MEP SSDL Protocol Framework

Savas Parastatidis[1], Jim Webber[2]
Savas@Parastatidis.name, Jim@Webber.name

Abstract

The Message Exchange Patterns (MEP) SSDL Protocol Framework defines a collection of XML Infoset element information items that represent commonly used simple exchange patterns.

SSDL documents

An Introduction to the SOAP Service Description Language [1]

SOAP Service Description Language (SSDL) v1.3 [2]

MEP SSDL Protocol Framework v1.3 [3] (this document)

CSP SSDL Protocol Framework v1.3 [4]

Rules SSDL Protocol Framework v1.3 [5]

Sequencing Constraints SSDL Protocol Framework v1.3 [6]

Status of this document

Version: 1.3

Date: April 2005

http://ssdl.org

Disclaimer

The contents of this document may not reflect the views of, and may not be endorsed by, the employers of the authors. This document is provided for informational purposes only. The authors and their employers will not accept any responsibility for any use or misuse of the information contained herein.

[1] School of Computing Science, University of Newcastle, Newcastle upon Tyne, NE1 7RU, UK
[2] ThoughtWorks Australia Pty. Ltd.

Table of Contents

1. Introduction

The MEP SSDL protocol framework defines the structure and semantics of elements that represent commonly used message exchange patterns. The current set of MEPs follows that found in the latest draft of the WSDL specification [7].

The MEP SSDL Protocol Framework does not demonstrate the full strength of SSDL [8] and the ability to describe arbitrary protocols. It is made available as a way to capture those MEPs that are defined by WSDL and thus allows SSDL to be used as a simple, SOAP-centric contract language in those places where WSDL may normally be deployed.

1.1. SSDL

The SOAP Service Description Language (SSDL) is a SOAP-centric contract description language for Web Services. It is meant as a means for exploring ideas in the areas of contract and protocol description and Web Services implementations using message-oriented programming abstractions.

1.1.1. Motivation

SOAP is the standard message transfer protocol for Web Services. However, the default description language for Web Services (WSDL) does not explicitly target SOAP but, instead, provides a generic framework for the description of network-exposed software artefacts. The work on SSDL aims to investigate the advantages/disadvantages of Web Services description when SOAP is assumed from the outset compared to the transfer-independent approach of WSDL. The use of formal models for describing message-based interactions is also a goal for SSDL. Finally, this work aims to demonstrate the benefits of focusing on message-orientation when architecting, designing, and building Web Services rather than on the interface and remote procedure call abstractions.

1.1.2. The Language

The SOAP Service Description Language provides the base framework for a range of protocol description frameworks which at one end of the spectrum can be a simpler, SOAP-focussed, direct replacement for WSDL MEPs while at the other end of the spectrum can enable formal validation and reasoning about the protocols that a Web Service supports.

The frameworks are componentised in a similar way to the WS-Policy suite of specifications, with a base SSDL framework providing the fundamental protocol building blocks for describing messages while other specifications utilise those building blocks to describe the way in which the messages participate in the protocols that a Web Service supports. This is illustrated in Figure 1.

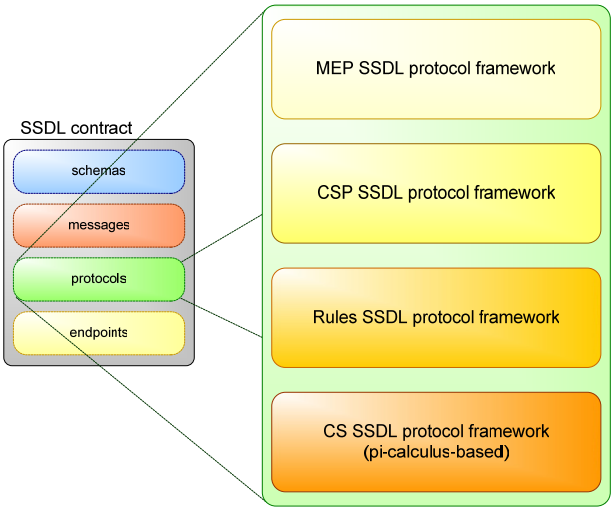

Figure 1 The SSDL Suite of Specifications

Four protocol description frameworks are provided with the base SSDL specification but it is expected that others will be implemented to meet different needs. The protocol description frameworks provided with the initial release of SSDL 1.0 are:

- MEP - The Message Exchange Patterns (MEP) Framework defines a collection of XML Infoset element information items that represent commonly used simple exchange patterns. The current set of message exchange patterns supported by the MEP framework is a superset of that found in the latest draft of the WSDL specification.
- CSP – The Communicating Sequential Processes (CSP) Framework defines a collection of XML Infoset element information items for defining a multi-message exchange using sequential process semantics, based on basic CSP semantics. Work is currently underway to make use of the full strength of the CSP in describing contracts.
- Rules - The Rules-based SSDL Protocol Framework defines a collection of XML Infoset element information items that can be used to describe a multi-message exchange protocol using conditions. The protocols captured using the Rules-based SSDL protocol framework can be validated for correctness, liveness, and other properties.
- SC - The Sequencing Constraints (SC) Protocol Framework defines a collection of XML Infoset element information items that can be used to describe a multi-party, multi-message exchange protocol using notations based on the pi-calculus. Protocols in the framework are specified using a sequential technique, specifying the legal set of actions at each stage of the protocol. The framework is intended to provide a simple way of specifying protocols but also have a formal basis to allow properties of the protocols to be determined.

1.1.3. Key Features

- SSDL assumes SOAP as the means of transferring messages between Web Services over arbitrary transport (and transfer) protocols. It has been designed to work harmoniously with all aspects of the underlying SOAP processing model. As a result, there is no need to define bindings for all possible transport protocols;
- SSDL assumes WS-Addressing as the standard means for embedding addressing information within SOAP envelopes and for binding those addresses onto underlying transport protocols;
- SSDL focuses on messages and protocols. As a result, there is no need for articles like ‘interface’, ‘inheritance’, and ‘operation’;
- XML Infoset is assumed as the underlying SSDL component model. There is no need (nor desire) to create a new component model simply for contract description;
- Modularisation of contracts is handled using XInclude. A shortcut mechanism is provided which is defined in terms of XInclude elements to simplify componentisation as far as is possible;
- SSDL promotes protocol framework extensibility. It allows different protocol description models to be plugged into the base SSDL framework which helps promote protocol-based integration and exposure of the messaging behaviour of a Web Service. Tools such as model checkers can verify the correctness of protocols defined in an SSDL contract, or automate the reasoning about the compatibility of Web Services. Hosting environments can even use the SSDL contract to validate the message exchanges between Web Services.

1.2. Goals

- Define a set of message exchange patterns that capture the same semantics as those defined by the WSDL spec

- Use WS-Addressing for message correlation where necessary

1.3. Example

```
<?xml version="1.0" encoding="utf-8" ?>
<ssdl:contract targetNamespace="http://example.org/service/contract"
               xmlns:ssdl="urn:ssdl:v1">
  <ssdl:schemas>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
               targetNamespace="http://exaxmple.org/service/schema.xsd">

      <xs:element name="AvailabilityCheckRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element  name="checkInDate" type="xs:date"/>
            <xs:element  name="checkOutDate" type="xs:date"/>
            <xs:element  name="roomType" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <xs:element name="AvailabilityCheckResponse" type="xs:double"/>

      <xs:element name="InvalidDataError" type="xs:string"/>
    </xs:schema>
  </ssdl:schemas>

  <ssdl:messages targetNamespace="http://example.org/service/messages"
                 xmlns:tns="http://exaxmple.org/service/schema.xsd">

    <ssdl:message name="AvailabilityCheckRequestMsg">
      <ssdl:body ref="tns:AvailabilityCheckRequest"/>
    </ssdl:message>

    <ssdl:message name="AvailabilityCheckResponseMsg">
      <ssdl:body ref="tns:AvailabilityCheckResponse"/>
    </ssdl:message>

    <ssdl:fault name="InvalidDataErrorFaultMsg">
      <ssdl:code value="Sender"/>
    </ssdl:fault>

  </ssdl:messages>

  <ssdl:protocols>
    <ssdl:protocol targetNamespace="http://example.org/service/protocol"
                   xmlns:msgs="http://example.org/service/messages"
                   xmlns:mep="urn:ssdl:mep:v1">

      <mep:in-out>
        <!-- first two messages are the request-response pair -->
        <ssdl:msgref ref="msgs:AvailabilityCheckRequestMsg"/>
        <ssdl:msgref ref="msgs:AvailabilityCheckResponseMsg"/>

        <!-- all other messages are faults -->
        <ssdl:msgref ref="msgs:InvalidDataErrorFaultMsg"/>
     </mep:in-out>

   </ssdl:protocol>

   <ssdl:endpoints>
     <ssdl:endpoint xmlns:wsa="http://www.w3.org/2004/12/addressing">
       <wsa:Address>http://example.org/service</wsa:Address>
     </ssdl:endpoint>
   </ssdl:endpoints>
</ssdl:contract>
```

Example 1: A simple contract

In the example, a request-response message exchange pattern is defined using the `mep:in-out` element information item.

1.4. Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "MAY", "MAY NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [9].

This specification uses properties from the XML Information Set [10]. Such properties are denoted by square brackets and in bold, e.g. **[namespace name]**.

This specification uses namespace prefixes throughout; they are listed in Table 1-1. Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [10]).

We use the pseudo-schema notation used in WSDL 2.0 Core [11] as a convenient description of the structure of a component.

1.5. Namespaces

These namespaces and their prefixes are used throughout this document.

Prefix	Namespace	Notes
ssdl	urn:ssdl:v1	
xs	http://www.w3.org/2001/XMLSchema	
wsa	http://www.w3.org/2004/12/addressing	
mep	urn:ssdl:mep:v1	Where elements are not qualified with a namespace prefix, urn:ssdl:mep:v1 is assumed

2. Protocol Framework Structure

```
<protocol>
  [ <mep:in-only/> |
    <mep:robust-in-only/> |
    <mep:in-out/> |
    <mep:in-optional-out/> |
    <mep:out-only/> |
    <mep:robust-out-only/> |
    <mep:out-in/> |
    <mep:out-optional-in/> ] *
</protocol>
```

The Rules SSDL Protocol Framework defines the following element information items in the **[children]** property of the `protocol` element information item:

- An OPTIONAL `in-only` element information item
- An OPTIONAL `robust-in-only` element information item
- An OPTIONAL `in-out` element information item
- An OPTIONAL `in-optional-out` element information item
- An OPTIONAL `out-only` element information item
- An OPTIONAL `robust-out-only` element information item
- An OPTIONAL `out-in` element information item
- An OPTIONAL `out-optional-in` element information item

2.1. in-only

```
<protocol>
  <mep:in-only>
    <msgref direction="in" />
  </mep:in-only>
</protocol>
```

The `in-only` information item has the following properties:

- A **[local name]** of "in-only"
- A **[namespace]** of "urn:ssdl:mep:v1"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "in"

An `in-only` element information item means that the referred message CAN be received.

2.2. robust-in-only

```
<protocol>
  <mep:robust-in-only>
    <msgref direction="in" />
    <msgref direction="out" /> +
  </mep:robust-in-only>
</protocol>
```

The `robust-in-only` information item has the following properties:

- A **[local name]** of "robust-in-only"
- A **[namespace]** of "urn:ssdl:mep:v1"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "in"
- One or more `ssdl:msgref` element information items with the **[normalised value]** of their `ref` attribute information item set to "out" and the **[normalised value]** of their `ref` attribute information item set to the qualified name of a fault message

A `robust-in-only` element information item means that the first referred message CAN be received. If a fault is triggered, one of the referred fault messages MAY be sent.

2.3. in-out

```
<protocol>
  <mep:in-out>
    <msgref direction="in" />
    <msgref direction="out" />
    <msgref direction="out" /> *
  </mep:in-out>
</protocol>
```

The `in-out` information item has the following properties:

- A **[local name]** of "in-out"
- A **[namespace]** of "urn:ssdl:mep:v1"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "in"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "out"

- Zero or more `ssdl:msgref` element information items with the **[normalised value]** of their `ref` attribute information item set to "out" and the **[normalised value]** of their `ref` attribute information item set to the qualified name of a fault message

An `in-out` element information item means that the first referred message CAN be received, and if that happens, the second referred message MUST be sent as a reply unless a fault is triggered in which case one of the referred fault messages MUST be sent instead.

The value of the WS-Addressing **[message id]** property of the first message MUST be set. The value of the WS-Addressing **[relationship]** property of the second message MUST be set to the value of the WS-Addressing **[message id]** property of the first message and the type of the **[relationship]** property MUST be set to *wsa:Reply*.

2.4. in-optional-out

```
<protocol>
  <mep:in-optional-out>
    <msgref direction="in" />
    <msgref direction="out" />
    <msgref direction="out" /> *
  </mep:in-optional-out>
</protocol>
```

The `in-optional-out` information item has the following properties:

- A **[local name]** of "in-optional-out"
- A **[namespace]** of "urn:ssdl:mep:v1"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "in"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "out"
- Zero or more `ssdl:msgref` element information items with the **[normalised value]** of their `ref` attribute information item set to "out" and the **[normalised value]** of their `ref` attribute information item set to the qualified name of a fault message

An `in-optional-out` element information item means that the first referred message CAN be received, and if that happens, the second referred message SHOULD be sent as a reply unless a fault is triggered in which case one of the referred fault messages SHOULD be sent instead.

The value of the WS-Addressing **[message id]** property of the first message MUST be set. If the second message is sent, the value of its WS-Addressing **[relationship]** property MUST be set to the value of the WS-Addressing **[message id]** property of the first message and the type of the **[relationship]** property MUST be set to *wsa:Reply*.

2.5. out-only

```
<protocol>
  <mep:out-only>
    <msgref direction="out" />
  </mep:out-only>
</protocol>
```

The `out-only` information item has the following properties:

- A **[local name]** of "out-only"
- A **[namespace]** of "urn:ssdl:mep:v1"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "out"

An `out-only` element information item means that the referred message CAN be sent.

2.6. robust-out-only

```
<protocol>
  <mep:robust-out-only>
    <msgref direction="out" />
    <msgref direction="in" /> +
  </mep:robust-out-only>
</protocol>
```

The `robust-out-only` information item has the following properties:

- A **[local name]** of "robust-out-only"
- A **[namespace]** of "urn:ssdl:mep:v1"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "out"
- One or more `ssdl:msgref` element information items with the **[normalised value]** of their `direction` attribute information item set to "in" and the **[normalised value]** of their `ref` attribute information item set to the qualified name of a fault message

A `robust-out-only` element information item means that the first referred message CAN be sent. If a fault is triggered, one of the referenced fault messages CAN be received.

2.7. out-in

```
<protocol>
  <mep:out-in>
    <msgref direction="out" />
    <msgref direction="in" />
    <msgref direction="in" /> +
  </mep:out-in>
</protocol>
```

The `out-in` information item has the following properties:

- A **[local name]** of "out-in"
- A **[namespace]** of "urn:ssdl:mep:v1"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "out"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "in"
- One or more `ssdl:msgref` element information items with the **[normalised value]** of their `direction` attribute information item set to "in" and the **[normalised value]** of their `ref` attribute information item set to the qualified name of a fault message

An `out-in` element information item means that the first referred message CAN be sent and if that happens, the second referred message MUST be received as a reply unless a fault is triggered in which case one of the following fault messages MUST be received instead.

The value of the WS-Addressing **[message id]** property of the first message MUST be set. The value of the WS-Addressing **[relationship]** property of the second message MUST be set to the value of the WS-Addressing **[message id]** property of the first message and the type of the **[relationship]** property MUST be set to *wsa:Reply*.

2.8. out-optional-in

```
<protocol>
  <mep:out-optional-in>
    <msgref direction="out" />
    <msgref direction="in" />
    <msgref direction="in" /> *
  </mep:out-optional-in>
</protocol>
```

The `out-optional-in` information item has the following properties:

- A **[local name]** of "out-optional-in"
- A **[namespace]** of "urn:ssdl:mep:v1"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "out"
- A REQUIRED `ssdl:msgref` element information item with the **[normalised value]** of its `direction` attribute information item set to "in"
- Zero or more `ssdl:msgref` element information items with the **[normalised value]** of their `direction` attribute information item set to "in" and the **[normalised value]** of their `ref` attribute information item set to the qualified name of a fault message

An `out-in` element information item means that the first referred message CAN be sent, and if that happens, the second referred message SHOULD be received as a reply unless a fault is triggered in which case one of the following fault messages SHOULD be received instead.

The value of the WS-Addressing **[message id]** property of the first message MUST be set. The value of the WS-Addressing **[relationship]** property of the second message MUST be set to the value of the WS-Addressing **[message id]** property of the first message and the type of the **[relationship]** property MUST be set to *wsa:Reply*.

References

[1] S. Parastatidis, J. Webber, S. Woodman, D. Kuo, and P. Greenfield, "An Introduction to the SOAP Service Description Language," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-898, 2005.

[2] S. Parastatidis, J. Webber, S. Woodman, D. Kuo, and P. Greenfield, "SOAP Service Description Language (SSDL)," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-899, 2005.

[3] S. Parastatidis and J. Webber, "MEP SSDL Protocol Framework," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-900, 2005.

[4] S. Parastatidis and J. Webber, "CSP SSDL Protocol Framework," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-901, 2005.

[5] D. Kuo, S. Parastatidis, and J. Webber, "Rules SSDL Protocol Framework," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-902, 2005.

[6] S. Woodman, S. Parastatidis, and J. Webber, "Sequencing Constraints SSDL Protocol Framework," School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-903, 2005.

[7] W3C, "Web Services Description Language (WSDL) Version 2.0 Part 2: Predefined Extensions." http://www.w3.org/TR/2004/WD-wsdl20-extensions-20040803/, 2004.

[8] S. Parastatidis and J. Webber, "SOAP Service Description Language," 2005.

[9] S. Bradner, "IETF RFC 2119: Key words for use in RFCs to Indicate Requirement Levels." http://www.ietf.org/rfc/rfc2119.txt: Internet Engineering Task Force, 1999.

[10] W3C, "XML Information Set." http://www.w3.org/TR/xml-infoset/, 2004.

[11] W3C, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," R. Chinnici, M. Gudgin, J.-J. Moreau, J. Schlimmer, and S. Weerawarana, Eds. http://www.w3.org/TR/2004/WD-wsdl20-20040803/, 2004.

Appendix A – XML Schema

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
    $Modtime: 5/03/05 14:56 $
    $Revision: 10 $
-->
<xs:schema
      elementFormDefault="qualified"
      targetNamespace="urn:ssdl:mep:v1"
      xmlns:ssdl="urn:ssdl:v1"
      xmlns:mep="urn:ssdl:mep:v1"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">

   <xs:import namespace="urn:ssdl:v1" />

   <xs:element name="in-only" type="mep:oneMsg-type"/>
   <xs:element name="robust-in-only" type="mep:manyMsgs-type"/>
   <xs:element name="in-out" type="mep:manyMsgs-type"/>
   <xs:element name="in-optional-out" type="mep:manyMsgs-type"/>
   <xs:element name="out-only" type="mep:oneMsg-type"/>
   <xs:element name="robust-out-only" type="mep:manyMsgs-type"/>
   <xs:element name="out-in" type="mep:manyMsgs-type"/>
   <xs:element name="out-optional-in" type="mep:manyMsgs-type"/>

   <xs:complexType name="oneMsg-type">
      <xs:sequence>
         <xs:element ref="ssdl:msgref" minOccurs="1" maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="manyMsgs-type">
      <xs:sequence>
         <xs:element ref="ssdl:msgref" minOccurs="2" maxOccurs="unbounded" />
      </xs:sequence>
   </xs:complexType>

</xs:schema>
```