

2 **Reference Model for Service Oriented**
3 **Architectures**

4 **Working Draft 10, 15 November 2005**

5 **Document identifier:**

6 wd-soa-rm-10

7 **Location:**

8 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

9 **Editors:**

10 C. Matthew MacKenzie, Adobe Systems Incorporated, mattm@adobe.com

11 Ken Laskey, MITRE Corporation, klaskey@mitre.org

12 Francis McCabe, Fujitsu Limited, fgm@fla.fujitsu.com

13 Peter Brown, Individual, peter@justbrown.net

14 Rebekah Metz, Booz Allen Hamilton, metz_rebekah@bah.com

15
16
17 **Abstract:**

18 This Reference Model for Service Oriented Architectures is an abstract framework for
19 understanding the significant entities and relationships between them within service-
20 oriented systems, and for the development of consistent standards or specifications
21 supporting that environment. It is based on core unifying concepts of SOA and may be
22 used by architects developing specific service oriented architectures or by those needing
23 to explain SOA principles. A reference model is not directly tied to any standards,
24 technologies or other concrete implementation details. It does seek to provide a common
25 semantics that can be used unambiguously across and between different
26 implementations.

27 While service-orientation may be a popular concept found in a broad variety of
28 applications and domains, we make no attempt to account for the use of the concepts
29 and relationships described in this specification outside of the software domain.

30 **Status:**

31 This document is updated periodically on no particular schedule. Send comments to the
32 editor(s).

33 Committee members should send comments on this specification to the [soa-
34 rm@lists.oasis-open.org](mailto:soa-rm@lists.oasis-open.org) list. Others should visit the SOA-RM TC home page at
35 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm, and record
36 comments using the web form available there.

37 For information on whether any patents have been disclosed that may be essential to
38 implementing this specification, and any offers of patent licensing terms, please refer to
39 the Intellectual Property Rights section of the SOA-RM TC web page at:

40 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

42
43
44

The errata page for this specification is at:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm.

Draft

Draft

44 Table of Contents

45	1	Introduction.....	4
46	1.1	What is a reference model.....	4
47	1.2	Audience.....	4
48	1.3	How to use the reference model.....	5
49	1.4	Notational Conventions.....	5
50	1.5	Relationships to Other Standards.....	5
51	2	Service Oriented Architecture.....	6
52	2.1	What is SOA?.....	6
53	2.2	How is Service Oriented Architecture different?.....	7
54	2.3	The Benefits of Service Oriented Architecture.....	8
55	3	The Reference Model.....	9
56	3.1	Overview of model.....	9
57	3.2	The Reference Model.....	10
58	3.2.1	Service.....	10
59	3.2.2	Service description.....	10
60	3.2.3	Descriptions and Metadata.....	12
61	3.3	Interacting with services.....	13
62	3.3.1	Data model.....	13
63	3.3.2	Behavioral model.....	15
64	3.3.3	Actualized Services.....	16
65	3.4	Real World Effect.....	17
66	3.5	Policies and Contracts.....	17
67	3.5.1	Service Policy.....	18
68	3.5.2	Service Contract.....	18
69	3.6	Visibility.....	19
70	3.6.1	Awareness.....	19
71	3.6.2	Willingness.....	19
72	3.6.3	Reachability.....	20
73	4	Conformance Guidelines.....	21
74	5	References.....	22
75	5.1	Normative.....	22
76	5.2	Non-Normative.....	22
77		Appendix A. Glossary.....	23
78		Appendix B. Acknowledgments.....	27
79		Appendix C. Notices.....	28
80			

81 1 Introduction

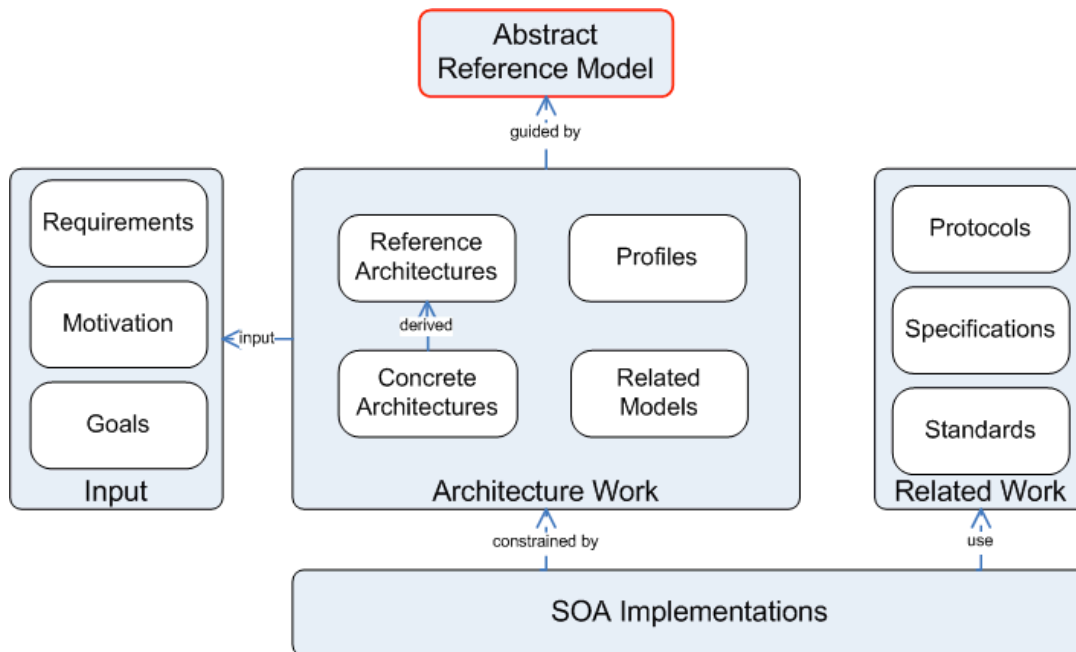
82 The notion of Service Oriented Architecture (SOA) has received significant attention within
83 the software design and development community. The result of this attention is the
84 proliferation of many conflicting definitions of SOA. Whereas SOA architectural patterns (or
85 *reference architectures*) may be developed to explain and underpin a generic design
86 template supporting a specific SOA, a reference model is intended to provide an even
87 higher level of commonality, with definitions that should apply to *all* SOA.

88 1.1 What is a reference model

89 A reference model is an abstract framework for understanding significant relationships
90 among the entities of some environment that enables the development of specific
91 architectures using consistent standards or specifications supporting that environment. A
92 reference model consists of a minimal set of unifying concepts, axioms and relationships
93 within a particular problem domain, and is independent of specific standards, technologies,
94 implementations, or other concrete details.

95 The purpose of a reference model is to provide a common conceptual framework that can
96 be used consistently across and between different implementations and is of particular use
97 in modeling specific solutions.

98 The goal of this reference model is to define the essence of service oriented architecture, and
99 emerge with a vocabulary and a common understanding of SOA. It provides a normative
100 reference for SOA as an abstract and powerful model, irrespective of the various and inevitable
101 technology evolutions that will impact SOA.



102

103 1.2 Audience

104 The intended audiences of this document include non-exhaustively:

- 105 • Architects and developers designing, identifying or developing a system based on the
106 service-oriented paradigm.

Draft

- 107 • Standards architects and analysts developing specifications that rely on service oriented
108 architecture concepts.
- 109 • Decision makers seeking a "consistent and common" understanding of service oriented
110 architecture.
- 111 • Users who need a better understanding of the concepts and benefits of service oriented
112 architecture.

113 1.3 How to use the reference model

114 New readers are encouraged to read this reference model in its entirety. Concepts are presented
115 in an order that the authors hope simplify understanding.

116 This section introduces the conventions, defines the audience and sets the stage for the rest of
117 the document. Non-technical readers are encouraged to read this information as it provides
118 background material necessary to understand the nature and usage of reference models.

119 Section 2 introduces the concept of SOA and identifies some of the ways that it differs from
120 previous paradigms for distributed systems. Section 2 offers guidance on the basic principles of
121 service oriented architecture. This can be used by non-technical readers to gain an explicit
122 understanding of the core principles of SOA and by architects as guidance for developing specific
123 service oriented architectures.

124 Section 3 introduces the Reference Model for SOA. In any framework as rich as SOA, it is difficult
125 to avoid a significant amount of cross referencing between concepts. This makes presentation of
126 the material subject to a certain amount of arbitrariness. We resolve this by initially discussing the
127 key concepts behind the reference model and then follow this by more detailed sections on the
128 main concepts. In the first more detailed section, *service* is defined along with *service description*.
129 There then follows a section about interaction between service participants, followed by sections
130 on service policies and expectations. Finally, the concept of service visibility is introduced.

131 Section 4 addresses compliance with this reference model.

132 The glossary provides a summary of the definitions made and used within the reference model
133 specification.

134 1.4 Notational Conventions

135 The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*,
136 and *optional* in this document are to be interpreted as described in [RFC2119].

137 References are surrounded with **[square brackets and are in bold text]**.

138 1.5 Relationships to Other Standards

139 Due to its nature, this reference model may have an implied relationship with any group that:

- 140 • Considers its work "service oriented";
- 141 • Makes (publicly) an adoption statement to use the Reference Model for SOA of this TC
142 as a base or inspiration for their work; and
- 143 • Standards or technologies that claim to be service oriented.

144 The reference model does not endorse any particular service-oriented architecture, or attest to
145 the validity of third party reference model conformance claims.

146

2 Service Oriented Architecture

147

2.1 What is SOA?

148

Service Oriented Architecture (SOA) is a paradigm for organizing and using distributed capabilities that may be under the control of different ownership domains. It is natural in such a context to think of one person's needs being met by capabilities offered by someone else; or, in the world of distributed computing, one computer agent's requirements being met by a computer agent belonging to a different owner. There is not necessarily a one-to-one correlation between needs and capabilities; the granularity of needs and capabilities vary from fundamental to complex, and any given need may require the combining of numerous capabilities while any single capability may address more than one need. The perceived value of SOA is that it provides a powerful framework for matching needs and capabilities and for combining capabilities to address those needs.

149

150

151

152

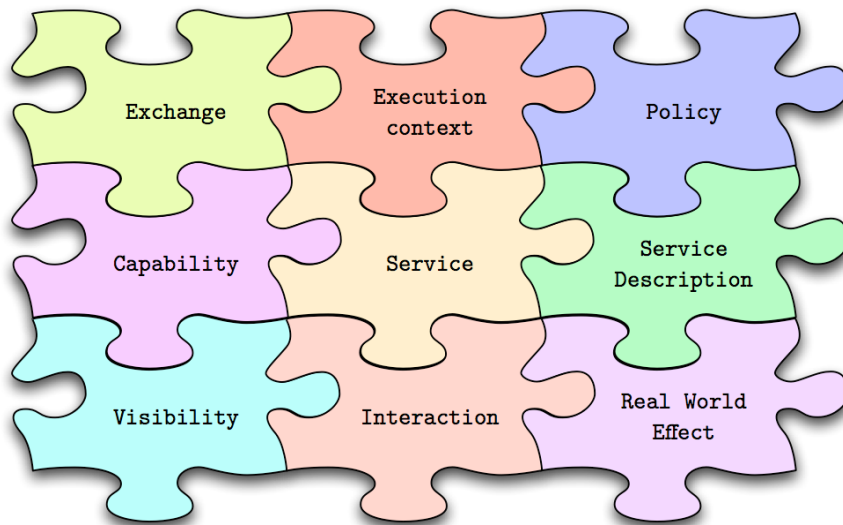
153

154

155

156

157



158

159

Visibility, interaction, and effect are key concepts for describing the SOA paradigm. **Visibility** refers to the capacity for those with needs and those with capabilities to be able to see each other to interact. Visibility is typically enhanced through the use of metadata to describe such aspects as functional and technical requirements, related constraints and policies, and mechanisms for interaction. For maximum visibility, metadata must be in a form in which its syntax and semantics are widely accessible and understandable.

160

161

162

163

164

165

Whereas visibility introduces the possibilities for matching needs to capabilities (and vice versa), **interaction** is the activity of *using* the capability. Typically mediated by the exchange of messages, an interaction proceeds through a series of information exchanges and invoked actions. There are many facets of interaction; but they are all grounded in a particular **execution context** – the set of technical and business elements that together form a path between those with needs and those with capabilities and that permit information to be exchanged, actions to be performed and provides a decision point for any policies and contracts that may be in force.

166

167

168

169

170

171

172

The purpose of using a capability is to realize one or more **real world effects**. At its core, an interaction is “an act” rather than “an object” and the result of an interaction is an effect (or a set/series of effects). We are careful to distinguish *public* actions and *private* actions; private actions are inherently unknowable by other parties. On the other hand, public actions result in changes to the *state* that is shared (at least) between those involved in the current execution context. Real world effects are, then, manifested in terms of changes to this shared state.

173

174

175

176

177

Draft

178 The expected effects, together with relevant preconditions associated with those effects, should
179 be made visible as part of the capability metadata and form an important part of the assessment
180 as to whether a given capability matches similarly described needs. It is not possible to describe
181 every possible effect of using a capability: indeed a cornerstone of SOA is that such knowledge is
182 not necessary.

183 A concept that is considered central to SOA has not yet been mentioned – that of **service**. Both
184 needs and capabilities exist outside of SOA. What distinguishes SOA is the perceived
185 improvement in bringing needs and capabilities together. **In SOA, services are the mechanism**
186 **by which needs and capabilities are brought together.** SOA is not the solution of domain
187 problems but rather a way of organizing a wider array of possibilities to generate a domain
188 solution. By itself, SOA does not provide a solution to a difficult domain problem where a
189 satisfactory solution does not already exist. SOA can, however, provide an organizing and
190 delivery paradigm that enables one to get more value from use of both solutions which are locally
191 “owned” and solutions under the control of others. It also enables one to express solutions in a
192 way that makes it easier to modify or evolve the identified solution or to try alternate domain
193 solutions.

194 The concepts of visibility, interaction, and effect apply directly to services in the same manner as
195 these were described for the general SOA paradigm. Visibility is promoted the **service**
196 **description** which contains the information necessary to interact with the service and describes
197 this in such terms as the service inputs, outputs, and associated semantics. The service
198 description also conveys what is accomplished when the service is invoked and the conditions for
199 invoking the service. In general, entities (people and organizations) offer capabilities through
200 services and act as **service providers**. Those with needs who make use of capabilities through
201 their associated services are referred to as **service consumers**. The service description allows
202 prospective consumers to decide if the service is suitable for their current needs and establish
203 whether a consumer satisfies the requirements, if any, of the service provider to be permitted
204 access.

205 Having described what is SOA, it is appropriate to note several things which are related but are
206 not necessary attributes or restrictions.

- 207 • SOA identifies necessary aspects of interactions involving multiple ownership domains;
208 however, it does not directly embody concepts relating to ownership.
- 209 • SOA is commonly implemented using Web services, but services can be made visible,
210 support interaction, and generate effects through other implementations.

211 In most discussions of SOA, the terms “loose coupling” and “coarse-grained” are commonly
212 applied as SOA concepts. However, these terms are subjective and without useful metrics to
213 indicate compliance. In terms of needs and capabilities, SOA is most effective when it focuses on
214 bringing solutions to bear, rather than on “fine-grained” pieces of a particular implementation that
215 may not be reusable beyond a particular solution. Granularity and coarseness are usually relative
216 to detail for the level of the problem being addressed (e.g. one that is more strategic compared
217 with another that considers the issues down to the algorithm level). Counting the number of
218 interfaces or the number or types of information exchanges connected to an interface does not
219 help define the optimum level of detail.

220 2.2 How is Service Oriented Architecture different?

221 How does this paradigm of Service Oriented Architecture differ from other approaches to
222 organizing and understanding IT assets? Essentially, there are two areas in which SOA
223 revolutionizes the framework of concepts that functions as a tool for addressing IT solutions.

224 First, SOA reflects the reality that ownership boundaries are a motivating consideration in the
225 architecture and design of systems. This recognition is evident in the core concepts of visibility,
226 interaction and effect. However, SOA does not itself address all the concepts associated with
227 ownership, ownership domains and actions communicated between legal peers. To fully account
228 for concepts such as trust, business transactions, authority, delegation and so on – additional

Draft

229 conceptual frameworks and architectural elements are required. Within the context of SOA,
230 these are likely to be represented within service descriptions and interfaces.

231 Second, SOA applies the lessons learned from commerce to the organization of IT assets to
232 facilitate the matching of capabilities and needs. That two or more entities come together within
233 the context of a single interaction implies the exchange of some type of value. This is the same
234 fundamental basis as trade itself, and suggests that as SOAs evolve away from interactions
235 defined in a point-to-point manner to a marketplace of services; the technology and concepts can
236 scale as successfully as the commercial marketplace.

237 Unlike Object Oriented Programming paradigms, where the focus is on packaging data with
238 operations, the central focus of SOA is the task or business function – getting something done.
239 This is a more viable basis for large scale systems because it is a better fit to the way human
240 activity itself is managed – by delegation and by trading.

241 **2.3 The Benefits of Service Oriented Architecture**

242 The main drivers for SOA-based architectures are the requirement to facilitate the manageable
243 growth of large-scale enterprise systems, the requirement to facilitate Internet-scale provisioning
244 and use of services and the requirement to reduce costs in organization to organization
245 cooperation.

246 The value of SOA is that it provides a simple scalable paradigm for organizing large networks of
247 systems that require interoperability to realize the value inherent in the individual components.
248 Indeed, SOA is scalable because it makes the fewest possible assumptions, including about the
249 network and also minimizes any trust assumptions that are often implicitly made in smaller scale
250 systems.

251 An architect using SOA principles is better equipped, therefore, to develop systems that are
252 scalable, evolvable and manageable. It should be easier to decide how to integrate functionality
253 across ownership boundaries. For example, a large company that acquires a smaller company
254 must determine how to integrate the acquired IT infrastructure into its overall IT portfolio.

255 Through this inherent ability to scale and evolve, SOA enables an IT portfolio which is also
256 adaptable to the needs of a specific problem domain or process architecture. The infrastructure
257 SOA encourages is also more agile and responsive than one built on an exponential number of
258 pair-wise interfaces. Therefore, SOA can also provide a solid foundation for business agility and
259 adaptability.

260 3 The Reference Model

261 A service oriented architecture represents a uniform means to offer, discover and interact with
262 capabilities to produce desired effects consistent with measurable preconditions and
263 expectations. This section introduces the main concepts within the SOA paradigm. A detailed
264 discussion of the concepts and their relationships are in the sections that follow.

265 3.1 Overview of model

266 A key concept of SOA is that of **service**. In general, entities (people and organizations) create
267 capabilities to solve or support a solution for the problems they face in the course of their
268 business. SOA is a way to organize the world around this key concept of service. The noun
269 "service" is defined in dictionaries as "The performance of work (a function) by one for another."
270 However, service, as the term is generally understood, also combines the following related ideas:

- 271 • The capability to perform work for another
- 272 • The specification of the work offered for another
- 273 • The offer to perform work for another

274 These concepts emphasize a distinction between a capability and the ability to bring that
275 capability to bear in the context of SOA, where the capability exists independently of SOA. The
276 term **service** should, therefore, be understood as a set of separate, yet interrelated and more
277 precise concepts. These concepts are an offer, interaction and effect.

278 The concept of an **offer** follows directly from the dictionary definition of service: 'by one' and 'for
279 another.' In general terms, an offer is a proposal; made by providers which may possess a
280 capability that address a need. In order to use a service, it is necessary to know that it exists,
281 what is accomplished if the service is invoked, how the service is invoked, and other
282 characteristics. Collectively this is the service **visibility**. When given an explicit searchable form,
283 this information allows, for example, prospective consumers to decide if the service is suitable for
284 their current needs and establish whether a consumer satisfies any requirements of the service
285 provider to be permitted access. This information constitutes the **service description**.

286 The convergence of a capability and a need results in an **interaction**. In an SOA, interaction is
287 effected by exchanging information between service providers and consumers. Typically this is
288 achieved by exchanging messages using a standardized protocol; however, there are many
289 modalities possible for interacting with services.

290 At its core, an interaction is "an act" rather than "an object." Therefore, interaction is the focus of
291 the interfaces and behavior necessary to support the interaction. Recall that interaction may, and
292 typically does, involve crossing ownership boundaries. SOA identifies some of the necessary
293 aspects of interactions involving multiple ownership domains; however, it does not directly
294 embody concepts relating to ownership.

295 The final key concept is the **real world effect** of using services; it is always the case that there is
296 an intended purpose to providing a service and similarly to using a service.

297 Given that there is often an ownership boundary between the service provider and consumer,
298 there is a natural distinction to be drawn between the public interactions with a service and the
299 private actions of both the service provider and consumer. This distinction maintains and
300 encourages independence of each service participant which, in turn, greatly enhances the
301 scalability and security attributes of SOA. Focus can be directed to the public aspects of using a
302 service by examining the **conditions** of using a service and the **expectations** that arise as a
303 result of using the service. Service conditions are loosely associated with the **service policies**
304 and the expectations with **service contracts**.

Draft

305 3.2 The Reference Model

306 3.2.1 Service

307 A **service** is a means to access to one or more capabilities, where the access is provided using a
308 prescribed interface and is exercised consistent with constraints and policies as specified by the
309 service description. A service is provided by one entity – the **service provider** – for use by
310 others, but the eventual consumers of the service may not be known to the service provider and
311 may demonstrate uses of the service beyond the scope originally conceived by the provider.

312 A service is invoked through a service interface (see Section 3.2.2.4), where the interface
313 comprises the specifics of how to access the underlying capabilities. There are no constraints on
314 what constitutes the underlying capability or how access is implemented by the service provider.
315 Thus, the service could carry out its described functionality through one or more automated
316 and/or manual processes that themselves could invoke other available services. A service is
317 opaque in that its implementation is typically hidden from the service consumer except for (1) the
318 data model exposed through the published service interface and (2) any information included as
319 metadata to describe aspects of the service which are needed by service consumers to
320 determine whether a given service is appropriate for the consumer's needs. The consequence of
321 invoking a service is a realization of one or more real world effects. The effects may include:

- 322 1. information returned in response to a request,
- 323 2. a change to the shared state of defined entities, or
- 324 3. some combination of (1) and (2).

325 Note, the **service consumer** in (1) does not typically know how the information is generated, e.g.
326 whether it is extracted from a database or generated dynamically; in (2), the service consumer
327 does not typically know how the state change is effected. In either case, the service consumer
328 would need to provide input parameters required by the service and the service would return
329 information, status indicators, or error descriptions, where both the input and output are as
330 described by the data model exposed through the published service interface. Note that the
331 service may be invoked without requiring information from the consumer (other than a command
332 to initiate action) and may accomplish its functions without providing any return or feedback to the
333 consumer.

334 The service concept above emphasizes a distinction between a capability that represents some
335 functionality created to address a need and the point of access to bring that capability to bear in
336 the context of SOA. It is assumed that capabilities exist outside of the SOA. In actual use,
337 maintaining this distinction may not be critical (i.e. the service may be talked about in terms of
338 being the capability) but the separation is pertinent in terms of a clear expression of the nature of
339 SOA and the value it provides.

340 3.2.2 Service description

341 The service description represents the information needed in order to use a service. It may be
342 considered part of or the complete set of the metadata (see Section 3.2.3) associated with a
343 service. In any case, the service description overlaps and shares many common properties with
344 service metadata. In most cases, there is no one "right" set of metadata but rather the metadata
345 content depends on the context and the needs of the parties using the associated entity. The
346 same holds for a service description. While there are certain elements that are likely to be part of
347 any service description, most notably the data model, many elements such as function and policy
348 may vary.

349 Best practice suggests that the service description should be represented using a standard,
350 referenceable format. Such a format facilitates the use of common processing tools (such as
351 discovery engines) that can, in turn, capitalize on the service description.

Draft

352 While the concept of a SOA supports use of a service without the service consumer needing to
353 know the details of the service implementation, the service description makes available critical
354 information that a consumer needs in order to decide whether or not to use a service. In
355 particular, a service consumer must possess the following items of information:

- 356 1. That the service exists and is **reachable** (i.e., the service is **visible** to the service
357 consumer and there are sufficient mechanisms in place for the service participants to be
358 able to interact);
- 359 2. That the service performs a certain function or set of functions;
- 360 3. That the service operates under a specified set of constraints and policies;
- 361 4. That the service will (to some implicit or explicit extent) comply with policies as prescribed
362 by the service consumer;
- 363 5. How to interact with the service in order to achieve the required objectives, including the
364 format and content of information exchanged between the service and the consumer and
365 the sequences of information exchange that may be expected.

366 Subsequent sections of this document will deal with these aspects of a service in detail but the
367 following subsections will describe the relationship of these information items to the service
368 description.

369 3.2.2.1 Service Reachability

370 A service description should include sufficient data to permit a service consumer and service
371 provider to exchange information. This might include metadata (such as the location of the
372 service and what information protocols it supports and requires) and information that allows the
373 service consumer to determine if the service is currently reachable or not.

374 3.2.2.2 Service Functionality

375 Item 2 relates to the need to unambiguously express the function(s) of the service and the real
376 world effects (see Section **Error! Reference source not found.**) that result from it being invoked.
377 This portion of the description needs to be expressed in a way that is generally understandable
378 by service consumers but able to accommodate a vocabulary that is sufficiently expressive for the
379 domain for which the service provides its functionality. The description of functionality may
380 include, among other possibilities, a textual description intended for human consumption or
381 identifiers or keywords referenced to specific machine-process-able definitions. For a full
382 description, it may be useful to indicate multiple identifiers or keywords from a number of different
383 collections of definitions.

384 Part of the description of functionality may include underlying technical assumptions that
385 determine the limits of functionality exposed by the service or of the underlying capability. For
386 example, the amounts dispensed by an automated teller machine (ATM) are consistent with the
387 assumption that the user is an individual rather than a business. To use the ATM, the user must
388 not only adhere to the policies and satisfy the constraints of the associated financial institution
389 (see Section 3.2.2.3 for how this relates to service description and Section **Error! Reference**
390 **source not found.** for a detailed discussion) but the user is limited to withdrawing certain fixed
391 amounts of cash and a certain number of transactions in a specified period of time. The financial
392 institution, as the underlying capability, does not have these limits but the service interface as
393 exposed to its customers does, consistent with its assumption of the needs of the intended user.
394 If the assumption is not valid, the user may need to use another service to access the capability.

395 3.2.2.3 Policies Related to a Service

396 Items 3 and 4 from Section 2.2.2 relate to the service description's support for associating
397 constraints and policies with a service and providing necessary information for prospective
398 consumers to evaluate if a service will act in a manner consistent with the consumer's constraints
399 and policies.

Draft

400 In some situations the consumer may similarly provide an indication of its constraints and policies
401 to support a service's need to do a similar evaluation of suitability. Thus, both prospective
402 consumers and providers are likely to use the service description to establish what Section 3.3.3
403 refers to as the **execution context**.

404 **3.2.2.4 Service Interface**

405 The service interface is the means referred to in Item 5 for interacting with a service. It includes
406 the specific protocols, commands, and information exchange by which actions are initiated that
407 result in the real world effects as specified through the service functionality portion of the service
408 description.

409 The specifics of the interface should be syntactically represented in a standard referenceable
410 format. These prescribe what information needs to be provided to the service in order to exercise
411 its functionality and/or the results of the service invocation to be returned to the service
412 consumer. This logical expression of the set of information items associated with the
413 consumption of the service is often referred to as the service's data model. It should be noted
414 that the particulars of the standard reference-able format is beyond the scope of the reference
415 model. However, requiring that mechanisms be available (in order to define and retrieve such
416 definitions) is fundamental to the SOA concept.

417 **3.2.2.5 An Example of Using Information Contained in the Service** 418 **Description**

419 The following example may help clarify the concepts related to service and service description.

420 A utility has the capacity to generate and distribute electricity (the underlying capability). A
421 consumer accesses electricity generated (the service) via a wall outlet (service interface). In order
422 to use the electricity, a consumer needs to understand what type of plug to use, which voltage is
423 used and possible limits to the load(service description). The utility presumes that the customer
424 will only connect devices that are compatible with the voltage provided; and the consumer in turn
425 assumes that compatible devices can be connected without damage or harm (service
426 assumptions).

427 A residential or business user will need to open an account with the utility in order to use the
428 supply (service contract) and the utility will meter usage and expects the consumer to pay for use
429 at the rate prescribed (service contract). Provided that the consumer utilizes the correct plugs
430 and does not overload the system (service policy), the consumer can receive electricity using the
431 service.

432 Another person (say, a visitor to someone else's house) may use a contracted supply without any
433 relationship with the utility or any requirement to also satisfy the initial service constraint but
434 would nonetheless be expected to be compatible with the service interface.

435 In certain situations (for example, excessive demand), a utility may limit supply or institute rolling
436 blackouts (service policy). A consumer might lodge a formal complaint if this occurred frequently
437 (consumer's implied policy). In this example, the underlying capability would still exist and be
438 usable even if every device were required to be hard-wired to the utility's equipment, but this
439 would result in a very different service and service interface.

440 **3.2.3 Descriptions and Metadata**

441 One of the hallmarks of a Service Oriented Architecture is the degree of documentation and
442 description associated with it; particularly *machine process-able descriptions* – otherwise known
443 as *metadata*.

444 The purpose of this metadata is to facilitate integration, particularly across ownership domains.
445 By providing public descriptions, it makes it possible for potential participants to construct
446 applications that use services and even offer compatible services. Standardizing the formats of

Draft

447 such metadata reduces the cost and burden of producing the descriptions necessary to promote
448 reuse and integration.

449 3.2.3.1 The roles of description

450 An important additional benefit of metadata – as opposed to informal natural language
451 descriptions – is its potential to facilitate automated software development. Both service providers
452 and service consumers can benefit from such automation – reducing the cost of developing such
453 systems.

454 For example, metadata can be used as a basis of discovery in dynamic systems. Metadata can
455 assist in managing a service, validating and auditing usage of services which may also be
456 simplified by rich metadata. It can also help ensure that requirements and expectations
457 (regarding the content of any data interchanged) are properly interpreted and fulfilled.

458 3.2.3.2 The limits of description

459 There are well-known theoretic limits on the effectiveness of descriptions – it is simply not
460 possible to specify, completely and unambiguously the precise semantics of a service.

461 There will always be unstated assumptions made by the describer of a service that must be
462 implicitly shared by readers of the description. This applies to machine processable descriptions
463 as well as to human readable descriptions.

464 Fortunately, complete precision is not necessary either – what is required is sufficient precision to
465 enable required functionality.

466 Another kind of limit of service descriptions is more straightforward: whenever a repository is
467 searched using any kind of query there is always the potential for *zero or more* responses.

468 There may be many reasons why a multiplicity of responses is returned: there might be several
469 versions of the service, there might be competing services that offer overlapping functionality and
470 there might be services from multiple different providers.

471 In the case that there is more than one response, this set of responses has to be converted into a
472 choice of a single service in order for a service consumer to ensure the required function
473 performed. In a multi-provider scenario, that choice must also take into account real world
474 aspects of the service – such as whether the service consumer can identify the provider, can or
475 should trust the provider, and whether the provider is reliable and timely in delivering the service
476 offered. It is unlikely that all such factors can be easily and securely encoded in descriptions and
477 search criteria.

478 3.3 Interacting with services

479 Interacting with a service involves exchanging information with the service and performing actions
480 against the service. In many cases, this is accomplished by sending and receiving messages, but
481 there are other modes possible that do not involve explicit message transmission. However, for
482 simplicity, we often refer to message exchange as the primary mode of interaction with a service.
483 The forms of information exchanged and understood, together with the mechanisms used to
484 exchange information, constitute the **service interface** – see Section 3.2.2.4.

485 The key concepts that are important in understanding what it is involved in interacting with
486 services are the **data model**, the **process model**, the **execution context** and the **expectations**
487 about the interaction.

488 3.3.1 Data model

489 The data model of a service is a characterization of the information associated with the use of the
490 service.

491 The scope of the data model includes the format of exchanged information, the structural
492 relationships within those documents and the definition of terms used. Typically, only information

Draft

493 about, and data potentially included in, an exchange with a service are generally considered as
494 being part of that service's data model.

495 There are two important aspects of a data model that are important in interpreting information
496 exchange – the structure of the information and the meaning assigned to elements of the
497 information. Particularly for information that is exchanged across an ownership boundary, the
498 interpretation of strings and other tokens in the information is a critical part of the semantics of the
499 interaction.

500 3.3.1.1 Structure

501 Understanding the representation, structure and form of information exchanged is a key initial
502 step in ensuring effective interactions with a service. There are several levels of such structural
503 information; ranging from the encoding of character data, through the use of formats such as
504 XML, SOAP and schema-based representations.

505 A described data model typically has a great deal to say about the form of messages, about the
506 types of the various components of messages and so on. However, pure “typed” information is
507 not sufficient to completely describe the appropriate interpretation of data.

508 3.3.1.2 Semantics and Ontology

509 The primary task of any communication infrastructure is to facilitate the exchange of information
510 and the exchange of intent. For example, a purchase order combines two somewhat orthogonal
511 aspects: the description of the items being purchased and the fact that one party intends to
512 purchase those items from another party. Even if for exchanges that do not cross any ownership
513 boundaries, exchanges with services have similar aspects: this is an update to the customer
514 profile with these changes.

515 Especially in the case where the exchanges are across ownership boundaries, a critical issue is
516 the interpretation of the data. This interpretation must be consistent between the participants in
517 the service interaction. Consistent interpretation is a stronger requirement than merely type (or
518 structural) consistency – the tokens in the data itself must also have a shared basis.

519 For example, there is often a huge potential for variability in representing street addresses. For
520 example, an address in San Francisco, California may have variations in the way the city is
521 represented: SF, San Francisco, San Fran, the City by the Bay are all alternate denotations of the
522 same city. For successful exchange of address information, all the participants must have a
523 consistent view of the meaning of the address tokens if address information is to be reliably
524 shared.

525 An ontology is a formal description of terms and the relationships between them in a given
526 context. It will include information about how terms should be interpreted within a given context,
527 constraints on and functions of valid values for the data and associated properties, as well as
528 information about possible relationships of some terms to other terms (hierarchical, class/sub-
529 class, associative, dependent, etc.).

530 The role of explicit ontologies in an SOA is to provide a firm basis for selecting correct
531 interpretations for elements of information exchanged. For example, an ontology can be used to
532 capture the alternate ways of expressing the name of a city as well as distinguishing a city name
533 from a street name.

534 Ontologies also provide a point of context to facilitate the *reinterpretation* of data – for example
535 that a 3/8” steel washer may be a potential replacement for a 1cm spacer. Such a reinterpretation
536 is effectively represented as a particular traversal of the graph of concepts and relationships
537 embodied in the ontology. How much automation of ontology walking is appropriate will depend
538 on the nature of the service and the service participants.

539 Note that, for the most part, it is not expected that service consumers and providers would
540 actually exchange ontologies in their interaction – the role of the ontology is a background one – it

Draft

541 facilitates sound interactions. Hence ontology references are mostly to be found in **service**
542 **descriptions**.

543 More specifically, and in order for a service to be consistent, the service should make consistent
544 use of terms as defined in an ontology. Specific domain semantics are beyond the scope of this
545 reference model; but there is a requirement that the service interface enable providers and
546 consumers to identify unambiguously those definitions that are relevant to their respective
547 domains.

548 3.3.2 Behavioral model

549 The second key requirement for successful interactions with services is knowledge of the process
550 or temporal aspects of interacting with the service. Loosely, this can be characterized as
551 knowledge of the actions on, responses to and temporal dependencies between actions on the
552 service.

553 For example, in a security-controlled access to a database service, the actions available to a
554 service consumer might include presenting credentials, requesting database updates and reading
555 results of queries. The security may be based on a challenge-response protocol. For example,
556 the initiator presents an initial token of identity, the responder presents a challenge and the
557 initiator responds to the challenge in a way that satisfies the service. Only after the user's
558 credentials have been verified will any action that queries and/or updates the database be
559 accepted. The sequences of actions involved are a critical aspect of the knowledge required for
560 successful use of the secured database service.

561 There are other aspects of the behavior of services that are important. These include, for
562 example, whether the service is transactional, idempotent or long running. As a particular
563 example, a service that supports updating an account balance with a transaction is typically
564 idempotent; i.e., the state of the account would not be affected should a subsequent interaction
565 be attempted for the same transaction.

566 3.3.2.1 Action model

567 The **action** model of a service is about the individual actions that may be invoked against the
568 service. Of course, a great portion of the behavior resulting from an action may be private;
569 however, the expected public view of a service surely includes the implied effects of actions.

570 For example, in a service managing a bank account, it is not sufficient to know that you need to
571 exchange a given message (with appropriate authentication tokens), in order to use the service. It
572 is also necessary to understand that using the service may actually affect the state of the account
573 (for example, withdrawing cash); that dependencies are involved (for example, a withdrawal
574 request must follow not precede an authentication); or that the data changes made have different
575 value in different contexts (for example, changing the data in a bank statement is not the same as
576 changing the actual data representing the amount in an account).

577 3.3.2.2 Process Model

578 The **process model** characterizes the temporal relationships between actions and events
579 associated with interacting with the service. It is fairly common to partition the process model
580 associated with a service into two levels: the particular sequences of operations needed to
581 achieve single service exchanges and longer term transactions. These two levels may be nested
582 – a long running transaction is often composed of sequences of exchange patterns.

583 Note that although the process model is an essential part of this Reference Model, its extent is
584 not completely defined. In some architectures the process model will include aspects that are not
585 strictly part of SOA – for example, in this reference model we do not address the orchestration of
586 multiple services – although orchestration and choreography may be part of the process model of
587 a given architecture. At a minimum, the process model must cover the interactions with the
588 service itself.

Draft

589 3.3.2.3 Higher-order attributes of processes

590 Beyond the straightforward mechanics of interacting with a service there are other, higher-order
591 attributes of services' process models that are also often important. These can include whether
592 the service is **idempotent**, whether the service is **long-running** in nature and whether it is
593 important to account for any **transactional** aspects of the service.

594 A service is idempotent if subsequent attempts to perform identical transactions are discounted.
595 For example, it often important that a bank will only cash a check once – subsequent attempts to
596 cash the same check should be ignored, rejected or initiate an alert process. Note that
597 idempotency is not the same as effect-free or stateless: a service that always returns the same
598 results is idempotent, but only by virtue of the fact that it does not change from invocation to
599 invocation.

600 Idempotency is an important attribute of a service in an environment where there is a significant
601 possibility that the interaction between the service provider and consumer may be interrupted –
602 whether by a network issue or simply one of the parties dropping out. A strategy for recovering
603 from such a breakdown is to attempt to repeat the interaction – an idempotent service is required
604 to ignore such repetitions should the transaction have been completed beforehand.

605 A service is long-running if the activities engendered by an interaction are likely to persist beyond
606 the immediate interaction itself. For example, a classic book selling service might be viewed as a
607 long-running service: the activity started by the purchase of the book may take days or weeks to
608 complete. It can be important to account for a long-running process as it has implications for the
609 kinds of infrastructure needed – both by the service provider and by the service consumer – in
610 order to be able to keep track of the progress of the interaction.

611 Often, once a business-level contract has been agreed on, it can be difficult or impossible to
612 simply cancel the consequences of the agreement. This is particularly an issue when the
613 agreement of several parties is necessary simultaneously. For example, booking a vacation may
614 require a flight ticket as well as a hotel room – without either component the result is not a
615 vacation. However, the airline typically will not have a relationship with the hotel. If there are no
616 hotel rooms available for the proposed vacation then the airline ticket will need to be canceled.

617 The process of reversing a previously completed transaction – backing out of the airline booking
618 for example – is likely to be quite different to the process for the original transaction; possibly
619 even involving a different service. Knowledge of such compensatory actions is a key aspect of
620 interacting with transactional services.

621 3.3.3 Actualized Services

622 The **execution context** of a service interaction is the set of infrastructure elements, process
623 entities and policy assertions that are deployed as part of the instantiated service interaction. In
624 effect, the execution context defines the point of contact between abstractions such as service
625 descriptions which are mostly about the potential for interaction and an actually executing service.
626 It is the point of measurement between the service description and reality, between theory and
627 practice.

628 The execution context is not limited to one side of the interaction; rather it concerns the totality of
629 the interaction – including the service provider, the service consumer and the common
630 infrastructure needed to mediate the interaction.

631 The execution context is central to many aspects of a service interaction. It defines, for example,
632 the decision point for any policy enforcement relating to the service interaction. Note that a policy
633 decision point is not necessarily the same as an enforcement point: an execution context is not by
634 itself something that lends itself to enforcement. On the other hand, any enforcement mechanism
635 of a policy is likely to take into account the particulars of the actual service interaction.

636 The execution context also allows us to distinguish services from one another. Different instances
637 of the same service – denoting interactions between a given service provider and different service

Draft

638 consumers for example – are distinguished by virtue of the fact that their execution contexts are
639 different.
640 Finally, the execution context is also the context in which the interpretation of data that is
641 exchanged takes place – it is where the *symbol grounding* happens as it were. A particular string
642 has a particular meaning in a service interaction in a particular context – the execution context.

643 3.4 Real World Effect

644 There is always a particular purpose associated with interacting with a service. Conversely, a
645 service provider (and consumer) often has a priori conditions that apply to its interactions. The
646 service consumer is trying to achieve some result by interacting with the service, as is the service
647 provider. At first sight, such a goal can often be expressed as “trying to get the service to do
648 something”. This is sometimes known as the **real world effect** of using a service. For example,
649 an airline reservation service can be used in order to book travel – the desired real world effect
650 being a seat on the right airplane.

651 The internal actions that a service providers and consumers perform as a result of participation in
652 service interactions are, by definition, private and fundamentally unknowable.¹ By unknowable we
653 mean both that external parties cannot see others’ private actions and, furthermore, should not
654 have explicit knowledge of them. Instead we focus on the state that is shared between the parties
655 – the **shared state**. Actions by service providers and consumers lead to modifications of this
656 shared state; and that in turn leads to modified **expectations** by the participants.

657 For example, when an airline has confirmed a seat for a passenger on a flight this represents a
658 fact that both the airline and the passenger share – it is part of their shared state. Thus the real
659 world effect of booking the flight is the modification of this shared state – the creation of the fact of
660 the booking. Flowing from the shared facts, both the passenger, the airline and interested third
661 parties may make inferences – for example, when the passenger arrives at the airport the airline
662 confirms the booking and permits the passenger onto the airplane (subject of course to the
663 passenger meeting the other requirements for traveling).

664 For the airline to know that the seat is confirmed it will likely require some private action to record
665 the reservation. By minimizing assumptions about how the airline fulfils its contracts, the potential
666 for smooth interoperation is maximized. Such minimization principles represent a key success
667 factor for scalability.

668 Note that there does not need to be a third party to act as a kind of *escrow* for the shared state
669 between service providers and consumers. The elements of the shared state are inferred from
670 the communication that has occurred between the participants together with other context as
671 necessary. Of course, in the case where adjudication is a possibility, it becomes prudent to *record*
672 the interaction – so that disputes can be arbitrated.

673 Although there is not necessarily a one-to-one correspondence, the natural container for the
674 conditions applying to a service is the **service policy**. Similarly, the natural container for the
675 expectations arising from a service is the **service contract**.

676 3.5 Policies and Contracts

677 A **policy** represents some constraint or condition on the use, deployment or description of an
678 owned entity as defined by any participant. A **contract**, on the other hand, represents an
679 agreement by two or more parties. Like policies, agreements are also about the conditions of use
680 of a service; they may also constrain the expected real world effects of using a service. The
681 reference model is focused primarily on the concept of policies and contracts as they apply to
682 services. We are not concerned with the form or expressiveness of any language used to
683 express policies and contracts.

¹ A similar analysis applies to service consumers: just how a consumer of a service decides which requests to make and which actions to perform is something that the service provider cannot determine.

Draft

684 3.5.1 Service Policy

685 A policy is a statement of the obligations, constraints or other conditions of use of a given service
686 that expresses intent on the part of a participant. More particularly, policies are a way for
687 expressing the relationship between the **execution context** and the **data and behavior models**
688 associated with the service.

689 Conceptually, there are three aspects of policies: the policy assertion, the policy owner
690 (sometimes referred to as the policy subject) and policy enforcement.

691 For example, the assertion: “All messages are triple-DES encrypted” is an assertion regarding the
692 forms of messages. As an assertion, it is measurable: it may be true or false depending on
693 whether the traffic is encrypted or not. Policy assertions are often about the way the service is
694 realized; i.e., they are about the relationship between the service and its execution context.

695 A policy always represents a participant’s point of view. An assertion becomes the policy of a
696 participant when they make it their policy. This linking is normally not part of the assertion itself.
697 For example, if the service consumer declares that “All messages are triple-DES encrypted”, then
698 that reflects the policy of the service consumer. This policy is one that may be asserted by the
699 service consumer independently of any agreement from the service provider.

700 Finally, a policy may be enforced. Techniques for the enforcement of policies depend on the
701 nature of the policy. Conceptually, service policy enforcement amounts to ensuring that the policy
702 assertion is consistent with the real world. This might mean preventing unauthorized actions to be
703 performed or states to be entered into; it can also mean initiating compensatory actions when a
704 policy violation has been detected. An unenforceable constraint is not a policy; it would be better
705 described as a wish.

706 Policies potentially apply to many aspects of SOA: security, privacy, manageability, Quality of
707 Service and so on. Beyond such infrastructure-oriented policies, participants may also express
708 business-oriented policies – such as hours of business, return policies and so on.

709 Policy assertions should be written in a form that is understandable to, and processable by, the
710 parties to whom the policy is directed. Policies may need to be automatically interpreted,
711 depending on the purpose and applicability of the policy and whether it might affect whether a
712 particular service is used or not.

713 A natural point of contact between service participants and policies associated with the service is
714 in the service description – see Section 3.2.2. It would be natural for the service description to
715 contain references to the policies associated with the service.

716 3.5.2 Service Contract

717 Where a policy is associated with the point of view of individual participants, a contract represents
718 an agreement between two or more participants. Like policies, contracts can cover a wide range
719 of aspects of services: quality of service agreements, interface and choreography agreements
720 and commercial agreements.

721 Thus, following the analysis above, a service contract is a measurable assertion that governs the
722 requirements and expectations of two or more parties. Unlike policy enforcement, which is
723 usually the responsibility of the policy owner, contract enforcement may involve resolving
724 disputes between the parties to the contract. The resolution of such disputes may involve appeals
725 to higher authorities.

726 Like policies, contracts may be expressed in a form that permits automated interpretation. Where
727 a contract is used to codify the results of a service interaction, it is good practice to represent it in
728 a machine processable form. This facilitates automatic service composition, for example. Where
729 a contract is used to describe over-arching agreements between service providers and
730 consumers, then the priority is likely to make such contracts readable by people.

Draft

731 3.6 Visibility

732 For a service provider and consumer to interact with each other they have to be able to 'see' each
733 other. This is true for any consumer/provider relationship – including in an application program
734 where one program calls another: without the proper libraries being present the function call
735 cannot complete. In the case of SOA visibility needs to be emphasized because it is not
736 necessarily obvious how service participants *can* see each other to interact.

737 Visibility is the relationship between service consumers and providers that is satisfied when they
738 are able to interact with each other. Preconditions to visibility are awareness – typically the
739 initiator in a service interaction must be aware of the other parties – willingness – the parties must
740 be predisposed to interactions – and ability – the participants must be able to exchange
741 information as part of a service interaction.

742 3.6.1 Awareness

743 A key aspect of visibility is awareness – both the service provider and the service consumer must
744 have information that would lead them to know of the other's existence. Technically, the prime
745 requirement is that the *initiator* of a service interaction has knowledge of the responder. The fact
746 of a successful initiation is often sufficient to inform the responder of the other's existence.

747 Awareness of service offerings is often mediated by various *discovery* mechanisms. For a service
748 consumer (say) to discover a service provider, the service provider must be capable of making
749 details of the service (notably service description and policies) available to potential consumers;
750 and consumers must be capable of finding that information.

751 Service discoverability requires that the **service description** and **policy** – or at least a suitable
752 subset thereof – be available in such a manner and form that, directly or indirectly, an awareness
753 of the existence and capabilities of the service can become known to potential consumers. The
754 extent to which the discovery is “pushed” by the service provider, “pulled” by a potential
755 consumer, subject to a probe or another method, will depend on many factors.

756 For example, a service provider may advertise and promote their service by either including it in a
757 service directory or broadcasting it to all consumers; potential consumers may broadcast their
758 particular service needs in the hope that a suitable service responds with a proposal or offer or a
759 service consumer might also “probe()” an entire network to determine if suitable services exist.
760 When the demand for a service is higher than the supply, then by advertising their needs,
761 potential consumers are likely to be more effective than service providers advertising offered
762 services.

763 One way or another, the potential consumer must acquire a sufficient description to evaluate
764 whether the service matches their expectations and, if so, the method for the consumer to
765 establish a contract and invoke the service.

766 3.6.2 Willingness

767 Associated with all service interactions is intent – it is an intentional act to initiate and to
768 participate in a service interaction. For example, if a service consumer discovers a service via its
769 description in a registry, and the consumer initiates an interaction, if the service provider does not
770 cooperate then there can be no interaction. In some circumstances it is precisely the correct
771 behavior for a service to fail to respond – for example, it is the classic defense against certain
772 denial-of-service attacks.

773 The extent of a service participant's willingness to engage in service interactions may be the
774 subject of policies. Those policies may be documented in the service description.

775 Of course, willingness on the part of service providers and consumers to interact is not the same
776 as a willingness to perform requested actions. A service provider that rejects all attempts to
777 cause it to perform some action may still be fully willing and engaged in interacting with the
778 consumer.

Draft

779 **3.6.3 Reachability**

780 A service consumer may have the intention of interacting with a service, and may even have all
781 the information needed to communicate with it. However, if the service is not reachable, for
782 example if there is not communication path between the consumer and provider, then, effectively,
783 the service is not visible to the consumer.

784 Reachability is the relationship between service participants where they are able to exchange
785 information as part of service interactions. Reachability is closely connected to the concept of
786 **execution context** (see Section 3.3.3) – an important requirement for an execution context is to
787 establish that service participants can communicate with each other.

Draft

788

4 Conformance Guidelines

789 The authors of this reference model envision that architects may wish to declare their architecture
790 is conformant with this reference model. Conforming to a Reference Model is not generally an
791 easily automatable task – given that the Reference Model’s role is primarily to define concepts
792 that are important to SOA rather than to give guidelines for implementing systems.

793 However, we do expect that any given Service Oriented Architecture will reference the concepts
794 outlined in this specification. As such, we expect that any design for a system that adopts the
795 SOA approach will

- 796 • Have entities that can be identified as services as defined by this Reference Model,
- 797 • Such entities will have descriptions associated with them,
- 798 • Service entities will have identifiable interaction models, including models of the
799 information exchanged by the services and the temporal behavior of the services
- 800 • It should be possible to identify a means by which consumers of services and providers
801 of services are able to engage; and
- 802 • That there will be identifiable aspects of service entities that correspond to the policies
803 relating to the conditions of use of services and to the expectations that result from
804 interacting with services.

805 It is not appropriate for this specification to identify *best practices* with respect to building SOA-
806 based systems. However, the ease with which the above elements can be identified within a
807 given SOA-based system could have significant impact on the scalability, maintainability and
808 ease of use of the system.

Draft

809 **5 References**

810 **5.1 Normative**

811 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
812 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
813

814 **5.2 Non-Normative**

815 [W3C WSA] W3C Working Group Note "Web Services Architecture",
816 <http://www.w3.org/TR/ws-arch/> , 11 February 2004

Draft

817 Appendix A. Glossary

818 **EDITOR'S NOTE TO THE READER:** This section is currently in flux. Please do not submit
819 comments/issues on/against this appendix.

820

821 Terms that are used within this Reference Model are often also found in other specifications. In
822 order to avoid potential ambiguity, this glossary locally scopes the definitions of those terms for
823 the purpose of this Reference Model and thus overrides any other definitions.

824

825 Action Model

826 The characterization of the permissible actions that may be invoked against a service.

827

828 Addressability

829 A state of knowledge of a participant whereby information exists that could, in principle,
830 permit a participant to interact with the addressable party. Addressability does not imply
831 reachability.

832

833 Awareness

834 A state whereby one party has knowledge of the existence of the other party. Awareness
835 does not imply addressability or reachability.

836

837 Architecture

838 A set of artifacts (that is: principles, guidelines, policies, models, standards and
839 processes) and the relationships between these artifacts, that guide the selection,
840 creation, and implementation of solutions aligned with business goals.

841 Software architecture is the structure or structures of an information system consisting of
842 entities and their externally visible properties, and the relationships among them.

843

844 Authentication

845 The act by which one entity establishes – to an agreed level of confidence – the identity
846 of another.

847

848 Awareness

849 Information that leads a service provider and/or consumer to be able to act on knowledge
850 of the other's existence.

851

852 Behavioral Model

853 The characterization of (and responses to, and temporal dependencies between) the
854 actions on a service.

855

856 Capability

857 A real-world effect that a service provider is able to provide to a service consumer.

Draft

858

859 (Service) Consumer

860 An entity which seeks to satisfy a particular need through the use capabilities offered by
861 means of a service.

862

863 Contract

864 The agreement between a service provider and a consumer, often including conditions of
865 use of a service and an indication of the expected real world effect.

866

867 Data Model

868 The characterization of the information that is associated with the use of a service.

869

870 Discoverability

871 The possibility that service consumers and service providers can be brought together,
872 and the mechanisms by which this is achieved.

873

874 Execution context

875 A set of technical and business elements that permit information to be exchanged and
876 actions to be performed when the “theory” of a service description, policies and contract
877 become the “practice” of an actual running service.

878

879 Framework

880 A set of assumptions, concepts, values, and practices that constitutes a way of viewing
881 the current environment.

882

883 Idempotency/Idempotent

884 A characteristic of a service whereby multiple attempts to change a state will always and
885 only generate a single change of state if the operation has been already been
886 successfully completed once.

887

888 Interaction

889 The activity involved in making using of a capability offered, usually across an ownership
890 boundary, in order to achieve a particular desired real-world effect.

891

892 Interface

893 The means by which the underlying capabilities of a service are accessed.

894

895 Message

896 A serialized set of data that is used to convey information and/or actions from one party
897 to another.

898

Draft

899 Metadata

900 A set of properties of a given entity which are intended to describe and/or indicate the
901 nature and characteristics of the entity.

902

903 Offer

904 An invitation to use the capabilities made available by a service provider in accordance
905 with some set of policies.

906

907 Ontology

908 A formal description of terms and the relationships between them in a given context.

909

910 Opaqueness

911 The extent to which an agent is able to interact successfully with a service without
912 detecting how the service is implemented.

913

914 Policy

915 A statement of obligations, constraints or other conditions of use of an owned entity as
916 defined by a participant.

917

918 Process Model

919 The characterization of the temporal relationships between actions and events
920 associated with interacting with a service.

921

922 (Service) Provider

923 An entity (person or organization) that offers the use of capabilities by means of a service

924

925 Reachability

926 The state in which a service is visible to potential consumers and capable of being
927 interacted with.

928

929 Real world effect

930 The actual result of using a service, rather than merely the capability offered by a service
931 provider

932

933 Reference Model

934 A reference model is an abstract framework for understanding significant relationships
935 among the entities of some environment that enables the development of specific
936 architectures using consistent standards or specifications supporting that environment.

937 A reference model is based on a small number of unifying concepts. A reference model is
938 not directly tied to any standards, technologies or other concrete implementation details,
939 but it does seek to provide a common semantics that can be used unambiguously across
940 and between different implementations.

941

Draft

942 Semantics

943 A conceptualization of the implied meaning of information, shared between the service
944 consumer and the service provider, that requires words and/or symbols within a usage
945 context.

946

947 Service

948 The means by which the needs of a consumer are brought together with the capabilities
949 of a provider.

950

951 Service description

952 A set of information describing a service, sufficient to allow a potential consumer to
953 ascertain, where appropriate:

- 954 - the identity of (and/or information about) the service provider;
- 955 - the policies, parameters and terms of use of the service;
- 956 - of the information necessary to interact with the service;
- 957 - what is accomplished when the service is invoked,
- 958 - and thus be able to use the service as intended by the provider.

959

960 Service Oriented Architecture (SOA)

961 A software architecture of services, policies, practices and frameworks in which
962 components can be reused and repurposed rapidly in order to achieve shared and new
963 functionality. It provides a uniform means to offer, discover, interact with and use
964 capabilities to produce desired effects consistent with measurable preconditions and
965 expectations.

966

967 Visibility

968 The capacity for those with needs and those with capabilities to be able to interact with
969 each other.

Draft

970 **Appendix B. Acknowledgments**

971 The following individuals were members of the committee during the development of this
972 specification:

973 [TODO: insert cte. Members]

974

Draft

975 **Appendix C. Notices**

976 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
977 that might be claimed to pertain to the implementation or use of the technology described in this
978 document or the extent to which any license under such rights might or might not be available;
979 neither does it represent that it has made any effort to identify any such rights. Information on
980 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
981 website. Copies of claims of rights made available for publication and any assurances of licenses
982 to be made available, or the result of an attempt made to obtain a general license or permission
983 for the use of such proprietary rights by implementers or users of this specification, can be
984 obtained from the OASIS Executive Director.

985 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
986 applications, or other proprietary rights, which may cover technology that may be required to
987 implement this specification. Please address the information to the OASIS Executive Director.

988 Copyright © OASIS Open 2005. *All Rights Reserved.*

989 This document and translations of it may be copied and furnished to others, and derivative works
990 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
991 published and distributed, in whole or in part, without restriction of any kind, provided that the
992 above copyright notice and this paragraph are included on all such copies and derivative works.
993 However, this document itself does not be modified in any way, such as by removing the
994 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
995 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
996 Property Rights document must be followed, or as required to translate it into languages other
997 than English.

998 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
999 successors or assigns.

1000 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1001 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1002 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1003 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1004 PARTICULAR PURPOSE.