



TestCases for the SCA Web Service Binding Specification Version 1.1

Committee Draft 01 / Public Review 01

1 July 2010

Specification URIs:

This Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-testcases-cd01.html>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-testcases-cd01.odt>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-testcases-cd01.pdf>
(Authoritative)

Previous Version:

N/A

Latest Version:

<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-testcases.html>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-testcases.odt>
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-testcases.pdf> (Authoritative)

Technical Committee:

OASIS Service Component Architecture / Bindings (SCA-Bindings) TC
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=sca-bindings

Chair(s):

Simon Holdsworth, IBM

Editor(s):

Mike Edwards, IBM

Related Work:

This document is related to:

- Service Component Architecture Web Services Specification Version 1.1
<http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-spec-cd04.pdf>

Declared XML Namespace(s):

<http://docs.oasis-open.org/ns/opencsa/scatests/200903>
<http://docs.oasis-open.org/ns/opencsa/scatests/2009032>
<http://test.sca.oasisopen.org/>

Abstract:

This document defines the TestCases for the SCA Web Service Binding specification.

The TestCases represent a series of tests that an SCA runtime must pass in order to claim conformance to the requirements of the SCA Web Service Binding specification.

Status:

This document was last revised or approved by the OASIS Service Component Architecture / Bindings (SCA-Bindings) TC on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at <http://www.oasis-open.org/committees/sca-bindings/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (.<http://www.oasis-open.org/committees/sca-bindings/ipr.php>)

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/sca-bindings/>

Notices

Copyright © OASIS® 2010. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

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

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

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS" is a trademark of [OASIS](http://www.oasis-open.org), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

Table of Contents

1		
2	1 Introduction.....	6
3	1.1 TestCase Structure.....	6
4	1.1.1 Test Client.....	6
5	1.1.2 Test Application.....	6
6	1.1.3 Test Artifacts Organization.....	7
7	1.2 Namespaces and Java Package Names.....	8
8	1.2.1 SCA Artifact Namespaces.....	8
9	1.2.2 WSDL Namespace.....	8
10	1.2.3 Java Package name.....	8
11	1.3 Terminology.....	8
12	1.4 Normative References.....	8
13	2 TestCases.....	9
14	2.1 Section 2.....	9
15	2.1.1 BWS_2001_TestCase.....	9
16	2.1.2 BWS_2002_TestCase.....	9
17	2.1.3 BWS_2003_TestCase.....	10
18	2.1.4 BWS_2004_TestCase.....	10
19	2.1.5 BWS_2005_TestCase.....	11
20	2.1.6 BWS_2006_TestCase.....	11
21	2.1.7 BWS_2007_TestCase.....	12
22	2.1.8 BWS_2008_TestCase.....	12
23	2.1.9 BWS_2009_TestCase.....	13
24	2.1.10 BWS_2010_TestCase.....	13
25	2.1.11 BWS_2011_TestCase.....	14
26	2.1.12 BWS_2012_TestCase.....	14
27	2.1.13 BWS_2013_TestCase.....	15
28	2.1.14 BWS_2014_TestCase.....	16
29	2.1.15 BWS_2015_TestCase.....	16
30	2.1.16 BWS_2016_TestCase.....	17
31	2.1.17 BWS_2017_TestCase.....	17
32	2.1.18 BWS_2018_TestCase.....	18
33	2.1.19 BWS_2019_TestCase.....	18
34	2.1.20 BWS_2020_TestCase.....	19
35	2.1.21 BWS_2021_TestCase.....	19
36	2.1.22 BWS_2022_TestCase.....	20
37	2.1.23 BWS_2023_TestCase.....	20
38	2.2 Section 4.....	21
39	2.2.1 BWS_4001_TestCase.....	21
40	2.2.2 BWS_4002_TestCase.....	21
41	2.2.3 BWS_4003_TestCase.....	22
42	2.2.4 BWS_4004_TestCase.....	22
43	2.2.5 BWS_4005_TestCase.....	23
44	2.2.6 BWS_4006_TestCase.....	23
45	2.2.7 BWS_4007_TestCase.....	24
46	2.2.8 BWS_4008_TestCase.....	24
47	2.3 Section 5.....	25
48	2.3.1 BWS_5001_TestCase.....	25
49	2.3.2 BWS_5002_TestCase.....	25
50	2.3.3 BWS_5003_TestCase.....	26
51	2.3.4 BWS_5004_TestCase.....	27

52	2.3.5 BWS_5005_TestCase.....	27
53	2.3.6 BWS_5006_TestCase.....	28
54	3 Cross Mapping of Test Assertions to TestCases.....	29
55	4 Catalog of Test Artifacts.....	32
56	4.1 Composite Files - lower level.....	32
57	4.2 Java Interfaces.....	32
58	4.3 Java Implementation Classes.....	33
59	4.4 WSDL Interface Files.....	33
60	5 Conformance.....	37
61	Appendix A.Acknowledgments.....	38
62	Appendix B.Revision History.....	39
63		

1 Introduction

This document defines the testcases for the SCA Web Service Bindings specification.

The tests described in this document are related to the Test Assertions described in the [SCA Web Service Binding Test Assertions document \[WS-TA\]](#).

1.1 TestCase Structure

The SCA Bindings testcases follow a standard structure. They are divided into two main parts:

1. Test Client, which drives the test and checks that the results are as expected
2. Test Application, which forms the bulk of the testcase and which consists of Composites, WSDL files, XSDs and code artifacts such as Java classes, organized into a series of SCA contributions

The basic idea is that the Test Application runs on the SCA runtime that is under test, while the Test Client runs as a standalone application, invoking the Test Application through one or more service interfaces.

1.1.1 Test Client

The test client is designed as a standalone application. The version built here is a Java application which uses the JUnit test framework, although in principle, the client could be built using another implementation technology.

The test client is structured to contain configuration information about the testcase, which consists of:

1. metadata identifying the Test Application in terms of the SCA Contributions that are used and the Composites that must be deployed and run
2. data indicating which service operation(s) must be invoked with input data and expected output data (including exceptions for expected failure cases)

The Java test client consists of a base runtime class, BaseJAXWSTestCase.java. Each actual testcase is implemented by a small class which extends the base runtime class. The bulk of the code required to run a test is held in the base runtime class. The small testcase class contains the configuration for the specific test, which it provides to the code in the base runtime class through a standard interface.

The Java test client base runtime class is structured so that there is a replaceable class called the RuntimeBridge, which is used to communicate with the SCA runtime under test, for the purposes of deploying and running the test application. Each SCA runtime provider can produce a version of this class. The code within the runtime bridge is likely to be highly proprietary and specific to the SCA runtime for which it is written. Which runtime bridge class is used at runtime is controlled by an environment variable or system variable with the name "OASIS_TESTENV_RUNTIME_BRIDGE_CLASS", which is read by the code in BaseJAXWSTestCase.

The Test Client defaults to using Web services to communicate with the test application.

1.1.2 Test Application

Each Test Application consists of one top level SCA Composite file and one or more other SCA Composite files and their associated artifacts (implementations, interface files), plus test client invocation application described above.

A typical test application has a design where the top level composite offers a single service to the client application over a Web services binding. The top level composite contains one component which offers

103 the service that is used by the client application. The top level composite then contains one or more other
104 components which are used by the first component.

105 All of the components in the top level composite are implemented by composites. These second level
106 composites then contain typically one component, implemented using a specific technology such as Java
107 POJO or a BPEL process. In some cases the implementation may be a third level composite.

108 The application is structured so that alternative technologies can be used. For example, replacing the
109 contents of the second-level or third-level composites allows different implementation technologies to be
110 tested – eg BPEL scripts or C++ classes may be used instead of Java classes. Similarly, the binding used
111 to connect from the top level composite to the client application may be changed from Web services to
112 JMS if required, simply by changing the binding on the <service/> of the top level composite.

113 Which implementation language to use for test artifacts is controlled by a system variable or environment
114 variable which is read by the test client application, with the name "OASIS_TESTENV_IMPL_LANG". This
115 variable can have one of the following values:

- 116 • "Java" - for Java implementations
- 117 • "BPEL" - for WS_BPEL process implementations
- 118 • "CPP" - for C++ implementations

119 The testcases are designed so that the range of implementation types can be expanded

120 **1.1.3 Test Artifacts Organization**

121 Note that the design of these testcases promotes reuse of artifacts between testcases, so that many
122 testcases share components. For example, components implementing simple invocable services are all
123 implemented using a single parameterized implementation artifact.

124 All the test artifacts are contained in a number of Contributions, which are simply filesystem directories
125 which are all peers in the filesystem hierarchy. The names of the directories are the names of the
126 Contributions and the names are significant. The names of Contributions containing implementation type
127 specific artifacts (such as Java classes) are also specially structured to allow for replacement of one type
128 of implementation artifact with another.

129 Broadly, Contribution names are as follows:

- 130 • BWS_nnnn - a contribution that is specific for a particular testcase, where "nnnn" is the number of
131 the testcase. Often this is required because a particular testcase involves artifacts that contain
132 errors that are statically checkable - an SCA runtime is permitted to reject such artifacts when
133 they are contributed and deployed and it is important to ensure that contributions containing
134 deliberate errors for one testcase do not interfere with the operation of other testcases.
- 135 • BWS_nnnn_Java - a contribution for a specific testcase where there is a need for language
136 specific artifacts that relate to that testcase alone
- 137 • BWS_General - a shared contribution containing implementation type independent artifacts that
138 can be used by many testcases.
- 139 • BWS_General_Java - a shared contribution containing implementation type dependent artifacts
140 for Java POJOs. These artifacts can include both Java classes and also SCA composites that
141 directly use Java classes.
- 142 • Contribution1, Contribution2, etc - contributions that are used by various testcases that are testing
143 the handling of SCA contributions

144 Note that the names of Contributions containing implementation specific artifacts ends with a name that is
145 specific to the implementation type - so "_Java" is used for Java implementations, "_BPEL" is used for
146 BPEL implementations, "_CPP" is used for C++ implementations (and so on). Note that the name

147 following the underscore matches the name used in the "OASIS_TESTENV_IMPL_LANG" variable used
148 to control execution of the test client. The concept is that where there is an implementation type specific
149 contribution, each implementation type must provide its own versions of the same basic artifacts.
150 Typically, this means that each contribution must contain the same set of Composites, but that the
151 implementation type dependent artifacts that these composites use will differ from implementation type to
152 implementation type.

153 Basically, the setting of the variable is used to select the suffix used for implementation type dependent
154 contributions. If the variable is set to "Java" then the contribution "General_Java" is selected, whereas if
155 the variable is set to "BPEL", the contribution "General_BPEL" is selected.

156 **1.2 Namespaces and Java Package Names**

157 The SCA Web Service Binding testcase suite makes use of some XML namespaces and Java package
158 names, as follows:

159 **1.2.1 SCA Artifact Namespaces**

160 These apply to artifacts such as Composites

161 <http://docs.oasis-open.org/ns/opencsa/scatests/200903>

162 <http://docs.oasis-open.org/ns/opencsa/scatests/2009032>

163 **1.2.2 WSDL Namespace**

164 <http://test.sca.oasisopen.org/>

165 **1.2.3 Java Package name**

166 For Java interface classes and for Java implementation classes

167 `org.oasisopen.sca.test`

168 **1.3 Terminology**

169 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
170 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as
171 described in IETF RFC 2119 [RFC 2119]

172 **1.4 Normative References**

- 173 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
174 RFC 2119, March 1997.
175 <http://www.ietf.org/rfc/rfc2119.txt>.
- 176 **[SCA-WS-TA]** OASIS Committee Draft 01, "Test Assertions for the SCA Web Service Binding
177 Version 1.1 Specification", July 2010.
178 <http://docs.oasis-open.org/sca-bindings/sca-wsbinding-1.1-test-assertions-cd01.pdf>
- 180 **[SCA-WSBinding]** OASIS Committee Draft 04, "Service Component Architecture Web Service
181 Binding Specification Version 1.1," May 2010.
182 <http://docs.oasis-open.org/opencsa/sca-bindings/sca-wsbinding-1.1-spec-cd04.pdf>
183

184 **2 TestCases**

185 **2.1 Section 2**

186 **2.1.1 BWS_2001_TestCase**

187

Testcase ID	BWS_2001_TestCase
Test Assertion	BWS-TA-20001
Description	Tests that the value of the @uri attribute of a <binding.ws/> subelement of a <component/> <reference/> element is an absolute URI.
Artifacts	BWS_2001_TestCase.java Test_BWS_2001.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl
Expected output	Negative test: "exception"

188

189 **2.1.2 BWS_2002_TestCase**

190

Testcase ID	BWS_2002_TestCase
Test Assertion	BWS-TA-20002
Description	Tests that a binding.ws/@wsdlElement attribute value points to an existing WSDL 1.1 element
Artifacts	BWS_2002_TestCase.java Test_BWS_2002.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl
Expected output	Negative test:

	"exception"
--	-------------

191

192 2.1.3 BWS_2003_TestCase

193

Testcase ID	BWS_2003_TestCase
Test Assertion	BWS-TA-20003
Description	Tests that a binding.ws/@wsdlElement attribute of a <service/> element of a <component/> does not point to a WSDL 1.1 service element
Artifacts	BWS_2003_TestCase.java Test_BWS_2003.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl
Expected output	Negative test: "exception"

194

195 2.1.4 BWS_2004_TestCase

196

Testcase ID	BWS_2004_TestCase
Test Assertion	BWS-TA-20004
Description	Tests that where a <component/> <reference/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL service element, that the set of available WSDL ports is not empty.
Artifacts	BWS_2004_TestCase.java Test_BWS_2004.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl
Expected output	Positive test: "BWS_2004 request service1 operation1 invoked service2 operation1 invoked"

197

198

2.1.5 BWS_2005_TestCase

199

Testcase ID	BWS_2005_TestCase
Test Assertion	BWS-TA-20005 BWS_TA_20010
Description	Tests that where a <component/> <reference/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL service element, and the WSDL service element does not have any ports that are supported by the SCA runtime, the SCA runtime raises an error.
Artifacts	BWS_2005_TestCase.java Test_BWS_2005.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl Service1InvalidService.wsdl in contribution BWS_2005 sca-variables.dtd in contribution BWS_2005
Expected output	Negative test: "exception"

200

201

2.1.6 BWS_2006_TestCase

202

Testcase ID	BWS_2006_TestCase
Test Assertion	BWS-TA-20006
Description	Tests that where a <component/> <reference/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL service element, and the WSDL service element has 2 or more WSDL Ports available, that the SCA Runtime uses exactly one port for each invocation through the reference
Artifacts	BWS_2006_TestCase.java Test_BWS_2006.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite

	TestComposite4.composite Service1.wsdl
Expected output	Positive test - 2 alternative outcomes: 1) "BWS_2006 request service1 operation1 invoked service2 operation1 invoked" 2) "BWS_2006 request service1 operation1 invoked service3 operation1 invoked"

203

204

2.1.7 BWS_2007_TestCase

205

Testcase ID	BWS_2007_TestCase
Test Assertion	BWS-TA-20007
Description	Tests that where a <component/> <reference/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL port element, that the WSDL portType element associated with the WSDL port element is compatible with the service interface.
Artifacts	BWS_2007_TestCase.java Test_BWS_2007.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl Service1InvalidService.wsdl
Expected output	Negative test: "exception"

206

207

2.1.8 BWS_2008_TestCase

208

Testcase ID	BWS_2008_TestCase
Test Assertion	BWS-TA-20008
Description	Tests that where a <component/> <service/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL port element, that the endpoint specified by the port element is exposed by the SCA runtime
Artifacts	BWS_2008_TestCase.java Test_BWS_2008.composite

	TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl Service1WithPort.wsdl
Expected output	Positive test: "BWS_2008 request service1 operation1 invoked service2 operation1 invoked"

209

210 **2.1.9 BWS_2009_TestCase**

211

Testcase ID	BWS_2009_TestCase
Test Assertion	BWS-TA-20009
Description	Tests that where a <component/> <reference/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL port element, that the portType specified by the WSDL port is a compatible superset of the interface on the reference
Artifacts	BWS_2009_TestCase.java Test_BWS_2009.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite9.composite Service1.wsdl Service1Superset.wsdl sca_variables.dtd in BWS_2009 contribution
Expected output	Positive test: "BWS_2009 request service1 operation1 invoked service2 operation1 invoked"

212

213 **2.1.10 BWS_2010_TestCase**

214

Testcase ID	BWS_2010_TestCase
Test Assertion	BWS-TA-20011

Description	Tests that where a <component/> <service/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL binding element, that the portType associated with the WSDL port is compatible with the interface on the service
Artifacts	BWS_2010_TestCase.java Test_BWS_2010.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite Service1.wsdl Service1WithBinding.wsdl
Expected output	Positive test: "BWS_2010 request service1 operation1 invoked service2 operation1 invoked"

215

216 **2.1.11 BWS_2011_TestCase**

217

Testcase ID	BWS_2011_TestCase
Test Assertion	BWS-TA-20012
Description	Tests that where a <component/> <service/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL binding element and the WSDL binding element is not supported by the SCA runtime, an error is raised
Artifacts	BWS_2011_TestCase.java Test_BWS_2011.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite Service1.wsdl Service1WithBinding.wsdl
Expected output	Negative test: "exception"

218

219 **2.1.12 BWS_2012_TestCase**

220

Testcase ID	BWS_2012_TestCase
-------------	-------------------

Test Assertion	BWS-TA-20013 BWS-TA-20015 (this testcase uses the @uri to identify the target service and so covers that aspect of that test assertion)
Description	Tests that where a <component/> <reference/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL binding element, that the portType specified by the WSDL binding is a compatible superset of the interface on the reference
Artifacts	BWS_2012_TestCase.java Test_BWS_2012.composite TestInvocation.wsdl TestClient_0002.composite TestComposite4.composite TestComposite9.composite Service1.wsdl Service1Superset.wsdl in BWS_2012 contribution sca_variables.dtd in BWS_2012 contribution
Expected output	Positive test: "BWS_2012 request service1 operation1 invoked service2 operation1 invoked"

221

222

2.1.13 BWS_2013_TestCase

223

Testcase ID	BWS_2013_TestCase
Test Assertion	BWS-TA-20014
Description	Tests that where a <component/> <reference/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL binding element, and the WSDL binding element is not supported by the SCA runtime, the SCA runtime raises an error.
Artifacts	BWS_2013_TestCase.java Test_BWS_2013.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl Service1InvalidService.wsdl in contribution BWS_2013 sca-variables.dtd in contribution BWS_2013

Expected output	Negative test: "exception"
-----------------	-------------------------------

224

2.1.14 BWS_2014_TestCase

225

226

Testcase ID	BWS_2014_TestCase
Test Assertion	BWS-TA-20015 (this testcase uses the EndpointReference element to identify the target service and so covers that aspect of that test assertion)
Description	Tests that where a <component/> <reference/> has a <binding.ws/> element with @wsdlElement attribute referencing a WSDL binding element, that when the endpoint address of the target service is supplied as an <wsa:EndpointReference/> element, the service can be successfully invoked
Artifacts	BWS_2014_TestCase.java Test_BWS_2014.composite TestInvocation.wsdl TestClient_0002.composite TestComposite4.composite TestComposite9.composite Service1.wsdl Service1Superset.wsdl in BWS_2014 contribution sca_variables.dtd in BWS_2014 contribution
Expected output	Positive test: "BWS_2014 request service1 operation1 invoked service2 operation1 invoked"

227

2.1.15 BWS_2015_TestCase

228

229

Testcase ID	BWS_2015_TestCase
Test Assertion	BWS-TA-20016
Description	Tests that where a <binding.ws/> element has a @wsdl:wsdlLocation attribute specified that it also has a @wsdlElement attribute specified
Artifacts	BWS_2015_TestCase.java Test_BWS_2015.composite TestInvocation.wsdl TestClient_0002.composite

	TestComposite1.composite TestComposite4.composite Service1.wsdl Service1InvalidService.wsdl in contribution BWS_2015 sca-variables.dtd in contribution BWS_2015
Expected output	Negative test: "exception"

230

231 **2.1.16 BWS_2016_TestCase**

232

Testcase ID	BWS_2016_TestCase
Test Assertion	BWS-TA-20017
Description	Tests that where a <binding.ws/> element has a @wsdli:wSDLLocation attribute specified, that the value of the attribute points to a WSDL 1.1 document
Artifacts	BWS_2016_TestCase.java Test_BWS_2016.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl Service1InvalidService.wsdl in contribution BWS_2016 sca-variables.dtd in contribution BWS_2016
Expected output	Negative test: "exception"

233

234 **2.1.17 BWS_2017_TestCase**

235

Testcase ID	BWS_2017_TestCase
Test Assertion	BWS-TA-20018
Description	Tests that where a <binding.ws/> element has a @uri attribute referencing the target service it does not also have a wsa:EndpointReference subelement referencing the target service
Artifacts	BWS_2017_TestCase.java

	Test_BWS_2017.composite TestInvocation.wsdl TestClient_0002.composite TestComposite9.composite TestComposite4.composite Service1.wsdl Service1Superset.wsdl in contribution BWS_2017 sca-variables.dtd in contribution BWS_2017
Expected output	Negative test: "exception"

236

237 **2.1.18 BWS_2018_TestCase**

238

Testcase ID	BWS_2018_TestCase
Test Assertion	BWS-TA-20019
Description	Tests that where a <binding.ws/> element is a subelement of a <service> <callback> element, it has no @uri attribute specified
Artifacts	BWS_2018_TestCase.java Test_BWS_2018.composite TestInvocation.wsdl TestClient_0002.composite TestComposite54.composite TestComposite55.composite Service1.wsdl Service5.wsdl Service5Callback.wsdl
Expected output	Negative test: "exception"

239

240 **2.1.19 BWS_2019_TestCase**

241

Testcase ID	BWS_2019_TestCase
Test Assertion	BWS-TA-20023
Description	Tests that a <binding.ws/> element conforms to the schema defined in

	sca-binding-webservice-1.1.xsd
Artifacts	BWS_2019_TestCase.java Test_BWS_2019.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl
Expected output	Negative test: "exception"

242

2.1.20 BWS_2020_TestCase

243

244

Testcase ID	BWS_2020_TestCase
Test Assertion	BWS-TA-20024
Description	Tests that where a <binding.ws/> subelement of a component <reference/> does not have a target defined, that the SCA runtime raises an error
Artifacts	BWS_2020_TestCase.java Test_BWS_2020.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl
Expected output	Negative test: "exception"

245

2.1.21 BWS_2021_TestCase

246

247

Testcase ID	BWS_2021_TestCase
Test Assertion	BWS-TA-20025
Description	Tests that where a <binding.ws/> subelement of a component <reference/> has a valid target defined by means of its @uri attribute, that the SCA runtime uses that target address when the component invokes an operation of the reference

Artifacts	BWS_2021_TestCase.java Test_BWS_2021.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl
Expected output	Positive test: "BWS_2021 request service1 operation1 invoked service2 operation1 invoked"

248

249 **2.1.22 BWS_2022_TestCase**

250

Testcase ID	BWS_2022_TestCase
Test Assertion	BWS-TA-20030
Description	Tests that where a <binding.ws/> subelement of a component <reference/> has a @wsdlElement attribute which references a Port element of a WSDL document and the Port element has an associated portType which contains an sca @callback extension attribute, but where the <reference/> interface declaration does not have a callback, that the portType is treated as being not compatible with the interface for the reference and that an error is raised.
Artifacts	BWS_2022_TestCase.java Test_BWS_2022.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl Service1WithCallback.wsdl
Expected output	Negative test: "exception"

251

252 **2.1.23 BWS_2023_TestCase**

253

Testcase ID	BWS_2023_TestCase
-------------	-------------------

Test Assertion	BWS-TA-20033
Description	Tests that the <bindingType/> for binding.ws binding includes the SOAP.v1_1 intent in its @mayProvides or @alwaysProvides attributes
Artifacts	BWS_2023_TestCase.java Test_BWS_2023.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite TestComposite4.composite Service1.wsdl
Expected output	Positive test: "BWS_2023 request service1 operation1 invoked service2 operation1 invoked"

254

255 **2.2 Section 4**

256 **2.2.1 BWS_4001_TestCase**

Testcase ID	BWS_4001_TestCase
Test Assertion	BWS-TA-40001
Description	Tests that where a <component/> <service/> element has a <binding.ws/> subelement to which the SOAP intent is applied, that the service endpoint both sends and receives messages in a version of the SOAP protocol
Artifacts	BWS_4001_TestCase.java Test_BWS_4001.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl TestComposite1.composite
Expected output	Positive test: "BWS_4001 request service1 operation1"

257

258 **2.2.2 BWS_4002_TestCase**

Testcase ID	BWS_4002_TestCase
Test Assertion	BWS-TA-40002
Description	Tests that where a <component/> <service/> element has a <binding.ws/> subelement to which the SOAP.v1_1 intent is applied, that the service

	endpoint both sends and receives messages in the SOAP v1.1 protocol
Artifacts	BWS_4002_TestCase.java Test_BWS_4002.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl TestComposite1.composite
Expected output	Positive test: "BWS_4002 request service1 operation1"

259

260 **2.2.3 BWS_4003_TestCase**

Testcase ID	BWS_4003_TestCase
Test Assertion	BWS-TA-40003
Description	Tests that where a <component/> <service/> element has a <binding.ws/> subelement to which the SOAP.V1.2 intent is applied, that the service endpoint both sends and receives messages in the SOAP v1.2 protocol
Artifacts	BWS_4003_TestCase.java Test_BWS_4003.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl TestComposite1.composite
Expected output	Positive test: "BWS_4003 request service1 operation1"

261

262 **2.2.4 BWS_4004_TestCase**

Testcase ID	BWS_4004_TestCase
Test Assertion	BWS-TA-40001
Description	Tests that where a <component/> <reference/> element has a <binding.ws/> subelement to which the SOAP intent is applied, that the reference both sends and receives messages in a version of the SOAP protocol
Artifacts	BWS_4004_TestCase.java Test_BWS_4004.composite TestInvocation.wsdl TestClient_0002.composite

	Service1.wsdl TestComposite4.composite
Expected output	Positive test: "BWS_4004 request service1 operation1 invoked request Client service1 operation1 invoked"

263

264 **2.2.5 BWS_4005_TestCase**

Testcase ID	BWS_4005_TestCase
Test Assertion	BWS-TA-40002
Description	Tests that where a <component/> <reference/> element has a <binding.ws/> subelement to which the SOAP.v1_1 intent is applied, that the reference both sends and receives messages in the SOAP v1.1 protocol
Artifacts	BWS_4005_TestCase.java Test_BWS_4005.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl TestComposite4.composite
Expected output	Positive test: "BWS_4005 request service1 operation1 invoked request Client service1 operation1 invoked"

265

266 **2.2.6 BWS_4006_TestCase**

Testcase ID	BWS_4006_TestCase
Test Assertion	BWS-TA-40003
Description	Tests that where a <component/> <reference/> element has a <binding.ws/> subelement to which the SOAP.V1.2 intent is applied, that the reference both sends and receives messages in the SOAP v1.2 protocol
Artifacts	BWS_4006_TestCase.java Test_BWS_4006.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl TestComposite4.composite
Expected output	Positive test:

	"BWS_4006 request service1 operation1 invoked request Client service1 operation1 invoked"
--	---

267

268

2.2.7 BWS_4007_TestCase

Testcase ID	BWS_4007_TestCase
Test Assertion	BWS-TA-40004
Description	Tests that where a <component/> <reference/> element has an explicit portType rpc-literal in form, it is accepted
Artifacts	BWS_4007_TestCase.java Test_BWS_4007.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl TestComposite4.composite Service1RPCLiteral.wsdl
Expected output	Positive test: "BWS_4007 request service1 operation1 invoked request Client service1 operation1 invoked"
Comment	All other (positive) testcases use a simple document literal wrapped form and so the acceptance of rpc-literal is otherwise not tested

269

270

2.2.8 BWS_4008_TestCase

Testcase ID	BWS_4008_TestCase
Test Assertion	BWS-TA-40004
Description	Tests that where <component/> <reference/> element has an explicit portType rpc-encoded in form, it is rejected
Artifacts	BWS_4008_TestCase.java Test_BWS_4008.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl TestComposite4.composite Service1RPCEncoded.wsdl
Expected output	Negative test: "exception"

271

272

273 2.3 Section 5

274 2.3.1 BWS_5001_TestCase

275

Testcase ID	BWS_5001_TestCase
Test Assertion	BWS-TA-50001
Description	<p>Tests that where an SCA reference with a bidirectional interface invokes a Web service, that the Web service message sent to the target service contains a Callback EPR, either in the wsa:From SOAP header or in the wsa:ReplyTo header</p> <p>In this case, an SCA component is a client to an external Web service that has a callback, so this checks the ability of the binding.ws implementation of the runtime to transmit a callback address in the right way and host the callback service for a reference</p>
Artifacts	<p>BWS_5001_TestCase.java</p> <p>Test_BWS_5001.composite</p> <p>TestInvocation.wsdl</p> <p>TestClient_0002.composite</p> <p>TestComposite54.composite</p> <p>Service5.wsdl</p> <p>Service5Callback.wsdl</p> <p>jaxwsClient.Service5Impl.java</p>
Expected output	<p>Positive test:</p> <p>"BWS_5001 request service1 operation1 invoked request Client service5 operation1 invoked service1 callback1 invoked"</p>

276

277 2.3.2 BWS_5002_TestCase

278

Testcase ID	BWS_5002_TestCase
Test Assertion	<p>BWS-TA-50001</p> <p>BWS-TA-50003</p>
Description	<p>Tests that where an SCA service with a bidirectional interface is invoked by a Web services client, that the Web service message sent to the target service contains a Callback EPR, either in the wsa:From SOAP header or in the wsa:ReplyTo header, which the service is able to use to invoke an operation on the callback interface</p> <p>In this case, an SCA component provides a service over binding.ws where</p>

	the service has a callback and an external client invokes the service and hosts the callback service.
Artifacts	BWS_5002_TestCase.java Test_BWS_5002.composite TestInvocation.wsdl TestClient_0002.composite TestComposite54.composite Service5.wsdl Service5Callback.wsdl jaxwsClient.Service5Impl.java
Expected output	Positive test: "BWS_5002 service1 operation1 invoked callback service5Callback callback1 invoked"

279

280 **2.3.3 BWS_5003_TestCase**

281

Testcase ID	BWS_5003_TestCase
Test Assertion	BWS-TA-50002
Description	Tests that where an SCA service with a bidirectional interface is invoked by a Web services client, that if the Web service invocation message sent to the target service contains a Callback EPR, of the form "http://www.w3.org/2005/08/addressing/anonymous" that the service raises an Invalid Addressing Header fault.
Artifacts	BWS_5003_TestCase.java Test_BWS_5003.composite TestInvocation.wsdl TestClient_0002.composite TestComposite54.composite Service5.wsdl Service5Callback.wsdl jaxwsClient.Service5Impl.java
Expected output	Negative test: "exception"

282

283

2.3.4 BWS_5004_TestCase

284

Testcase ID	BWS_5004_TestCase
Test Assertion	BWS-TA-50004 BWS-TA-50007
Description	Tests that where an SCA service with a bidirectional interface is invoked by a Web services client, that a callback operation invocation follows the rules defined in Section 3.3 of the WS-Addressing 1.0 - Core specification - deals with case where there are no reference parameters on forward call
Artifacts	BWS_5004_TestCase.java Test_BWS_5004.composite TestInvocation.wsdl TestClient_0002.composite TestComposite54.composite Service5.wsdl Service5Callback.wsdl jaxwsClient.Service5Impl.java
Expected output	Positive test: "BWS_5004 service1 operation1 invoked callback service5Callback callback1 invoked headers OK"

285

286

2.3.5 BWS_5005_TestCase

287

Testcase ID	BWS_5005_TestCase
Test Assertion	BWS-TA-50005 BWS-TA-50006
Description	Tests that where an SCA service with a bidirectional interface is invoked by a Web services client, and the client invocation message contains a wsa:MessageID SOAP header block, that an invocation of a callback operation contains a wsa:RelatesTo SOAP header block
Artifacts	BWS_5005_TestCase.java Test_BWS_5005.composite TestInvocation.wsdl TestClient_0002.composite TestComposite54.composite Service5.wsdl

	Service5Callback.wsdl jaxwsClient.Service5Impl.java
Expected output	Positive test: "BWS_5005 service1 operation1 invoked callback service5Callback callback1 invoked headers OK"

288

289 **2.3.6 BWS_5006_TestCase**

290

Testcase ID	BWS_5006_TestCase
Test Assertion	BWS-TA-50004
Description	Tests that where an SCA service with a bidirectional interface is invoked by a Web services client, that a callback operation invocation follows the rules defined in Section 3.3 of the WS-Addressing 1.0 - Core specification - deals with case where there are reference parameters on forward call
Artifacts	BWS_5006_TestCase.java Test_BWS_5006.composite TestInvocation.wsdl TestClient_0002.composite TestComposite54.composite Service5.wsdl Service5Callback.wsdl jaxwsClient.Service5Impl.java
Expected output	Positive test: "BWS_5006 service1 operation1 invoked callback service5Callback callback1 invoked headers OK"

291

3 Cross Mapping of Test Assertions to TestCases

Test Assertion	Test Cases
BWS-TA-20001	BWS_2001_TestCase
BWS-TA-20002	BWS_2002_TestCase
BWS-TA-20003	BWS_2003_TestCase
BWS-TA-20004	BWS_2004_TestCase
BWS-TA-20005	BWS_2005_TestCase
BWS-TA-20006	BWS_2006_TestCase
BWS-TA-20007	BWS_2007_TestCase
BWS-TA-20008	BWS_2008_TestCase
BWS-TA-20009	BWS_2009_TestCase
BWS-TA-20010	BWS_2005_TestCase
BWS-TA-20011	BWS_2010_TestCase
BWS-TA-20012	BWS_2011_TestCase
BWS-TA-20013	BWS_2012_TestCase
BWS-TA-20014	BWS_2013_TestCase
BWS-TA-20015	BWS_2012_TestCase BWS_2014_TestCase
BWS-TA-20016	BWS_2015_TestCase
BWS-TA-20017	BWS_2016_TestCase
BWS-TA-20018	BWS_2017_TestCase
BWS-TA-20019	BWS_2018_TestCase
BWS-TA-20020	Covered by a series of other testcases that use one or more of each of the attributes of the <binding.ws/> element
BWS-TA-20021	Optional function - no test
BWS-TA-20022	Optional function - no test
BWS-TA-20023	BWS_2019_TestCase
BWS-TA-20024	BWS_2020_TestCase
BWS-TA-20025	BWS_2021_TestCase
BWS-TA-20026	Untestable as this requires a specific non-WSDL interface type, and there is no mandatory support for such an interface type
BWS-TA-20027	Untestable as this requires a specific non-WSDL interface type, and there is no mandatory support

	for such an interface type
BWS-TA-20028	Optional function - no test
BWS-TA-20029	Optional function - no test
BWS-TA-20030	BWS_2022_TestCase
BWS-TA-20031	BWS_2012_TestCase (depends on the SOAP 1.1 binding being supported)
BWS-TA-20032	Optional function - no test
BWS-TA-20033	BWS_2023_TestCase
BWS-TA-20034	Optional function - no test
BWS-TA-20035	Untestable - there is no standard way of producing a conflict between attached intent(s) and the configuration of a <binding.ws/> element

294

Test Assertion	Test Cases
BWS-TA-40001	BWS_4001_TestCase BWS_4004_TestCase
BWS-TA-40002	BWS_4002_TestCase BWS_4005_TestCase
BWS-TA-40003	BWS_4003_TestCase BWS_4006_TestCase
BWS-TA-40004	BWS_4007_TestCase BWS_4008_TestCase
BWS-TA-40005	Untestable - or test not required since the assumption behind the TA pervades many other tests in the test suite
BWS-TA-40007	Optional function - no test

295

Test Assertion	Test Cases
BWS-TA-50001	BWS_5001_TestCase BWS_5002_TestCase
BWS-TA-50002	BWS_5003_TestCase
BWS-TA-50003	BWS_5002_TestCase
BWS-TA-50004	BWS_5004_TestCase BWS_5006_TestCase
BWS-TA-50005	BWS_5005_TestCase
BWS-TA-50006	BWS_5005_TestCase
BWS-TA-50007	BWS_5004_TestCase
BWS-TA-50008	Untestable as it requires the support of some other callback protocol
BWS-TA-50009	SCA runtimes are not required to support or understand WS Policy - untestable
BWS-TA-50010	Optional function - no test
BWS-TA-50011	Optional function - no test

297

4 Catalog of Test Artifacts

298

299

4.1 Composite Files - lower level

Name	Valid	Description
TestComposite1 <i>"Impl Specific"</i>	Y	1 service, interface Service1 0 references service1Impl
TestComposite4 <i>"Impl Specific"</i>	Y	1 service, interface Service1 1 reference (1..1), interface Service1 service1Impl2
TestComposite9 <i>"Impl Specific"</i>	Y	1 service, interface = Service1Superset 0 references service1SupersetImpl
TestComposite54 <i>"Impl Specific"</i>	Y	1 service, interface = Service1 1 reference with a callback - Service5 & Service5Callback service1Callback5Impl
TestComposite55 <i>"Impl Specific"</i>	Y	1 service with callback - Service5 & Service5Callback 0 references service5Impl
TestClient_0002 <i>"Impl Specific"</i>	Y	Test client invocation composite 1 service, interface = TestInvocation 1 reference (1..1), interface = Service1

300

301

4.2 Java Interfaces

302

Name	Description
Service1.java	
Service1Superset.java	
Service5.java	
Service5Callback.java	
TestInvocation.java	

303 **4.3 Java Implementation Classes**

304

Name	Description
Service1Impl.java	
Service1Impl2.java	
Service1SupersetImpl.java	
Service1Callback5Impl.java	
Service5Impl.java	
TestException.java	
ASM_0002_Client.java	

305 **4.4 WSDL Interface Files**

306

Name	Description
Service1.wsdl	Service1 interface 1 operation: - "operation1" string input, string output

Service1RPCLiteral.wsdl	Service1 interface in RPC Literal form
Service1Superset.wsdl	Service1Superset interface - superset of Service1 2 operations: - "operation1" string input, string output - "operation2" string input, string output
Service5.wsdl	Service5 interface which has a callback interface Service5Callback 1 operation: - "operation1" string input, string output
Service5Callback.wsdl	Service5Callback interface, which is the callback interface for Service5 1 operation: - "callback1" string input, string output
Service1InvalidService.wsdl (BWS_2005 contribution)	Service1 interface - this is Service1 with an invalid WSDL Binding (invalid transport)
Service1InvalidPort.wsdl (BWS_2007 contribution)	Service1 interface - this is Service1 with a PortType incompatible with the regular Service1 PortType
Service1WithPort.wsdl (BWS_2008 contribution)	Service1 interface - this is Service1 with a Port which includes a target address for the service
Service1Superset.wsdl (BWS_2009 contribution)	Service1Superset interface - with additional WSDL Service, Port & Binding elements
Service1WithBinding.wsdl (BWS_2010 contribution)	Service1 interface - with a valid WSDL Binding element
Service1InvalidService.wsdl (BWS_2011 contribution)	Service1 interface - this is Service1 with a WSDL Service made invalid through the use of an invalid WSDL Binding element

Service1Superset.wsdl (BWS_2012 contribution)	Service1Superset interface - with additional WSDL Service, Port & Binding elements
Service1InvalidService.wsdl (BWS_2013 contribution)	Service1 interface - this is Service1 with a WSDL Service made invalid through the use of an invalid WSDL Binding element
Service1Superset.wsdl (BWS_2014 contribution)	Service1Superset interface - with additional WSDL Service, Port & Binding elements
Service1WithBinding.wsdl (BWS_2015 contribution)	Service1 interface - with a valid WSDL Binding element
Service1WithBinding.wsdl (BWS_2016 contribution)	Service1 interface - with a valid WSDL Binding element
Service1Superset.wsdl (BWS_2017 contribution)	Service1Superset interface - with additional WSDL Service, Port & Binding elements
Service1WithCallback.wsdl (BWS_2022 contribution)	Service1 interface - PortType has a sca:callback attribute added which points to a callback interface Service1Callback, which is contained in the same WSDL document
Service1RPCEncoded.wsdl (BWS_4008 contribution)	Service1 interface in RPC Encoded form
sca_variables.dtd (BWS_General contribution)	DTD holding service addresses which can be modified to suit an SCA runtime
sca_variables.dtd (BWS_2005 contribution)	holds service address for <i>TEST_BWS_2005Component2/Service1</i>
sca_variables.dtd (BWS_2008 contribution)	holds service address for <i>TEST_BWS_2008Component2/Service1Special</i>
sca_variables.dtd (BWS_2009 contribution)	holds service address for <i>TEST_BWS_2009Component2/Service1Superset</i>
sca_variables.dtd (BWS_2011 contribution)	holds service address for <i>TEST_BWS_2011Component2/Service1</i>
sca_variables.dtd (BWS_2012 contribution)	holds service address for

	<i>TEST_BWS_2012Component2/Service1Super set</i>
sca_variables.dtd (BWS_2013 contribution)	holds service address for <i>TEST_BWS_2013Component2/Service1</i>
sca_variables.dtd (BWS_2014 contribution)	holds service address for <i>TEST_BWS_2014Component2/Service1Super set</i>
sca_variables.dtd (BWS_2015 contribution)	holds service address for <i>TEST_BWS_2015Component2/Service1</i>
sca_variables.dtd (BWS_2016 contribution)	holds service address for <i>TEST_BWS_2016Component2/Service1</i>
sca_variables.dtd (BWS_2017 contribution)	holds service address for <i>TEST_BWS_2017Component2/Service1Super set</i>
sca_variables.dtd (BWS_2021 contribution)	holds service address for <i>TEST_BWS_2021Component2/Service1</i>
sca_variables.dtd (BWS_2022 contribution)	holds service address for <i>TEST_BWS_2022/Component2/Service1</i>
sca_variables.dtd (BWS_2023 contribution)	holds service address for <i>TEST_BWS_2023Component2/Service1</i>
sca_variables.dtd (BWS_4008 contribution)	holds service address for service exposed by JAXWS test runner client application
TestClient.wsdl	TestClient interface used to invoke SCA services
TestInvocation.wsdl	TestInvocation interface supplied by SCA component to allow client application to invoke test artifacts 1 operation: - "invokeTest" string input, string output

308 **5 Conformance**

309 There are no conformance statements relating to the TestCases.

310

311 **Appendix A. Acknowledgments**

312 The following individuals have participated in the creation of this specification and are gratefully
313 acknowledged

314 **Participants:**

315

Participant Name	Affiliation
Mike Edwards	IBM
Eric Johnson	TIBCO
Anish Karmarkar	Oracle
Simon Holdsworth	IBM

316

317 **Appendix B. Revision History**

318

Revision	Date	Editor	Changes Made
wd01	13/05/10	Mike Edwards	Initial version
wd02	17/05/10	Mike Edwards	Added testcases BWS_2002 to BWS_2006
wd03	01/06/10	Mike Edwards	Added testcases BWS_2007 to BWS_2023
wd04	07/06/10	Mike Edwards	Editorial changes Added testcases BWS_4001 to BWS_4007
wd05	14/06/10	Mike Edwards	Modified BWS_5001 Added BWS_5002 to BWS_5006
cd01	01/07/10	Mike Edwards	All changes accepted Frontmatter edited to conform to OASIS requirements Completed artifact catalog in Section 4

319