Initial report on emerging technology standards, including assessment and recommendations for support

Elmar Dorner
Christian Drumm
Joachim Quantz

June 28th, 2004
EXECUTIVE SUMMARY

This deliverable presents an overview on emerging technology standards in the area of Web Services. Grounded by the reflection of “what is a standard?” the document focuses on the technology layer of Web Services. The assortment and description of new technologies in this document orients itself at the goals of the DIP SWS Architecture. Within these boundaries a recommendation for support of standards is given. The document closes with a summary of the presented standards and proposed actions for the recommendations for the DIP project.

The deliverable contributes to two major planned DIP results: A Standard proposal and an Open Source SWS Architecture. For both of them the knowledge of current and emerging technology standards is a mandatory pre-requisite.
Document Information

<table>
<thead>
<tr>
<th>IST Project Number</th>
<th>FP6 – 507483</th>
<th>Acronym</th>
<th>DIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full title</td>
<td>Data, Information, and Process Integration with Semantic Web Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project URL</td>
<td><a href="http://dip.semanticweb.org">http://dip.semanticweb.org</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document URL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EU Project officer</td>
<td>Daniele Rizzi</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deliverable Number</th>
<th>7.1</th>
<th>Title</th>
<th>Initial report on emerging technology standards, including assessment and recommendations for support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work package Number</td>
<td>7</td>
<td>Title</td>
<td>Technology Watch and Standardization</td>
</tr>
</tbody>
</table>

| Date of delivery Contractual | M 6 | Actual | 30-June-04 |
| Status | Version 1.0 | final |
| Nature | Prototype ☑ Report ☑ Dissemination ☑ |
| Dissemination Level | Public ☑ Consortium ☑ |

Authors (Partner) | Elmar Dorner (SAP), Christian Drumm (SAP), Joachim Quantz (Berlecon) |
| Responsible Author | Dr. Elmar Dorner |
| Email | Elmar.dorner@sap.com |
| Partner | SAP |
| Phone | +49 721/6902-31 |

Abstract (for dissemination) | The deliverable contributes to two major planned DIP results: A Standard proposal and an Open Source SWS Architecture. For both of them the knowledge of current and emerging technology standards is a mandatory pre-requisite. |
|
| Keywords | Web Services, standardization, technical layer |
# Version Log

<table>
<thead>
<tr>
<th>Issue Date</th>
<th>Rev No.</th>
<th>Author</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>21-Apr-04</td>
<td>001</td>
<td>ED</td>
<td>Initial document</td>
</tr>
<tr>
<td>26-Apr-04</td>
<td>002</td>
<td>JQ</td>
<td>Included input from Berlecon</td>
</tr>
<tr>
<td>3-May-04</td>
<td>003</td>
<td>JQ, ED</td>
<td>New document structure based on Concall with WP7 partners</td>
</tr>
<tr>
<td>17-May-04</td>
<td>004</td>
<td>CD</td>
<td>Technical Web Service Standards, Messaging, Metadata, Reliable Messaging, Transactions</td>
</tr>
<tr>
<td>26-May-04</td>
<td>005</td>
<td>ED</td>
<td>Introduction to Standardization</td>
</tr>
<tr>
<td>8-Jun-04</td>
<td>006</td>
<td>CD, JQ, ED</td>
<td>Included input from Concall, more input, …</td>
</tr>
<tr>
<td>11-Jun-04</td>
<td>007</td>
<td>JQ, ED</td>
<td>Final text input, formatting, send to reviewers</td>
</tr>
<tr>
<td>21-Jun-04</td>
<td>008</td>
<td>JQ, ED</td>
<td>Revise document based on reviewers comments</td>
</tr>
<tr>
<td>23-Jun-04</td>
<td>009</td>
<td>WP7</td>
<td>Concall discussing final changes to deliverable</td>
</tr>
<tr>
<td>24-Jun-04</td>
<td>010</td>
<td>JQ, CD, ED</td>
<td>Final text changes</td>
</tr>
<tr>
<td>28-Jun-04</td>
<td>011</td>
<td>ED</td>
<td>Final formatting</td>
</tr>
</tbody>
</table>
# Project Consortium Information

<table>
<thead>
<tr>
<th>Partner</th>
<th>Acronym</th>
<th>Contact</th>
</tr>
</thead>
</table>
| National University of Ireland Galway       | NUIG    | Prof. Dr. Christoph Bussler  
Digital Enterprise Research Institute (DERI)  
National University of Ireland, Galway  
Galway, Ireland  
Email: chris.bussler@deri.ie  
Tel: +353 91 512460 |
| Fundacion De La Innovacion.Bankinter        | Bankinter | Jose Luis Bas  
Fundacion de la Innovation. Bankinter,  
Paseo Castellana, 29  
28046 Madrid, Spain  
Email: ilbas@bankinter.es  
Tel: 916234238 |
| Berlecon Research GmbH                      | Berlecon | Dr. Thorsten Wichmann  
Berlecon Research GmbH,  
Oranienburger Str. 32  
10117 Berlin, Germany  
Email: tw@berlecon.de  
Tel: +49 30 2852960 |
| British Telecommunications Plc.             | BT      | Dr John Davies  
BT Exact (Orion Floor 5 pp12)  
Adastral Park Martlesham,  
Ipswich IP5 3RE, United Kingdom  
Email: john.nj.davies@bt.com  
Tel: +44 1473 609583 |
| Swiss Federal Institute of Technology, Lausanne | EPFL    | Prof. Karl Aberer  
Distributed Information Systems Laboratory  
École Polytechnique Fédérale de Lausanne  
Bât. PSE-A  
1015 Lausanne, Switzerland  
Email: Karl_Aberer@epfl.ch  
Tel: +41 21 693 4679 |
| Essex County Council                         | Essex   | Mary Rowlett,  
Essex County Council,  
PO Box 11, County Hall, Duke Street,  
Chelmsford, Essex, CM1 1LX, United Kingdom.  
Email: maryr@essexcc.gov.uk  
Tel: +44 (0)1245 436524 |
| Forschungszentrum Informatik                | FZI     | Andreas Abecker  
Forschungszentrum Informatik  
Haid-und-Neu Strasse 10-14  
76131 Karlsruhe, Germany  
Email: abecker@fzi.de  
Tel: +49 721 9654 0 |
<table>
<thead>
<tr>
<th>Organization</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFI</td>
<td>Prof. Dieter Fensel</td>
</tr>
<tr>
<td></td>
<td>Institute of computer science</td>
</tr>
<tr>
<td></td>
<td>University of Innsbruck</td>
</tr>
<tr>
<td></td>
<td>Technikerstr. 25</td>
</tr>
<tr>
<td></td>
<td>A-6020 Innsbruck, Austria</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:dieter.fensel@uibk.ac.at">dieter.fensel@uibk.ac.at</a></td>
</tr>
<tr>
<td></td>
<td>Tel: +43 512 5076485</td>
</tr>
<tr>
<td>ILOG SA</td>
<td>Christian de Sainte Marie</td>
</tr>
<tr>
<td></td>
<td>9 Rue de Verdun, 94253</td>
</tr>
<tr>
<td></td>
<td>Gentilly, France</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:csma@ilog.fr">csma@ilog.fr</a></td>
</tr>
<tr>
<td></td>
<td>Tel: +33 1 49082981</td>
</tr>
<tr>
<td>Inubit AG</td>
<td>Torsten Schmale,</td>
</tr>
<tr>
<td></td>
<td>inubit AG</td>
</tr>
<tr>
<td></td>
<td>Lützowstraße 105-106</td>
</tr>
<tr>
<td></td>
<td>D-10785 Berlin, Germany</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:ts@inubit.com">ts@inubit.com</a></td>
</tr>
<tr>
<td></td>
<td>Tel: +49 30726112 0</td>
</tr>
<tr>
<td>iSOCO</td>
<td>Dr. V. Richard Benjamins, Director R&amp;D</td>
</tr>
<tr>
<td></td>
<td>Intelligent Software Components, S.A.</td>
</tr>
<tr>
<td></td>
<td>Pedro de Valdivia 10</td>
</tr>
<tr>
<td></td>
<td>28006 Madrid, Spain</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:rbenjamins@isoco.com">rbenjamins@isoco.com</a></td>
</tr>
<tr>
<td></td>
<td>Tel. +34 913 349 797</td>
</tr>
<tr>
<td>Net Dynamics</td>
<td>Peter Smolle</td>
</tr>
<tr>
<td></td>
<td>Net Dynamics Internet Technologies GmbH &amp;.</td>
</tr>
<tr>
<td></td>
<td>Co KG</td>
</tr>
<tr>
<td></td>
<td>Prinz-Eugen-Strasse 68-70</td>
</tr>
<tr>
<td></td>
<td>A-1040 Wien, Austria</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:peter.smolle@netdynamics-tech.com">peter.smolle@netdynamics-tech.com</a></td>
</tr>
<tr>
<td></td>
<td>Tel.: +43 1 503982615</td>
</tr>
<tr>
<td>OU</td>
<td>Dr. John Domingue</td>
</tr>
<tr>
<td></td>
<td>Knowledge Media Institute,</td>
</tr>
<tr>
<td></td>
<td>The Open University, Walton Hall, Milton Keynes, MK7 6AA,</td>
</tr>
<tr>
<td></td>
<td>United Kingdom</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:j.b.domingue@open.ac.uk">j.b.domingue@open.ac.uk</a></td>
</tr>
<tr>
<td></td>
<td>Tel.: +44 1908 655014</td>
</tr>
<tr>
<td>SAP AG</td>
<td>Dr. Elmar Dorner</td>
</tr>
<tr>
<td></td>
<td>SAP Research, CEC Karlsruhe</td>
</tr>
<tr>
<td></td>
<td>SAP AG</td>
</tr>
<tr>
<td></td>
<td>Vincenz-Priessnitz-Str. 1</td>
</tr>
<tr>
<td></td>
<td>76131 Karlsruhe, Germany</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:elmar.dorner@sap.com">elmar.dorner@sap.com</a></td>
</tr>
<tr>
<td></td>
<td>Tel: +49 721 6902 31</td>
</tr>
<tr>
<td>Sirma AI Ltd.</td>
<td>Atanas Kiryakov,</td>
</tr>
<tr>
<td></td>
<td>Ontotext Lab, - Sirma AI EAD,</td>
</tr>
<tr>
<td></td>
<td>Office Express IT Centre, 3rd Floor</td>
</tr>
<tr>
<td></td>
<td>135 Tzarigradsko Chausse,</td>
</tr>
<tr>
<td></td>
<td>Sofia 1784, Bulgaria</td>
</tr>
<tr>
<td></td>
<td>Email: <a href="mailto:atanas.kiryakov@sirma.bg">atanas.kiryakov@sirma.bg</a></td>
</tr>
<tr>
<td></td>
<td>Tel.: +359 2 9768 303</td>
</tr>
<tr>
<td>Organization</td>
<td>Name</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Tiscali Österreich GmbH</td>
<td>Dieter Haacker</td>
</tr>
<tr>
<td>Unicorn Solution Ltd.</td>
<td>Jeff Eisenberg</td>
</tr>
<tr>
<td>Vrije Universiteit Brussel</td>
<td>Carlo Wouters,</td>
</tr>
</tbody>
</table>

Initial report on emerging technology standards, including assessment and recommendations for support
# Table of Contents

## Executive Summary ........................................................................................................1

## Table of Contents ............................................................................................................ VII

## Table of Figures ................................................................................................................. VIII

## 1 Introduction .................................................................................................................... 1

## 2 Established Standards .................................................................................................... 6
  2.1 SOAP .................................................................................................................. 6
  2.2 WSDL ............................................................................................................... 7
  2.3 UDDI .................................................................................................................. 8

## 3 Messaging ...................................................................................................................... 9
  3.1 WS-Addressing ................................................................................................. 9
  3.2 WS-Eventing ..................................................................................................... 10
  3.3 WS-ReliableMessaging ..................................................................................... 11

## 4 Metadata ....................................................................................................................... 12
  4.1 WS-Policy ......................................................................................................... 12
  4.2 WS-Discovery ................................................................................................. 13
  4.3 WS-MetadataExchange ..................................................................................... 14

## 5 Security .......................................................................................................................... 15
  5.1 WS-Security ...................................................................................................... 16
  5.2 WS-Trust .......................................................................................................... 17
  5.3 WS-SecureConversation ................................................................................... 18
  5.4 WS-Federation .................................................................................................. 18

## 6 Transactions .................................................................................................................. 20
  6.1 Business Transaction Protocol ........................................................................... 21
  6.2 WS-Coordination .............................................................................................. 22
  6.3 WS-AtomicTransaction .................................................................................... 22
  6.4 WS-BusinessActivity ........................................................................................ 23

## 7 Choreography/orchestration ......................................................................................... 24
  7.1 BPEL4WS ........................................................................................................... 24
  7.2 WS-CDL ............................................................................................................ 26

## 8 Management .................................................................................................................. 28
  8.1 Web Services Distributed Management ............................................................. 28
Initial report on emerging technology standards, including assessment and recommendations for support

8.2 WSMF .......................................................................................................................... 29
8.3 WS-Manageability ........................................................................................................ 30
9 RECOMMENDATIONS FOR DIP .................................................................................... 31

ACRONYMS......................................................................................................................... 33
REFERENCES .................................................................................................................... 34

TABLE OF FIGURES
Figure 1: Standardization Stack...................................................................................... 1
Figure 2: View on standards ............................................................................................. 2
Figure 3: Standardization Organizations ......................................................................... 3
Figure 4: Web Services components and operations ......................................................... 4
Figure 5: Web Service technology stack .......................................................................... 4
Figure 6: IBM/Microsoft Web Services security model ..................................................... 15
Figure 7: Summary of Web Service Protocols adoption ................................................... 31
Figure 8: Summary of recommendations for DIP ............................................................. 32
1 INTRODUCTION

James Maguire stated in an article in the NewsFactor Network: …But the sheer volume of standards work also serves to impede Web Services adoption. "This is the first time in 15 years that anyone had to think about a protocol other than TCP/IP. Now all of sudden they give us 22 different protocols, standards and specifications." [1]

Figure 1 shows on the left side a simplified three layer classification of the huge number of so called “standards” in the area of Web Services. Within this document we focus on the Technical Standards layer. A more detailed view of these standards is shown on the right side of the Figure. It depicts a common categorization that distinguishes several working areas, e.g. Management, Security, and so on. In addition, fine-grained topics in these areas are shown [2, 3]. Emerging higher layer standards (Framework Standards and Business Standards) that could be of relevance for the DIP project are discussed in an other corresponding DIP deliverable (D7.2 “Report on relevant E-Business/E-Commerce formats/standards including assessment and requirements for mapping/mediation”, available at milestone M6 of the DIP project).

The term “standard” is not only used for real standards, which are supposed to be specified by governmental bodies. It is used in a broader sense as a guideline documentation that reflects agreements on practices, or operations by nationally or internationally recognized industrial, professional, trade associations or governmental bodies.

Whether or not a standard is really a standard can be answered from at least two perspectives. The one is whether or not it is relevant or already adopted in the industry. The other is whether it has been accepted by a formal standards organization. An evaluation for some well-known technical standards is given by the following table.

![Standardization Stack Diagram](image-url)
It is important to note that many technical standards are not developed or ratified by a formal, internationally accredited standards organization (“de jure”), but rather by industry consortia or even by individual groups of vendors (“de facto” or “industry consortium”). Another aspect of technical standards is their relevance in “real-life”. There are cases where standards are ratified by a formal standards organization, but lack actual support in the industry. Others are in their “emerging” phase or reached a “mainstream” status. Figure 2 summarizes the above said and classifies accordingly some of the actual activities.

<table>
<thead>
<tr>
<th>Level of Standardization</th>
<th>Relevance</th>
<th>mainstream</th>
<th>emerging</th>
<th>less relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>de facto</td>
<td>SOAP 1.1, WSDL 1.1</td>
<td>WS-Addressing, WS-Policy, WS-ReliableMessaging</td>
<td>WSCL (HP), WS-Inspection</td>
<td></td>
</tr>
<tr>
<td>industry consortium</td>
<td>HTTP, XML 1.0, Basic Profile 1.0, UDDI 2.0, WS-Security, WS-Metadata-Exchange</td>
<td>SOAP 1.2, UDDI 3.0, WSBPEL, WSDL 2.0, XML 1.1, WS-Eventing, WS-Discovery</td>
<td>ebXML Infrastructure, WS-Reliability</td>
<td></td>
</tr>
<tr>
<td>de jure</td>
<td>-</td>
<td>CCTS (UN/CEFACT), XML Naming and Design Rules (UN/CEFACT)</td>
<td>SGML (ISO), EDIFACT-XSD mapping (ISO)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: View on standards

An overview of "Standardization Organizations” can be found in Figure 3. Again, the only one that deserves this term is in fact “UN/CEFACT”. All others are building around industry consortia.

<table>
<thead>
<tr>
<th>Name</th>
<th>Focus</th>
<th>Founded</th>
<th>Membership</th>
<th>Important Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPMI</td>
<td>Business Process Management</td>
<td>2000</td>
<td>50+</td>
<td>BPML, BPMN</td>
</tr>
<tr>
<td>DMTF</td>
<td>System Management</td>
<td>1992</td>
<td>150+</td>
<td>CIM, WBEM</td>
</tr>
<tr>
<td>IETF</td>
<td>Core Internet Protocols</td>
<td>1986</td>
<td>unknown</td>
<td>HTTP, SSL, MIME</td>
</tr>
<tr>
<td>Organization</td>
<td>Focus Area</td>
<td>Year</td>
<td>Members</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------------</td>
<td>------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>JCP</td>
<td>Java Technologies</td>
<td>1995</td>
<td>650+</td>
<td></td>
</tr>
<tr>
<td>Liberty Alliance</td>
<td>Identity Management</td>
<td>2001</td>
<td>150+</td>
<td></td>
</tr>
<tr>
<td>OAGi</td>
<td>ERP Business Standards</td>
<td>1995</td>
<td>40+</td>
<td></td>
</tr>
<tr>
<td>OASIS</td>
<td>Web Service Protocols and e-Business Technologies</td>
<td>1993</td>
<td>600+</td>
<td></td>
</tr>
<tr>
<td>OMG</td>
<td>Modeling Notation and Methodology</td>
<td>1989</td>
<td>800+</td>
<td></td>
</tr>
<tr>
<td>UN/CEFACT</td>
<td>Core Business Standards</td>
<td>1996</td>
<td>1500+ individuals</td>
<td></td>
</tr>
<tr>
<td>W3C</td>
<td>Core XML and Internet Protocols</td>
<td>1994</td>
<td>400+</td>
<td></td>
</tr>
<tr>
<td>WS-I</td>
<td>Web Service Protocol Profiling</td>
<td>2002</td>
<td>150+</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: Standardization Organizations

The remainder of the document is structured as follows:

Chapter two of this document reveals the basic components and operations required in Service Oriented Architectures. XML Schema and WSDL (Web Services Description Language) are formal languages for the description of Web Services, SOAP is used to invoke Web Services, and UDDI (Universal Description, Discovery and Integration) is used to publish or to find a Web Service description by means of a registry (Figure 4). These have effectively become de facto standards, with effectively universal acceptance and widespread implementation by vendors.
On the other hand, this very simplistic approach can't easily be extended to handle new standards, e.g. for security, choreography, management, and so on. In order to show the big picture of the Web Services architecture as it currently evolves, the picture needs to be somewhat more complex (Figure 5) [4]. In the following chapters we describe for each of these areas the current trends, who is working on which topics, and the relevance for the DIP project.

Figure 4: Web Services components and operations

Figure 5: Web Service technology stack
The last chapter of the document summarizes the presented emerging technologies. It gives recommendations, which of them are of interest for usage in the DIP project. Furthermore it categorizes the actions to be taken for each of the recommended standards/work in progress. Clearly this is a process that not ends with this deliverable. Therefore, another deliverable (D7.3) in the DIP project describes a standardization strategy and forums to be targeted from within the DIP project. This ensures a continuous monitoring and updated recommendations during the whole DIP project life-span.

---

1 The deliverable D7.3 “Standardization strategy and forums to be targeted” is available at milestone M6 and will be updated for M18 of the DIP project
2 ESTABLISHED STANDARDS

2.1 SOAP

Purpose:
SOAP\(^2\) Version 1.2 provides the definition of the XML-based information which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment.

Players involved:
SOAP was originally developed by Ariba, IBM, and Microsoft and is now developed by the XML Protocol Working Group at W3C.

Brief Description:
Part 1 of SOAP Version 1.2 defines the SOAP envelope, which is a construct that defines an overall framework for representing the contents of a SOAP message, identifying who should deal with all or part of it, and whether handling such parts are optional or mandatory. It also defines a protocol binding framework, which describes how the specification for a binding of SOAP onto another underlying protocol may be written.

Part 2 of SOAP Version 1.2 defines a data model for SOAP, a particular encoding scheme for data types which may be used for conveying remote procedure calls (RPC), as well as one concrete realization of the underlying protocol binding framework defined in Part 1. This binding allows the exchange of SOAP messages either as payload of a HTTP POST request and response, or as a SOAP message in the response to a HTTP GET.

SOAP is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information. SOAP is silent on the semantics of any application-specific data it conveys, as it is on issues such as the routing of SOAP messages, reliable data transfer, firewall traversal, etc. However, SOAP provides the framework by which application-specific information may be conveyed in an extensible manner. Also, SOAP provides a full description of the required actions taken by a SOAP node on receiving a SOAP message. [5]

Status:
The W3C has released SOAP Version 1.2 as a recommendation in June 2003. SOAP is supported by all major software vendors offering Web Services functionality. The WS-I Basic Profile 1.0 contains further restrictions to SOAP to ensure interoperability.

\(^2\) Originally, SOAP was used as an acronym for Simple Object Acces Protocoll. This acronym reading was explicitly dismissed, however, when SOAP 1.1 was released as a W3C standard recommendation.
**DIP Recommendation:**

As SOAP is an established and widely used standard it is strongly recommended to use it for message exchange in DIP. WSDL descriptions of DIP component interfaces should thus contain a SOAP binding. It is also recommended to use SOAP in conformance with the WS-I Profile 1.0.

---

2.2 WSDL

**Purpose:**

The Web Service Description Language (WSDL) [41] provides a model for describing Web Services interfaces via a “programming-language independent API”.

**Players involved:**

WSDL was originally developed by BEA, IBM, and Microsoft and is now developed by W3C.

**Brief Description:**

In WSDL a Web Service is described on two levels: the abstract level and the concrete level.

At an abstract level, WSDL describes a Web Service in terms of the messages it sends and receives. Messages are described independent of a specific wire format, such as SOAP, by using a type system, typically XML Schema.

An operation associates a message exchange pattern with one or more messages. A message exchange pattern identifies the sequence and cardinality of messages sent and/or received as well as whom they are logically sent to and/or received from. An interface groups together operations without any commitment to transport or wire format.

At a concrete level, a binding specifies transport and wire format details for one or more interfaces. An endpoint associates a network address with a binding. And finally, a service groups together endpoints that implement a common interface.

**Status:**

WSDL is currently available in Version 1.1 as a W3C recommendation. Currently the W3C works on version 2.0, which will most likely be the next recommendation. A working draft of version 2.0 can be found on the W3C web site. WSDL is supported by all major software vendors offering Web Services functionality. The WS-I Basic Profile 1.0 contains further restrictions to WSDL to ensure interoperability.

**DIP Recommendation:**

As WSDL is an established and widely accepted standard it is strongly recommended to use it for describing Web Services interfaces of DIP components. It is also recommended to use WSDL in conformance with the WS-I Profile 1.0.
2.3 UDDI

Purpose:
The Universal Description, Discovery, and Integration (UDDI) project "creates a platform-independent, open framework for describing services, discovering businesses, and integrating business services using the Internet, as well as an operational registry that is available today.

Players involved:
UDDI is developed by the OASIS Technical Committee UDDI, which comprises, amongst others, the following companies as members: Cisco, Fujitsu, IBM, Intel, Microsoft, OpenStorm, SAP, and Sun.

Brief Description:
The Universal Description, Discovery and Integration (UDDI) protocol is one of the major building blocks required for successful Web services. UDDI creates a standard interoperable platform that enables companies and applications to quickly, easily, and dynamically find and use Web services over the Internet. UDDI also allows operational registries to be maintained for different purposes in different contexts. UDDI is a cross-industry effort driven by major platform and software providers, as well as marketplace operators and e-business leaders within the OASIS standards consortium.

UDDI version 3 is a “meta service” for locating web services by enabling robust queries against rich metadata. Expanding on the foundation of versions 1 and 2 of UDDI, version 3 offers the industry a specification for building flexible, interoperable XML Web services registries useful in private as well as public deployments. With a vast array of enhancements – including multi-registry topologies, increased security features, improved WSDL support, a new subscription API and core information model advances – the version 3 specification offers clients and implementers a comprehensive and complete blueprint of a description and discovery foundation for a diverse set of web service architectures. [8]

Status:
UDDI Version 3.0 has been released by the OASIS TC UDDI.org in July 2002.

DIP Recommendation:
UDDI is the established standard for Web Services registries. It is strongly recommended to base development of registry functionality in DIP on UDDI3.

3 DIP Deliverable D4.2 (Publishing) describes the usage of UDDI as basis for the DIP Registry.
3 MESSAGING

The current de-facto standards for Web Services, as they are described in Figure 4, lack certain features (e.g. transport independent addressing, reliable messaging) in the area of message delivery and exchange. To overcome these shortcomings three new standards, namely WS-Addressing, WS-Eventing and WS-ReliableMessaging, are proposed by different consortia.

The WS-Addressing specification aims to provide a transport neutral mechanism for the addressing of endpoints to enable a transport protocol independent implementation of Web Services. Building on that specification WS-Eventing defines a protocol to enable publish/subscribe scenarios using Web Services. Finally WS-ReliableMessaging defines a protocol to enable reliable message delivery even when an unreliable transport protocol is used.

3.1 WS-Addressing

Purpose:
WS-Addressing defines a transport protocol independent addressing mechanisms for: Message header elements and References of endpoints.

Players involved:
The Web Service Addressing was developed by BEA, IBM, Microsoft, and Tibco.

Brief Description:
To achieve a transport neutral mechanism to address Web Services the WS-Addressing specification defines two constructs, Endpoint References and Message Information Headers, which contain information normally provided by a specific transport mechanism or messaging systems. By using these constructs, the information is now provided in a uniformed way and can be processed independently of transport or application.

Endpoint References are intended to support dynamic scenarios that are currently not appropriately supported by WSDL 1.1. In particular these scenarios are:

- Dynamic generation of endpoint references
- Identification of a specific service instance
- Flexible and dynamic exchange of endpoint information.

Message Information Headers augment a message with a number of abstract properties that enable the identification and location of the endpoints involved in the interaction.

WS-Addressing is important to enable certain scenarios, e.g. pub/sub scenarios, mediation, etc. [9]

Status:
Currently the WS-Addressing exists only as a specification, which is not yet submitted to any standardization body. It is provided for free from the authors.
WS-Addressing is re-used in several other specifications, e.g. WS-Policy, WS-ReliableMessaging. By this, it is much likely that WS-Addressing will become a de-facto standard in the future.

**DIP Recommendation:**

Further monitoring is advisable as the specification currently evolves. It may gain in relevance if other specifications (WS-Policy, WS-ReliableMessaging) in the area of Web Services mature as they already stated to use WS-Addressing.

### 3.2 WS-Eventing

**Purpose:**
The WS-Eventing standard describes a protocol to enable Web Services to subscribe or to accept subscription to event notifications.

**Players involved:**
WS-Eventing is developed by BEA, Microsoft, and Tibco.

**Brief Description:**
Web Services often want to receive messages when events occur in other services and applications. A mechanism for registering interest is needed because the set of Web Services interested in receiving such messages is often unknown in advance or will change over time. This specification defines a protocol for one Web Service (called an "event sink") to register interest (called a "subscription") with another Web Service (called an "event source") in receiving messages about events (called "notifications"). To improve robustness, the subscription is leased by an event source to an event sink, and the subscription expires over time. An event source may allow an event sink to renew the subscription.

WS-Eventing utilizes capabilities of published specifications such as WS-Addressing (for the exchange of endpoint references), WS-Security, and WS-ReliableMessaging. It works both with SOAP 1.1 and SOAP 1.2 message envelopes. [10, 11]

**Status:**
Version 1.0 was released in January 2004. The companies plan to submit the specification for consideration by an industry standards body such as W3C or OASIS. This submission will happen after seeking feedback and testing for interoperability.

As a counterpart of WS-Eventing an industry consortium with HP, IBM, and others announced the creation of a proposed Web Services standard called WS-Notification, which allows messaging systems to generate notifications of Web Services “events”. There are also plans to submit the specification to OASIS for standardization purposes. [12, 13]

**DIP Recommendation:**

A future fully-fledged Web Service infrastructure will definitely support mechanisms for sending and processing events. Currently it is not clear whether WS-Eventing or WS-Notification will gain more acceptances. It is not an urgent issue for DIP to decide for one of both at the moment. Further monitoring with low effort is advisable.
3.3 WS-ReliableMessaging

**Purpose:**
The specification describes a protocol that allows messages to be delivered reliably between distributed applications in the presence of software component, system, or network failures.

**Players involved:**
WS-ReliableMessaging is a specification by IBM, BEA, Microsoft and Tibco.

**Brief Description:**
WS-ReliableMessaging may not be mistaken to define a message format. It rather enhance messaging by reliability in the sense that it provides tags that can be populated by message senders to state that one or several messages should be acknowledged, or that messages should be delivered in a fixed order, or that duplicates must be eliminated. [6]

**Status:**
The first version was published in early 2003. Since March 2004 an updated version is made available. WS-ReliableMessaging is competing specification to WS-Reliability being finalized by OASIS WS-Reliable Messaging TC.

**DIP Recommendation:**
The fact that there are two competing proposals accentuates the necessity of a mechanism to exchange messages in a reliable manner within a Web Service Infrastructure. In terms of the DIP infrastructure development this functionality will be provided by the supporting framework. Therefore there is currently no need for the DIP project to favor or decide for one of the both. Later in the project as more scenarios arise within the context of the DIP infrastructure it may be the case that a more specific need for reliable messaging grows.
4 Metadata

To enable interoperability between different Web Services in a large, highly dynamic environment like the internet, Web Services need to be able to exchange different kinds of metadata, like e.g. the interface description in WSDL or information about the capabilities of a given service. Currently no standard mechanisms for exchanging this data are provided by the de-facto standards.

The specifications that will be described in this section provide functionality to exchange information on requirements and capabilities (WS-Policy), a protocol to retrieve metadata from a Web Service and a mechanism to enable a-hoc discovery and advertisement (WS-Discovery).

4.1 WS-Policy

Purpose:
WS-Policy will describe how senders and receivers can specify their requirements and capabilities.

Players involved:
The Web Services Policy is an initiative of several companies: BEA Systems, IBM, Microsoft, and SAP AG

Brief Description:
*WS-Policy* is a set of related specifications that together enable the seamless description and communication of Web Service policies:

- Web Services Policy Framework (WS-Policy) provides a general-purpose model and corresponding syntax to describe and communicate the policies of a Web Service. WS-Policy defines a base set of constructs that can be used and extended by other Web Services specifications to describe a broad range of service requirements, preferences, and capabilities.

- Web Services Policy Assertions Language (WS-PolicyAssertions) specifies a set of common message policy assertions that can be used within a policy expression. The specification defines general messaging-related assertions for use within the WS-Policy Framework.

- Web Services Policy Attachment (WS-PolicyAttachment) specifies three specific attachment mechanisms for using policy expressions with existing XML Web Service technologies. Specifically, it defines how to associate policy expressions with WSDL type definitions and UDDI entities. It also defines how to associate implementation-specific policy with all or part of a WSDL portType when exposed from a specific implementation.

Web Services Policy fits into the core Web Service architecture since it is built on top of XML, XML Schema, WSDL, and UDDI. [14, 15]
Status:

Web Services Policy Version 1.1 as of 28 May 2003 supersedes the former Version 1.0, which was published 18 December 2002. The Web Service Policy 1.1 authors plan to submit the specifications to a standards organization in the near future.

DIP Recommendation:

WS-Policy is a framework and provides only a small number of general-purpose assertions itself. Thus, the success of WS-Policy highly depends on its usage in concrete domains such as security and reliable messaging. WS-Policy also "competes" to some extent with the features and properties approach introduced by SOAP 1.2 and adopted by WSDL 2.0. Therefore ongoing monitoring of the development of WS-Policy and the related proposals is necessary for DIP.

4.2 WS-Discovery

Purpose:

The WS-Discovery specification provides an ad hoc mechanism for services to advertise themselves and for service consumers to track them down.

Players involved:

WS-Discovery is developed by Microsoft, with the assistance of BEA Systems, Canon, and Intel.

Brief Description:

This specification defines a multicast discovery protocol to locate services. The primary mode of discovery is a client searching for one or more target services. To find a target service by the type of the target service, a scope in which the target service resides, or both, a client sends a probe message to a multicast group; target services that match the probe send a response directly to the client. To locate a target service by name, a client sends a resolution request message to the same multicast group, and again, the target service that matches sends a response directly to the client. [16]

To minimize the need for polling, when a target service joins the network, it sends an announcement message to the same multicast group. By listening to this multicast group, clients can detect newly-available target services without repeated probing.

The WS-Discovery specification acts somewhat like the UDDI (Universal Description, Discovery and Integration) standard, except it will focus on devices and systems that are not always connected to the network. WS-Discovery is complementary to UDDI and will work with other protocols, such as WS-Eventing, WS-Addressing, WS-Security and WS-ReliableMessaging.

Status:

The current version dated February 2004 of WS-Discovery is provided for use as-is and for review and evaluation only.
DIP Recommendation:

The specification is in a very early stage. The network optimization based on multicast communication seems promising. Future versions may evolve and be beneficial for the DIP architecture. Monitoring of the working group is advisable.

4.3 WS-MetadataExchange

Purpose:
The WS-MetadataExchange specification defines a "bootstrap mechanism" for exchanging messages based on metadata. It describes a mechanism that can be used in order to discover WS-Policy, WSDL and Schema information about a URI addressable end-point.

Players involved:
The Web Services Metadata Exchange is an initiative of BEA Systems, IBM, Microsoft, and SAP AG.

Brief Description:

Web Services use metadata to describe what other endpoints need to know to interact with them. Specifically, WS-Policy describes the capabilities, requirements, and general characteristics of Web Services; WSDL describes abstract message operations, concrete network protocols, and endpoint addresses used by Web Services; XML Schema describes the structure and contents of XML-based messages received and sent by Web Services.

To bootstrap communication with a Web Service, this specification defines three request-response message pairs to retrieve these three types of metadata: one retrieves the WS-Policy associated with the receiving endpoint or with a given target namespace, another retrieves either the WSDL associated with the receiving endpoint or with a given target namespace, and a third retrieves the XML Schema with a given target namespace. Together these messages allow efficient, incremental retrieval of a Web Service's metadata. [17, 18]

Status:
The specification dated March 2004 is an initial public draft release and is provided for review and evaluation only.

DIP Recommendation:

Currently for the DIP architecture a repository is planned. This component would already provide most of the functionality that a WS-MetadataExchange protocol is supposed to do. Therefore this specification is of minor interest for the DIP project.
5 SECURITY

IBM, Microsoft and other major software companies have defined a Web Services security model that guarantees end-to-end communication security. Figure 6 illustrates the building blocks and the dependencies of the model.

![Diagram of IBM/Microsoft Web Services security model](image)

**Figure 6: IBM/Microsoft Web Services security model**

The architecture uses the existing W3C XML standard recommendations XML-Encryption [19], XML-KeyManagementSystem [20], and XML-Signature [21] as a basis for a family of Web Service security standards. This section will briefly present the following standards:

- WS-Security
- WS-Trust
- WS-SecureConversation
- WS-Federation

The WS-Policy standard has been discussed in the section on meta-data standards above and the WS-Authorization standard has not yet been specified.
5.1 WS-Security

**Purpose:**
WS-Security proposes a standard set of SOAP extensions that can be used when building secure Web Services to implement integrity and confidentiality. WS-Security provides three main mechanisms: security token propagation, message integrity, and message confidentiality.

**Players involved:**
WS-Security has been originally developed by IBM, Microsoft, and VeriSign. It is now being developed by the OASIS Technological Committee Web Services Security.

**Brief Description:**
WS-Security proposes a standard set of SOAP extensions that can be used when building secure Web Services to implement integrity and confidentiality. WS-Security is flexible and is designed to be used as the basis for the construction of a wide variety of security models including PKI, Kerberos, and SSL. Specifically WS-Security provides support for multiple security tokens, multiple trust domains, multiple signature formats, and multiple encryption technologies.

This specification provides three main mechanisms: security token propagation, message integrity, and message confidentiality. These mechanisms by themselves do not provide a complete security solution. Instead, WS-Security is a building block that can be used in conjunction with other Web Service extensions and higher-level application-specific protocols to accommodate a wide variety of security models and encryption technologies.

These mechanisms can be used independently (e.g., to pass a security token) or in a tightly integrated manner (e.g., signing and encrypting a message and providing a security token hierarchy associated with the keys used for signing and encryption). [23, 24]

**Status:**
Version 1.0 of the standard has been released in April 2002. Development is now continued at the OASIS TC Web Services security. Currently, April 2004, “OASIS Web Service Security 1.0” is available. It comprises several parts: “SOAP Message Security V1.0”, “Username Token Profile V1.0”, “X.509 Token Profile V1.0”, and “Schema files V1.0” [22]. Due to the wide-ranging support for the standard it is expected to be taken up by most major software vendors within the next years.

**DIP Recommendation:**
As security is not a central issue for DIP, WS-Security is not directly relevant for DIP. However, compatibility of DIP solutions with WS-Security should be ensured.
5.2 WS-Trust

**Purpose:**
WS-Trust defines extensions building on WS-Security to provide a framework for requesting and issuing security tokens, and to broker trust relationships.

**Players involved:**

**Brief Description:**
WS-Trust defines the basic mechanisms for providing secure messaging. This specification uses these basic mechanisms and defines additional primitives and extensions for security token exchange to enable the issuance and dissemination of credentials within different trust domains. In order to secure a communication between two parties, the two parties must exchange security credentials (either directly or indirectly). However, each party needs to determine if they can "trust" the asserted credentials of the other party. In the WS-Trust specification extensions to WS-Security are defined providing:

- methods for issuing and exchanging security tokens,
- ways to establish and access the presence of trust relationships

Using these extensions, applications can engage in secure communication designed to work with the general Web Services framework, including WSDL service descriptions, UDDI businessServices and bindingTemplates, and SOAP messages. [42]

**Status:**
WS-Trust Version 1.1 was released in May 2004. This specification is supported by Microsoft’s WSE 2.0 (Web Services Enhancement), an add-on to Visual Studio .NET. [25]

**DIP Recommendation:**
Since Security is not a primary topic in DIP and WS-Trust is not yet widely supported by products or used in applications, its relevance to DIP is limited.
5.3 WS-SecureConversation

Purpose:
The Web Services Secure Conversation Language (WS-SecureConversation) is built on top of the WS-Security and WS-Trust models to provide secure communication between services.

Players involved:

Brief Description:
The Web Services Secure Conversation Language (WS-SecureConversation) is built on top of the WS-Security and WS-Policy models to provide secure communication between services. WS-Security focuses on the message authentication model but not a security context, and thus is subject several forms of security attacks. This specification defines mechanisms for establishing and sharing security contexts, and deriving keys from security contexts, to enable a secure conversation.

By using the SOAP extensibility model, modular SOAP-based specifications are designed to be composed with each other to provide a rich messaging environment. As such, WS-SecureConversation by itself does not provide a complete security solution. WS-SecureConversation is a building block that is used in conjunction with other Web service and application-specific protocols (for example, WS-Security) to accommodate a wide variety of security models and technologies. [26]

Status:
An updated specification of WS-SecureConversation was released in May 2004. This specification is supported by Microsoft’s WSE 2.0 (Web Services Enhancement), an add-on to Visual Studio .NET. [25].

DIP Recommendation:
Since Security is not a primary topic in DIP and WS-SecureConversation is not yet widely supported by products or used in applications, its relevance to DIP is limited.

5.4 WS-Federation

Purpose:
WS-Federation defines mechanisms that are used to enable identity, account, attribute, authentication, and authorization federation across different trust realms.

Players involved:
WS-Federation was developed by BEA, IBM, Microsoft, RSA Security and VeriSign.
**Brief Description:**

WS-Federation defines mechanisms to allow different security realms to federate using different or like mechanisms by allowing and brokering trust of identities, attributes, authentication between participating Web services.

The mechanisms defined in WS-Federation can be used by passive and active requestors. The Web service requestors are assumed to understand the new security mechanisms and be capable of interacting with Web service providers.

WS-Federation defines the model and framework for federation; subsequent documents define profiles which detail how different requestors apply this model. [27]

**Status:**


**DIP Recommendation:**

Since Security is not a primary topic in DIP and WS-Federation is not yet widely supported by products or used in applications, its relevance to DIP is limited.
6 TRANSACTIONS

Essentially, a transaction is a set of service operations each of which should be executed only if all other operations belonging to the same transaction are also executed. Classical examples are modifications in a database or financial transactions. The implementation of money transfer from one account to another has to ensure that the amount transferred is only deducted from the balance of the source account if it is added to the balance of the target account, and vice versa. Thus, if one or more operations belonging to a transaction fail, no operation belonging to that transaction will be executed.

Traditionally this is achieved by locking resources (see the description of 2-Phase-Commit below for details). Roughly put, locking resources freezes the systems involved in a transaction to make sure that the operations belonging to a transaction are executed “together” and no other operations “interfere”. For obvious reasons, this approach is not really an option in the case of long-living transactions in loosely-coupled systems. Instead, compensations are used to undo operations if the transaction they belong to does not succeed.

The traditional approach to transaction, for example in the context of database systems, is 2-Phase-Commit. As the name suggests, it proceeds in two phases:

1. In the first phase the transaction coordinator initiates the transaction. Each operation or system involved in the transaction informs the coordinator whether it can execute or not. This information is binding, i.e. having indicated the ability to execute means that the system has to ensure execution in the second step.

2. If all operations/systems involved have indicated their ability to execute the coordinator sends the signal execute (commit). If one or more operations do not indicate their ability, which can also be due to time-outs, the entire transaction is aborted (Rollback).

2-Phase-Commit ensures the so-called ACID properties (see, for example [29]).

- Atomicity: transactions are atomic (all or none).
- Consistency: a consistent state of the underlying system is guaranteed at all times.
- Isolation: transactions are isolated from another; race conditions ensure that multiple transaction instances attempts do not collide.
- Durability: once a transaction commits, its updated survive, even if the system goes down.

Transaction standards developed in the context of Web Services aim at developing solutions which are more appropriate for loosely coupled systems. The basic idea is to relax the ACID properties and support long-living transaction based on compensation techniques.

There are two competing approaches to transaction standards in Web Services:

- The Business Transaction Protocol developed by the OASIS Technical Committee.
- The specification family WS-Coordination, WS-AtomicTransaction, WS-BusinessActivity developed by BEA, IBM, and Microsoft. WS-AtomicTransaction and WS-BusinessActivity were initially parts of a joint specification called WS-Transaction.

6.1 Business Transaction Protocol

**Purpose:**

The Business Transaction Protocol, or “BTP,” provides a common understanding and a way to communicate guarantees and limits on guarantees between organizations. The formal rules are necessary for the distribution of parts of business processes outside the boundaries of an organization. BTP solves part of the problem for developers of loosely coupled transactions—the coordination and forcing a consistent termination portions. Expertise in the design of compensating actions is still required, but these compensations are local rather than distributed.

**Players involved:**

BTP is developed through an OASIS Technical Committee. The BTP specification has been developed by BEA, Bowstreet, Choreology, Entrust, Hewlett-Packard, Interwoven, IONA, Oracle, SeeBeyond, Sun, Talking Blocks, Alex Ceponkus, Gordon Hamilton, and Bill Pope.

**Brief Description:**

The BTP (Business Transaction Protocol), approved as an OASIS Technical Committee draft in June 2002, distinguishes two types of transactions: atoms and cohesions. Whereas the former ones behave like ACID transactions the latter ones are more flexible. A central coordinator reviews the status of each operation belonging to a cohesion. Even if some operation cannot successfully commit, the coordinator can still decide to allow the other operations to commit. It is up to the autonomous business participants to use compensation operations to enable “role-back” in the case of failed transactions. [30]

**Status:**

BTP has been released as an OASIS TC draft in June 2002. Since then the TC is working on an updated version.

**DIP Recommendation:**

Although BTP is still actively developed at OASIS, it does not seem to be much supported outside the OASIS TC. Since transactions will probably not be addressed within DIP, relevance of BTP in the context of DIP is rather limited.
6.2 WS-Coordination

**Purpose:**
Web Services Coordination (WS-Coordination) describes an extensible framework for providing protocols that coordinate the actions of distributed applications. It is the basis for WS-AtomicTransaction and WS-BusinessActivity.

**Players involved:**
WS-Coordination was developed by BEA, IBM, and Microsoft.

**Brief Description:**
WS-Coordination describes an extensible framework for providing protocols that coordinate the actions of distributed applications. Such coordination protocols are used to support a number of applications, including those that need to reach consistent agreement on the outcome of distributed activities.

The framework defined by WS-Coordination enables an application service to create a context needed to propagate an activity to other services and to register for coordination protocols. The framework enables existing transaction processing, workflow, and other systems for coordination to hide their proprietary protocols and to operate in a heterogeneous environment.

Additionally WS-Coordination describes a definition of the structure of context and the requirements for propagating context between cooperating services. [31]

**Status:**
WS-Coordination was released in September 2003.

**DIP Recommendation:**
Since transactions will probably not be addressed within DIP, relevance of WS-Coordination in the context of DIP is limited. If transactions are to be addressed in DIP, however, a solution should rather be based on WS-Coordination and WS-BusinessActivity than on BTP.

6.3 WS-AtomicTransaction

**Purpose:**
Web Services Atomic Transaction (WS-AtomicTransaction) provides the definition of the atomic transaction coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification.

**Players involved:**
WS-AtomicTransaction has been developed by BEA, IBM, and Microsoft.

**Brief Description:**
WS-AtomicTransaction defines ACID transactions using Web Services to allow interoperability with older ACID-based systems within an enterprise. It defines three specific agreement coordination protocols for the atomic transaction coordination type: completion, volatile two-phase commit, and durable two-phase commit. Developers can
use any or all of these protocols when building applications that require consistent agreement on the outcome of short-lived distributed activities that have the all-or-nothing property. [32]

**Status:**


**DIP Recommendation:**

Since it is very unlikely that ACID-style transactions will be addressed within DIP at all, relevance of WS-AtomicTransaction in the context of DIP is rather limited.

### 6.4 WS-BusinessActivity

**Purpose:**

WS-BusinessActivity provides the definition of the business activity coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification.

**Players involved:**

WS-BusinessActivity has been developed by BEA, IBM, and Microsoft.

**Brief Description:**

WS-BusinessActivity defines two specific agreement coordination protocols for the business activity coordination type: BusinessAgreementWithParticipantCompletion and BusinessAgreementWithCoordinatorCompletion. Developers can use any or all of these protocols when building applications that require consistent agreement on the outcome of long-running distributed activities.

BusinessActivities (BA) are similar to cohesions in BTP in that compensating actions are used to undo partially completed work and that business logic determines the success or failure of a particular BA. BA are based on BPEL4WS (Business Process Execution Language for Web Services) and use the compensation mechanisms provided by BPEL. [33]

**Status:**

WS-BusinessActivity was released in January 2004 and replaces Part I of the WS-Transaction specification released August 2002.

**DIP Recommendation:**

Since transactions will probably not be addressed within DIP, relevance of WS-BusinessActivity in the context of DIP is limited. If transactions are to be addressed in DIP, however, a solution should rather be based on WS-Coordination and WS-BusinessActivity than on BTP.
7 CHOREOGRAPHY/ORCHESTRATION

Standards for choreography and orchestration address the issue of composing individual service operation into complex processes. They usually take WSDL as a starting point for the definition of the operations’ interfaces. They then add language constructs for combining these WSDL operations, e.g. by defining sequences, parallel executions, or conditional branching. These language constructs also allow the specification of data exchange between the operations involved in the complex process. Another important aspect of composite services is the treatment of exceptions and faults.

Two major standards are emerging in this area: on the one hand, BPEL, which is developed by OASIS and already supported by some software products; on the other hand, WS-CDL, which has been submitted to the W3C.

7.1 BPEL4WS

Purpose:

BPEL4WS (Business Process Execution Language for Web Services), or BPEL for short, defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its partners.

Players involved:

BPEL4WS was originally developed by BEA, IBM, and Microsoft, who were joined by SAP and Siebel Systems for the development of version 1.1. Development is currently continued within the OASIS Technical Committee Web Services Business Process Execution Language. [7]

Brief Description:

BPEL4WS defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its partners. The interaction with each partner occurs through Web Service interfaces, and the structure of the relationship at the interface level is encapsulated in what is called a partner link. The BPEL4WS process defines how multiple service interactions with these partners are coordinated to achieve a business goal, as well as the state and the logic necessary for this coordination. BPEL4WS also introduces systematic mechanisms for dealing with business exceptions and processing faults. Finally, BPEL4WS introduces a mechanism to define how individual or composite activities within a process are to be compensated in cases where exceptions occur or a partner requests reversal. [34]

BPEL4WS is layered on top of several XML specifications: WSDL 1.1, XML Schema 1.0, and XPath1.0. WSDL messages and XML Schema type definitions provide the data model used by BPEL4WS processes. XPath provides support for data manipulation. All external resources and partners are represented as WSDL services. BPEL4WS provides extensibility to accommodate future versions of these standards, specifically the XPath and related standards used in XML computation.

A business process is defined by four major sections in a process definition:

- The <variables> section defines the data variables used by the process, providing their definitions in terms of WSDL message types, XML Schema
simple types, or XML Schema elements. Variables allow processes to maintain state data and process history based on messages exchanged.

- The `<partnerLinks>` section defines the different parties that interact with the business process in the course of processing the order. Each partner link is characterized by a partner link type and a role name. This information identifies the functionality that must be provided by the business process and by the partner service for the relationship to succeed, that is, the portTypes that the purchase order process and the partner need to implement.

- The `<faultHandlers>` section contains fault handlers defining the activities that must be performed in response to faults resulting from the invocation of the assessment and approval services. In BPEL4WS, all faults, whether internal or resulting from a service invocation, are identified by a qualified name. In particular, each WSDL fault is identified in BPEL4WS by a qualified name formed by the target namespace of the WSDL document in which the relevant portType and fault are defined, and the name of the fault. It is important to note, however, that because WSDL 1.1 does not require that fault names be unique within the namespace where the operation is defined, all faults sharing a common name and defined in the same namespace are indistinguishable. In spite of this serious WSDL limitation, BPEL4WS provides a uniform naming model for faults, in the expectation that future versions of WSDL will provide a better fault-naming model.

- The rest of the process definition contains the description of the normal behavior for handling a purchase request. The major elements of this description are: `<receive>`, `<reply>`, `<invoke>`, `<assign>`, `<throw>`, `<terminate>`, `<wait>`, `<empty>`, `<sequence>`, `<switch>`, `<while>`, `<pick>`, `<flow>`, `<scope>`, `<compensate>`.

**Status:**

Version 1.1 of BPEL4WS has been published in May 2003. The OASIS TC Web Services Business Process Execution Language is now responsible for further development of BPEL. Major vendors, such as BEA, IBM, Microsoft, SAP, webMethods have already announced BPEL support in upcoming product releases.

**DIP Recommendation:**

BPEL is highly relevant for DIP as it provides the very basic framework for modeling complex processes with Web Services and thus for the task of composition. Moreover, it is widely accepted and will already be supported by major software vendors in upcoming product releases.
7.2 WS-CDL

Purpose:
WS-CDL is a declarative, XML based language for defining the complementary and observable behavior of a set of collaborating web services. The observable behaviors are defined from a global point of view and not from the point of view of a particular partner (as with WS-BPEL abstract processes). As a result the rules of the collaboration can be defined and agreed by the partners involved.

Players involved:
The Web Services Choreography Description Language (WS-CDL), developed by Oracle, Commerce One, and Novell, is the latest choreography language proposal from the Web Services Choreography Working Group of W3C.

Brief Description:
Web Service Choreographies are part of the process layer of the Web Services Stack [35, 43]. With WSDL, which is part of the description layer, the functionality of a participant in a collaboration can be described but only based on a stateless client-server model. This approach is not appropriate for complex, multi-party, peer-to-peer collaboration - the WS-CDL language makes it possible to describe such collaborations. The WS-CDL language is independent of specific executable business process (orchestration) languages such as [34] or [36] and is designed so that it complements such languages by describing the external behavior of each of the participants in a business collaboration.

As stated in [35] the goals of the WS-CDL language are to permit:

- **Reusability, Composability and Modularity** – choreography definitions can be combined out of existing ones and the same choreography definition can be reused by different partners
- **Multi-party, cooperative exchanges** – a choreography describes the sequence of message exchange between arbitrary number of independent partners
- **Information Alignment** – the choreography description makes it possible for the participating partners to exchange state information and thus synchronize their state in the choreography life-line
- **Exception handling and Transactional behavior** – within a choreography exception handlers can be defined and the transactional
- **Semantics** – choreography definitions can include human readable component descriptions as well as semantic descriptions in languages such as RDF or OWL

The WS-CDL conceptual model is comprised of the following entities:

- **Participants, Roles and Relationships.** A Participant identifies a party in multi-party business collaboration. A participant is related to one or more roles that it plays during the collaboration (for example an Organization may take a Buyer role when buying goods and a Seller role when selling them, [35]) A Role represents the observable behavior of a participant in a multiparty collaboration. The Role element is associated with one or more WSDL interfaces, each representing a subset of the observable behavior of participant. A Relationship
represents an association existing between two roles and the possible ways these roles may interact. Relationships are always between two roles but more than one relationship may be defined between the same roles.

- **Types, Variables and Tokens – Information Types** are an abstraction used to describe the type of information used within the choreography activities without making direct references to data types defined in a WSDL / XML Schema documents. **Variables** can be used to exchange information (from/to a message) and state information. Variables are associated with a type (Information Type or a Channel Type). A **Token** is an alias for a piece of data in a variable or a message. Tokens have types and Token Locators are used to select the relevant information for a token based on XPath expressions.

- **Channels** – channels (channel types) specify the way two participants may exchange information. A channel is always associated with a participant Role that is the target of a choreography Interaction.

- **Activities, Work Units and Ordering Structures** – Activities are the lowest level components within a choreography. Activities can be divided into Basic Activities (units that perform actual work, such as Interaction, Perform, Assign and NoAction activities), Ordering Structures which combine activities and other ordering structures in a nested way by specifying the correct order of executing the activities (such as Sequence, Parallel and Choice) and Work Units (which describe activities with explicit constraints and data dependencies that must be satisfied before the activity can be executed)

- **Choreographies** – a choreography defines the observable behavior of a set of collaborating parties. A choreography may contain several enclosed choreographies but there is always one top-level choreography defined per package (WS-CDL document). A choreography may contain an optional exception handling block specifying the actions that should be performed in exceptional circumstances. A choreography may contain a finalizer block containing actions that provide the means to semantically rollback the effects of the completed actions in exceptional conditions (e.g. compensation). A choreography contains an Activity that performs the actual work (and which in turn may be comprised of many activities).

**Status:**

Version 1.0 of WS-CDL has been published as a working draft of the Web Services Choreography Working Group in April 2004.

**DIP Recommendation:**

WS-CDL is a promising initiative based on existing efforts like WSCI ([37]). Its purpose is to describe choreographies involving multiple participants. The language contains some means of attaching arbitrary metadata to a choreography (with the help of the **description** element) and extensions that use RDF/OWL to semantically enrich a choreography description are possible and can be further explored. It is thus recommended to closely monitor future development with respect to WS-CDL and to take it into account for composition in DIP.
8 MANAGEMENT

Web Services Management is the only area in which standard development is not clearly dominated by IBM and Microsoft, although IBM is involved in the development of one of the standards. There are currently two main proposals for Web Services Management:

- Hewlett-Packard’s Web Services Management Framework
- WS-Manageability developed by Computer Associates, IBM, and Talking Blocks

Both specifications have been submitted as input to the OASIS Technical Committee Web Services Distributed Management, which is aiming at providing a standard in this area.

8.1 Web Services Distributed Management

Purpose:
The purpose of the Web Services Distributed Management (WSDM) TC at OASIS is to define web services management, including using web services architecture and technology to manage distributed resources. The TC will also develop the model of a web service as a manageable resource.

Players involved:
Actional Corporation, AmberPoint, BEA, BMC, Computer Associates, Hewlett-Packard, IBM, Novell, OpenNetwork, Oracle, Sun, Tibco, and webMethods are among the companies represented in the TC.

Brief Description:
The WSDM TC’s purpose is to define web services management. This includes using web services architecture and technology to manage distributed resources. The TC will also develop the model of a web service as a manageable resource. It will collaborate with various evolving activities within other standards groups, including, but not limited to, DMTF (working with its technical work groups regarding relevant CIM Schema), GGF (on the OGSA common resource model and OGSI regarding infrastructure), and W3C (the web services architecture committee). Also liaison with other OASIS TCs, including the security TC and other management oriented TCs. [38]

Status:
WSDM is still in the development phase. Hewlett-Packard has submitted WSMF (see below) as input to WSDM in July 2003. IBM, Computer Associates, and Talking Blocks have submitted Web Service Manageability (WS-Manageability) as input to WSDM in September 2003.

DIP Recommendation:
As WSDM is still under development and the need for using a management standard in DIP is not yet apparent, WSDM has only limited relevance for DIP. If Web Services Management is to be addressed in DIP, it is recommended to closely monitor WSDM
activities, as there is a high chance that an emerging standard from this TC will be widely accepted and used.

8.2 WSMF

**Purpose:**
The Web Services Management Framework (WSMF) is a logical architecture for managing computing resources, including Web services themselves, through Web services.

**Players involved:**
WSMF is developed by Hewlett-Packard.

**Brief Description:**
This framework is based on the concept of managed objects and their relationships. In this view, a managed object represents a managed resource and exposes a set of management interfaces. Through the management interfaces, the underlying resource can be managed and controlled. Similarly, relationships among managed objects represent relationships among underlying resources. To better support important IT domains (such as EAI, ERP, SCM, Web services), WSMF is model-neutral and is designed to be applied to many different domains with varying management requirements. It also is platform-neutral, allowing for consistent management of J2EE, .NET, and other platforms. [39]

WSMF currently contains three loosely coupled specifications that provide the necessary components of a management stack:

- Web Services Events (WS-Events): an XML syntax and a set of processing rules for advertising, subscribing, producing and consuming Web Services Events.

- Foundation: a solution for using Web services to expose manageability information (and the model for that information) for a manageable resource. Using WSMF-Foundation, manageability information can be described, exposed, discovered, and accessed. Manageability interfaces can be extended to address the management capabilities of resources that are domain-specific. This specification uses WS-Events.

- Web Services Management: a solution for providing manageability information with the effect that Web services can be managed by a management system. This solution relies on the WSMF-Foundation and can be further extended as required to support additional features of Web services.

**Status:**
WSMF version 2.0 has been released by HP in July 2003 and submitted as input to the OASIS TC WSDM.

**DIP Recommendation:**
As WSMF is not yet widely accepted or used in applications and the need for using a management standard in DIP is not yet apparent, it has only limited relevance for DIP. If Web Services Management is to be addressed in DIP, it is recommended to closely
monitor whether WSMF is incorporated in the OASIS WSDM standard and to use that standard instead of WSMF.

8.3 WS-Manageability

**Purpose:**
The WS-Manageability specification introduces the general concepts of a manageability model in terms of manageability topics and the aspects used to define them.

**Players involved:**
WS-Manageability has been developed by Computer Associates, IBM, and Talking Blocks.

**Brief Description:**
The WS-Manageability specification begins by introducing the general concepts of a manageability model in terms of manageability topics, (identification, configuration, state, metrics, and relationships) and the aspects (properties, operations and events) used to define them. These abstract concepts apply to understanding and describing the manageability information and behavior of any IT resource, not just Web services.

The manageability model for Web services endpoint is defined as concrete models in UML using the topics and aspects concepts, without implying any particular implementation or locus of implementation. Appropriate manageability interfaces are defined based on the UML manageability models. While some parts of this model may be useful for modeling the manageability of any IT resource, this specification is focused exclusively on the requirements of Web service endpoints and does not propose a complete generic resource manageability model. The OASIS WSDM "Management Using Web Services" specification may incorporate these more generic models. [40]

**Status:**
WS-Manageability has been released in September 2003 and submitted as input to the OASIS TC WSDM.

**DIP Recommendation:**
As WS-Manageability is not yet widely accepted or used in applications and the need for using a management standard in DIP is not yet apparent, it has only limited relevance for DIP. If Web Services Management is to be addressed in DIP, it is recommended to closely monitor whether WS-Manageability is incorporated in the OASIS WSDM standard and to use that standard instead of WS-Manageability.
9 RECOMMENDATIONS FOR DIP

Though the proposal of various Web Services protocols has been a fast moving area, their transition into actual open standards is inevitably much slower. There are only a few protocols that have, or are close to completing the standards process proper. Some key proposals have yet to be submitted to any standards body. Figure 7 summarizes the current status of the most relevant Web Services Protocols. For DIP it is advisable to continuously monitor of what are currently the two main standards groups involved in Web Services: World Wide Web Consortium (W3C) and Organization for the Advancement of Structured Information Standards (OASIS).

![Table of Web Service Protocols]

A summary of the presented standardization activities in this document is given in Figure 8. The relevance of the proposals for the DIP project is categorized in “low”, “medium”, and “high”. The proposed activity distinguishes between four categories.

- The simplest one is “–“ (“do nothing”), which goes together with the fact that this standardization effort has no relevance for the DIP project at all. Therefore no activity is necessary.
- “Wait” till one of several competitive, emerging technologies in the same area will gain enough support. For DIP these technologies have only low relevance.
- “Monitor (framework)” means that DIP should monitor the ongoing work with low effort. These standards are only related to DIP due to the fact that future frameworks will offer/integrate these technologies.
- “Monitor (compatibility)” means that DIP should monitor the ongoing work with medium effort. These standards are related to DIP due to the fact that the DIP project should assure compatibility.
- “Monitor (development)” means that DIP should monitor the ongoing work with high effort. These standards are related to DIP due to the fact that there is need
for components in the DIP project that implement such functionality. This can either result in adopting standards or actively contribute to the development.

<table>
<thead>
<tr>
<th>Area</th>
<th>Proposal</th>
<th>Relevance for DIP</th>
<th>Proposed Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging</td>
<td>WS-Addressing</td>
<td>M</td>
<td>Monitor (framework)</td>
</tr>
<tr>
<td></td>
<td>WS-Eventing / WS-Notification</td>
<td>L</td>
<td>Wait</td>
</tr>
<tr>
<td></td>
<td>WS-ReliableMessaging / WS-Reliability</td>
<td>L</td>
<td>Wait</td>
</tr>
<tr>
<td>Metadata</td>
<td>WS-Policy</td>
<td>M</td>
<td>Monitor (framework)</td>
</tr>
<tr>
<td></td>
<td>WS-Discovery</td>
<td>M</td>
<td>Monitor (devel.)</td>
</tr>
<tr>
<td></td>
<td>WS-MetadataExchange</td>
<td>L</td>
<td>–</td>
</tr>
<tr>
<td>Security</td>
<td>WS-Security</td>
<td>M</td>
<td>Monitor (compatibility)</td>
</tr>
<tr>
<td></td>
<td>WS-Trust</td>
<td>L</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>WS-SecureConversation</td>
<td>L</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>WS-Federation</td>
<td>L</td>
<td>–</td>
</tr>
<tr>
<td>Transactions</td>
<td>Business Transaction Protocol</td>
<td>L</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>WS-Coordination</td>
<td>L</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>WS-AtomicTransaction</td>
<td>L</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>WS-BusinessActivity</td>
<td>L</td>
<td>–</td>
</tr>
<tr>
<td>Choreography / Orchestration</td>
<td>BPEL4WS</td>
<td>H</td>
<td>Monitor (devel.)</td>
</tr>
<tr>
<td></td>
<td>WS-CDL</td>
<td>H</td>
<td>Monitor (devel.)</td>
</tr>
<tr>
<td>Management</td>
<td>WS-Distributed Management</td>
<td>M</td>
<td>Monitor (framework)</td>
</tr>
<tr>
<td></td>
<td>WSMF</td>
<td>L</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>WS-Manageability</td>
<td>L</td>
<td>–</td>
</tr>
</tbody>
</table>

*Figure 8: Summary of recommendations for DIP*
ACRONYMS

ASAP: Asynchronous Service Access Protocol
BPEL: Business Process Execution Language
BPEL4WS: Business Process Execution Language for Web Services
BPMI: The Business Process Management Initiative
DIP: Data, Information, and Process Integration with Semantic Web Services
DMTF: Distributed Management Task Force
IETF: Internet Engineering Task Force
JCP: Java Community Process
MTOM: Message Transmission Optimization Mechanism
OAGi: Office Automation Group, Inc.
OASIS: Organization for the Advancement of Structured Information Standards
OMG: Object Management Group
SOAP: Simple Object Access Protocol
UDDI: Universal Description, Discovery and Integration
UN/CEFACT: United Nations Centre for Trade Facilitation and Electronic Business
W3C: World Wide Web Consortium
WS-CAF: Web Services Composite Application Framework
WS-CDL: Web Services Choreography Description Language
WSDL: Web Service Definition Language
WSDM: Web Services Distributed Management
WS-I: Web Services Interoperability Organization
WSMF: Web Service Management Framework
WSRP: Web Services Standard for Remote Portals
XML: Extensible Markup Language
REFERENCES


[8] *Universal Description, Discover and Integration (UDDI)*. http://www.uddi.org/


