ecma INTERNATIONAL

Technical Report

ECMA TR/90

1st Edition / December 2005

Session Management, Event Notification, and Computing Function Services - Amendments for ECMA-348

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

Technical Report
ECMA TR/90
1st Edition / December 2005

Session Management, Event Notification, and Computing Function Services - Amendments for ECMA-348

Introduction

ECMA-348 provides a Web Service (WS) interface for CSTA using Web Service Description Language (WSDL) 1.1.

CSTA (ECMA-269) requires applications to establish a session, which provides the context for all services and event transactions. ECMA-366 specifies Services for Web Service (WS) based session management. This Technical Report proposes to amend ECMA-348 to use ECMA-366 for session management.

WSDL 1.1 does not support concrete bindings for Notification operations. ECMA-348 uses these operations for CSTA events. This Technical Report proposes to amend ECMA-348 to establish an event channel and three event-sink types using WS-Addressing and WS-Eventing.

WSDL 1.1 does not define fault messages with one-way operation types that are needed for switching function services without a positive acknowledgement. ECMA-348 uses notification operations to convey the fault to those service requests. This Technical Report handles such notifications in the same way as events.

WSDL 1.1 does not support concrete bindings for Solicit-response operations. To allow proper implementation of the Computing Function Services, this Technical Report proposes to add a WSDL interface to ECMA-348 for the Computing Function in addition to the Switching Function WSDL.

When the WS-Addressing and WS-Eventing specifications have become stabilised, a next edition of ECMA-348 should be drafted.

This Ecma Technical Report has been adopted by the General Assembly of December 2005.

Table of contents

1 Scope

ECMA-348 2nd edition specifies a Web service interface for CSTA (ECMA-269) in Web Services Description Language (WSDL) version 1.1. It provides WSDL definitions for solicit-response and notification operations, referred in this Technical Report as outbound operations. These operations are used in ECMA-348 to convey events, some negative responses and computing function services. However, WSDL 1.1 does not specify concrete bindings for those outbound operations.

In particular, WSDL 1.1 does not provide a mechanism for transmitting events, e.g. specifying where the switching function should send CSTA events. This Technical Report proposes to amend ECMA-348 to establish an event channel and event-sink interface using WS-Addressing and WS-Eventing.

This Technical Report proposes to add a Computing Function WSDL interface for proper support of the Computing Function Services of ECMA-348.

CSTA also requires an application context before any services or events can be exchanged, to that end, this Technical Report proposes that the next edition of ECMA-348 uses WS-Session (ECMA-366) Services.

This Technical Report describes amendments to the 2nd edition of ECMA-348 to be effectuated when the WS-Addressing and WS-Eventing specifications become stabilised.

2 References

ECMA-269	Services for Computer Supported Telecommunications Applications (CSTA) Phase III, 6th Edition (June 2004): http://www.ecma-international.org/publications/standards/Ecma-269.htm
ECMA-323	XML Protocol for Computer Supported Telecommunication Applications (CSTA) Phase III, 3rd Edition (June 2004): http://www.ecma-international.org/publications/standards/Ecma-323.htm
ECMA-348	Web Services Description Language (WSDL) for CSTA Phase III, 2nd Edition (June 2004): http://www.ecma-international.org/publications/standards/Ecma-348.htm
ECMA-354	Application Session Services (June 2004): http://www.ecma-international.org/publications/standards/Ecma-354.htm
ECMA-366	WS-Session - Web Service Specification of Application Session Services (June 2005): http://www.ecma-international.org/publications/standards/Ecma-366.htm
W3C-SOAP	Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
W3C-WSDL	Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001: http://www.w3.org/TR/wsdl
WS-Eventing	Web Service Eventing (WS-Eventing) from Oasis, August, 2004, by IBM, Microsoft, BEA, Computer Associates, TIBCO Software, and Sun Microsystems http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-eventing.asp
WS-Eventing-XSD	WS-Eventing XSD: http://schemas.xmlsoap.org/ws/2004/08/eventing
WS-Eventing-WSDL	WS-Eventing WSDL: http://schemas.xmlsoap.org/ws/2004/08/eventing/eventing.wsdl
WS-Addressing	Web Service Addressing (WS-Addressing) from W3C: WS-Addressing 1.0 Core: W3C Working Draft (February 15, 2005): http://www.w3.org/TR/2005/WD-ws-addr-core-20050215

WS-Addressing 1.0 SOAP Binding: W3C last call Working Draft (February 15, 2005): http://www.w3.org/TR/2005/WD-ws-addr-soap-20050215/
WS-Addressing 1.0 WSDL Binding: W3C Working Draft (February 15, 2005): http://www.w3.org/TR/2005/WD-ws-addr-wsdl-20050215/

3 Definitions

Outbound operations - solicit-response and notification operations as defined in 6.4 of ECMA-348.

4 Acronyms and Abbreviations

CF	Computing Function as defined in ECMA-269
CF-SP	Computing Function - Service Provider
CF-SR	Computing Function - Service Requester
SF	Switching Function as defined in ECMA-269
SF-SR	Switching Function - Service Requester
SF-SP	Switching Function - Service Provider
SP	Service Provider as defined in ECMA-348
SR	Service Requestor as defined in ECMA-348
WS	Web Service as defined in W3C-WSDL
WSDL	Web Services Description Language as defined in W3C-WSDL

5 Web Service Based Application Session Management

ECMA-269, on which ECMA-348 is based, requires a computing function (CF) to establish a session (i.e. application association) with the switching function (SF) before any CSTA message can be exchanged.

ECMA-366 defines Web services for application session management. This Technical Report proposes to use these services in ECMA-348.

6 Event Sink Interface Definitions for ECMA-348

ECMA-348 describes four categories of operations from the perspective of the switching function, in which it performs the role of service provider.

The computing function receives three types of outbound operations from the service provider:

1. Event (Notification)
2. One-way fault notification with no acknowledgement (Notification)
3. Request with acknowledgement (Solicit-response)

This Technical Report proposes to add a computing function WSDL to ECMA-348, which leads to the situation that computing function and switching function can be both a service requester and a service provider (i.e. CF-SR, CF-SP, SF-SR, and SF-SP).

The service requester has to specify the event sink to service provider in order to receive event notification. Two general approaches can be used to define these outbound operations for the service requester interface, namely a *Loosely Coupled* interface and a *Tightly Coupled* interface.

6.1 Loosely Coupled (generic) Event Sink Interface

Loosely coupled event sink Interfaces define operations for the categories of events they receive. The loosely coupled event sink interface is based on a “wrapped” message delivery model as defined in WS-Eventing. Events from each category are delivered to their specific event sink interface inside category specific SOAP message. It has the advantage that the computing function interface is loosely-coupled with the switching function interface. It only needs to expose one operation per event category.

The loosely coupled *generic* event sink defines one operation for all categories of events it receives. The loosely coupled *generic* event sink is defined as follows:

```
<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://www.ecma-international.org/standards/ecma-366/ws-session/generic_sink"
    targetNamespace="http://www.ecma-international.org/standards/ecma-366/ws-session/generic_sink">
        <types>
            <xs:schema targetNamespace="http://www.ecma-international.org/standards/ecma-366/ws-
session/generic_sink">
                <xs:complexType name="EventType" mixed="true">
                    <xs:sequence>
                        <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:schema>
        </types>
        <message name=”notifyEvent”>
            <part name="parameter" type="tns:EventType"/>
        </message>
        <portType name="GenericSinkPortType">
            <operation name="NotifyEvent">
                <input message="tns:notifyEvent"/>
            </operation>
        </portType>
</definitions>
```

6.2 Tightly Coupled (typed) Event Sink Interface

The tightly coupled interface defines operations that are exact reversal of the outbound operations as defined in ECMA-348. Each solicit-response operation in ECMA-348 is reversed into a request-response operation, and each notification operation is reversed into a one-way operation for CF service requester interface.

Instead of providing a complete CF service requester side interface, we illustrate the tightly coupled event sink interface for two notifications:

```
<operation name="CSTA-DisplayUpdated-event">
    <input message="tns:displayUpdatedEvent"/>
</operation>

<operation name="CSTA-Delivered-event">
    <input message="tns:deliveredEvent"/>
</operation>
```

This is a reversal of the following outbound operations defined in ECMA-348:

```
<operation name="CSTA-DisplayUpdated-event">
    <output message="tns:displayUpdatedEvent"/>
</operation>

<operation name="CSTA-Delivered-event">
    <output message="tns:deliveredEvent"/>
</operation>
```

Both the service requester and the service provider can perform message (type) validation. But any change to the service provider interface will require the corresponding change in the service requester interface.

6.3 Combined (typed+generic) Event Sink Interface

The event sink can be a combination of typed and loosely coupled event interfaces. The switching function service provider should send the event notifications only to the matched typed event sink interface of the computing function, if existing. Otherwise, the event notification should be delivered to the loosely coupled event interface of the computing function.

A combined (typed+generic) event sink interface is illustrated below in which the generic interface WSDL is imported to a typed interface WSDL.

```
<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:csta=”http://www.ecma-international.org/standards/ecma-323/csta/ed3”
    xmlns:tns="http://www.example.com/csta_combined_sink"
    targetNamespace="http://www.example.com/csta_combined_sink">

<import namespace="http://www.ecma-international.org/standards/ecma-366/ws-session/generic_sink"
location="http://www.ecma-international.org/standards/ecma-366/ws-session/generic_sink/ws-session-
generic-sink-wsdl.wsdl" />

<types>
    <xs:schema targetNamespace="http://www.example.com/csta_combined_sink">
    <xs:import namespace="http://www.ecma-international.org/standards/ecma-323/csta/ed3"
schemaLocation="http://www.ecma-international.org/standards/ecma-323/csta/ed3/csta.xsd"/>
    </xs:schema>
</types>

<message name="displayUpdatedEvent">
    <part name="parameter" element="csta:DisplayUpdatedEvent"/>
</message>
<message name="deliveredEvent">
    <part name="parameter" element="csta:DeliveredEvent"/>
</message>

<portType name="CSTATypedSinkPortType">
    <operation name="CSTA-DisplayUpdated-event">
        <input message="tns:displayUpdatedEvent"/>
    </operation>
     <operation name="CSTA-Delivered-event">
        <input message="tns:deliveredEvent"/>
    </operation>
</portType>
</definitions>
```

6.4 Event Sink Interface Design

Both loosely coupled and tightly coupled event sink interfaces can be used for event notification. But they are based on two different architectures. In tightly coupled event sink interface, the event sink interface is a mirrored reversal of the service provider interface. Changes made to the service provider interface can impact all its service requesters.

On the other hand, loosely coupled event sink interface can allow the service requester and service provider to evolve separately as long as they maintain the loosely coupled event interface relation. A resource constrained service requester may dispatch the service event notifications received to a more resource rich platform for processing, and it needs only a very light and stable event sink interface for the event notification to pass through.

The tightly coupled event sink interface has the advantage of strict type checking by both the service provider and service requester. On the other hand, the loosely coupled event sink interface allows the service requester to do late binding with other distributed processing resources.

Both tightly coupled and loosely coupled event interfaces can be applied, and in many cases, a combined (typed+generic) event sink interface is preferred. The tightly coupled event sink interface can be used for operations which are more stable and loosely coupled event sink interface can be used for operations which are more dynamic and subject to change. WS-Session Annex C specifies a default generic (loosely coupled) event sink interface for all subscribed events within the session, and the typed (tightly coupled) event interface as optional.

Service providers should support the combined (typed+generic) event sink interface, and the service requester should specify its event sink interface type declaratively to the service provider during the event subscription as defined in Clause 6. A use case of combined event sink interface is provided in WS-Session Annex C for ApplicationSessionTerminated service subscription, and the event sink interface type should be declared in the event sink endpoint reference as specified in WS-Addressing.

7 WS Interface for Computing Function Services

ECMA-348 defines solicit-response operations for computing functions services such as Call Detail Report Services. WSDL 1.1 does not provide the concrete binding for these operations.

In addition to the WSDL definition for the switching function in 2nd edition of ECMA-348, the computing function should also have its WSDL definition. Only those operations, which are modelled in ECMA-348 as solicit-response operations, are included in the computing function WSDL. The computing function WSDL contains the operations, which are the reversal of the corresponding operations defined in the switching function WSDL.

Reversal means that the solicit-response operation that contains an output element followed by an input element as defined in ECMA-348 must be reversed into an input element followed by an output element in the computing function WSDL.

For instance, ECMA-348 defines CSTA-Call-Data-Recording-Services operation as follows:

```
<operation name="CSTA-CDR-Notification">
    <output message="tns:cDRNotificationResponse"/>
    <input message="tns:cDRNotification"/>
    <fault name="FaultName" message="tns:negResponse"/>
</operation>
```

The corresponding CF operation is defined as follows, which is the reversal of the operation defined in the service provider WSDL interface of ECMA-348.

```
<operation name="CSTA-CDR-Notification">
    <input message="tns:cDRNotificationResponse"/>
    <output message="tns:cDRNotification"/>
    <fault name="FaultName" message="tns:negResponse"/>
</operation>
```

8 Subscription Patterns for ECMA-348

CSTA switching functions can be viewed as hosting various event sources, such as WS-Session, CSTA monitors, and registrations. Event sources can be enabled using a CSTA Service Request, e.g. a MonitorStart Service Request creates a monitor. Switching functions may notify computing functions of these events following subscription. The computing function may link each of these event sources to its event sink.

WS-Eventing can be used to manage the many-to-many relationships between the Event sinks and sources through subscriptions as illustrated in Figure 1.

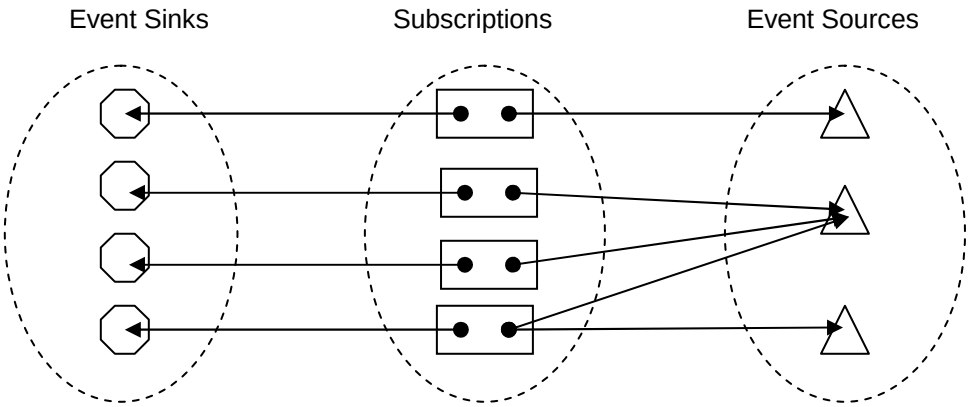

Figure 1 - Subscription patterns for ECMA-348

The two subscription patterns for using CSTA with WS-Eventing, without requiring any modification to the XML Schemas of ECMA-348 or Application Session Services (WS-Session, ECMA-354) are:

1) Source-Sink: In this pattern, the computing function creates the CSTA sessions, monitors, and registrations using CSTA Service Requests before it subscribes to these entities one by one and it provides event sink for each subscription.

2) Sink-Source: In this pattern, the CF Service Requester enables a pool of event sinks by using the Subscribe message of WS-Eventing before the computing function creates the CSTA event sources using CSTA Service Requests augmented with references to the subscription managers from the preceding subscriptions.

8.1 Source-Sink Subscription Pattern

WS-Eventing defines the semantics for the **Source-Sink** pattern.

However, when a monitor is created and it begins to fire events before the Subscribe operation is completed, the events for the monitor may be lost because the switching function has no place to send the events. If the events were to be queued until an event sink would be linked to it, not only would it consume resources but also events might lose their critical time value in real-time communication. Using Snapshot Service Requests, computing functions can “catch up” with the state of calls and devices, as illustrated in Figure 2 where the X in SnapShotX/SnapShotXResponse can either be “Call” or “Device” (dotted lines indicate optional messages).

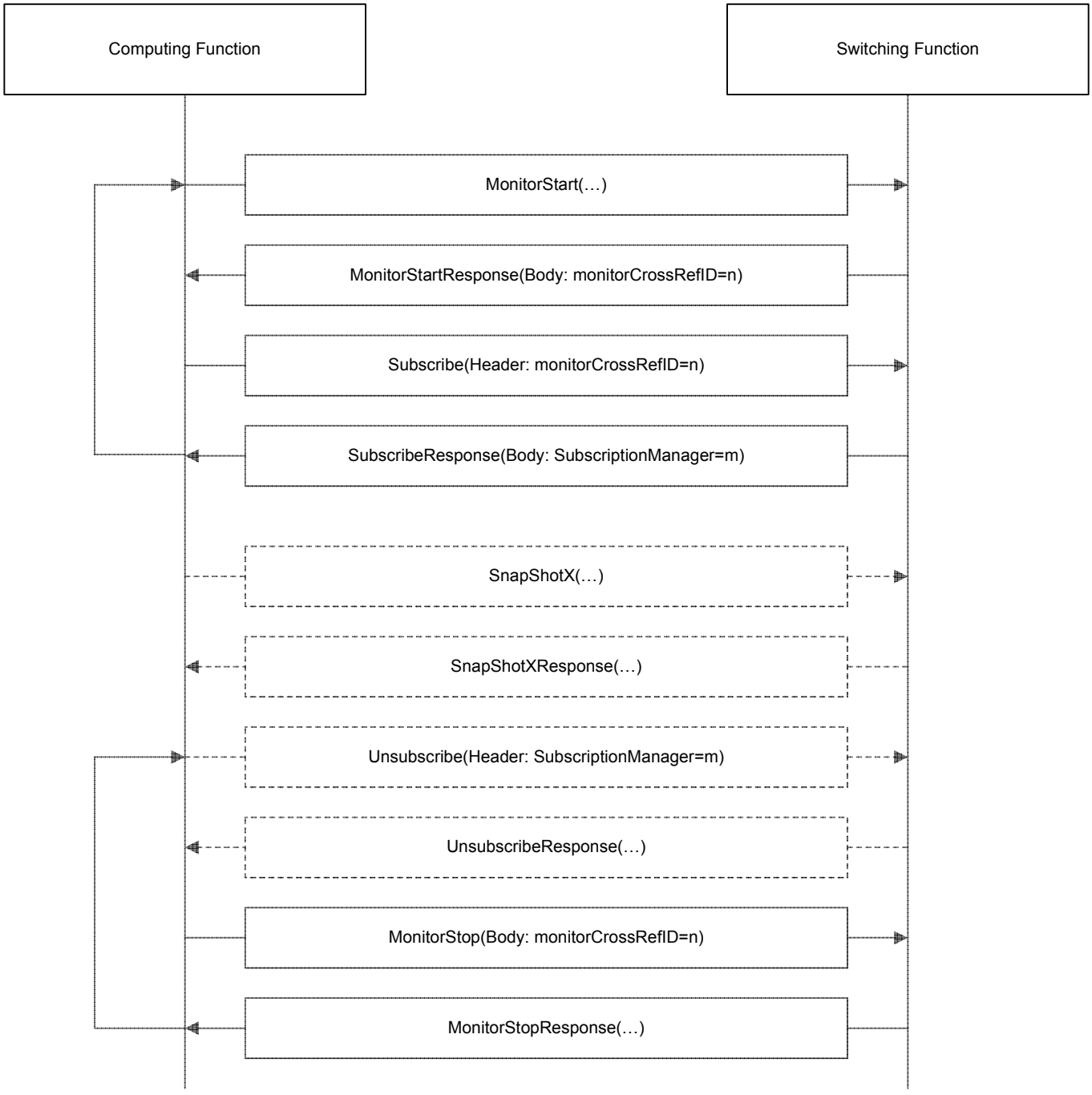

Figure 2 - Source-Sink Subscription Pattern

The use of Application Session Services as defined in ECMA-366 is the exception to the rule, where StartApplicationSession must be used to create a Session before Subscribing to the ApplicationSessionTerminated event. If the application session were to terminate abnormally before the ApplicationSessionTerminated event can be subscribed to, later subscriptions to this event will result in a corresponding Fault message that hints to the occurance of the ApplicationSessionTerminated event.

8.2 Sink-Source Subscription Pattern

The **Sink-Source** pattern avoids the timing problem of the **Source-Sink** pattern because computing functions subscribe to event sources before they create and enable event sources. In

this pattern, the switching function uses the SubscriptionManager in the SOAP header of the CSTA Request such as MonitorStart, to link the event source, such as a monitor, to its sink or sinks.

In sink-source pattern subscriptions within a session, the session is not the intended event source but a context for the subscription message. To make such distinction within the framework of WS-Eventing, additional elements must be used in the SOAP header of subscription messages to identify the intended event source. Detailed SOAP message examples are given in 8.6.

The loosely coupled event sink interface as specified in 6.1 is needed when one CSTA message can create several sources. The sink-source interaction pattern is illustrated in Figure 3.

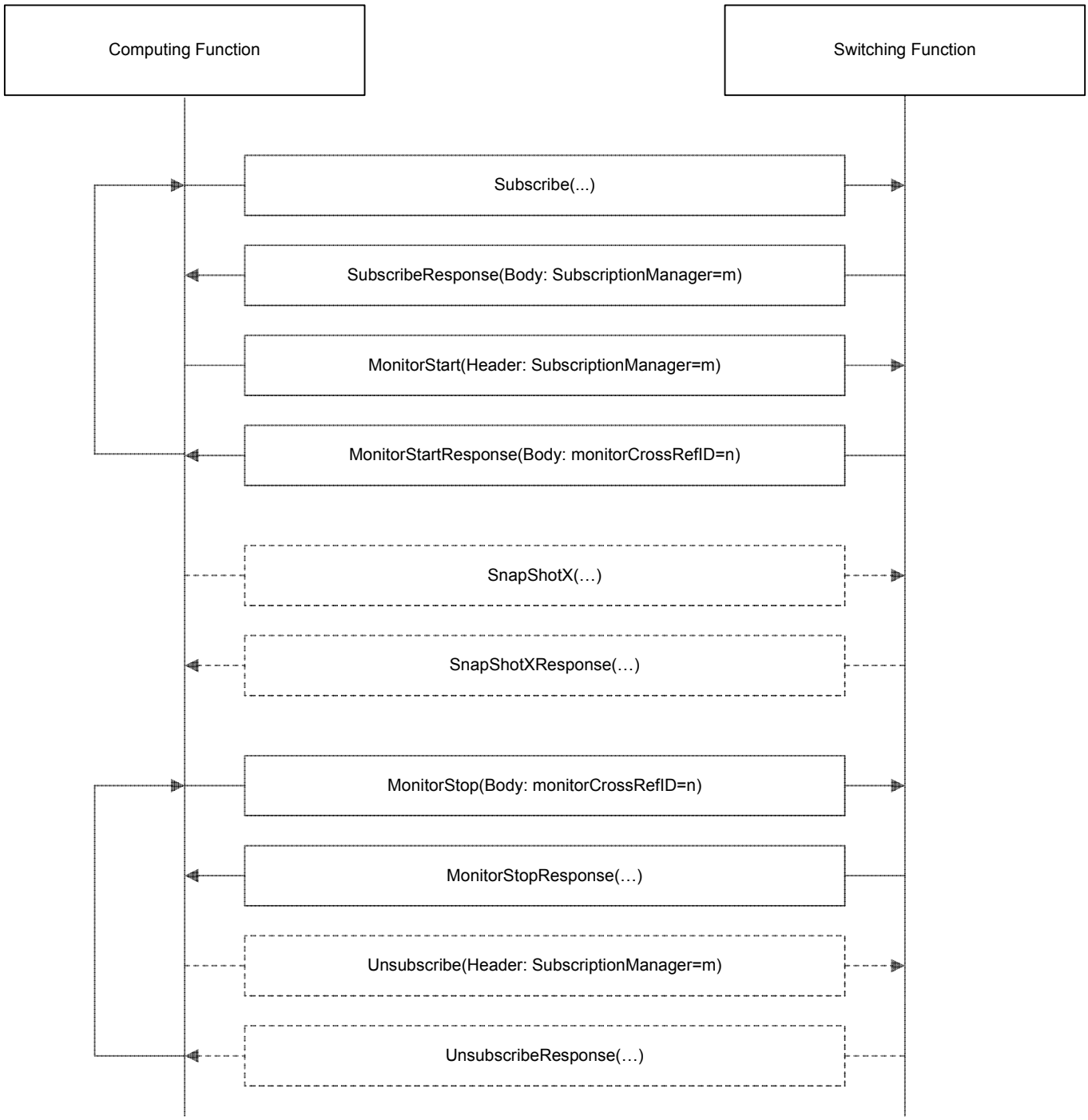

Figure 3 - Interaction of the Sink-Source Subscription Pattern

A loosely coupled generic event sink using the sink-source subscription model can be established easily through the mechanism of default subscription as defined in 8.3.

8.3 Default Subscription

The loosely coupled generic event sink interface for ApplicationSessionTerminated event of WS-Session may be used as the default sink for other event sources within the Application Session (monitor, registration, etc.) for which a subscription is not (yet) defined.

8.4 Life Cycle of Subscriptions

For the Source-Sink pattern, an event subscription becomes effective once the WS-Eventing Subscribe message returns with a positive response. For the Sink-Source pattern, an event subscription becomes effective when the CSTA message with reference to the subscription returns with a positive response.

A subscription for an event source can be deleted once its source is removed. For instance, if a monitor is removed for any reason, all subscriptions on the monitor are deemed invalid and cannot be renewed. A subscription can be deleted explicitly by using the WS-Eventing Unsubscribe message before or after the event source is removed.

8.5 SOAP Subscription Messages

As discussed in previous sections, the subscription to WS-Session follows the source-sink pattern, and the subscriptions to CSTA event sources can follow the sink-source pattern. Figure 4 illustrates using both patterns of subscriptions, to establish sessions and many-to-many relationships among event sinks and sources.

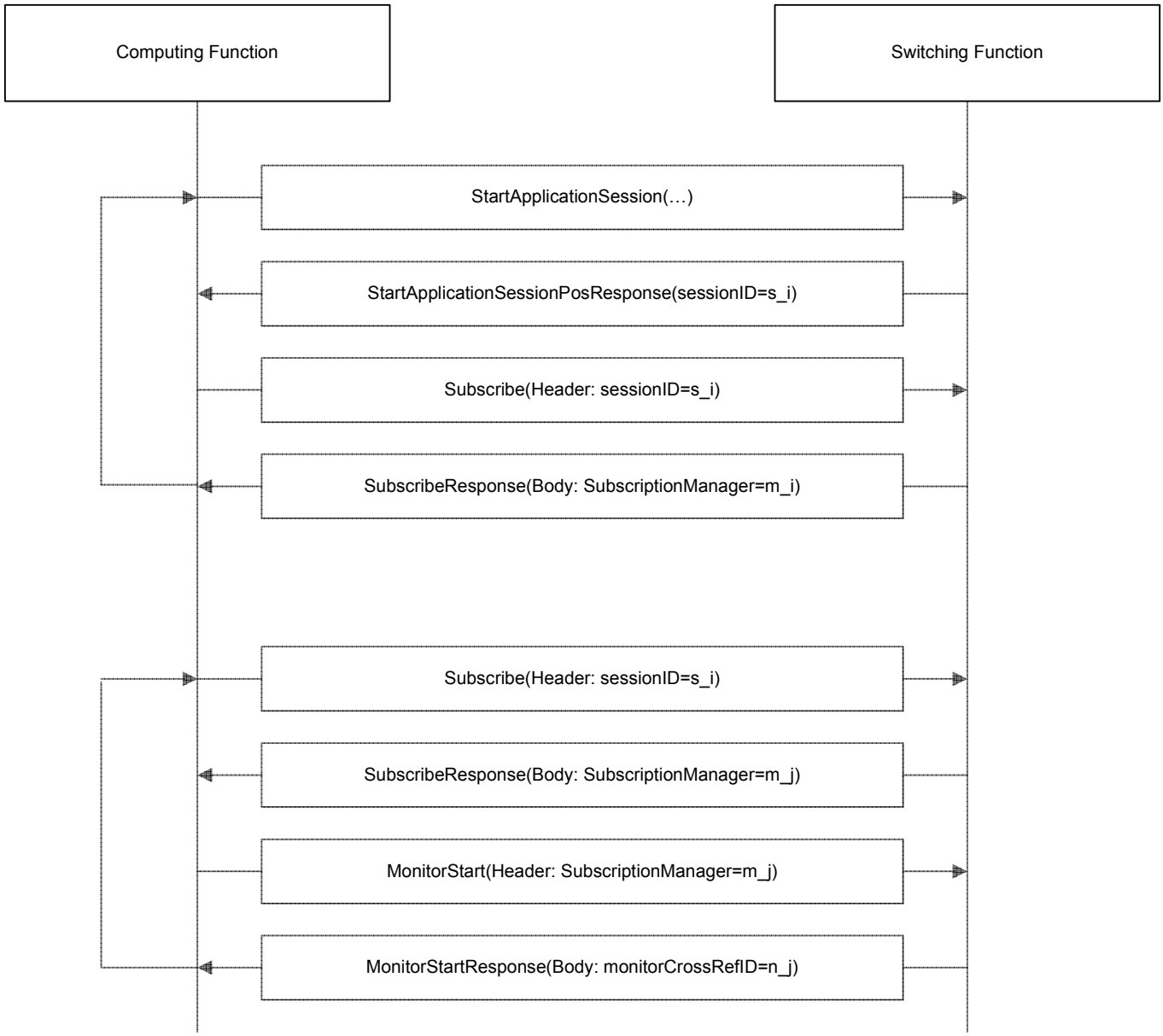

Figure 4 - combined Source-sink and sink-source Subscription pattern with WS-Session

8.6 Examples of SOAP Subscription Messages

The following is an example WS-Eventing Subscribe message to create a source-sink subscription pattern to an Application Session. The event source is identified by the aps:sessionID element.

```
<S:Envelope
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2005/02/addressing"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:aps="http://www.ecma-international.org/standards/ecma-354/appl_session"
    xmlns:esi="http://www.ecma-international.org/standards/ecma-366/event_sink_interface">
        <S:Header>
            <wsa:To>http://www.example.com/CSTA_Server</wsa:To>
            <aps:sessionID>5555</aps:sessionID>
            <wsa:MessageID>abcd</wsa:MessageID>
            <wsa:ReplyTo>
                <wsa:Address>http://www.example.com/CSTA_requester</wsa:Address>
            </wsa:ReplyTo>
            <wsa:Action>http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe</wsa:Action>
        </S:Header>
        <S:Body>
            <wse:Subscribe>
                <wse:Delivery>
                    <wse:NotifyTo>
                        <wsa:Address>http://www.example.com/CSTA_sink</wsa:Address>
                        <esi:interface type="generic|typed|typed+generic" />
                    </wse:NotifyTo>
                </wse:Delivery>
        </wse:Subscribe>
        </S:Body>
</S:Envelope>
```

A positive response to the Subscription request returns a SubscriptionManager element that uniquely identifies the subscription with a wse:Identifier element.

```
<S:Envelope
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2005/02/addressing"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:aps="http://www.ecma-international.org/standards/ecma-354/appl_session" >
        <S:Header>
            <aps:sessionID>5555</aps:sessionID>
            <wsa:To>http://www.example.com/CSTA_requester</wsa:To>
            <wsa:RelatesTo>abcd</wsa:RelatesTo>
            <wsa:Action>http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscribeResponse</wsa:Action>
        </S:Header>
        <S:Body >
            <wse:SubscribeResponse>
                <wse:SubscriptionManager>
                    <wsa:Address>http://www.example.com/CSTA_Subscription</wsa:Address>
                    <wsa:ReferenceParameters>
                        <wse:Identifier>1234</wse:Identifier>
                    </wsa:ReferenceParameters>
                </wse:SubscriptionManager>
            </wse:SubscribeResponse>
        </S:Body>
</S:Envelope>
```

For the sink-source subscription pattern, the following SOAP message should be used within a session to create a subscription for Monitors. The empty csta:monitorCrossRefID element in the SOAP header designates that the subscription is for a type of event source within the session.

```
<S:Envelope
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2005/02/addressing"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:aps="http://www.ecma-international.org/standards/ecma-354/appl_session"
    xmlns:csta="http://www.ecma-international.org/standards/ecma-323/csta/ed3"
    xmlns:esi="http://www.ecma-international.org/standards/ecma-366/event_sink_interface">
        <S:Header>
            <aps:sessionID>5555</aps:sessionID>
            <csta:monitorCrossRefID />
            <wsa:To>http://www.example.com/CSTA_Server</wsa:To>
            <wsa:MessageID>abcd</wsa:MessageID>
            <wsa:ReplyTo>
                <wsa:Address>http://www.example.com/CSTA_requester</wsa:Address>
            </wsa:ReplyTo>
            <wsa:Action>http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe</wsa:Action>
        </S:Header>
        <S:Body>
            <wse:Subscribe>
                <wse:Delivery>
                    <wse:NotifyTo>
                        <wsa:Address>http://www.example.com/CSTA_sink</wsa:Address>
                        <esi:interface type="generic|typed|typed+generic" />
                    </wse:NotifyTo>
                </wse:Delivery>
            </wse:Subscribe>
        </S:Body>
</S:Envelope>
```

The positive response to the subscription request above returns a SubscriptionManager element that uniquely identifies the subscription with a wse:Identifier element. An example of a subsequent MonitorStart Service Request whose SOAP header contains a reference to the event subscription is illustrated below.

```
<S:Envelope
    xmlns:csta="http://www.ecma-international.org/standards/ecma-323/csta/ed3"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2005/02/addressing"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:aps="http://www.ecma-international.org/standards/ecma-354/appl_session" >
        <S:Header>
            <aps:sessionID>5555</aps:sessionID>
            <invokeID>1</invokeID>
            <wse:SubscriptionManager>
                <wsa:Address>http://www.example.com/CSTA_Subscription</wsa:Address>
                <wsa:ReferenceParameters>
                    <wse:Identifier>1234</wse:Identifier>
                </wsa:ReferenceParameters>
            </wse:SubscriptionManager>
        </S:Header>
        <S:Body >
            <csta:MonitorStart><!-- omitted --></csta:MonitorStart>
        </S:Body>
</S:Envelope>
```

CSTA registration requests, such as RouteRegister, can refer to the subscription (manager) in the same way. A successful completion of the request indicates that the event sink or service requester service of the object (session, monitor, or registration) is defined in the subscription. If the SubscriptionManager is invalid, a negative response should be returned.

A sink-source pattern subscription within a session for subsequent CSTA RouteRegister message is illustrated below.

```
<S:Envelope
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2005/02/addressing"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:aps="http://www.ecma-international.org/standards/ecma-354/appl_session"
    xmlns:csta="http://www.ecma-international.org/standards/ecma-323/csta/ed3"
    xmlns:esi="http://www.ecma-international.org/standards/ecma-366/event_sink_interface">
        <S:Header>
            <aps:sessionID>5555</aps:sessionID>
            <csta:routeRegisterReqID />
            <wsa:To>http://www.example.com/CSTA_Server</wsa:To>
            <wsa:MessageID>abcd</wsa:MessageID>
            <wsa:ReplyTo>
                <wsa:Address>http://www.example.com/CSTA_requester</wsa:Address>
            </wsa:ReplyTo>
            <wsa:Action>http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe</wsa:Action>
        </S:Header>
        <S:Body>
            <wse:Subscribe>
                <wse:Delivery>
                    <wse:NotifyTo>
                        <wsa:Address>
                            http://www.example.com/CSTA_client
                        </wsa:Address>
                        <esi:interface type="typed" />
                    </wse:NotifyTo>
                </wse:Delivery>
            </wse:Subscribe>
        </S:Body>
</S:Envelope>
```

An example of CSTA RouteRegister message associated with a subscription created by sink-source pattern subscription is given below.

```
<S:Envelope
    xmlns:csta="http://www.ecma-international.org/standards/ecma-323/csta/ed3"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2005/02/addressing"
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:aps="http://www.ecma-international.org/standards/ecma-354/appl_session" >
    <S:Header>
        <aps:sessionID>5555</aps:sessionID>
        <invokeID>2</invokeID>
        <wse:SubscriptionManager>
            <wsa:Address>http://www.example.com/CSTA_Subscription</wsa:Address>
            <wsa:ReferenceParameters>
                <wse:Identifier>5678</wse:Identifier>
            </wsa:ReferenceParameters>
        </wse:SubscriptionManager>
    </S:Header>
    <S:Body >
        <csta:RouteRegister><!-- omitted --></csta:RouteRegister>
    </S:Body>
</S:Envelope>
```

An example of notification message of CSTA DeliveredEvent event to a generic event sink is given below.

```
<S:Envelope
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2005/02/addressing"
    xmlns:aps="http://www.ecma-international.org/standards/ecma-354/appl_session"
    xmlns:csta="http://www.ecma-international.org/standards/ecma-323/csta/ed3">
    <S:Header>
        <aps:sessionID>5555</aps:sessionID>
        <invokeID>9999</invokeID>
        <wsa:To> http://www.example.com/CSTA_sink </wsa:To>
        <wsa:Action>http://www.ecma-international.org/standards/ecma-366/generic_sink/GenericSinkPort
Type/NotifyEvent</wsa:Action>
    </S:Header>
    <S:Body>
        <csta:DeliveredEvent>
            <csta:monitorCrossRefID>1</csta:monitorCrossRefID>
            <!-- omitted →
        </csta:DeliveredEvent>
    </S:Body>
</S:Envelope>
```

An example notification operation carrying a DeliveredEvent to a typed event sink is given below.

```
<S:Envelope
      xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:wsa="http://schemas.xmlsoap.org/ws/2005/02/addressing"
      xmlns:aps="http://www.ecma-international.org/standards/ecma-354/appl_session"
      xmlns:csta="http://www.ecma-international.org/standards/ecma-323/csta/ed3">
      <S:Header>
            <aps:sessionID>5555</aps:sessionID>
            <invokeID>9999</invokeID>
            <wsa:To> http://www.example.com/CSTA_sink </wsa:To>
            <wsa:Action>http://www.example.com/csta_combined_sink/CSTATypedSinkPortType/CSTA-
Delivered-event</wsa:Action>
      </S:Header>
      <S:Body>
            <csta:DeliveredEvent>
                  <csta:monitorCrossRefID>2</csta:monitorCrossRefID>
                  <!—omitted →
            </csta:DeliveredEvent>
      </S:Body>
</S:Envelope>
```