remote service on another machine, a resource can implement both the process instance and the observer interfaces.

Mapping Control Mechanisms. A process instance may run for days, months, or even years. Because of the asynchronous nature of workflow, mechanisms for controlling and managing it using HTTP need to be mapped out. Keeping an HTTP connection alive long enough to perform changes and monitor progress of a long-running workflow process, while doable, is not something the hypertext protocol was designed to support.

SWAP assumes that the workflow engine has the ability to map the protocol methods and translate the parameters to the appropriate workflow process instances and to return the appropriate values. Here's how it works.

Let's say that a process definition is offered at a specific URI. An HTTP request to the URI invokes a new process instance. A given workflow implementation has complete control over how it creates the URI that names the target process. Names serve as a unique ID for the process involved. They are used as a unique service ID for other requests to that process instance. For example, an HTTP request can provide data to the process instance through name-value pairs. Another request could retrieve the current state of the process instance, or pause, resume, or terminate it. Every interaction specifies a method, the standard HTTP header field that controls what action the server takes.

There is still some debate in the working group about whether adding methods particular to SWAP is the appropriate approach. This column describes the current thinking, but the same functionality could easily be supporting using just the HTTP POST method and passing the SWAP-specific directives in the parameter values.

All parameters, including scoping and name spaces, are put into an XML encoding to be sent with the appropriate calls. In the example of Figure 1, the root node has a list of attributes, including interfaces, process instance key, valid states, actual state, and data. From this XML snippet, we can determine the values and state of that process instance as well as other possible instances.

The example shows a hypothetical purchase order made by the SWAP working group chair using the configuration information provided by an online PC manufacturer. Upon the request, SWAP creates a trackable process instance but doesn't begin executing the process. The method call

```
>>Request
CREATEPROCESSINSTANCE /submit/order?proc=10 HTTP/1.1
Host: www.widget-makers.com
Content-Type: text/xml
Content-Length: xxxx
Authorization: Digest username="skreddy"
    realm="skreddy@oracle.com", ...

<?xml version="1.0" ?>
<d:swap xmlns:d="SWAP:">
<d:observer>http://www.ics.uci.edu/pub/ietf/swap/chair.html</d:observer>

<d:name>equipment-purchase-process</d:name>
<d:subject>procurement</d:subject>
<d:description>New equipment purchase</d:description>
<d:contextData>
    <z:processor xmlns:z=http://conf.pcmanufact.com>
        pentiumII
    </z:processor>
     <z:memory xmlns:z=http://conf.pcmanufact.com>
        <z:size>256 Meg</z:size>
        <z:speed>60 ns</z:speed>
        <z:type>DRAM</z:type>
    </z:memory>
     <!-- "name, billing address, etc." -->
</d:contextData>
<d:startImmediately>no</d:startImmediately><!-- "available?" -->
</d:swap>
</xml>
```

Figure 1. XML encoding of SWAP content.

```
>>Response
HTTP/1.1 207 Multi-Status
Content-Type: text/xml
Content-Length: xxxxx

<?xml version="1.0" ?>
<d:swap xmlns:d="SWAP:">
<d:multistatus>
    <d:response>
        <d:processInstance>
            <d:href>
http://www.widget-makers.com/status?proc=10.1
            </d:href>
            <?-- "other items as defined by process" -->
        </d:processInstance>
        <d:propstat>
            <d:prop><z:size/</d:prop>
            <d:status>HTTP/1.1 409 Conflict</d:status>
            <d:comment>part
unavailable</d:comment>
        </d:propstat>
        <d: propstat>
            <?-- "other resource properties" -->
        </d:propstat>
    </d:response>
</d:multistatus>
</d:swap>
</xml>
```

Figure 2. XML return values for a SWAP call.

in this case returns the HTTP status code indicating multiple return values as seen in Figure 2.

It turns out that the memory size conflicts with the order configuration. The customer can then choose to terminate the purchase order or amend it using the appropriate PROPFIND and PROPPATCH methods for assigning particular values.

Conclusions

SWAP has the potential to provide interoperability between workflow systems using the Web and the Internet, allowing distributed workflow solutions to be more easily deployed and accessed than at present. SWAP's simple workflow model and implementation approach allow business processes to be structured more naturally. This will encourage people to use the network and computing infrastructure already in place in their company's work culture.

SWAP allows abstraction through its support of process and subprocess execution across multiple machines and between various workflow vendor implementations. Its standardization will provide a framework not just for sharing documents but also for coordinating the activities that surround them. By limiting the protocol's scope, the SWAP working group hopes to quickly define a usable protocol specification that leverages work to date in both the workflow and Web communities. The goal is to increase the Web's potential as an infrastructure for building and deploying workflow solutions and also to alleviating the difficulty of constructing such solutions.

An early prototype of a SWAP-enabled HTTP server was built at Netscape, however, no reference implementation has yet been made. Many of the workflow vendors participating in the SWAP working group support HTTP in some capacity, so discussions about interoperability demos are in the works. In the meantime, the protocol is being iteratively revised, and the working group openly welcomes participation. ■

ACKNOWLEDGMENT

The authors would like to acknowledge Keith Swenson, Surendra Reddy, Richard Heim, Lisa Lippert, and members of the SWAP mailing list for their comments and feedback.

REFERENCES

1. K. Swenson, "Simple Workflow Access Protocol (SWAP)", Strawman document, August, 1998. http://www.ietf.org/internet-drafts/draft-swenson-swap-prot-00.txt.

Gregory Alan Bolcer received his PhD and BS from the University of California, Irvine and his MS from the University of Southern California. He is the founder and CEO of Endeavors Technology, Inc., a supplier of Web-Based workflow solutions.

Gail Kaiser is a professor of computer science and director of the Programming Systems Library at Columbia University. She is on the IEEE Internet Computing editorial board.