ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στα Δίκτυα Υπηρεσιών

Διάλεξη 15η: WS-Addressing and Notification

Χρήστος Νικολάου
Τμήμα Επιστήμης Υπολογιστών

Introduction to Service Networks

WSRF, Addressing and Notification

Christos Nikolaou
Transformation Services Laboratory
CSD/UoC

Addressing and Notification

Based on Chapter 7 of:

- ***FOUNDATIONS AND PRINCIPLES OF WEB SERVICES:***
  - ***AN HOLISTIC VIEW (CONCEPTS, TECHNOLOGIES, ARCHITECTURES AND STANDARDS)***
- ***Michael P. Papazoglou***
- ***(INFOLAB/CRISM, Tilburg University, The Netherlands)***

Introduction

Real-life SOA implementations need to rely on event processing and notification.

Notification pattern for SOAs: an information-providing service sends messages to one or more interested parties.

- In this pattern the message frequently carries information about an event that has occurred rather than a request that some specific action should occur.
- This pattern unifies the principles and concepts of SOA with those of event-based programming.

critical components of the notification infrastructure:

- mechanisms for handling stateful resources as well as
- transport neutral mechanisms for addressing Web services and messages.
- concentrate on a standard Web services approach to notification using topic-based publish/subscribe mechanisms.

key concepts:

- Representing stateful resources in a Web services environment.
- The concept of an end-point reference.
- Mechanisms for routing messages and addressing Web services.
- Event-processing and notification mechanisms for Web services.
- The WS-Notification family of specifications.
- Peer-to-peer notifications and topic-based notification patterns.

Web Services and Stateful Resources-1

the effect of a computation on volatile storage could be described as taking a series snapshots, called **states**.

A program's state refers to its ability to "remember" information between successive computations.

Usually, procedure (method) invocations cause these computations and change the state of a program (or of an object).

An interaction spans two modes: **stateful** and **stateless**. A stateful interaction keeps track of client state between method invocations, whereas a stateless interaction does not.

Web Services and Stateful Resources-2

Web services must also often provide their users with the ability to access and manipulate state.

State concerns data values that persist across, and evolve as a result of, Web service interactions.

Web services are typically implemented by stateless components such as Java servlets or Enterprise Java Bean stateless session beans.

While Web services implementation is typically stateless, Web services interfaces must frequently allow for the manipulation of state.

- For example, an online order fulfillment process must maintain state concerning purchase order status, purchases or orders made by specific customers, and the system itself: its current location, load, and performance.
- Web service interfaces that allow requestors to query purchase order status, make or cancel orders, change order status, and manage the order processing system must necessarily provide access to this state.

Web Services and Stateful Resources-3

In the Web services world, state is any piece of information, outside the contents of a Web services request message, which a Web service needs in order to properly process the request.

While Web services successfully implement applications that manage state, it is also desirable to define Web service conventions to enable:

- the discovery of,
- introspection on,
- and interaction with stateful resources in standard and interoperable ways [Foster 2004].
- The information that forms the state is often bundled together in what is known as a *stateful resource*.

A stateful resource exhibits three major properties [Foster 2004]:

- it is a specific set of state data expressible as an XML document that defines the type of the resource;
- it has a well-defined identity and lifecycle;
- and is known to, and acted upon, by one or more Web services.

Web Services and Stateful Resources-4

What are examples of stateful resources?

Stateful resources are elements that can be modeled and range from physical entities (such as servers) to logical constructs (such as business agreements and contracts).

Access to these stateful resources enables customers to realize business efficiencies including just in time procurement with multiple suppliers, systems outage detection and recovery and workload balancing.

A stateful resource can also be a collection or group of other stateful resources.

The state of a specific resource may be implemented as an actual XML document that is stored in memory, in the file system, in a database, or in some XML repository.

Alternatively, the same resource may be implemented as a logical projection over data constructed or composed dynamically from programming language objects (such as an EJB entity bean) or

from data returned by executing a command on a private communications channel to a traditional procedural application or data system.

Web Services and Stateful Resources-5

Several aspects of stateful resources are of interest to Web services applications that need to reason about state. These include [Graham 2004a]:

- How a stateful resource is used as the data context for the execution of Web service message exchanges;
- How stateful resources can be created, assigned an identity, and destroyed;
- How the elements of a stateful resource can be modified;
- How the type definition of a stateful resource can be associated with the interface description of a Web service to enable well-formed queries against the resource via its Web service interface, and the state of the stateful resource can be queried and modified via Web service message exchanges; and
- How a stateful resource is identified and referenced by other components in the system.

All these aspects are addressed by elements of the WS-Resource Framework (WS-RF), which is concerned primarily with the creation, addressing, inspection, and lifetime management of stateful resources. The relationship between Web services and stateful resources sits at the core of the WS-RF.

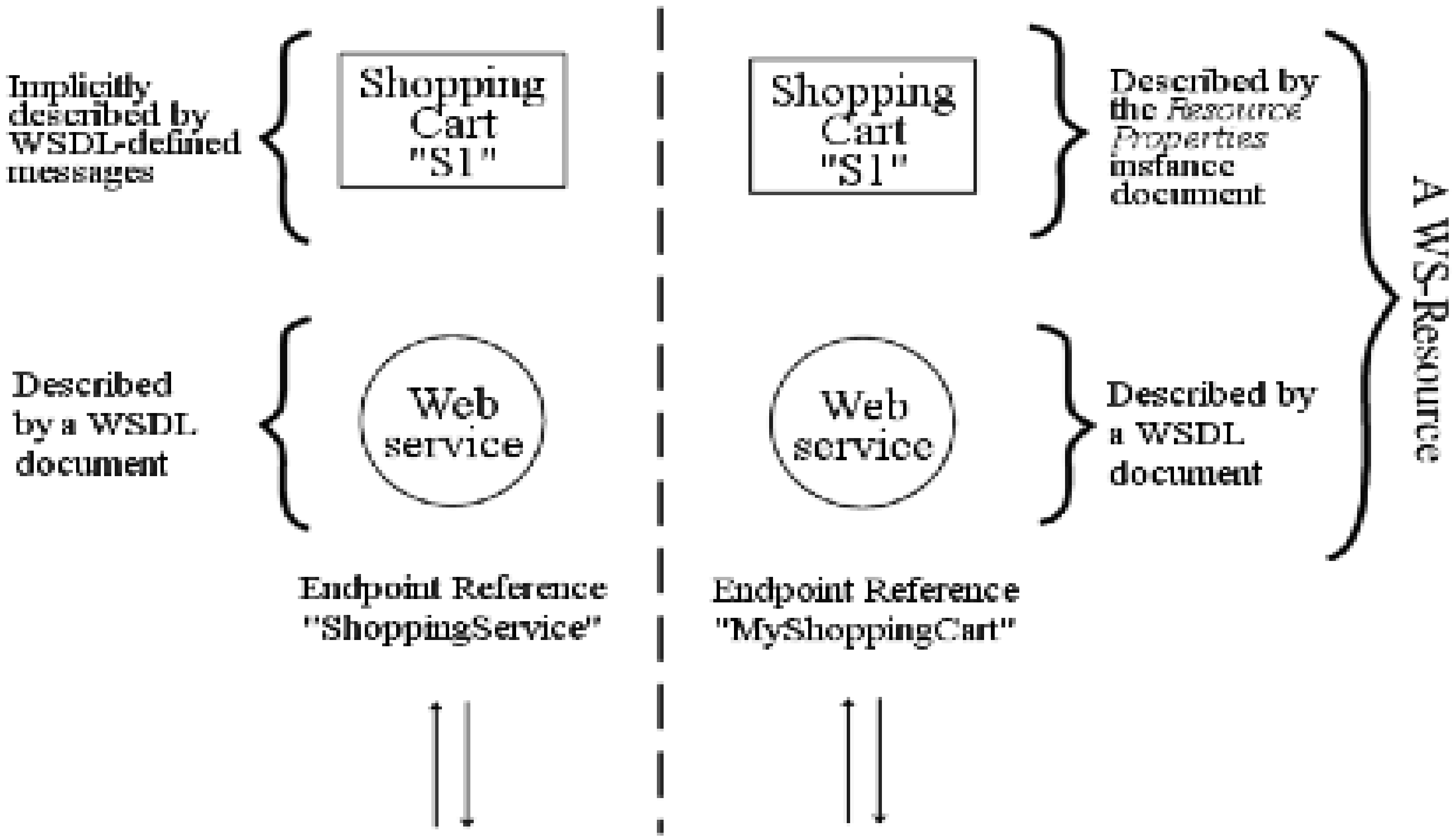
Web Services and Stateful Resources-6

There are many motivations behind the WS-RF specifications [Foster 2004].

The most notable contribution is the intersection of grid computing and Web services standards and their alignment with SOA principles.

- In particular, WS-RF helps define how Web services can maintain stateful information, aiming to unify grid and Web services.
- Through this alignment with the Web services stack, grid services can use existing Web services standards, such as WS-Notification, WS-Addressing, and WS-Security, and build extensions for extended capabilities such as service state data, lifetime, grouping, and reference management.
- grid services are particular types of services that are used to represent stateful resources in the context of grid computing. In WS-RF, these stateful resources are called WS-Resources.
- A grid service has an identity, service data, and lifetime management mechanisms, while a WS-Resource has a name, resource properties, and lifetime management mechanisms.

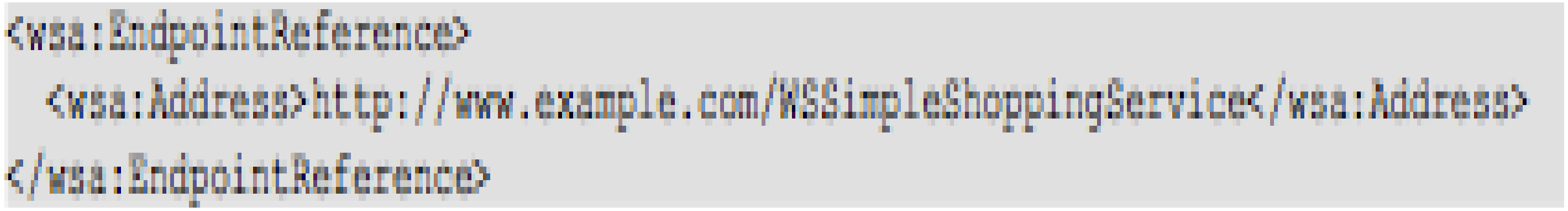
Non-WSRF Web Services and WSRF


Implicitly described by WSDL-defined messages
Shopping Cart "S1"
Described by a WSDL document
Web service
Endpoint Reference "ShoppingService"
Shopping Cart "S1"
Described by the Resource Properties instance document
Web service
Described by a WSDL document
A WS-Resource
Endpoint Reference "MyShoppingCart"

Source: WSRF Primer

Non WSRF vs WSRF cont...

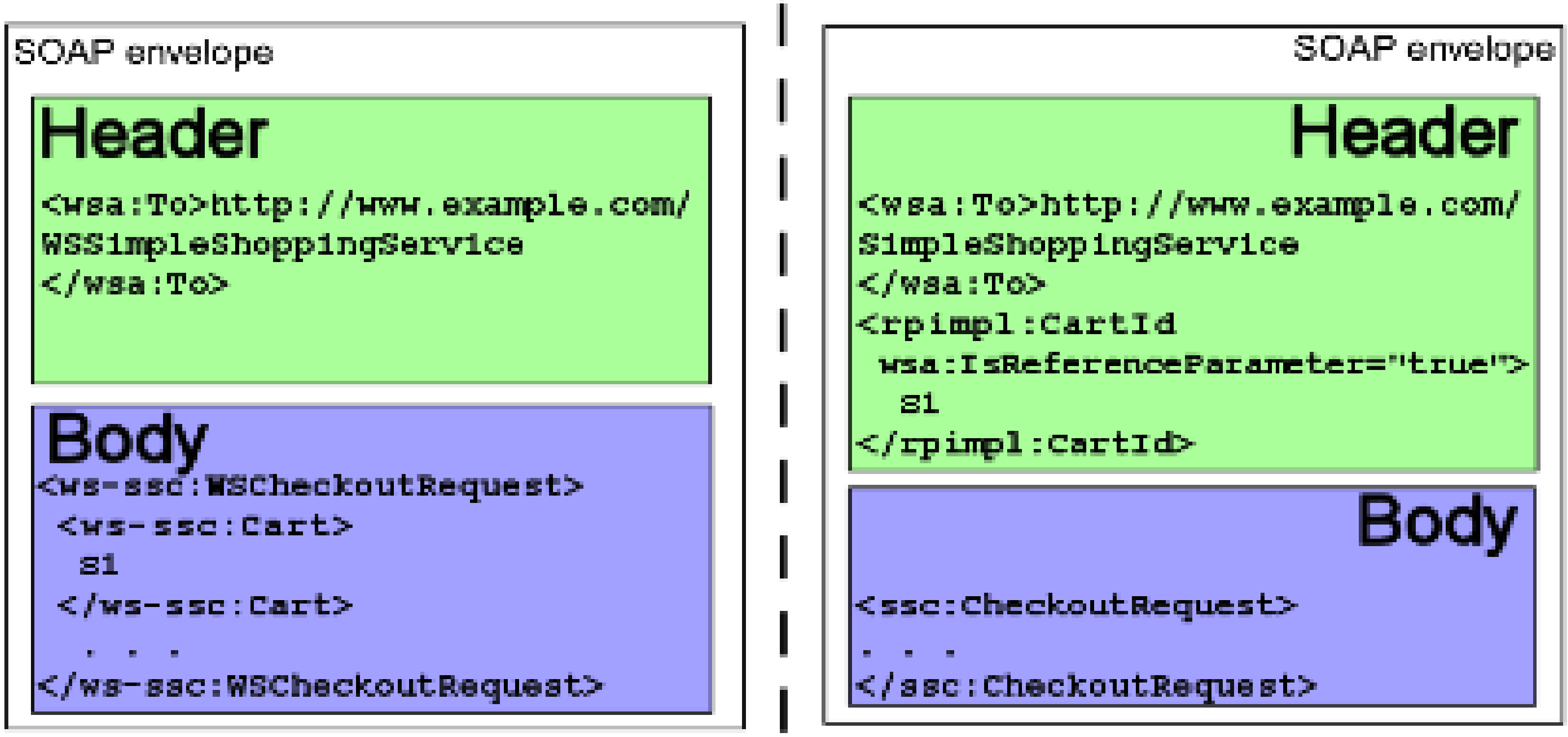
Previous Figure shows a comparison of a simple shopping service built with non-WSRF Web services (on the left) and WSRF (on the right). The left-hand side of the figure shows a ShoppingCart resource called ‘S1’ accessed through a Web service at a particular location. The location is described by a WS-Addressing [WS-Addressing] EndpointReference (EPR), which contains the URI of the Web service endpoint as follows:



```
<wsa:EndpointReference>
  <wsa:Address>http://www.example.com/WSSimpleShoppingService</wsa:Address>
</wsa:EndpointReference>
```

This EPR is known as “ShoppingService” by the requester.

The Shopping Cart Example – SOAP Messages



```
<wsa:To>http://www.example.com/
WSSimpleShoppingService
</wsa:To>
```

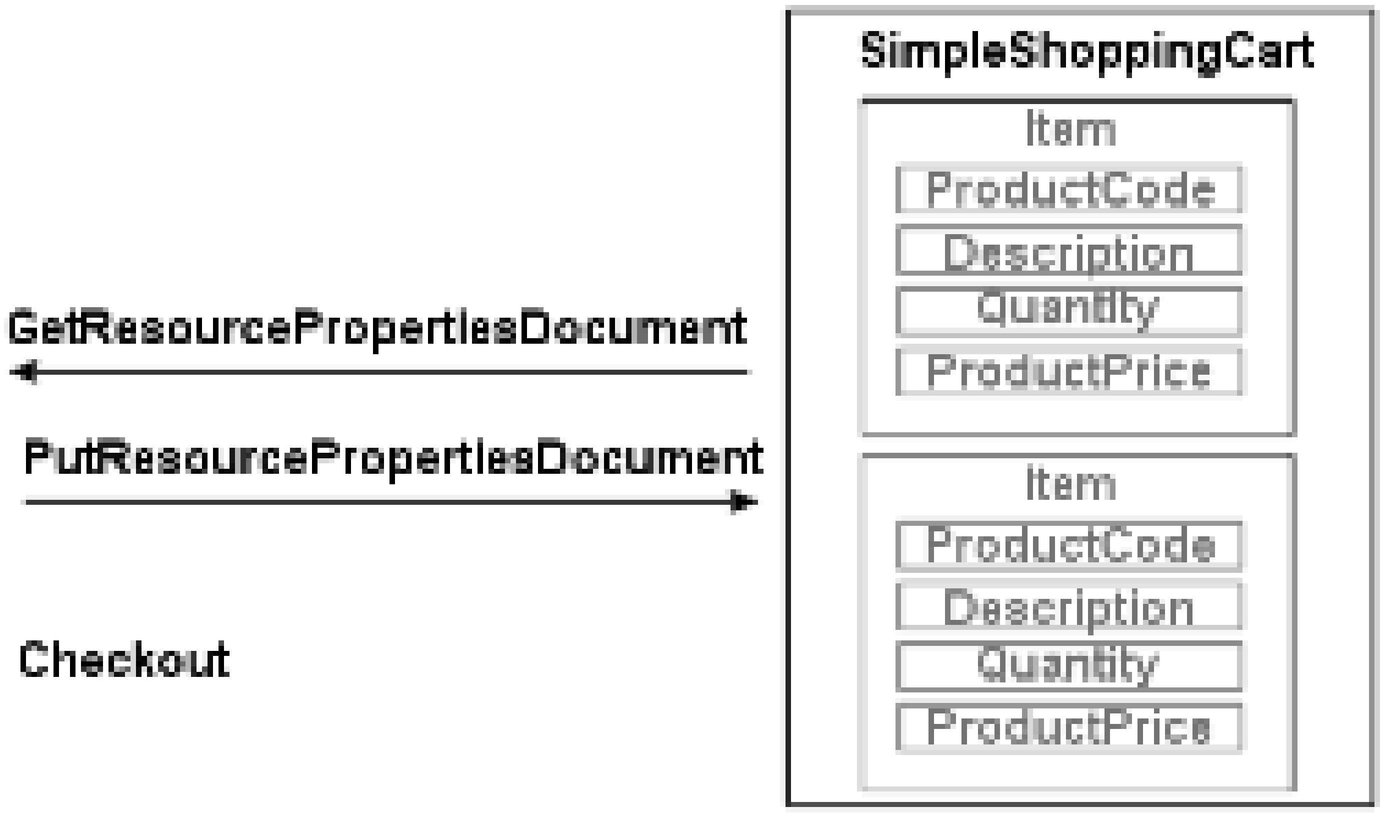
```
<ws-ssc:WSCheckoutRequest>
 <ws-ssc:Cart>
  s1
 </ws-ssc:Cart>
  . . .
</ws-ssc:WSCheckoutRequest>
```

```
<wsa:To>http://www.example.com/
SimpleShoppingService
</wsa:To>
<rpimpl:CartId
 wsa:IsReferenceParameter="true">
  s1
</rpimpl:CartId>
```

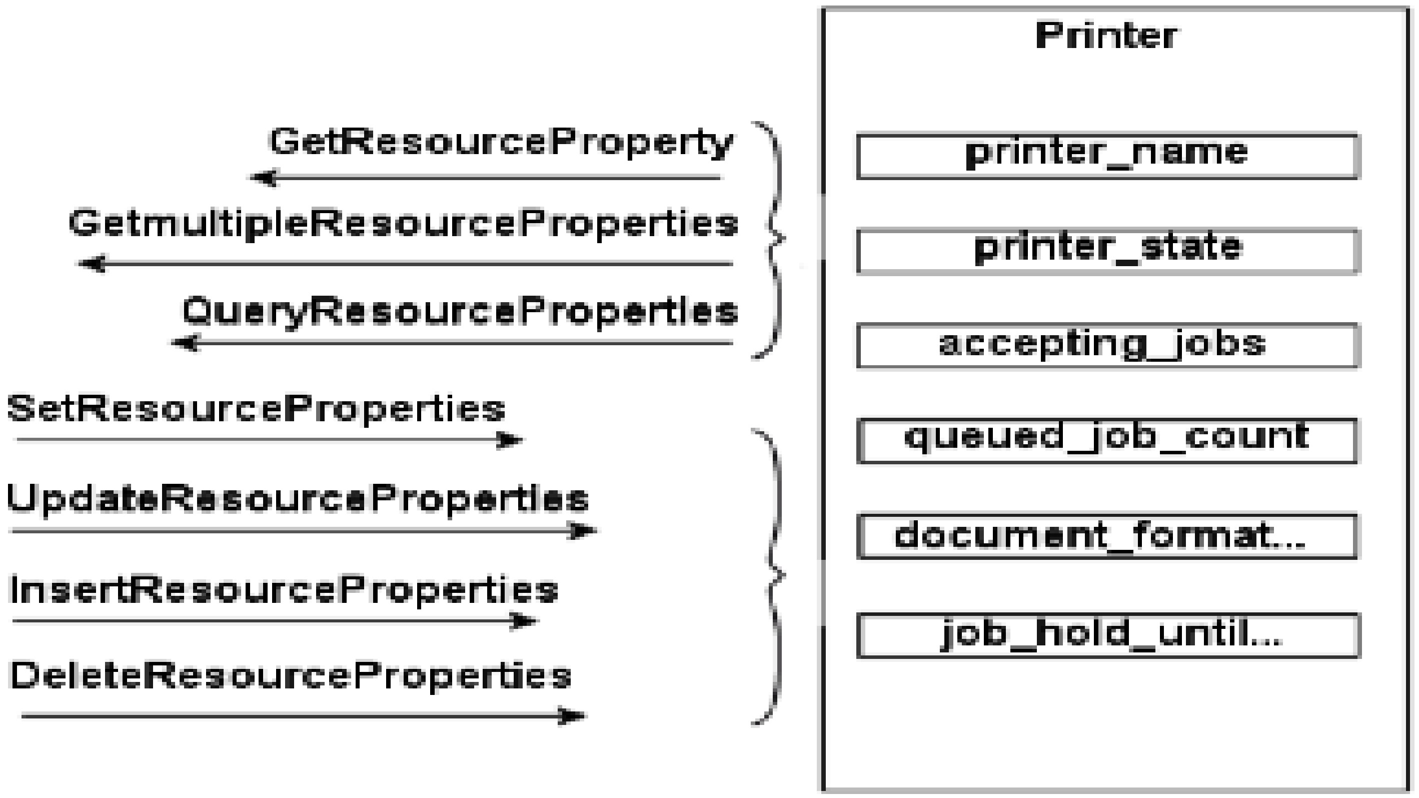
```
<ssc:CheckoutRequest>
. . .
</ssc:CheckoutRequest>
```

Source: WSRF Primer

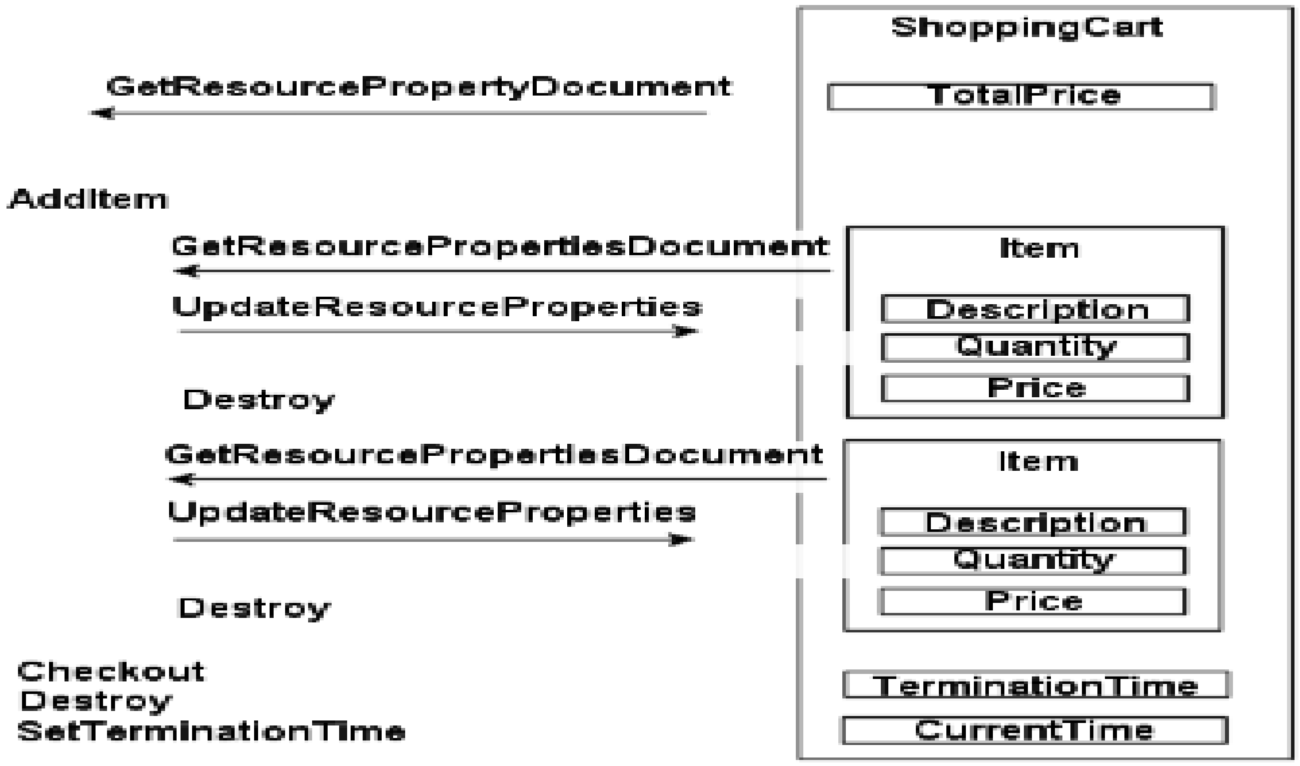
The Shopping Cart Example – The Resource



The Printer as a Resource



The Shopping Cart as multiple WS Resources



Summary of WS-Resource Framework specifications

Name	Description
WS-ResourceProperties	Describes associating stateful resources and Web services to produce WS-Resources, and how elements of publicly visible properties of a WS-Resource are, retrieved, changed, and deleted.
WS-ResourceLifetime	Allows a requestor to destroy a WS-Resource either immediately or at a scheduled future point in time.
WS-RenewableReferences	Annotate a WS-Addressing endpoint reference with information needed to retrieve a new endpoint reference when the current reference becomes invalid.
WS-ServiceGroup	Creates and uses heterogeneous by-reference collections of Web services.
WS-BaseFault	Describes a base fault type used for reporting errors.
WS-Notification family of specifications	Standard approaches to notification using a topic-based publish and subscribe pattern.

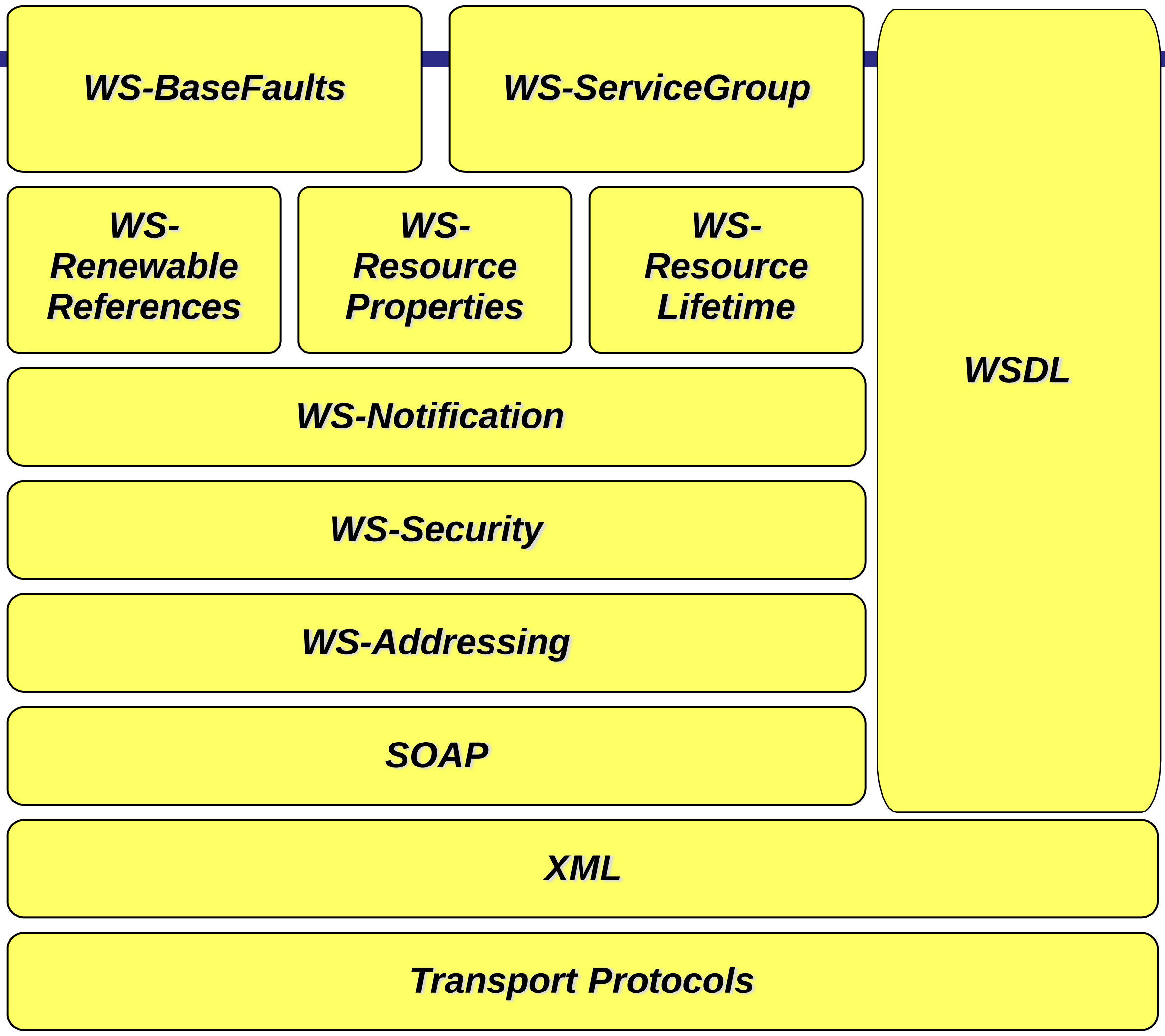
WS-Resource Framework-1

The WS-Resource Framework is a set of six Web services specifications.

The first five specifications in Table 1 are named collectively the WS-Resource Framework [Czajkowski 2004],

the WS-Notification family of specifications addresses notification (event) subscription and delivery, see next slide for relationship with other WS standards.

WS-Resource Framework



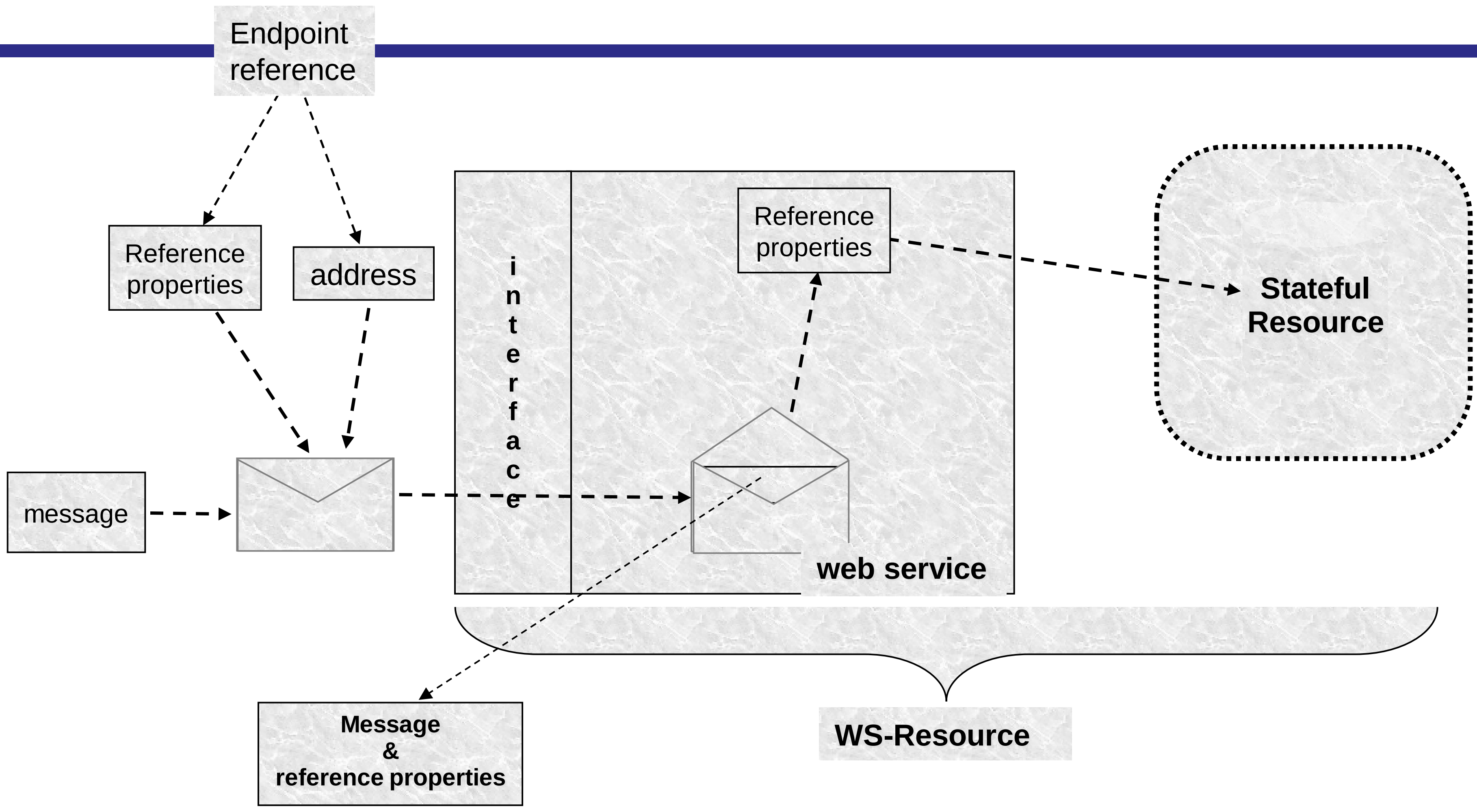
WS-Resource Framework-2

The WS-RF codifies the relationship between Web services and stateful resources in terms of what is known as an implied resource pattern.

An *implied resource pattern* is a set of conventions on Web services technologies, particularly XML, WSDL, and WS-Addressing, which helps route Web services messages and deliver them to their true destination.

– In particular, the term implied resource pattern describes the way WS-Addressing is used to associate a stateful resource with the execution of message exchanges implemented by a Web service.
– This is illustrated in the next slide, where it is shown how an implied resource pattern helps establish a specific kind of relationship between a Web service and one or more stateful resources.

WS-Resource Framework-3



WS-Resource Framework-4

The term “implied” is used because the identity of the stateful resource associated with a given message exchange is not part of the request message itself.

Instead, the stateful resource is implicitly associated with the execution of the message exchange.

This can occur either statically or dynamically [Foster 2004].

Finally, the term “pattern” is used to indicate that the relationship between Web services and stateful resources is codified by a set of conventions on existing Web services technologies, in particular XML, WSDL, and WS-Addressing.

A stateful resource that is taking part in this implied resource pattern is called WS-Resource.

The WS-RF describes the WS-Resource definition and also how to make the properties of a WS-Resource accessible through a Web service interface, and to manage and reason about a WS-Resource’s lifetime.

WS-Resource Framework-5

A stateful resource is associated with the Web service statically if the association is made when a Web service is deployed.

Alternatively, when the association is made during execution of message exchanges the stateful resource is dynamically associated with the Web service.

When the association is performed dynamically then the stateful resource identifier is used to designate that the implied stateful resource may be encapsulated in the WS-Addressing endpoint reference that is used to address the target Web service at its endpoint.

WS-Addressing-1

Normally, Web services are invoked by the service endpoint information that is provided by WSDL.

- For instance, a WSDL service port has a location address, which identifies the endpoint.

But for stateful Web services or in cases where we want to add more dynamic information to the address including instance information, policy, complex binding, and so on [Joseph 2004], we need a client or runtime system to uniquely identify a service at runtime.

This binding-specific information could include a unique identifier.

Currently, there is no standard way by which this information can be exchanged and then mapped to the runtime engine while the service is accessed.

The WS-Addressing specification tackles this problem by providing a lightweight mechanism for identifying and describing the endpoint information and mapping that information into the SOAP message headers.

- WS-Addressing provides transport-neutral mechanisms to address Web services and messages.
- WS Addressing was introduced in order to standardize the notion of a pointer to a Web service [Graham 2004a].

WS-Addressing-2

WS-Addressing defines how message headers direct messages to a service, provides an XML format for exchanging endpoint references, and defines mechanisms to direct replies or faults to a specific location.

It also enables messaging systems to support message transmission through networks that include processing nodes such as endpoint managers, firewalls, and gateways in a transport-neutral manner.

The WS-Addressing standard designates that addressing and action information normally embedded in communication transport headers should be placed within the SOAP envelope.

WS-Addressing-3

The WS-Addressing standard defines a schema for a portable address construct known as an **endpoint reference**.

An endpoint reference provides an address, not an identity, of its target and is defined as an XML type. The address construct is a URI that is used to provide the logical address of an endpoint, and appears in the header block of every SOAP message targeted at that endpoint.

A service endpoint in WS-Addressing follows the implied resource pattern by providing three critical pieces of information required at run-time for interaction with an endpoint:

– a base address,
– sets of reference properties,
– and reference parameters.

Reference properties and reference parameters are collections of arbitrary XML elements used to complement the base address construct by providing additional routing or processing information for messages.

Reference properties help address collections of WSDL entities that share a common URL and scope. Endpoint references must contain the address along with meta-data descriptions, such as service name, port name, port type, and WS-Policy statements that describe the requirements, capabilities, and preferences of the service. Such properties allow discovery of contract details and policies.

WS-Addressing-4

To enable interacting with endpoints WS-Addressing defines a set of headers in SOAP messages that are sent to relevant Web services endpoints. More specifically, it defines a set of four headers ReplyTo, FaultTo, RelatesTo, MessageID used to dynamically define the message flow between different endpoints.

This kind of support not only enables further independence of a SOAP message from its communication protocol, but also defines a means by which Web services may pass references to themselves and other services.

The sample code in the following slide illustrates the use of these WS-Addressing mechanisms in a SOAP message being sent from the site http://myclient.com/business/someClient to the site http://www.plastics_supply.com/purchasing.

SOAP message with WS-Addressing

```
<Soap:Envelope xmlns:Soap="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsa="http://www.w3.org/2004/12/addressing">
  <Soap:Header>
    <wsa:MessageID>
       uuid:SomeUniqueMessageIdString
    </wsa:MessageID>
    <wsa:ReplyTo>
       <!-- End point reference for intended receiver of message reply -->
      <wsa:Address> http://myclient.com/business/someClient </wsa:Address>
   </wsa:ReplyTo>
   <!-- End point reference for ultimate receiver of message -->
    <wsa:To> http://www.plastics_supply.com/purchasing </wsa:To>
    <wsa:Action> http://www.plastics_supply.com/SubmitPO </wsa:Action>
  </Soap:Header>
  <Soap:Body>
     <!-- The message body of the SOAP request appears here -->
     <SubmitPO> … </SubmitPO>
  </Soap:Body>
</Soap:Envelope>
```

WS-Addressing-5

The next example shows a sample WS-Addressing endpoint reference.

The service's URI is specified in the <Address> element of a purchase order service. The <Address> element contains the transport-specific address of the Web service. In this case it is an HTTP URL.

The WS-Addressing <EndpointReference> includes a <ReferenceProperties> child element that identifies the resource to be associated with the execution of all message exchanges performed using this <EndpointReference>.

For instance, the reference property in the example indicates that the purchase order is from a version of the service that provides purchase services with discount prices available for premium customers.

WS-Addressing-6

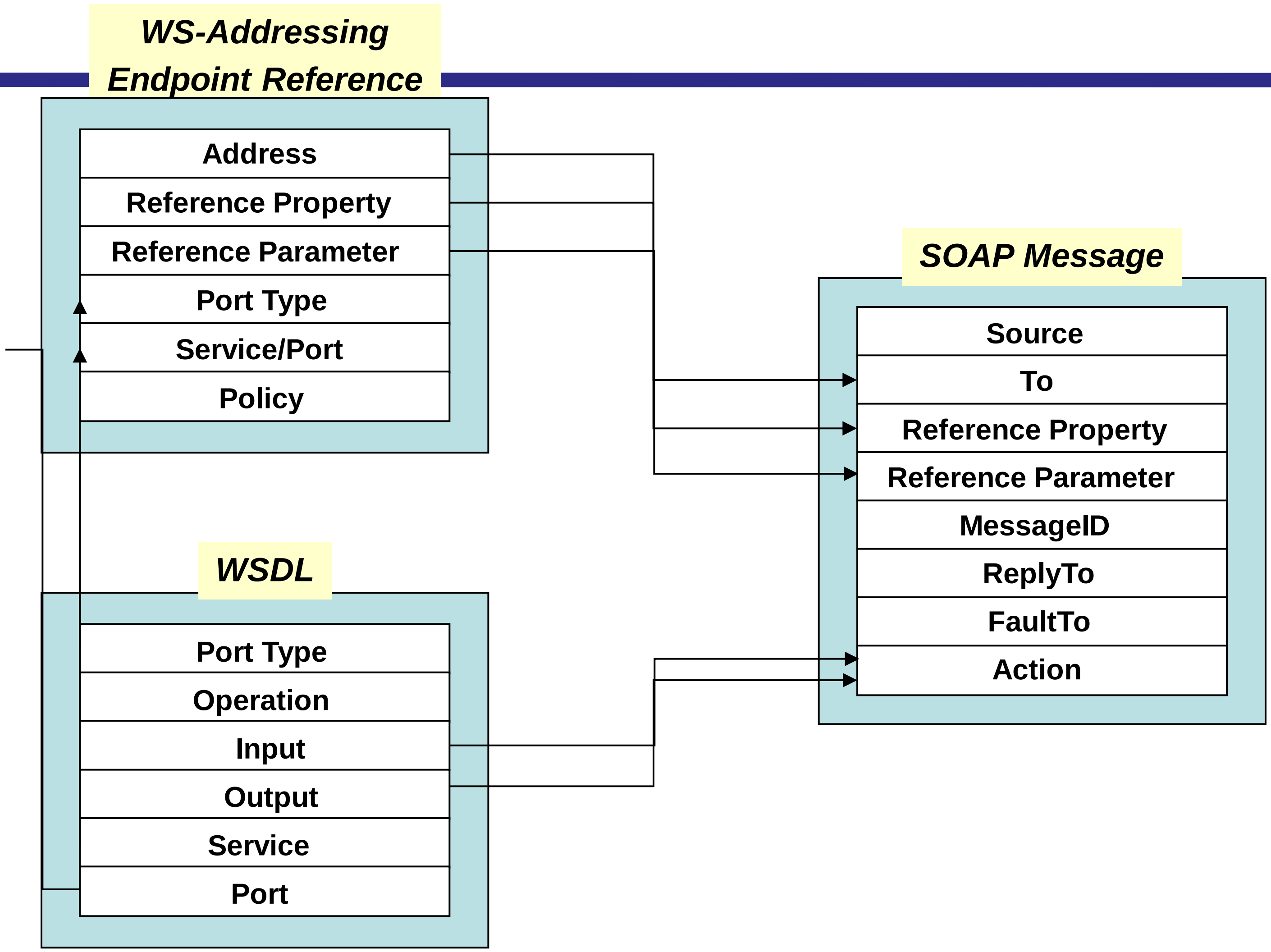
Two endpoint references that share the same URI but specify different reference property values represent two different services. Reference properties are used to dispatch a request to the appropriate service. For example, an application might deploy two different versions of a service and have requests specify a target version in their reference parameters. One service version may target basic service-level customers while the other could target premium service-level customers.

Finally, policies may be included in an endpoint to facilitate easier processing by the consuming application. This is achieved by using the <Policy> element in the example, which describes the behavior, requirements, and non-functional capabilities of the endpoint according to the WS-Policy specification.

Specifying an endpoint reference

```
<wsa:EndpointReference
   xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:tns="http://supply.com/PurchaseService/wsdl"
>
   <wsa:Address> http://supply.com/PurchaseService/wsdl </wsa:Address>
   <wsa:PortType> tns:PurchaseOrderPortType </wsa:PortType>
   <wsa:ServiceName PortName="tns:PurchaseOrderPort">
        tns:PurchaseOrderService
   </wsa:ServiceName>
   <wsa:ReferenceProperties>
       <tns:CustomerServiceLevel> Premium </tns:CustomerServiceLevel>
   </wsa:ReferenceProperties>
   <wsp:Policy>
       <!-- policy statement omitted for brevity -->
   </wsp:Policy>
</wsa:EndpointReference>
```

WS-Addressing: Relationship with SOAP, WSDL fields/structure



WS-Resource-1

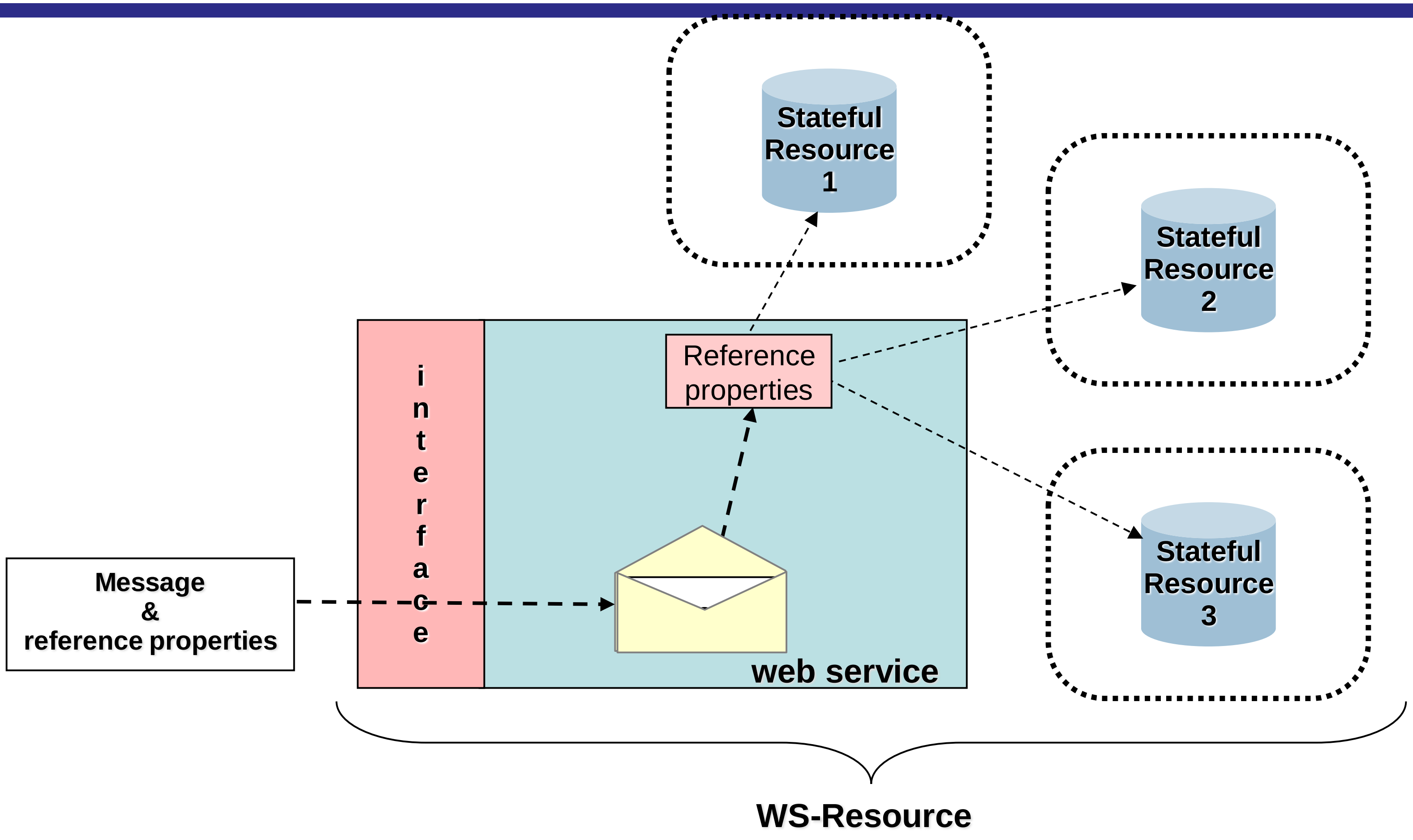
A WS-Resource is defined as the combination of a Web service and a stateful resource [Czajkowski 2004]. This combination is expressed as an association of an XML document with defined type with a Web services <PortType> element.

With the WS-Resources construct the relationship between Web services and stateful resources is encoded in the way that the properties of a WS-Resource are accessible through a Web service interface.

A WS-Resource has a network-wide endpoint reference to identify the Web service and a set of reference properties to uniquely identify the specific resource through the WS-Addressing reference properties.

For example, the next slide shows three stateful resources associated with a single Web service. The stateful resource component of a WS-Resource is identified through the use of a stateful resource identifier carried in the reference properties component in WS-Addressing.

Web service and associated WS-resources



WS-Resource-2

Each WS-Resource has at least one form of identity that identifies that unique WS-Resource within the context of the Web service that provides access to that WS-Resource [Foster 2004].

This identity is not the same as the identity of the Web service, but the Web service can construct an address for an associated WS-Resource by including the local identity information of a WS-Resource into the reference properties portion of a WS-Addressing endpoint reference.

Such an endpoint reference is then to be said WS-Resource-qualified. A WS-Resource qualified endpoint reference can be made available to other entities in a distributed system, which can subsequently use it to direct requests to the WS-Resource.

Using the WS-Resource construct it is possible to completely separate a Web service from its state which results in a stateless Web service. Such a stateless service acts upon stateful resources, provides access and manipulates a set of logical stateful resources based on messages it sends and receives.

To exemplify the use of WS-Resource consider the purchase order service. Here, we assume that purchase orders need to be stateful entities that need to be modeled as WS-Resources that maintain the state associated with the processing of each purchase order request.

The PurchaseOrderPortType

```
<wsdl:message name="POMessage">
    <wsdl:part name="PurchaseOrder" type=“tns:POType"/>
    <wsdl:part name=“CustomerInfo” type=“tns:CustomerInfoType”/>
</wsdl:message>
<wsdl:message name="InvMessage">
    <wsdl:part name=“Invoice" type=“tns:InvoiceType"/>
</wsdl:message>
<wsdl:portType name=“PurchaseOrderPortType">
    <wsdl:operation name=“SendPurchase">
        <wsdl:input message="tns:POMessage"/>
          <wsdl:output message="tns:InvMessage"/>
    </wsdl:operation>
</wsdl:portType>
```

WS-Resource-3

According to the WS-Resource definition given earlier in this subsection a WS-Resource can be defined as the composition of the purchase order Web service and a stateful resource accessed according the implied resource pattern.

The operation SendPurchase in the previous listing then becomes a factory operation to create a new purchase order WS-Resource.

The WS-Resource Framework defines **the factory pattern** that supports an operation that creates and returns endpoint references for one or more new WS-Resources.

The response is no longer an invoice, but rather an endpoint reference pointing to a purchase order WS-Resource.

This purchase order WS-Resource is created as a result of the SendPurchase operation issued by a client. This allows an application to implement the purchase order service by using the implied resource pattern. This simply requires changing the output (response) message in the previous listing as follows:

WS-Resource-4

```
<wsdl:message name="InvMessage">
    <wsdl:part name="POEndPointReference"
element="tns:POReference"/>
</wsdl:message>
```

The POReference element can be defined in the type elements section of the WSDL interface definition file as follows:

```
<xsd: element name="POReference"
    type="wsa: EndPointReferenceType"/>
</xsd:message
```

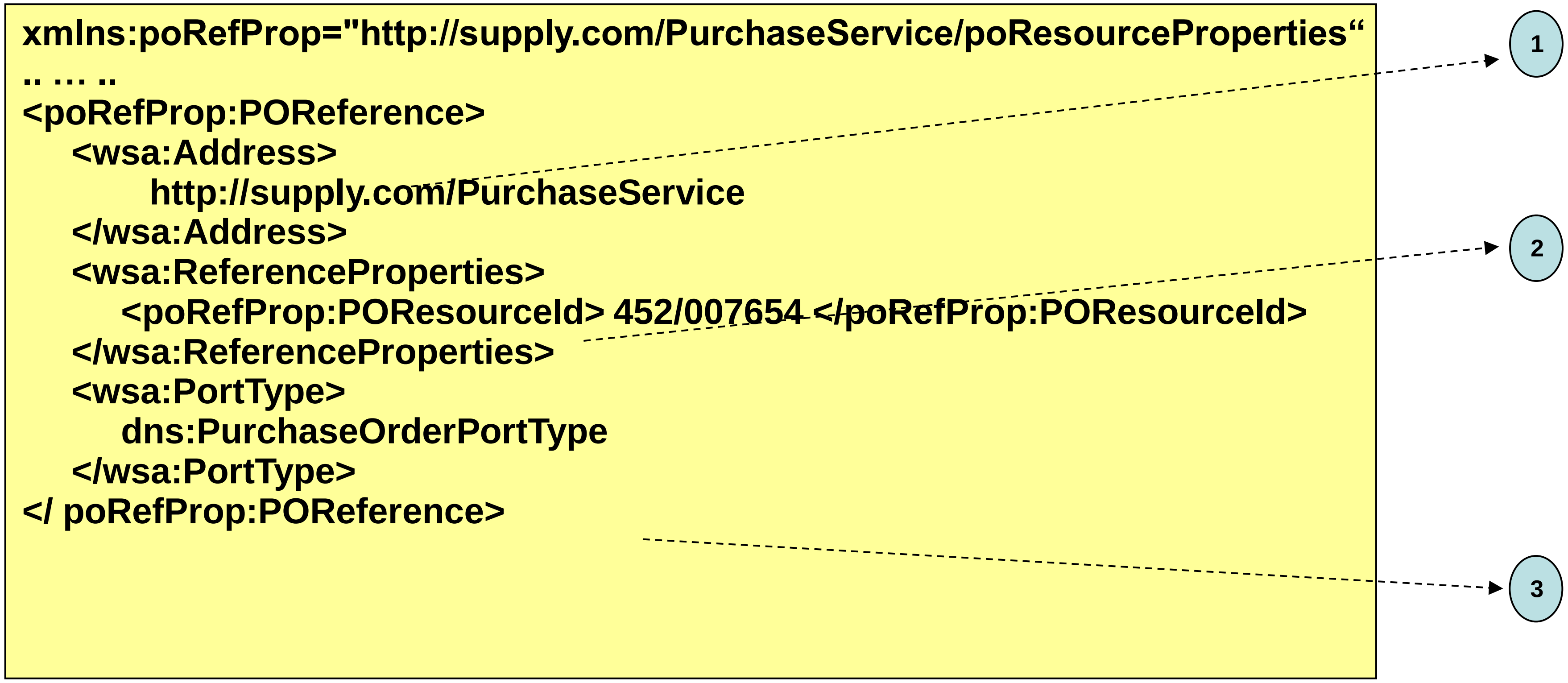
WS-Resource-5

The <PortType> element responsible for purchase order WS-resources provides operations to allow a client to query the contents of a purchase order that the client submitted, query the status of a submitted purchase order, cancel a purchase order and so on.

The content of the response message to a SendPurchase operation contains a WS-Resource qualified endpoint reference to a purchase order WS-Resource, see next figure. This figure identifies three important constructs of the endpoint reference to the purchase order WS-Resource. These are:

- The <Address> element in WS-Addressing that identifies a URL to a Web service that can perform operations on the purchase order WS-Resource.
- The reference properties element that uniquely identifies the specific resource specified by the <Address> element, i.e. the identifier of the purchase order stateful resource created after performing the SendPurchase operation.
- The interface of the Web service, which is identified by the <Address> element. The Web service interface is described by the annotated PurchaseOrderPortType that we shall examine next.

Constructs of the endpoint reference to a purchase order created after the SendPurchase operation



```
xmlns:poRefProp="http://supply.com/PurchaseService/poResourceProperties“
.. …. ..
<poRefProp:POReference>
    <wsa:Address>
          http://supply.com/PurchaseService
    </wsa:Address>
    <wsa:ReferenceProperties>
        <poRefProp:POResourceId> 452/007654 </poRefProp:POResourceId>
    </wsa:ReferenceProperties>
    <wsa:PortType>
        dns:PurchaseOrderPortType
    </wsa:PortType>
</ poRefProp:POReference>
```

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Χρήστος Νικολαου. «**Εισαγωγή στα Δίκτυα Υπηρεσιών. Διάλεξη 15η: WS-Addressing and Notification**». Έκδοση: 1.0. Ηράκλειο/Ρέθυμνο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση: https://elearn.uoc.gr/course/view.php?id=416/