The SOAP Service Description Language: A High Level Overview

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

Introduction

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.

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.

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.

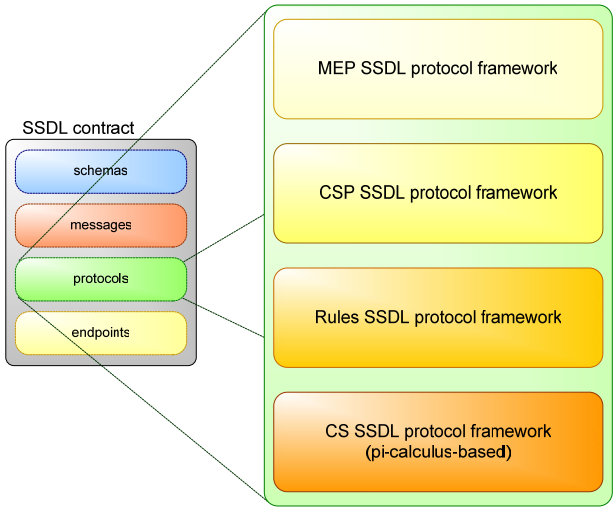

Figure 1 The SSDL Suite of Specifications

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

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.

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.

Current Status

SSDL v1.0 and v1.0 versions of four protocol frameworks have now been publicly released. In addition, the originators of the specifications have developed next-generation tool support

(currently available only on the .NET 2.0 beta platform) which enables SSDL contracts to be consumed and appropriate code to be generated which can be used for the implementation of Web Services or communication with remote Web Services. The tooling does not try to abstract Web Services as objects and message exchanges as method calls – the APIs and programming abstractions presented to developers expose programmatic representations of messages to the user code via events for incoming messages, and provides methods for the explicit sending of messages, which are represented as objects, for outgoing messages.

While these specifications have been reviewed in private by a number of distributed systems researchers and practitioners, SSDL has not yet been submitted for public review. With the public release of SSDL v1.0 we are explicitly soliciting feedback from the community on the strengths, weaknesses, and general utility of the approach.

Contact Details

SSDL was a collaborative effort by a number of researchers and practitioners worldwide. The central point of contact for the SSDL community is via the ssdl.org web site, at http://www.ssdl.org. For further details please contact Jim Webber (jim@webber.name) and Savas Parastatidis (savas@parastatidis.name).

People

The following people have contributed to the SSDL work:

- Simon Woodman (School of Computing Science, University of Newcastle upon Tyne, UK)
- Dean Kuo (CSIRO Information and Communication Technology (ICT) Centre, CSIRO, Australia)
- Paul Greenfield (CSIRO Information and Communication Technology (ICT) Centre, CSIRO, Australia)

Acknowledgements

The authors would like to thank Alan Fekete (School of Information Technologies, University of Sydney, Australia), Surya Nepal (CSIRO Information and Communication Technology Centre, CSIRO, Australia), and Paul Watson (School of Computing Science, University of Newcastle upon Tyne, UK) for their feedback on this work.