TestCases for the SCA_J Common Annotations and APIs Version 1.1 Specification

Committee Draft 01

19 July 2010

Specification URIs:

This Version:

http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-testcases-cd01.html

http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-testcases-cd01.odt

http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-testcases-cd01.pdf(Authoritative)

Previous Version:

N/A

Latest Version:

http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-testcases.html

http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-testcases.odt

http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-testcases.pdf (Authoritative)

Technical Committee:

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

Chair(s):

David Booz, IBM
Mark Combellack Avaya Inc

Editor(s):

Mike Edwards, IBM
David Booz, IBM

Related Work:

This document is related to:

- Service Component Architecture Java Common Annotations and APIs Specification Version 1.1
  http://docs.oasis-open.org/opencsa/sca-j/sca-javacaa-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 Java Common Annotations and APIs Assembly 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 Java Common Annotations and APIsAssembly specification.

Status:

This document was last revised or approved by the OASIS Service Component Architecture / J (SCA-J) 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-j/

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-j/ipr.php.

The non-normative errata page for this specification is located at

http://www.oasis-open.org/committees/sca-j/

Notices

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", "Service Component Architecture" are trademarks of OASIS, 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 Introduction

This document defines the TestCases for the SCA Assembly specification.

The tests described in this document are related to the Test Assertions described in the SCA Assembly Test Assertions document [CAA-TA].

1.1 TestCase Structure

The SCA J CAA 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.

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. The client is structured to permit Web services to be replaced by some other binding (eg JMS) should the SCA runtime under test not support Web services as a binding technology.

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

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

The application is structured so that alternative technologies can be used. For example, replacing the contents of the second-level or third-level composites allows different Java implementation technologies to be tested – eg POJOs or Spring Application Contexts may be used. Similarly, the binding used to connect from the top level composite to the client application may be changed from Web services to JMS if required, simply by changing the binding on the <service/> of the top level composite.

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

- "POJO" - for Java implementations
- "Spring" - for Spring implementations

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

Test Artifacts Organization

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

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

Broadly, Contribution names are as follows:

- JCA_nnnn - a contribution that is specific for a particular testcase, where "nnnn" is the number of the testcase. Often this is required because a particular testcase involves artifacts that contain errors that are statically checkable - an SCA runtime is permitted to reject such artifacts when they are contributed and deployed and it is important to ensure that contributions containing deliberate errors for one testcase do not interfere with the operation of other testcases.
- JCA_nnnn_POJO - a contribution for a specific testcase where there is a need for language specific artifacts that relate to that testcase alone
- JCA_General - a shared contribution containing implementation type independent artifacts that can be used by many testcases.
- JCA_General_POJO - a shared contribution containing implementation type dependent artifacts for Java POJOs. These artifacts can include both Java classes and also SCA composites that directly use Java classes.

Note that the names of Contributions containing implementation specific artifacts ends with a name that is specific to the implementation type - so "_POJO" is used for Java POJO implementations, "_Spring" is used for Spring implementations (and so on). Note that the name following the underscore matches the name used in the "OASIS_TESTENV_IMPL_LANG" variable used to control execution of the test client. The concept is that where there is an implementation type specific contribution, each implementation type

must provide its own versions of the same basic artifacts. Typically, this means that each contribution must contain the same set of Composites, but that the implementation type dependent artifacts that these composites use will differ from implementation type to implementation type.

Basically, the setting of the variable is used to select the suffix used for implementation type dependent contributions. If the variable is set to "POJO" then the contribution "JCA_General_POJO" is selected, whereas if the variable is set to "Spring", the contribution "JCA_General_Spring" is selected.

1.2 Namespaces and Java Package Names

The SCA Assembly testcase suite makes use of some XML namespaces and Java package names, as follows:

SCA Artifact Namespaces

These apply to artifacts such as Composites

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

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

WSDL Namespace

http://test.sca.oasisopen.org/

Java Package name

For Java interface classes and for Java implementation classes

`org.oasisopen.sca.test`

1.3 Terminology

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

1.4 Normative References

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

[CAA-TA] OASIS Committee Draft 01, **"**Test Assertions for the SCA_J Common Annotations and APIs Version 1.1 Specification", July 2010. http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-test-assertions-cd01.pdf

1.5 Non-normative References

N/A

2 TestCases

2.1 Section 2

JCA_2001_TestCase

Testcase ID	JCA_2001_TestCase
Test Assertion	JCA-TA-2001
Description	Tests that a Java service which is marked remotable does not use method overloading
Artifacts	JCA_2001_TestCase.java Test_JCA_2001.composite TestInvocation.wsdl TestClient_0002.composite Service1Overload.java service1OverloadImpl.java
Expected output	Negative test: “exception”

JCA_2002_TestCase

Testcase ID	JCA_2002_TestCase
Test Assertion	JCA-TA-2002
Description	Tests that a stateless scoped Java implementation instance is not dispatched on more than a single thread at one time
Artifacts	JCA_2002_TestCase.java Test_JCA_2002.composite TestInvocation.wsdl TestClient_0002.composite Service1.java ParallelService.java parallelServiceClientImpl.java parallelServiceImpl.java
Expected output	Positive test:

	"JCA_2002 request service1 parallel service invocation successful"

151

152 JCA_2003_TestCase

153

Testcase ID	JCA_2003_TestCase
Test Assertion	JCA-TA-2003
Description	Tests that an implementation instance annotated with @Scope("STATELESS") is invoked only once through one business method during the lifetime of the implementation instance
Artifacts	JCA_2003_TestCase.java Test_JCA_2003.composite TestInvocation.wsdl TestClient_0002.composite MultipleService.java multipleServiceClientImpl.java multipleServiceImpl.java
Expected output	Positive test: "JCA_2003 request service1 multiple service invocation successful"

154

155 JCA_2004_TestCase

156

Testcase ID	JCA_2004_TestCase
Test Assertion	JCA-TA-2004
Description	Tests that where there is a Domain-level component implementation marked as COMPOSITE scope, that all its clients appear to interact with a single runtime instance of the class.
Artifacts	JCA_2004_TestCase.java Test_JCA_2004.composite TestInvocation.wsdl TestClient_0002.composite Service1.java service1CoordinatorImpl.java service1Impl2.java

	service1CompositeImpl.java
Expected output	Positive test: "JCA_2004 request serviceCoordinator operation1 invoked service1 operation1 invoked serviceComposite operation1 Initial service2 operation1 invoked serviceComposite operation1 1 service3 operation1 invoked serviceComposite operation1 2 service4 operation1 invoked serviceComposite operation1 3"

157

158 JCA_2005_TestCase

159

Testcase ID	JCA_2005_TestCase
Test Assertion	JCA-TA-2005
Description	Tests that where a component implementation is marked with COMPOSITE scope and with @EagerInit, that the implementation instance is created and initialized when the component is started.
Artifacts	JCA_2005_TestCase.java Test_JCA_2005.composite TestInvocation.wsdl TestClient_0002.composite Service1.java compositeEagerInitImpl.java service1CompositeImpl.java
Expected output	Positive test: "JCA_2005 request serviceComposite2 operation1 EagerInit called"

160

161 JCA_2006_TestCase

162

Testcase ID	JCA_2006_TestCase
Test Assertion	JCA-TA-2006
Description	Tests that where a component implementation has a method marked with @Init, that this method is called when the implementation instance is created
Artifacts	JCA_2006_TestCase.java Test_JCA_2006.composite TestInvocation.wsdl TestClient_0002.composite

	Service1.java service1InitCheckerImpl.java service1InitImpl.java
Expected output	Positive test: "JCA_2006 request serviceComposite1 operation1 Init check succeeded"

163

164 JCA_2007_TestCase

165

Testcase ID	JCA_2007_TestCase
Test Assertion	JCA-TA-2007
Description	Tests that for a Java implementation with COMPOSITE scope, that multiple invocations of service operations which overlap in time can run within a single instance of the implementation on separate Java threads.
Artifacts	JCA_2007_TestCase.java Test_JCA_2007.composite TestInvocation.wsdl TestClient_0002.composite Service1.java ParallelService.java parallelCompositeClientImpl.java parallelCompositeServiceImpl.java
Expected output	Positive test: "JCA_2007 request serviceCompositeClient COMPOSITE service invocation successfully used by multiple threads simultaneously"

166

167 JCA_2008_TestCase

168

Testcase ID	JCA_2008_TestCase
Test Assertion	JCA-TA-2008
Description	Tests that for a Java implementation class with COMPOSITE scope offering a service, used as the implementation of a component within a composite which is itself the implementation of a Domain level component, all clients of the component service appear to interact with a single instance of the implementation class
Artifacts	JCA_2008_TestCase.java

	Test_JCA_2008.composite TestInvocation.wsdl TestClient_0002.composite CompositeScope.composite Service1.java service1CoordinatorImpl.java service1Impl2.java service1CompositeImpl.java
Expected output	Positive test: "JCA_2008 request serviceComposite operation1 invoked service1 operation1 invoked serviceComposite operation1 Initial service2 operation1 invoked serviceComposite operation1 1 service3 operation1 invoked serviceComposite operation1 2 service4 operation1 invoked serviceComposite operation1 3"

169

170 JCA_2009_TestCase

171

Testcase ID	JCA_2009_TestCase
Test Assertion	JCA-TA-2011, JCA-TA-10004
Description	Tests that where one component is a client of a service provided by a second component, both with Java implementations and which both run in the same JVM, and the client reference is marked with @AllowsPassByReference but the service implementation methods are not marked with @AllowsPassByReference that invocations of the service use "pass by value" semantics.
Artifacts	JCA_2009_TestCase.java Test_JCA_2009.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service4.java service1Impl7.java service4Impl.java
Expected output	Positive test: "JCA_2009 request service1 operation1 invoked service2 operation1 invoked request+1"

172

173 JCA_2010_TestCase

174

Testcase ID	JCA_2010_TestCase
Test Assertion	JCA-TA-2011, JCA-TA-10004
Description	Tests that where one component is a client of a service provided by a second component, both with Java implementations and which both run in the same JVM, and the service implementation methods are not marked with @AllowsPassByReference but the client reference is not marked with @AllowsPassByReference that invocations of the service use "pass by value" semantics.
Artifacts	JCA_2010_TestCase.java Test_JCA_2010.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service4.java service1Impl7b.java service4Impl1.java
Expected output	Positive test: "JCA_2010 request service1 operation1 invoked service2 operation1 invoked request+1"

175

176

177 2.2 Section 3

178 JCA_3001_TestCase

179

Testcase ID	JCA_3001_TestCase
Test Assertion	JCA-TA-3001
Description	Tests that a Java Interface class name is fully qualified on a service <interface.java> element
Artifacts	JCA_3001_TestCase.java Test_JCA_3001.composite TestInvocation.wsdl

	TestClient_0002.composite ASM_0002_Client.java Service1.java Service1Impl.java
Expected output	Negative test: “exception”

180

181 JCA_3002_TestCase

182

Testcase ID	JCA_3002_TestCase
Test Assertion	JCA-TA-3001
Description	Tests that a Java Interface is fully qualified on a reference <interface.java> element
Artifacts	JCA_3002_TestCase.java Test_JCA_3002.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java Service1Impl.java Service1Impl2.java
Expected output	Negative test: “exception”

183

184 JCA_3003_TestCase

185

Testcase ID	JCA_3003_TestCase
Test Assertion	JCA-TA-3002, JCA-TA-10031
Description	Tests that callback interfaces on a service are specified in their fully qualified form
Artifacts	JCA_3003_TestCase.java Test_JCA_3003.composite TestInvocation.wsdl

	TestClient_0002.composite ASM_0002_Client.java Service1.java service1CalbackImpl.java Service3WithCallback.java Service3Callback.java service3Impl1.java
Expected output	Negative test: “exception”

186

187 ## JCA_3004_TestCase

188

Testcase ID	JCA_3004_TestCase
Test Assertion	JCA-TA-3002, JCA-TA-10031
Description	Tests that callback interfaces on a reference are specified in their fully qualified form
Artifacts	JCA_3004_TestCase.java Test_JCA_3004.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service1CalbackImpl.java Service3WithCallback.java Service3Callback.java service3Impl1.java
Expected output	Negative test: “exception”

189

190 ## JCA_3005_TestCase

191

Testcase ID	JCA_3005_TestCase

Test Assertion	JCA-TA-3003, JCA-TA-10031
Description	Tests that callback interfaces specified in the composite match the callback interface specified in the service interface class
Artifacts	JCA_3005_TestCase.java Test_JCA_3005.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service1CalbackImpl.java Service3WithCallback.java Service3Callback.java service3Impl1.java
Expected output	Positive test: "JCA_3005 request service1 operation1 invoked service3 operation1 invoked service1 callback1 invoked"

192

193 ## JCA_3006_TestCase

194

Testcase ID	JCA_3006_TestCase
Test Assertion	JCA-TA-3003, JCA-TA-10031
Description	Tests that callback interfaces specified in the composite match the callback interface specified in the service interface class
Artifacts	JCA_3006_TestCase.java Test_JCA_3006.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service1CalbackImpl.java Service3WithCallback.java Service3Callback.java service3Impl1.java
Expected output	Negative test:

	"exception"

JCA_3007_TestCase

Testcase ID	JCA_3007_TestCase
Test Assertion	JCA-TA-3003, JCA-TA-10031
Description	Tests that callback interfaces specified in the composite match the callback interface specified in the reference interface class
Artifacts	JCA_3007_TestCase.java Test_JCA_3007.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service1CalbackImpl.java Service3WithCallback.java Service3Callback.java service3Impl1.java
Expected output	Negative test: "exception"

JCA_3008_TestCase

Testcase ID	JCA_3008_TestCase
Test Assertion	JCA-TA-3004
Description	Tests that <interface.java/> conforms to the schema
Artifacts	JCA_3008_TestCase.java Test_JCA_3008.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java

	service1Impl.java
Expected output	Negative test: "exception"

201

202 JCA_3009_TestCase

203

Testcase ID	JCA_3009_TestCase
Test Assertion	JCA-TA-3005
Description	Tests that remotable attribute matches @Remotable annotation
Artifacts	JCA_3009_TestCase.java Test_JCA_3009.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service1Impl.java
Expected output	Negative test: "exception"

204

205 JCA_3010_TestCase

206

Testcase ID	JCA_3010_TestCase
Test Assertion	JCA-TA-3005
Description	Tests that remotable attribute matches @Remotable annotation
Artifacts	JCA_3010_TestCase.java Test_JCA_3010.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service1Impl.java
Expected output	Positive test:

	"JCA_3010 request service1 operation1 invoked"

JCA_3011_TestCase

Testcase ID	JCA_3011_TestCase
Test Assertion	JCA-TA-3006
Description	Tests that a service interfaces doesn't contain forbidden annotations
Artifacts	JCA_3011_TestCase.java Test_JCA_3011.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java JCA3011Service.java JCA3011serviceImpl.java
Expected output	Negative test: "exception"

JCA_3012_TestCase

Testcase ID	JCA_3012_TestCase
Test Assertion	JCA-TA-3007
Description	Tests that callback interfaces don't contain forbidden annotations
Artifacts	JCA_3012_TestCase.java Test_JCA_3012.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service3WithCallback.java Service1.java service1CallbackImpl.java Service3Callback.java JCA3012Service3WithCallback.java

	JCA3012Service3Callback.java JCA3012service3Impl1.java
Expected output	Negative test: "exception"

213

214 JCA_3013_TestCase

215

Testcase ID	JCA_3013_TestCase
Test Assertion	JCA-TA-3008
Description	Tests that where two Java interfaces are otherwise compatible, every method present in both interfaces where annotated with @OneWay in one interface is also annotated with @OneWay in the other interface
Artifacts	JCA_3013_TestCase.java Test_JCA_3013.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java ServiceOneWayMissing.java serviceOneWayImpl.java Service1OneWayMissingCallerImpl.java ServiceOneWay.java
Expected output	Negative test: "exception"

216

217 JCA_3014_TestCase

218

Testcase ID	JCA_3014_TestCase
Test Assertion	JCA-TA-3009
Description	Tests that where an <interface.java/> element references a Java interface class which contains a @WebService annotation with a non-empty wsdlLocation property pointing at a WSDL document, it is treated as if it were an <interface.wsdl/> element with an @interface attribute identifying the portType in the WSDL document mapped from the Java interface class
Artifacts	JCA_3014_TestCase.java Test_JCA_3014.composite

	TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java Service3Operations.java Service3OperationsWSDL.java Service3OperationsWSDLImpl.java Service1Calls3OperationsImpl.java Service3OperationsWSDL.wsdl
Expected output	Negative test: "exception"

219

220

221 2.3 Section 4

222 JCA_4001_TestCase

223

Testcase ID	JCA_4001_TestCase
Test Assertion	JCA-TA-4001, JCA-TA-10005, JCA-TA-10006
Description	Tests that for a stateless Java implementation, that all lifecycle stages are performed in the correct sequence and that no stage occurs out of sequence.
Artifacts	JCA_4001_TestCase.java Test_JCA_4001.composite TestInvocation.wsdl TestClient_0002.composite JCA_0002_Client.java Service1.wsdl DataStore.java lifecycleControllerImpl.java service1StatelessLifecycleImpl.java service1Impl.java dataStoreCompositeImpl.java
Expected output	"JCA_4001 request Constructing property1 injected reference1 injected property2 injected reference2 injected Init invoked Init completed

	serviceLifecycle operation1 invoked service1 operation1 invoked service2 operation1 invoked service3 operation1 invoked prop1 prop2 prop3 Destroy invoked"

224

225 ## JCA_4002_TestCase

226

Testcase ID	JCA_4002_TestCase
Test Assertion	JCA-TA-4004
Description	Tests that where a stateless Java implementation throws an exception from its Constructor method, that the implementation transitions to the Terminating state
Artifacts	JCA_4002_TestCase.java Test_JCA_4002.composite TestInvocation.wsdl TestClient_0002.composite JCA_0002_Client.java Service1.wsdl DataStore.java lifecycleControllerImpl.java service1LifecycleExceptionsImpl.java service1Impl.java dataStoreCompositeImpl.java
Expected output	"JCA_4002 request Constructing exception thrown"

227

228 ## JCA_4003_TestCase

229

Testcase ID	JCA_4003_TestCase
Test Assertion	JCA-TA-4010
Description	Tests that when a stateless Java implementation throws an exception when a property is injected, that the implementation transitions to the Destroying state
Artifacts	JCA_4003_TestCase.java Test_JCA_4003.composite TestInvocation.wsdl TestClient_0002.composite JCA_0002_Client.java

	Service1.wsdl DataStore.java lifecycleControllerImpl.java service1LifecycleExceptionsImpl.java service1Impl.java dataStoreCompositeImpl.java
Expected output	"JCA_4003 request Constructing property1 injected reference1 injected property2 injected exception thrown Destroy invoked"

230

231 JCA_4004_TestCase

232

Testcase ID	JCA_4004_TestCase
Test Assertion	JCA-TA-4012
Description	Tests that where a Java implementation invokes an injected reference while in the initializing phase and the target of the reference has not been initialized, that the implementation receives a ServiceUnavailableException
Artifacts	JCA_4004_TestCase.java Test_JCA_4004.composite TestInvocation.wsdl TestClient_0002.composite JCA_0002_Client.java Service1.wsdl DataStore.java lifecycleControllerImpl.java service1LifecycleExceptionsImpl.java service1UninitImpl.java service1Impl.java dataStoreCompositeImpl.java
Expected output	"JCA_4004 request Constructing property1 injected reference1 injected property2 injected reference2 injected Init invoked calling uninitialized service ServiceUnavailable exception received Init completed serviceLifecycle operation1 invoked service1 operation1 invoked service2 operation1 invoked service3 operation1 invoked prop1 prop2 prop3 Destroy invoked"

233

JCA_4005_TestCase

Testcase ID	JCA_4005_TestCase
Test Assertion	JCA-TA-4015
Description	Tests that where a Java implementation throws an exception from the method marked with the @Init annotation, that the implementation transitions to the Destroying state
Artifacts	JCA_4005_TestCase.java Test_JCA_4005.composite TestInvocation.wsdl TestClient_0002.composite JCA_0002_Client.java Service1.wsdl DataStore.java lifecycleControllerImpl.java service1LifecycleExceptionsImpl.java service1Impl.java dataStoreCompositeImpl.java
Expected output	"JCA_4005 request Constructing property1 injected reference1 injected property2 injected reference2 injected Init invoked exception thrown Destroy invoked"

JCA_4006_TestCase

Testcase ID	JCA_4006_TestCase
Test Assertion	JCA-TA-4019
Description	Tests that a Java implementation in the Destroying state which invokes a method on a reference whose target service has already been destroyed receives an InvalidServiceException
Artifacts	JCA_4006_TestCase.java Test_JCA_4006.composite TestInvocation.wsdl TestClient_0002.composite JCA_0002_Client.java Service1.wsdl DataStore.java

	lifecycleControllerImpl.java service1StatelessLifecycleImpl.java service1Impl.java dataStoreCompositeImpl.java
Expected output	"JCA_4006 request Constructing property1 injected reference1 injected property2 injected reference2 injected Init invoked Init completed serviceLifecycle operation1 invoked service1 operation1 invoked service2 operation1 invoked service3 operation1 invoked prop1 prop2 prop3 Destroy invoked"

239

240 ## JCA_4007_TestCase

241

Testcase ID	JCA_4007_TestCase
Test Assertion	JCA-TA-4022
Description	Tests that where a Java implementation throws an exception from its method marked with the @Destroy annotation, that the implementation transitions to the terminated state
Artifacts	JCA_4007_TestCase.java Test_JCA_4007.composite TestInvocation.wsdl TestClient_0002.composite JCA_0002_Client.java Service1.wsdl DataStore.java lifecycleControllerImpl.java service1LifecycleExceptionsImpl.java service1Impl.java dataStoreCompositeImpl.java
Expected output	"JCA_4007 request Constructing property1 injected reference1 injected property2 injected reference2 injected Init invoked Init completed serviceLifecycle operation1 invoked service1 operation1 invoked service2 operation1 invoked service3 operation1 invoked prop1 prop2 prop3 Destroy invoked exception thrown"

242

243 ## JCA_4008_TestCase

244

Testcase ID	JCA_4008_TestCase

Test Assertion	JCA-TA-4024
Description	Tests that when a stateless Java implementation throws an exception when a reference is injected, that the implementation transitions to the Destroying state
Artifacts	JCA_4008_TestCase.java Test_JCA_4008.composite TestInvocation.wsdl TestClient_0002.composite JCA_0002_Client.java Service1.wsdl DataStore.java lifecycleControllerImpl.java service1LifecycleExceptionsImpl.java service1Impl.java dataStoreCompositeImpl.java
Expected output	"JCA_4008 request Constructing property1 injected reference1 injected property2 injected reference2 injected exception thrown Destroy invoked"

245

246 2.4 Section 5

247 JCA_7001_TestCase

248

Testcase ID	JCA_7001_TestCase
Test Assertion	JCA-TA-7001 JCA-TA-7002
Description	Tests that where a Java implementation offers a bidirectional service and has fields and setter methods annotated with @Callback, when the bidirectional service is invoked, a callback reference object is injected into those fields and setter methods which have a type which is the interface for the callback service but null is injected into fields and setter methods which have a different type.
Artifacts	JCA_7001_TestCase.java Test_JCA_7001.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service3WithCallback.java Service3Callback.java

	Service7WithCallback.java Service7Callback.java service1CallbackImpl.java MultipleCallbacksImpl.java
Expected output	"JCA_7001 request service1 operation1 invoked service2 operation1 invoked callback1 invoked other callbacks null";

249

250 JCA_7002_TestCase

251

Testcase ID	JCA_7002_TestCase
Test Assertion	JCA-TA-7002
Description	Tests that where a Java implementation offers a bidirectional service and also has one non-bidirectional service and has fields and setter methods annotated with @Callback, when the non-bidirectional service is invoked, null is injected into all the fields and setter methods annotated with @Callback
Artifacts	JCA_7002_TestCase.java Test_JCA_7002.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service3WithCallback.java Service3Callback.java Service7WithCallback.java Service7Callback.java service1CallbackImpl.java MultipleCallbacksImpl.java
Expected output	"JCA_7002 request service1 operation1 invoked service2 operation1 invoked callback1 invoked other callbacks null";

252

253 JCA_7003_TestCase

254

Testcase ID	JCA_7003_TestCase
Test Assertion	JCA-TA-7003 JCA-TA-7004 JCA-TA-7005 JCA-TA-7007

Description	Tests that where a Java interface class is an asynchronous service mapping of a WSDL portType containing request/response operations that the class is annotated with the "asyncInvocation" intent
Artifacts	JCA_7003_TestCase.java Test_JCA_7003.composite TestInvocation.wsdl TestClient_0002.composite Service1AsyncServer.java Service1AsyncServerImpl.java
Expected output	"JCA_7003 request service1 operation1 invoked asynchronously"

255

256 JCA_7004_TestCase

257

Testcase ID	JCA_7004_TestCase
Test Assertion	JCA-TA-7003
Description	Tests that where a Java interface class is an asynchronous service mapping of a WSDL portType containing request/response operations that the class is annotated with the "asyncInvocation" intent - this is the negative version of JCA_7003_Testcase where the asyncInvocation intent is left off the interface class
Artifacts	JCA_7004_TestCase.java Test_JCA_7004.composite TestInvocation.wsdl TestClient_0002.composite Service1AsyncServerError.java Service1AsyncServerErrorImpl.java
Expected output	"exception"

258

259 JCA_7005_TestCase

260

Testcase ID	JCA_7005_TestCase
Test Assertion	JCA-TA-7006
Description	Tests that when an asynchronous service implementation invokes the sendResponse method of the supplied ResponseDispatch.object when that method has already been called once, the method throws an IllegalStateException
Artifacts	JCA_7005_TestCase.java

	Test_JCA_7005.composite TestInvocation.wsdl TestClient_0002.composite Service1AsyncServer.java Service1AsyncServerMultiReponseImpl.java asyncControllerImpl.java dataStoreCompositeImpl.java
Expected output	"JCA_7005 request service1 operation1 invoked asynchronously IllegalStateException thrown on second invocation of ResponseDispatch.sendResponse() method"

261

262 JCA_7006_TestCase

263

Testcase ID	JCA_7006_TestCase
Test Assertion	JCA-TA-9068 JCA-TA-9069
Description	Tests that when an asynchronous service implementation invokes the sendFault method of the supplied ResponseDispatch.object that the exception is sent to the client of the service operation and if the sendFault method is subsequently invoked a second time when that method has already been called once for a given ResponseDispatch object, the method throws an IllegalStateException
Artifacts	JCA_7006_TestCase.java Test_JCA_7006.composite TestInvocation.wsdl TestClient_0002.composite Service1AsyncServer.java Service1AsyncServerMultiFaultImpl.java asyncControllerImpl.java dataStoreCompositeImpl.java
Expected output	"JCA_7005 request service1 operation1 invoked asynchronously IllegalStateException thrown on second invocation of ResponseDispatch.sendResponse() method"

264

265

2.5 Section 8

JCA_8001_TestCase

Testcase ID	JCA_8001_TestCase
Test Assertion	JCA-TA-8001
Description	Tests that intent annotations use the @Intent annotation in the definition of the annotation. In this testcase, two mutually exclusive intents are defined in the domain. One of them is also defined as a Java annotation using the @Intent definition. TEST_JCA_8001Component1 requires both mutually exclusive intents (one in the SCDL, one in the Java implementation class) and should flag an error that mutually exclusive intents are not allowed.
Artifacts	JCA_8001_TestCase.java Test_JCA_8001.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java Service1Intent.java TestIntent2.java definitions.xml
Expected output	Negative test: "exception"

JCA_8002_TestCase

Testcase ID	JCA_8002_TestCase
Test Assertion	JCA-TA-8002
Description	Tests that intent annotations cannot be used on methods which are not reference setter methods.
Artifacts	JCA_8002_TestCase.java Test_JCA_8002.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java

	Service1.java TestIntent2.java service1BadIntent.java
Expected output	Negative test: "exception"

273

274 JCA_8003_TestCase

275

Testcase ID	JCA_8003_TestCase
Test Assertion	JCA-TA-8003
Description	Tests that intent annotations cannot be used on fields which are not references.
Artifacts	JCA_8003_TestCase.java Test_JCA_8003.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java TestIntent2.java service1BadIntent.java
Expected output	Negative test: "exception"

276

277 JCA_8004_TestCase

278

Testcase ID	JCA_8004_TestCase
Test Assertion	JCA-TA-8004
Description	Tests that intent annotations cannot be used on constructor parameters which are not references.
Artifacts	JCA_8004_TestCase.java Test_JCA_8004.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java

	Service1.java TestIntent2.java service1BadIntent.java
Expected output	Negative test: "exception"

279

280 JCA_8005_TestCase

281

Testcase ID	JCA_8005_TestCase
Test Assertion	JCA-TA-8002, JCA-TA-8003, JCA-TA-8004
Description	Tests that intent annotations can be used to annotate every place that a reference is allowed (setter method, field, or constructor parameter).
Artifacts	JCA_8005_TestCase.java Test_JCA_8005.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java TestIntent2.java service1GoodIntent.java
Expected output	Positive test: "JCA_8005 request service1 operation1 invoked service2 operation1 invoked service3 operation1 invoked service4 operation1 invoked"

282

283 JCA_8006_TestCase

284

Testcase ID	JCA_8006_TestCase
Test Assertion	JCA-TA-8005
Description	Tests that intent annotations are merged together (unioned) when present on the same Java element. In this testcase, two mutually exclusive intents are placed on the same Java element. When the runtime correctly merges them, an error should result.
Artifacts	JCA_8006_TestCase.java Test_JCA_8006.composite TestInvocation.wsdl

	TestClient_0002.composite ASM_0002_Client.java Service1.java TestIntent1.java TestIntent2.java service1BadIntent.java definitions.xml
Expected output	Negative test: "exception"

285

286 ## JCA_8007_TestCase

287

Testcase ID	JCA_8007_TestCase
Test Assertion	JCA-TA-8006
Description	Tests that intent annotations are correctly merged when they appear at different levels in a Java interface class. PolicySet1 satisfies testIntent3 which is the only intent that should be applicable to the service after the intents in Service5Intents are normalized.
Artifacts	JCA_8007_TestCase.java Test_JCA_8007.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java TestIntent3.java TestIntent4.java service5Impl2.java service5Impl.java Service5Intents.java definitions.xml
Expected output	Positive test: "JCA_8007 request service1 operation1 invoked service2 operation1 invoked"

288

289 ## JCA_8008_TestCase

290

Testcase ID	JCA_8008_TestCase
Test Assertion	JCA-TA-8007
Description	Tests that policySets cannot be used on methods which are not reference setter methods.
Artifacts	JCA_8008_TestCase.java Test_JCA_8008.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service1BadPolicySet.java definitions.xml
Expected output	Negative test: "exception"

291

292 ## JCA_8009_TestCase

293

Testcase ID	JCA_8009_TestCase
Test Assertion	JCA-TA-8008
Description	Tests that policySets cannot be used on fields which are not references.
Artifacts	JCA_8009_TestCase.java Test_JCA_8009.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service1BadPolicySet.java definitions.xml
Expected output	Negative test: "exception"

294

295

Testcase ID	JCA_8010_TestCase
Test Assertion	JCA-TA-8009
Description	Tests that policySets cannot be used on constructor parameters which are not references.
Artifacts	JCA_8010_TestCase.java Test_JCA_8010.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service1BadPolicySet.java definitions.xml
Expected output	Negative test: "exception"

296

297 JCA_8011_TestCase

298

Testcase ID	JCA_8011_TestCase
Test Assertion	JCA-TA-8010
Description	Tests that policySet annotations are correctly merged when they appear at different levels in a Java interface class. PolicySet1 satisfies testIntent3 and PolicySet2 satisfies testIntent5.
Artifacts	JCA_8010_TestCase.java Test_JCA_8010.composite TestInvocation.wsdl TestClient_0002.composite ASM_0002_Client.java Service1.java service6Impl.java service6Impl2.java Service6PolicySets.java TestIntent3.java TestIntent4.java definitions.xml

Expected output	Positive test: "JCA_8011 request service1 operation1 invoked service2 operation1 invoked"

299

300

2.6 Section 9

JCA_9001_TestCase

303

Testcase ID	JCA_9001_TestCase
Test Assertion	JCA-TA-9001
Description	Tests that if an invocation of the getService() method of the ComponentContext API is made for a reference that is 0..n or 1..n multiplicity, that the caller receives an IllegalArgumentException
Artifacts	JCA_9001_TestCase.java Test_JCA_9001.composite TestInvocation.wsdl TestClient_0002.composite Service1.java service1ContextImpl1.java service1Impl.java
Expected output	Positive test: "JCA_9001 request invokerService operation1 invoked IllegalArgumentException received"

304

JCA_9002_TestCase

306

Testcase ID	JCA_9002_TestCase
Test Assertion	JCA-TA-9002
Description	Tests that if an invocation of the getRequestContext() method of the ComponentContext API is made during the execution of a Java business method of a service operation on the same Java thread used to invoke the business method, then a non-null value is returned for the RequestContext
Artifacts	JCA_9002_TestCase.java Test_JCA_9002.composite

	TestInvocation.wsdl TestClient_0002.composite Service1.java service1ContextImpl1.java service1Impl.java
Expected output	Positive test: "JCA_9002 request service1 operation1 invoked RequestContext - serviceName: Service1"

307

308 JCA_9003_TestCase

309

Testcase ID	JCA_9003_TestCase
Test Assertion	JCA-TA-9003
Description	Tests that if an invocation of the getServiceReference() method of the RequestContext API is made during the execution of a Java business method, then the method returns a ServiceReference object that represents the service that was invoked
Artifacts	JCA_9003_TestCase.java Test_JCA_9003.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1Superset.java service1RequestContextImpl2.java
Expected output	Positive test: "JCA_9003 request service1 operation1 invoked ServiceReference obtained: service1 checkService operation2 invoked"

310

311 JCA_9004_TestCase

312

Testcase ID	JCA_9004_TestCase
Test Assertion	JCA-TA-9004
Description	Tests that if an invocation of the getServiceReference() method of the RequestContext API is made during the execution of a Java business method of a callback, then the method returns a ServiceReference object that represents the callback that was invoked
Artifacts	JCA_9004_TestCase.java

	Test_JCA_9004.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service3WithCallback.java Service3Callback.java service1CallbackContextImpl1.java service3Impl1.java
Expected output	Positive test: "JCA_9004 request serviceComposite1 operation1 invoked callbackService operation1 invokedserviceComposite1 callback1 invoked ServiceReference obtained: serviceComposite1 check callback1 invoked"

313

314 JCA_9005_TestCase

315

Testcase ID	JCA_9005_TestCase
Test Assertion	JCA-TA-9005
Description	Tests that if an invocation of the getRequestContext() method of the ComponentContext API is made during the execution of a Java business method of a callback operation on the same Java thread used to invoke the business method, then a non-null value is returned for the RequestContext
Artifacts	JCA_9005_TestCase.java Test_JCA_9005.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service3WithCallback.java Service3Callback.java service1CallbackContextImpl2.java service3Impl1.java
Expected output	Positive test: "JCA_9005 request serviceComposite1 operation1 invoked callbackService operation1 invoked serviceComposite1 callback1 invoked RequestContext - serviceName: reference1"

316

JCA_9006_TestCase

Testcase ID	JCA_9006_TestCase
Test Assertion	JCA-TA-9070 JCA-TA-9006 JCA-TA-9007 JCA-TA-9008 JCA-TA-9014
Description	Tests invocations of the ComponentContext.getServiceReference() method o invoked with a referenceName parameter for a reference of multiplicity 0..n o invoked for a reference which does not have the interface of the type in the businessInterface parameter o invoked with a referenceName value that does not match the name of any of the references of the component o invoked for a reference with multiplicity 0..1 which has no target service configured o invoked for a valid reference with a target service configured
Artifacts	JCA_9006_TestCase.java Test_JCA_9006.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1Impl.java Service1CCgetServiceReferenceImpl.java
Expected output	"JCA_9006 request service1 operation1 invoked getServiceReference expected IllegalArgumentException received for 0..n ref expected IllegalArgumentException received for invalid businessInterface expected IllegalArgumentException received for invalid reference name expected null ServiceReference for unwired 0..1 reference expected non-null ServiceReference for wired 1..1 reference service2 operation1 invoked" ;

JCA_9007_TestCase

Testcase ID	JCA_9007_TestCase
Test Assertion	JCA-TA-9009
Description	Tests invocations of the ComponentContext.getURI() method: o invocation returns a String containing the absolute URI of the component in the SCA Domain

Artifacts	JCA_9007_TestCase.java Test_JCA_9007.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1CCgetURIImpl.java
Expected output	"JCA_9007 request service1 operation1 invoked getURI getURI() returned URI = TEST_JCA_9007Component1"

322

323 JCA_9008_TestCase

324

Testcase ID	JCA_8_TestCase
Test Assertion	JCA-TA-9001 JCA-TA-9010 JCA-TA-9011 JCA-TA-9012 JCA-TA-9013
Description	Tests invocations of the ComponentContext.getService() method: o method throws an IllegalArgumentException if the referenceName reference has multiplicity 0..n o method returns a proxy object with the businessInterface for the reference referenceName, where that reference has a target service configured o method returns null for a 0..1 reference with no target service configured o method throws an IllegalArgumentException where the component does not have a reference with the name referenceName o method throws an IllegalArgumentException where the the referenceName reference does not have the interface defined in the businessInterface parameter
Artifacts	JCA_9008_TestCase.java Test_JCA_9008.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1CCgetServiceImpl.java
Expected output	"JCA_9008 request service1 operation1 invoked getService expected IllegalArgumentException received for 0..n ref expected IllegalArgumentException received for invalid businessInterface expected IllegalArgumentException received for invalid reference name expected null Service for unwired 0..1 reference expected non-null Service for wired

	1..1 reference service2 operation1 invoked"

JCA_9009_TestCase

Testcase ID	JCA_9009_TestCase
Test Assertion	JCA-TA-9015 JCA-TA-9016 JCA-TA-9017 JCA-TA-9018 JCA-TA-9019
Description	Tests invocations of the ComponentContext.getServices() method: o invocation returns a Collection containing a proxy object for each target service of the reference each implementing the interface supplied in the businessInterface parameter o invocation returns an empty Collection where the reference named by referenceName exists but has no target services configured o invocation throws an IllegalArgumentExeption when the named reference exists and has a multiplicity of 0..1 or 1..1 o invocation throws an IllegalArgumentExeption when the named reference does not exist o invocation throws an IllegalArgumentExeption when the named reference does not have an interface compatble with the interface in the businessInterface parameter
Artifacts	JCA_9009_TestCase.java Test_JCA_9009.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1CCgetServicesImpl.java
Expected output	"JCA_9009 request service1 operation1 invoked getServices expected IllegalArgumentException received for 1..1 ref expected IllegalArgumentException received for invalid businessInterface expected IllegalArgumentException received for invalid reference name expected empty Collection for unwired 0..n reference expected non-empty Collection for wired 0..n reference service2 operation1 invoked service3 operation1 invoked"

JCA_9010_TestCase

Testcase ID	JCA_9010_TestCase

Test Assertion	JCA-TA-9020 JCA-TA-9021 JCA-TA-9022 JCA-TA-9023 JCA-TA-9024
Description	Tests invocations of the ComponentContext.getServiceReferences() method: o invocation returns a Collection containing a ServiceReference object for each target service of the reference each implementing the interface supplied in the businessInterface parameter o invocation returns an empty Collection where the reference named by referenceName exists but has no target services configured o invocation throws an IllegalArgumentExeption when the named reference exists and has a multiplicity of 0..1 or 1..1 o invocation throws an IllegalArgumentExeption when the named reference does not exist o invocation throws an IllegalArgumentExeption when the named reference does not have an interface compatble with the interface in the businessInterface parameter
Artifacts	JCA_9010_TestCase.java Test_JCA_9010.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1CCgetServiceReferencesImpl.java
Expected output	"JCA_9010 request service1 operation1 invoked getServiceReferences expected IllegalArgumentException received for 1..1 ref expected IllegalArgumentException received for invalid businessInterface expected IllegalArgumentException received for invalid reference name expected empty Collection for unwired 0..n reference expected non-empty Collection for wired 0..n reference service2 operation1 invoked service3 operation1 invoked"

331

332 JCA_9011_TestCase

333

Testcase ID	JCA_9011_TestCase
Test Assertion	JCA-TA-9025 JCA-TA-9026
Description	Tests invocations of the ComponentContext.createSelfReference(businessInterface) method: o invocation returns a ServiceReference typed by the interface of the businessInterface parameter for one of the services offered by the component that has that interface

	o method thorws an IllegalArgumentException if the component has no service which has the interface specified by the businessInterface parameter
Artifacts	JCA_9011_TestCase.java Test_JCA_9011.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1CCcreateSelfReferenceAImpl.java
Expected output	"JCA_9011 request service1 operation1 invoked createSelfReference expected IllegalArgumentException received for invalid businessInterface" expected non-null ServiceReference service1 operation2 invoked"

334

335 ## JCA_9012_TestCase

336

Testcase ID	JCA_9012_TestCase
Test Assertion	JCA-TA-9027 JCA-TA-9028 JCA-TA-9029
Description	Tests invocations of the ComponentContext.createSelfReference(businessInterface, serviceName) method: o invocation returns a ServiceReference typed by the interface of the businessInterface parameter for the service offered by the component that has that interface and has the name specified by serviceName o method throws an IllegalArgumentException if the service named in the serviceName parameter does not have the interface specified by the businessInterface parameter o method throws an IllegalArgumentException if the component does not have a service with the name specified in the serviceName parameter
Artifacts	JCA_9012_TestCase.java Test_JCA_9012.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1CCcreateSelfReferenceBImpl.java
Expected output	"JCA_9012 request service1 operation1 invoked createSelfReference expected IllegalArgumentException received for invalid businessInterface expected IllegalArgumentException received for invalid service name expected non-null ServiceReference service1 operation2 invoked"

337

338 JCA_9013_TestCase

339

Testcase ID	JCA_9013_TestCase
Test Assertion	JCA-TA-9030 JCA-TA-9031 JCA-TA-9032 JCA-TA-9033
Description	Tests invocations of the ComponentContext.getProperty method: o invocation returns an object of the type specified by the type parameter containing the value in the component configuration for the property named by the propertyName parameter o method returns null for a property which has no value specified in the component configuration o method throws an IllegalArgumentException if the component does not have a property with the name specified in the propertyName parameter o method throws an IllegalArgumentException if the property named in the propertyName parameter does not have a type compatible with the supplied type parameter
Artifacts	JCA_9013_TestCase.java Test_JCA_9013.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1CCgetPropertyImpl.java
Expected output	"JCA_9013 request service1 operation1 invoked getProperty expected IllegalArgumentException received for invalid property type" expected IllegalArgumentException received for invalid property name expected null value for unconfigured optional property expected non-null property value for required property: 2.456"

340

341 JCA_9014_TestCase

342

Testcase ID	JCA_9014_TestCase
Test Assertion	JCA-TA-9034 JCA-TA-9035
Description	Tests invocations of the ComponentContext.cast method: o invocation returns a ServiceReference object typed by the same interface as specified by the reference proxy object supplied in the target parameter

	o method throws an IllegalArgumentException if the supplied target parameter is not an SCA reference proxy object
Artifacts	JCA_9014_TestCase.java Test_JCA_9014.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1CCcastImpl.java
Expected output	"JCA_9014 request service1 operation1 invoked cast expected IllegalArgumentException received for invalid reference proxy expected non-null ServiceReference ServiceReference has correct business interface"

343

344 JCA_9015_TestCase

345

Testcase ID	JCA_5_TestCase
Test Assertion	JCA-TA-9039 JCA-TA-9040 JCA-TA-9041 JCA-TA-9042 JCA-TA-9043 JCA-TA-9044 JCA-TA-9045 JCA-TA-9046 JCA-TA-9047 JCA-TA-9048
Description	Tests invocations of the RequestContext.method: o getServiceName invocation returns the name of the service for which the operation is being processed when called from a thread processing a service operation o getServiceName method returns null when called from a thread not processing a service operation or callback operation o getCallbackReference returns a ServiceReference typed by the interface of the callback when invoked by a thread processing an operation of a bidirectional service o getCallbackReference returns null when invoked by a thread processing an operation of a service which is not bidirectional o getCallbackReference returns null when invoked by a thread not processing an operation of a service o getCallback returns a reference proxy typed by the interface of the callback when invoked by a thread processing an operation of a bidirectional service

	o getCallback returns null when invoked by a thread processing an operation of a service which is not bidirectional o getCallback returns null when invoked by a thread not processing an operation of a service o getServiceReference method returns a ServiceReference representing the callback service when invoked from a thread processing a callback operation o getServiceReference method returns null when called from a thread not involved in processing either a service operation or a callback operation
Artifacts	JCA_9015_TestCase.java Test_JCA_9015.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1RCgetServiceNameImpl.java
Expected output	

346

347 JCA_9016_TestCase

348

Testcase ID	JCA_9016_TestCase
Test Assertion	JCA-TA-9051 JCA-TA-9052 JCA-TA-9053 JCA-TA-9054 JCA-TA-9055 JCA-TA-9056 JCA-TA-9057 JCA-TA-9058 JCA-TA-9059 JCA-TA-9060
Description	Tests invocations of the SCAClientFactory API class: o invocation of newInstance(URI) with a valid URI value returns an object which implements the SCAClientFactory class for the SCA Domain identified by the domainURI parameter o invocation of newInstance(URI) with an invalid URI value causes a NoSuchDomainException to be thrown o invocation of newInstance(Properties, URI) with a valid URI value returns an object which implements the SCAClientFactory class for the SCA Domain identified by the domainURI parameter o invocation of newInstance(Properties, URI) with an invalid URI value causes a NoSuchDomainException to be thrown o invocation of newInstance(Classloader, URI) with a valid URI value

	returns an object which implements the SCAClientFactory class for the SCA Domain identified by the domainURI parameter o invocation of newInstance(Classloader, URI) with an invalid URI value causes a NoSuchDomainException to be thrown o invocation of newInstance(Properties, Classloader, URI) with a valid URI value returns an object which implements the SCAClientFactory class for the SCA Domain identified by the domainURI parameter o invocation of newInstance(Properties, Classloader, URI) with an invalid URI value causes a NoSuchDomainException to be thrown o invocation of getService with a serviceURI that identifies a service in the Domain which has the business interface identified bby the interfaze parameter returns a reference proxy object which can be used to invoke operations on the identified service o invocation of getService with a serviceURI for which there is no service in the domain throws a NoSuchServiceException
Artifacts	JCA_9016_TestCase.java Test_JCA_9016.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1SCAClientImpl.java
Expected output	"JCA_9016 request service1 operation1 invoked SCAClient expected non-null client factory expected exception received for invalid URI: NoSuchDomainException expected non-null client factory expected exception received for invalid URI: NoSuchDomainException expected non-null client factory expected exception received for invalid URI: NoSuchDomainExceptionexpected non-null client factory expected exception received for invalid URI: NoSuchDomainException expected non-null reference proxyservice2 operation1 invoked expected NoSuchServiceException exception received for getService()"

2.7 Section 10

JCA_10001_TestCase

Testcase ID	JCA_10001_TestCase
Test Assertion	JCA-TA-10001
Description	Tests that Java annotations are not used in improper locations. This testcase has an @Property annotation on a method parameter where the method is not a constructor.
Artifacts	JCA_10001_TestCase.java Test_JCA_10001.composite

	TestInvocation.wsdl TestClient_0002.composite Service1.wsdl service1BadAnnotation.java Service1.java
Expected output	

353

354 JCA_10002_TestCase

355

Testcase ID	JCA_10002_TestCase
Test Assertion	JCA-TA-10002
Description	Tests that Java annotations are not used on static methods. This testcase has an @Property annotation on a static setter method.
Artifacts	JCA_10002_TestCase.java Test_JCA_10002.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl service1StaticAnnotation.java Service1.java
Expected output	

356

357 JCA_10003_TestCase

358

Testcase ID	JCA_10003_TestCase
Test Assertion	JCA-TA-10003
Description	Tests that Java annotations are not used on static fields. This testcase has an @Property annotation on a static field.
Artifacts	JCA_10003_TestCase.java Test_JCA_10003.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl service1StaticAnnotation.java

	Service1.java
Expected output	

359

360 JCA_10004_TestCase

361

Testcase ID	JCA_10004_TestCase
Test Assertion	JCA-TA-10031
Description	Tests that the @Callback annotation has to be specified with any parameters when used as the injection point of a callback reference.
Artifacts	JCA_10004_TestCase.java Test_JCA_10004.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1CalbackImpl.java service3BadCallbackImpl.java Service3Callback.java Service3WithCallback.java
Expected output	

362

363 JCA_10005_TestCase

364

Testcase ID	JCA_10005_TestCase
Test Assertion	JCA-TA-10005
Description	Tests that @Constructor works correctly with annotated parameters.
Artifacts	JCA_10005_TestCase.java Test_JCA_10005.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1ConstrImpl.java
Expected output	Positive:

JCA_10006_TestCase

Testcase ID	JCA_10006_TestCase
Test Assertion	JCA-TA-10005
Description	Tests that the runtime raises an error when an @Constructor annotated constructor does not have all of it's parameters annotated.
Artifacts	JCA_10006_TestCase.java Test_JCA_10006.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1BadConstrImpl.java
Expected output	

JCA_10007_TestCase

Testcase ID	JCA_10007_TestCase
Test Assertion	JCA-TA-10006
Description	Tests that the runtime raises an error when an @Destroy method has a non-void return and parameters.
Artifacts	JCA_10007_TestCase.java Test_JCA_10007.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1BadDestroyImpl.java
Expected output	

JCA_10008_TestCase

Testcase ID	JCA_10008_TestCase
Test Assertion	JCA-TA-10009
Description	Tests that the runtime raises an error when an @Init method has a non-void return and parameters.
Artifacts	JCA_10008_TestCase.java Test_JCA_10008.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1BadInitImpl.java
Expected output	

374

375 JCA_10009_TestCase

376

Testcase ID	JCA_10009_TestCase
Test Assertion	JCA-TA-10011
Description	Tests that the runtime raises an error when an @Property annotated field is marked as final.
Artifacts	JCA_10009_TestCase.java Test_JCA_10009.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1BadPropImpl.java
Expected output	

377

378 JCA_10010_TestCase

379

Testcase ID	JCA_10010_TestCase
Test Assertion	JCA-TA-10012
Description	Tests that @Property specifies a name when used with @Constructor.
Artifacts	JCA_10010_TestCase.java Test_JCA_10010.composite

	TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1ConstrBadPropImpl.java
Expected output	

380

381 JCA_10011_TestCase

382

Testcase ID	JCA_10011_TestCase
Test Assertion	JCA-TA-10013
Description	Tests that @Property is not required=true when used with @Constructor.
Artifacts	JCA_10011_TestCase.java Test_JCA_10011.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1ConstrBadPropImpl.java
Expected output	

383

384 JCA_10012_TestCase

385

Testcase ID	JCA_10012_TestCase
Test Assertion	JCA-TA-10014
Description	Tests that multi valued properties are introspected correctly.
Artifacts	JCA_10012_TestCase.java Test_JCA_10012.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1MultiPropImpl.java

Expected output	Positive:

386

387 JCA_10013_TestCase

388

Testcase ID	JCA_10013_TestCase
Test Assertion	JCA-TA-10016
Description	Tests that intent annoations can have qualifiers.
Artifacts	JCA_10013_TestCase.java Test_JCA_10013.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1IntentQualifier.java
Expected output	Positive:

389

390 JCA_10014_TestCase

391

Testcase ID	JCA_10014_TestCase
Test Assertion	JCA-TA-10018
Description	Tests that @Reference has a name when used with @Constructor.
Artifacts	JCA_10014_TestCase.java Test_JCA_10014.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1ConstrBadRefImpl.java service1Impl.java
Expected output	Negative:

392

393 JCA_10015_TestCase

394

Testcase ID	JCA_10015_TestCase
Test Assertion	JCA-TA-10019
Description	Tests that @Reference has required=true when used with @Constructor.
Artifacts	JCA_10015_TestCase.java Test_JCA_10015.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1ConstrBadRefImpl.java service1Impl.java
Expected output	Negative:

395

396 JCA_10016_TestCase

397

Testcase ID	JCA_10016_TestCase
Test Assertion	JCA-TA-10022
Description	Tests that @Reference(required=false) is introspected as multiplicity 0..n.
Artifacts	JCA_10016_TestCase.java Test_JCA_10016.composite TestInvocation.wsdl TestClient_0002.composite Service1.wsdl Service1.java service1OptMultiRef.java
Expected output	Positive:

398

399

401 JCA_10044_TestCase

402

Testcase ID	JCA_10044_TestCase
Test Assertion	JCA-TA-10024
Description	Tests that where a component reference with multiplicity of 0..1 is not wired, that the reference injected into the implementation is null
Artifacts	JCA_10044_TestCase.java Test_JCA_10044.composite TestInvocation.wsdl TestClient_0002.composite Service1.java service1Impl6.java
Expected output	Positive test: "JCA_10044 request service1 operation1 invoked reference is null"

403 JCA_10045_TestCase

404

Testcase ID	JCA_10045_TestCase
Test Assertion	JCA-TA-10025
Description	Tests that where a component reference with multiplicity of 0..n is not wired, that the reference injected into the implementation is an empty array or collection
Artifacts	JCA_10045_TestCase.java Test_JCA_10045.composite TestInvocation.wsdl TestClient_0002.composite Service1.java service1Impl4.java
Expected output	Positive test: "JCA_10045 request service1 operation1 invoked reference array is empty"

405

406 JCA_10046_TestCase

407

Testcase ID	JCA_10046_TestCase
Test Assertion	JCA-TA-10045
Description	Tests that where the Java interface of a service is marked remotable, the interface is translatable into a WSDL portType
Artifacts	JCA_10046_TestCase.java Test_JCA_10046.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Unmappable.java unmappableServiceImpl.java
Expected output	Negative test: "exception"

408

409 JCA_10047_TestCase

410

Testcase ID	JCA_10047_TestCase
Test Assertion	JCA-TA-10046
Description	Tests that the @Scope annotation is only valid when applied to a component implementation class
Artifacts	JCA_10047_TestCase.java Test_JCA_10047.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite Service1.wsdl
Expected output	Negative test: "exception"

411

412 JCA_10048_TestCase

413

Testcase ID	JCA_10048_TestCase
Test Assertion	JCA-TA-10047
Description	Tests that where a class is annotated with a @Service annotation which declares a set of services, that the implementation class implements all of

	the methods of all the declared service interfaces
Artifacts	JCA_10048_TestCase.java Test_JCA_10048.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite Service1.wsdl
Expected output	Negative test: "exception"

414

415 JCA_10049_TestCase

416

Testcase ID	JCA_10049_TestCase
Test Assertion	JCA-TA-10050
Description	Tests that where a @Service annotation has a @names attribute, that the number of entries in the array of @names is the same as the number of entries in the array of the @value attribute
Artifacts	JCA_10049_TestCase.java Test_JCA_10049.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite Service1.wsdl
Expected output	Negative test: "exception"

417

418 JCA_10050_TestCase

419

Testcase ID	JCA_10050_TestCase
Test Assertion	JCA-TA-10055
Description	Tests that where an implementation class declares 2 or more services, that each service interface class has a distinct Java simple name
Artifacts	JCA_10050_TestCase.java Test_JCA_10050.composite TestInvocation.wsdl

	TestClient_0002.composite TestComposite1.composite Service1.wsdl
Expected output	Negative test: "exception"

420

421 ## JCA_10051_TestCase

422

Testcase ID	JCA_10051_TestCase
Test Assertion	JCA-TA-10057
Description	Tests that a @Service annotation has at least one element in its @value array
Artifacts	JCA_10051_TestCase.java Test_JCA_10051.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite Service1.wsdl
Expected output	Negative test: "exception"

423

424 ## JCA_10052_TestCase

425

Testcase ID	JCA_10052_TestCase
Test Assertion	JCA-TA-10058
Description	Tests that all the entries in the @names array of a @Service annotation are unique
Artifacts	JCA_10052_TestCase.java Test_JCA_10052.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite Service1.wsdl
Expected output	Negative test:

	"exception"

2.8 Section 11

JCA_11001_TestCase

Testcase ID	JCA_11001_TestCase
Test Assertion	JCA-TA-11001
Description	Tests that a Java interface which does not have @WebService annotation which is mapped to WSDL by an SCA runtime is treated as if it did contain a @WebService annotation
Artifacts	JCA_11001_TestCase.java Test_JCA_11001.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite Service1.java Service1ImplWSService.wsdl service1Impl.java service1Impl2.java
Expected output	Positive test: "JCA_11001 request service1 operation1 invoked service2 operation1 invoked"

JCA_11002_TestCase

Testcase ID	JCA_11002_TestCase
Test Assertion	JCA-TA-11002
Description	Tests that a Java interface containing a @org.oasisopen.sca.annotation.OneWay annotation is mapped to WSDL as if it contained a @javax.jws.OneWay annotation
Artifacts	JCA_11002_TestCase.java Test_JCA_11002.composite TestInvocation.wsdl

	TestClient_0002.composite Service1.java ServiceOneWay.java service1OneWayCallerImpl.java serviceOneWayImpl.java
Expected output	Positive test: "JCA_11002 request service1 operation1 invoked service2 operation1 invoked OneWay method inonly previously called with testInvoke"

JCA_11003_TestCase

Testcase ID	JCA_11003_TestCase
Test Assertion	JCA-TA-11003
Description	Tests that if a Java interface mapped from a WSDL interface by an SCA runtime contains a @WebService annotation, the interface is treated as if it had a @Remotable annotation
Artifacts	JCA_11003_TestCase.java Test_JCA_11003.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1WS.java service1ImplWS.java
Expected output	Positive test: "JCA_11003 request service1 operation1 invoked"

JCA_11004_TestCase

Testcase ID	JCA_11004_TestCase
Test Assertion	JCA-TA-11004
Description	Tests that if a Java interface used as a service interface contains JAXB 2.1 datatypes and annotations, the SCA runtime is able to map the interface to WSDL.
Artifacts	JCA_11004_TestCase.java Test_JCA_11004.composite TestInvocation.wsdl TestClient_0002.composite

	Service1.wsdl ServiceJAXB.java service1JAXBCallerImpl.java serviceJAXBImpl.java
Expected output	Positive test: "JCA_11004 request service1 operation1 invoked service2 operation1 invoked JAXB parameters received"

438 ## JCA_11005_TestCase

439

Testcase ID	JCA_11005_TestCase
Test Assertion	JCA-TA-11006
Description	Tests that a Java interface used to declare an SCA service interface in an <interface.java/> element does not contain the additional client-side asynchronous polling and callback methods defined by JAX-WS.
Artifacts	JCA_11005_TestCase.java Test_JCA_11005.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1WithAsyncMethods.java service1Impl.java
Expected output	Negative test: "exception"

440 ## JCA_11006_TestCase

441

Testcase ID	JCA_11006_TestCase
Test Assertion	JCA-TA-11006
Description	Tests that where a Java interface used to declare an SCA reference interface contains the additional client side asynchronous polling and callback methods defined by JAX-WS, that the interface is treated as valid and is used for the reference.
Artifacts	JCA_11006_TestCase.java Test_JCA_11006.composite TestInvocation.wsdl TestClient_0002.composite

	Service1.java Service1WithAsyncMethods.java service1AsyncRefImpl.java service1Impl.java
Expected output	Positive test: "JCA_11006 request service1 operation1 invoked service2 operation1 invoked"

JCA_11007_TestCase

Testcase ID	JCA_11007_TestCase
Test Assertion	JCA-TA-11008
Description	Tests that where a Java implementation class has an SCA reference which uses an interface which contains the additional client side asynchronous polling and callback methods defined by JAX-WS, that the introspected reference element in the component type for that implementation has an interface that does not contain the additional asynchronous polling and callback methods.
Artifacts	JCA_11007_TestCase.java Test_JCA_11007.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1WithAsyncMethods.java service1AsyncRefImpl.java service1Impl.java
Expected output	Positive test: "JCA_11007 request service1 operation1 invoked service2 operation1 invoked"

JCA_11008_TestCase

Testcase ID	JCA_11008_TestCase
Test Assertion	JCA-TA-11009
Description	Tests that where a Java implementation class has an SCA reference which uses an interface which contains the additional client side asynchronous polling and callback methods defined by JAX-WS, that the implementation class is able to use the asynchronous polling and callback methods with the semantics described in the JAX-WS 2.1 specification.
Artifacts	JCA_11008_TestCase.java

	Test_JCA_11008.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1WithAsyncMethods.java service1AsyncRefImpl.java service1Impl.java
Expected output	Positive test: "JCA_11008 request service1 operation1 invoked service2 operation1 invoked service3 operation1 invoked"

446

447 JCA_11009_TestCase

448

Testcase ID	JCA_11009_TestCase
Test Assertion	JCA-TA-11013
Description	Tests that where a Java implementation class has an @WebService annotation with the @endpointInterface attribute has its (Java) service interface defined by the interface referenced by the @endpointInterface attribute
Artifacts	JCA_11009_TestCase.java Test_JCA_11009.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service3Operations.java Service1Calls3OperationsImpl.java Service3OperationsWebServiceImpl.java
Expected output	Positive test: "JCA_11009 request service1 operation1 invoked"

449

450 JCA_11010_TestCase

451

Testcase ID	JCA_11010_TestCase
Test Assertion	JCA-TA-11014

Description	Tests that where a Java service interface contains a @WebParam annotation with its @header attribute set to "true", that the interface is treated as if it had the SOAP intent applied
Artifacts	JCA_11010_TestCase.java Test_JCA_11010.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1Impl.java Service1WithWebParam.java definitions.xml (in contribution JCA_General_POJO)
Expected output	Negative test: "exception"

452

453 JCA_11011_TestCase

454

Testcase ID	JCA_11011_TestCase
Test Assertion	JCA-TA-11015
Description	Tests that where a Java service interface contains a @WebResult annotation with its @header attribute set to "true", that the interface is treated as if it had the SOAP intent applied
Artifacts	JCA_11011_TestCase.java Test_JCA_11011.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1Impl.java Service1WithWebResult.java definitions.xml (in contribution JCA_General_POJO)
Expected output	Negative test: "exception"

455

456 JCA_11012_TestCase

457

Testcase ID	JCA_11012_TestCase
Test Assertion	JCA-TA-11020
Description	Tests that where a Java service interface contains a @SOAPBinding annotation with its @header attribute set to "true", that the interface is treated as if it had the SOAP intent applied
Artifacts	JCA_11012_TestCase.java Test_JCA_11012.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1Impl.java Service1WithWebSoapBinding.java definitions.xml (in contribution JCA_General_POJO)
Expected output	Negative test: "exception"

458

459 ## JCA_11013_TestCase

460

Testcase ID	JCA_11013_TestCase
Test Assertion	JCA-TA-11016
Description	Tests that where a Java service implementation class contains a @ServiceMode annotation that the service interface is treated as if it had the SOAP intent applied
Artifacts	JCA_11013_TestCase.java Test_JCA_11013.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1WithServiceModeImpl.java definitions.xml (in contribution JCA_General_POJO)
Expected output	Negative test: "exception"

461

462 JCA_11014_TestCase

463

Testcase ID	JCA_11014_TestCase
Test Assertion	JCA-TA-11017
Description	Tests that where a Java interface used to the define the interface of an SCA service does not contains an @WebServiceClient annotation
Artifacts	JCA_11014_TestCase.java Test_JCA_11014.composite TestInvocation.wsdl TestClient_0002.composite Service1WithWebServiceClient.java service1Impl.java
Expected output	Negative test: "exception"

464

465 JCA_11015_TestCase

466

Testcase ID	JCA_11015_TestCase
Test Assertion	JCA-TA-11018
Description	Tests that where a Java service implementation class contains a @WebServiceProvider annotation, the class is treated as if it is annotated with an SCA @Remotable annotation
Artifacts	JCA_11015_TestCase.java Test_JCA_11015.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1WithWebServiceProviderImpl.java
Expected output	Positive test: "JCA_11015 request service1 operation1 invoked"

467

468 JCA_11016_TestCase

469

Testcase ID	JCA_11016_TestCase
Test Assertion	JCA-TA-11019
Description	Tests that where a Java service implementation class contains a @WebServiceProvider annotation and the @wsdlLocation attribute of that annotation is declared, referencing a WSDL document, that the class is treated as if its service interface is defined by the referenced WSDL document
Artifacts	JCA_11016_TestCase.java Test_JCA_11016.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1WithWebServiceProviderImpl.java
Expected output	Negative test: "exception"

470

471 JCA_11017_TestCase

472

Testcase ID	JCA_11017_TestCase
Test Assertion	JCA-TA-9066 JCA-TA-9068 JCA-TA-11009 JCA-TA-11010
Description	Tests that where a client component Java implementation has a reference that uses the async callback client API to invoke a service which is an "asyncInvocation" asynchronous service, where the response to an operation is received some arbitrary time after the request is made, the client is able to make the invocation successfully and get back the result - covers both a regular response and a fault response.
Artifacts	JCA_11017_TestCase.java Test_JCA_11017.composite TestInvocation.wsdl TestClient_0002.composite Service1.java Service1AsyncServer.java

	service1AsyncRefImpl.java Service1AsyncServerImpl.java
Expected output	Positive test: "JCA_11017 request service1 operation1 invoked Future returned service2 operation1 invoked asynchrono org.oasisopen.sca.test.BusinessFault1: service2 operation1 invoked asynchronously"

473

3 Cross Mapping of Test Assertions to TestCases

Test Assertion	Test Cases
JCA-TA-2001	JCA_2001_TestCase
JCA-TA-2002	JCA_2002_TestCase
JCA-TA-2003	JCA_2003_TestCase
JCA-TA-2004	JCA_2004_TestCase
JCA-TA-2005	JCA_2005_TestCase
JCA-TA-2006	JCA_2006_TestCase
JCA-TA-2007	JCA_2007_TestCase
JCA-TA-2008	JCA_2008_TestCase
JCA-TA-2009	Optional - not tested
JCA-TA-2010	JCA_2009_TestCase
JCA-TA-2011	JCA_2010_TestCase

Test Assertion	Test Cases
JCA-TA-3001	JCA_3001_TestCase JCA_3002_TestCase
JCA-TA-3002	JCA_3003_TestCase JCA_3004_TestCase
JCA-TA-3003	JCA_3005_TestCase JCA_3006_TestCase JCA_3007_TestCase
JCA-TA-3004	JCA_3008_TestCase
JCA-TA-3005	JCA_3009_TestCase JCA_3010_TestCase
JCA-TA-3006	JCA_3011_TestCase
JCA-TA-3007	JCA_3012_TestCase
JCA-TA-3008	JCA_3013_TestCase
JCA-TA-3009	JCA_3014_TestCase

Test Assertion	Test Cases
JCA-TA-4001	JCA_4001_TestCase
JCA-TA-4002	JCA_4001_TestCase
JCA-TA-4003	JCA_4001_TestCase
JCA-TA-4004	JCA_4002_TestCase

JCA-TA-4005	JCA_4001_TestCase
JCA-TA-4006	JCA_4001_TestCase
JCA-TA-4007	JCA_4001_TestCase
JCA-TA-4008	JCA_4001_TestCase
JCA-TA-4009	JCA_4001_TestCase
JCA-TA-4010	JCA_4003_TestCase
JCA-TA-4011	JCA_4001_TestCase
JCA-TA-4012	JCA_4004_TestCase
JCA-TA-4013	JCA_4001_TestCase
JCA-TA-4014	JCA_4001_TestCase
JCA-TA-4015	JCA_4005_TestCase
JCA-TA-4016	JCA_4001_TestCase
JCA-TA-4017	JCA_4001_TestCase
JCA-TA-4018	JCA_4001_TestCase
JCA-TA-4019	untestable
JCA-TA-4020	JCA_4001_TestCase
JCA-TA-4021	JCA_4001_TestCase
JCA-TA-4022	JCA_4007_TestCase
JCA-TA-4023	JCA_4001_TestCase
JCA-TA-4024	JCA_4008_TestCase

478

479

Test Assertion	Test Cases
JCA-TA-7001	JCA_7001_TestCase
JCA-TA-7002	JCA_7002_TestCase
JCA-TA-7003	JCA_7003_TestCase JCA_7004_TestCase
JCA-TA-7004	JCA_7003_TestCase
JCA-TA-7005	JCA_7003_TestCase
JCA-TA-7006	JCA_7005_TestCase JCA_7006_TestCase
JCA-TA-7007	JCA_7003_TestCase

480

Test Assertion	Test Cases
JCA-TA-8001	JCA_8001_TestCase

JCA-TA-8002	JCA_8002_TestCase JCA_8005_TestCase
JCA-TA-8003	JCA_8003_TestCase JCA_8005_TestCase
JCA-TA-8004	JCA_8004_TestCase JCA_8005_TestCase
JCA-TA-8005	JCA_8006_TestCase
JCA-TA-8006	JCA_8007_TestCase
JCA-TA-8007	JCA_8008_TestCase
JCA-TA-8008	JCA_8009_TestCase
JCA-TA-8009	JCA_8010_TestCase
JCA-TA-8010	JCA_8011_TestCase

481

482

Test Assertion	Test Cases
JCA-TA-9001	JCA_9001_TestCase JCA_9008_TestCase
JCA-TA-9002	JCA_9002_TestCase
JCA-TA-9003	JCA_9003_TestCase
JCA-TA-9004	JCA_9004_TestCase
JCA-TA-9005	JCA_9005_TestCase
JCA-TA-9006	JCA_9006_TestCase
JCA-TA-9007	JCA_9006_TestCase
JCA-TA-9008	JCA_9006_TestCase
JCA-TA-9009	JCA_9007_TestCase
JCA-TA-9010	JCA_9008_TestCase
JCA-TA-9011	JCA_9008_TestCase
JCA-TA-9012	JCA_9008_TestCase
JCA-TA-9013	JCA_9008_TestCase
JCA-TA-9014	JCA_9006_TestCase
JCA-TA-9015	JCA_9009_TestCase
JCA-TA-9016	JCA_9009_TestCase
JCA-TA-9017	JCA_9009_TestCase
JCA-TA-9018	JCA_9009_TestCase
JCA-TA-9019	JCA_9009_TestCase
JCA-TA-9020	JCA_9010_TestCase

JCA-TA-9021	JCA_9010_TestCase
JCA-TA-9022	JCA_9010_TestCase
JCA-TA-9023	JCA_9010_TestCase
JCA-TA-9024	JCA_9010_TestCase
JCA-TA-9025	JCA_9011_TestCase
JCA-TA-9026	JCA_9011_TestCase
JCA-TA-9027	JCA_9012_TestCase
JCA-TA-9028	JCA_9012_TestCase
JCA-TA-9029	JCA_9012_TestCase
JCA-TA-9030	JCA_9013_TestCase
JCA-TA-9031	JCA_9013_TestCase
JCA-TA-9032	JCA_9013_TestCase
JCA-TA-9033	JCA_9013_TestCase
JCA-TA-9034	JCA_9014_TestCase
JCA-TA-9035	JCA_9014_TestCase
JCA-TA-9036	Untestable - no standard means of configuring a service to have a JAAS subject
JCA-TA-9037	Untestable - no standard means of configuring a service to have a JAAS subject
JCA-TA-9038	Untestable - no standard means of configuring a service to have a JAAS subject
JCA-TA-9039	JCA_9015_TestCase
JCA-TA-9040	JCA_9015_TestCase
JCA-TA-9041	JCA_9015_TestCase
JCA-TA-9042	JCA_9015_TestCase
JCA-TA-9043	JCA_9015_TestCase
JCA-TA-9044	JCA_9015_TestCase
JCA-TA-9045	JCA_9015_TestCase
JCA-TA-9046	JCA_9015_TestCase
JCA-TA-9047	JCA_9015_TestCase
JCA-TA-9048	JCA_9015_TestCase
JCA-TA-9049	JCA_9015_TestCase
JCA-TA-9050	JCA_9014_TestCase
JCA-TA-9051	JCA_9016_TestCase
JCA-TA-9052	JCA_9016_TestCase
JCA-TA-9053	JCA_9016_TestCase

JCA-TA-9054	JCA_9016_TestCase
JCA-TA-9055	JCA_9016_TestCase
JCA-TA-9056	JCA_9016_TestCase
JCA-TA-9057	JCA_9016_TestCase
JCA-TA-9058	JCA_9016_TestCase
JCA-TA-9059	JCA_9016_TestCase
JCA-TA-9060	JCA_9016_TestCase
JCA-TA-9061	Impossible to obtain a SCAClientFactory instance with an invalid domainURI - untestable
JCA-TA-9062	Untestable since this is a protected method.
JCA-TA-9063	Impossible to obtain a SCAClientFactory instance with an invalid domainURI - untestable
JCA-TA-9064	Untestable unless we make the vendor implementation of SCAClientFactoryFinder a conformance target
JCA-TA-9065	Untestable unless we make the vendor implementation of SCAClientFactoryFinder a conformance target
JCA-TA-9066	JCA_7003_TestCase
JCA-TA-9067	JCA_7005_TestCase
JCA-TA-9068	JCA_7006_TestCase
JCA-TA-9069	JCA_7006_TestCase
JCA-TA-9070	JCA_9006_TestCase

483

484

Test Assertion	Test Cases
JCA-TA-10001	JCA_10001_TestCase
JCA-TA-10002	JCA_10002_TestCase
JCA-TA-10003	JCA_10003_TestCase
JCA-TA-10004	JCA_2009_TestCase JCA_2010_TestCase
JCA-TA-10031	JCA_10004_TestCase
JCA-TA-10005	JCA_4001_TestCase JCA_10005_TestCase JCA_10006_TestCase
JCA-TA-10006	JCA_4001_TestCase JCA_10007_TestCase
JCA-TA-10007	JCA_4001_TestCase

JCA-TA-10008	JCA_2005_TestCase
JCA-TA-10009	JCA_10008_TestCase
JCA-TA-10010	JCA_4001_TestCase
JCA-TA-10011	JCA_10009_TestCase
JCA-TA-10012	JCA_10005_TestCase JCA_10010_TestCase
JCA-TA-10013	JCA_10005_TestCase JCA_10011_TestCase
JCA-TA-10014	JCA_10012_TestCase
JCA-TA-10015	JCA_2001_TestCase
JCA-TA-10016	JCA_10013_TestCase
JCA-TA-10017	JCA_2001_TestCase
JCA-TA-10018	JCA_10014_TestCase
JCA-TA-10019	JCA_10015_TestCase
JCA-TA-10020	JCA_2001_TestCase
JCA-TA-10021	JCA_2001_TestCase
JCA-TA-10022	JCA_10016_TestCase
JCA-TA-10023	JCA_2008_TestCase
JCA-TA-10024	JCA_10044_TestCase
JCA-TA-10025	JCA_10045_TestCase
JCA-TA-10026	optional requirement - no test
JCA-TA-10027	optional requirement - no test
JCA-TA-10028	optional requirement - no test
JCA-TA-10029	optional requirement - no test
JCA-TA-10030	requires undeploy API - no test
JCA-TA-10031	no TA
JCA-TA-10032	no TA
JCA-TA-10033	requires redeploy API - no test
JCA-TA-10034	optional requirement - no test
JCA-TA-10035	requires undeploy API - no test
JCA-TA-10036	requires undeploy API - no test
JCA-TA-10037	requires redeploy API - no test
JCA-TA-10038	requires deploy/redeploy API - no test
JCA-TA-10039	requires deploy/redeploy API - no test
JCA-TA-10040	requires deploy/redeploy API - no test
JCA-TA-10041	requires deploy/redeploy API - no test

JCA-TA-10042	requires deploy/redeploy API - no test
JCA-TA-10043	requires deploy/redeploy API - no test
JCA-TA-10044	requires optional support for rewiring - no test
JCA-TA-10045	JCA_10046_TestCase
JCA-TA-10046	JCA_10047_TestCase
JCA-TA-10047	JCA_10048_TestCase
JCA-TA-10050	JCA_10049_TestCase
JCA-TA-10055	JCA_10050_TestCase
JCA-TA-10056	requires deploy/redeploy API - no test
JCA-TA-10057	JCA_10051_TestCase
JCA-TA-10058	JCA_10052_TestCase

485

486

Test Assertion	Test Cases
JCA-TA-11001	JCA_11001_TestCase
JCA-TA-11002	JCA_11002_TestCase
JCA-TA-11003	JCA_11003_TestCase
JCA-TA-11004	JCA_11004_TestCase
JCA-TA-11005	JCA_11005_TestCase
JCA-TA-11006	JCA_11006_TestCase
JCA-TA-11007	optional - no test case
JCA-TA-11008	JCA_11007_TestCase
JCA-TA-11009	JCA_11008_TestCase
JCA-TA-11010	JCA_7003_TestCase JCA_7004_TestCase
JCA-TA-11011	JCA_11003_TestCase
JCA-TA-11012	JCA_3014_TestCase
JCA-TA-11013	JCA_11009_TestCase
JCA-TA-11014	JCA_11010_TestCase
JCA-TA-11015	JCA_11011_TestCase
JCA-TA-11016	JCA_11013_TestCase
JCA-TA-11017	JCA_11014_TestCase
JCA-TA-11018	JCA_11015_TestCase
JCA-TA-11019	JCA_11016_TestCase
JCA-TA-11020	JCA_11012_TestCase
JCA-TA-11021	Tested by a range of the testcases in this test suite

	- default mappings - mappings influenced by JAXWS annotations - JAXB type mapping are all dealt with in various different testcases, so there is no need to create a new specific testcase for this general requirement.

487

4 Catalog of Test Artifacts

4.1 Composite Files - lower level

Name	Valid	Description
Marked *"Impl Specific"* if it is implementation type specific		*Services and References use interface Service1 unless described otherwise*
CompositeScope.composite	**Y**	Composite containing a component with a composite scope implementation (service1CompositeImpl.java) invoked by a variety of other components
TestClient_002.composite	**Y**	Test invocation composite that presents the TestInvocation interface to the test runner client.

4.2 Java Interfaces

Name	Description
Constants.java	Policy related constants declared in an interface
DataStore.java	Data store interface used as service interface for data store component
MultipleService.java	Interface with a large number of operations, used to check multiple invocations to a service component
ParallelService.java	
Service1.java	Interface with 1 operation - "operation1", string input, string output
Service1AsyncServer.java	SCA Async Server form of the Service1 interface
Service1Superset.java	A superset of the Service1 interface - "operation1", string input, string output - "operation2", string input, string output
Service1WithAsyncMethods.java	Service1 service interface - this version has all the additional JAXWS async client methods added
Service1WithFaults.java	Version of the Service1 interface with a series of declared business faults on operation1

Service1WithFaultsAsyncServer.java	Service1 service interface with with a series of declared business faults on operation1 - Asynchronous server version
Service1WS.java	Service1 service interface - a variant which has a @WebService annotation - and no @Remotable annotation
Service2.java	A test service interface, designed to be incompatible with the Service1 interface 1 operation "operation2", int input, int output
Service3Callback.java	A remotable service interface used for a Callback from Service3 1 operation "callback1", string input, string output
Service3Operations.java	A service with 3 operations - operation1: input String response String - operation2: input String response String - operation3: input String response String
Service3OperationsWSDL.java	A service with 3 operations - with a @WebService annotation that references a WSDL - operation1: input String response String - operation2: input String response String - operation3: input String response String
Service3WithCallback.java	A bi-directional service interface with callback interface Service3Callback - 1 operation "operation3", string input, string output
Service4.java	A remotable service interface containing 1 operation with a (Java) mutable parameter 1 operation "operation1" StringBuffer input, string output
Service5Intents.java	1 operation "operation1", string input, string output - entire interface is annotated with TestIntent4 - operation1 is annotated with TestIntent3
Service6PolicySets.java	Service interface - 1 operation "operation1", string input, string output - interface is annotated with policy set PolicySet1 - operation is annotated with policy set PolicySet2
Service7Callback.java	Remotable service interface used for a Callback from Service7

	- 1 operation "service7Callback1" string input, string output
Service7WithCallback.java	A bi-directional service interface with callback interface Service7Callback 1 operation "operation7", string input, string output
ServiceJAXB.java	ServiceJAXB service interface - 1 operation "operation1", JAXB_Class1 input, JAXB_Class2 output This interface has JAXB classes as parameters
ServiceOneWay.java	ServiceOneWay service interface 2 operations: - "inonly1" a OneWay method, string input - "operation1" a Request/Response method, string input, string output
ServiceOneWayMissing.java	ServiceOneWay service interface but with the inonly1 operation missing its @OneWay annotation 2 operations - "inonly1" a OneWay method missing its @OneWay annotation, string input - "operation1" a Request/Response method, string input, string output
TestInvocation.java	Basic interface to invoke testcases 1 operation - "invokeTest", string input, string output
TestIntent1.java	Intent annotation for test:testIntent1
TestIntent2.java	Intent annotation for test:testIntent2
TestIntent3.java	Intent annotation for test:testIntent3
TestIntent4.java	Intent annotation for test:testIntent4
TestIntent5.java	Intent annotation for test:testIntent5
TestIntent6.java	Intent annotation for test:testIntent6

4.3 Java Implementation Classes

Name	Description

	Services and references use interface Service1 unless otherwise described
ASM_0002_Client.java	1 service implementing TestInvocation 0 references
asyncControllerImpl.java	1 service 2 references (1..1) - reference1 with interface Service1 - dataStore with interface DataStore Acts as an invoker for an Async service and retrieves results from the DataStore
asyncControllerWithFaultsImpl.java	1 service 2 references (1..1) - reference1 with interface Service1WithFaults - dataStore with interface DataStore Acts as an invoker for an Async service which can throw Faults and retrieves results from the DataStore
BusinessFault1.java	A business fault class
BusinessFault2.java	A business fault class
compositeEagerInitImpl.java	1 service 1 reference (1..1) Implementation is COMPOSITE scope and is marked with @EagerInit - reference is invoked from method marked with @Init
dataStoreCompositeImpl.java	1 service with DataStore interface 0 references COMPOSITE scope - acts as a store for data under a key
lifecycleControllerImpl.java	1 service 2 references - reference1 (1..1) with interface Service1 - dataStore (1..1) with interface DataStore Acts as a controller component for a class that undergoes a series of lifecycle tests - the results are stored into a data store component and are retrieved at the end by this implementation

MultipleCallbacksImpl.java	3 services - Service1 with interface Service1 - Service3WithCallback with interface Service3WithCallback - Service7WithCallback with interface Service7WithCallback 0 references
multipleServiceClientImpl.java	1 service 1 reference - reference1 (1..1) with interface MultipleService Invokes all the operations of MultipleService in sequence for a single reference
multipleServiceImpl.java	1 service with interface MultipleService 0 references Implementation with many service operations that checks if a stateless instance ever gets invoked more than once
parallelCompositeClientImpl.java	1 service with interface Service1 1 reference 2 properties - "invocationCount" - number of parallel invocations - "maxWaitTime" - max time to wait for a response from the reference invocation - "reference1" (1..1) with interface ParallelService reference1 is invoked multiple times in parallel on multiple threads
parallelCompositeServiceImpl.java	1 service with interface ParallelService 0 references Implementation has COMPOSITE scope
parallelServiceClientImpl.java	1 service with interface Service1 1 reference 2 properties - "invocationCount" - number of parallel invocations - "maxWaitTime" - max time to wait for a response from the reference invocation - "reference1" (1..1) with interface ParallelService

	reference1 is invoked multiple times in parallel on multiple threads
parallelServiceImpl.java	1 service with interface ParallelService 0 references Implementation has STATELESS scope
service1AsyncRefImpl.java	1 service with interface Service1 1 reference (1..1) "reference1" with JAXWS async client form of Service1 inerface 2 properties - "invokeMethods" - array of strings indicating how to invoke methods on reference1 - "invokeParm" - array of strings which are the inpit parameters to the invocations in invokeMethods
Service1AsyncServerImpl.java	1 service with interface Service1AsyncServer 0 references Async service implementation of Service1, with response sent back on a separate thread.
Service1AsyncServerMultiFaultImpl.java	1 service with interface Service1WithFaultsAsyncServer 1 reference (1..1) with interface DataStore Async service implementation of Service1WithFaults, which calls ResponseDispatch.sendFault() method more than once
Service1AsyncServerMultiResponseImpl.java	1 service with interface Service1AsyncServer 1 reference (1..1) with interface DataStore Async service implementation of Service1, which calls ResponseDispatch.sendResponse() method more than once
service1CallbackContextImpl1.java	1 service with interface Service1 1 reference (1..1) with interface Service3WithCallback which has a Callback interface Service3Callback Reference gets called when service1 operation1 is invoked
service1CallbackContextImpl2.java	1 service with interface Service1 1 reference (1..1) with interface Service3WithCallback which has a Callback interface Service3Callback

	The callback operation's method accesses the getRequestContext() method of the ComponentContext API during the processing of a callback method and checks that returned RequestContext is not null
service1CallbackImpl.java	1 service with interface Service1, 1 reference (1..1) with interface Service3WithCallback which has a Callback interface Service3Callback which is implemented by this class Reference gets called when service1 operation1 is invoked
Service1Calls3OperationsImpl.java	1 service with interface Service1, 1 reference (1..1) with interface Service3Operations
Service1CCcastImpl.java	1 service with interface Service1 1 reference "reference1" with interface Service1 Implementation performs invocations of the cast() method of the ComponentContext API
Service1CCcreateSelfReferenceAImpl.java	1 service with interface Service1 0 references Implementation performs invocations of the createSelfReference(Class<B>) method of the ComponentContext API
Service1CCcreateSelfReferenceBImpl.java	1 service with interface Service1 0 references Implementation performs invocations of the createSelfReference(Class<B>, String) method of the ComponentContext API
Service1CCgetPropertyImpl.java	1 service with interface Service1 0 references Implementation performs invocations of the getProperty() method of the ComponentContext API
Service1CCgetServiceImpl.java	1 service with interface Service1 0 references Implementation performs invocations of the getService() method of the ComponentContext API
Service1CCgetServiceReferenceImpl.java	Java component implementation with 1 service with interface Service1

	3 references - requiredRef (1..1) with interface Service1 - singledRef (0..1) with interface Service1 (intended to be unwired) - multiRef (0..n) with interface Service1 Implementation performs invocations of the getServiceReference() method of the ComponentContext API
Service1CCgetServiceReferencesImpl.java	Java component implementation with 1 service with interface Service1 4 references - requiredRef (1..1) with interface Service1 - singledRef (0..1) with interface Service1 (intended to be unwired) - multiRef (0..n) with interface Service1 - multiRef2 (0..n) with interface Service1 (intended to be unwired) Implementation performs invocations of the getServiceReferences() method of the ComponentContext API
Service1CCgetServicesImpl.java	Java component implementation with 1 service with interface Service1 4 references - requiredRef (1..1) with interface Service1 - singledRef (0..1) with interface Service1 (intended to be unwired) - multiRef (0..n) with interface Service1 - multiRef2 (0..n) with interface Service1 (intended to be unwired) Implementation performs invocations of the getServices() method of the ComponentContext API
Service1CCgetURIImpl.java	1 service with interface Service1 0 references Implementation performs invocations of the getURI() method of the ComponentContext API
service1CompositeImpl.java	1 service with interface Service1 0 references

	Implementation is COMPOSITE scope
service1ConstrImpl.java	1 service with interface Service1 0 references Implementation has constructor with parameter annotated with @Property
service1ContextImpl1.java	1 service with interface Service1 1 reference (1..n) with interface Service1 all configured wires get called when service1 operation1 is invoked Iimplementation invokes the ComponentContext getService() API
service1CoordinatorImpl.java	1 service with interface Service1 1 reference (1..n) with interface Service1 all configured wires get called when service1 operation1 is invoked Implementation acts as a coordinator between each of the invocations - it sends different data as input to each reference invocation - it records all the output data from all the invocations
service1GoodIntent.java	1 service with interface Service1 3 references Implementation has all references annotated with TestIntent2
service1Impl.java	1 service with interface Service1 0 references
service1Impl2.java	1 service with interface Service1, 1 reference (1..1) with interface Service1 reference gets called when service1 operation1 is invoked
service1Impl4.java	1 service with interface Service1 1 reference (0..n) with interface Service1 all configured wires get called when service1 operation1 is invoked - reports an empty reference array if it is unwired
service1Impl6.java	1 service with interface Service1 1 reference (0..1) with interface Service1

	configured wire gets called when service1 operation1 is invoked, if present otherwise, this implementation reports its absence
service1Impl7.java	1 service with interface Service1, 1 reference (1..1) with interface Service4 - reference gets called when service1 operation1 is invoked - reference is marked with @AllowsPassByReference
service1Impl7b.java	1 service with interface Service1, 1 reference (1..1) with interface Service4 - reference gets called when service1 operation1 is invoked - reference is not marked @AllowsPassByReference and the implementation checks for alteration of input parameters
service1ImplWS.java	Java implementation - NOT used as an SCA component implementation, but instead used in the process of generating a WSDL interface through the standard JDK wsgen tool, which follows JAXWS 2.1 rules 1 service with interface Service1WS 0 references
service1InitCheckerImpl.java	1 service with interface Service1 1 reference (1..1) with interface Service1 2 properties - invocationCount - expectedResponse The implementation invokes the reference invocationCount times and checks for the response expectedResponse
service1InitImpl.java	1 service with interface Service1 0 references Implementation has a method marked with @Init - the response from the operation1 method of Service1 differs depending on whether the @Init method has been called before the operation1 method is called
service1Intent.java	1 service with interface Service1 0 references Implementation class is annotated with TestIntent2

service1IntentQualifier.java	1 service with interface Service1 0 references Implementation class is annotated with TestIntent6.qual1
service1JAXBCallerImpl.java	1 service with interface Service1, 1 reference (1..1) with interface ServiceJAXB reference gets called when service1 operation1 is invoked
service1LifecycleExceptionsImpl.java	1 service with interface Service1 1 reference (1..1) with interface Service1 Java STATELESS component implementation which tests the overall Lifecycle sequence of an SCA Java implementation class
service1MultiPropImpl.java	1 service with interface Service1 0 references 1 property which is multi-valued
service1OneWayCallerImpl.java	1 service with interface Service1, 1 reference (1..1) with interface ServiceOneWay - reference OneWay operation gets called first when service1 operation1 is invoked - then reference request/response operation gets called
Service1OneWayMissingCallerImpl.java	1 service with interface Service1, 1 reference (1..1) with interface ServiceOneWayMissing - reference OneWay operation gets called first when service1 operation1 is invoked - then reference request/response operation gets called
service1OptMultiRef.java	1 service with interface Service1 1 reference (0..n) with interface Service1
Service1RCgetServiceNameImpl.java	1 service with interface Service1 2 references - singleRef (1..1) with interface Service1 - callback3Ref (1..1) with interface Service3WithCallback Implementation performs invocations of the RequestContext getServiceName() method of the

	API
service1RequestContextImpl1.java	1 service with interface Service1 0 references Implementation invokes the ComponentContext getRequestContext() API and checks that the returned value is a valid RequestContext
service1RequestContextImpl2.java	1 service with interface Service1 0 references This implementation invokes the getServiceReference() method of the RequestContext API and checks that the returned ServiceReference represents the Service offered by this component
Service1SCAClientImpl.java	1 service with interface Service1 1 reference "reference1" with interface Service1 Implementation performs invocations of the SCAClient API
service1StatelessLifecycleImpl.java	1 service with interface Service1 1 reference (1..1) with interface Service1 Java STATELESS component implementation which tests the overall Lifecycle sequence of an SCA Java implementation class
service1SupersetImpl.java	1 service with interface Service1Superset 0 references
service1UninitImpl.java	1 service with interface Service1 0 references Implementation refuses to initialize - it throws an exception from its @Init method
service1WSImpl.java	1 service with interface Service1WS 0 references
service2Impl.java	1 service with interface Service2 0 references
service3Impl1.java	1 service with interface Service3WithCallback 1 callback using interface Service3Callback 0 references
Service3OperationsWebServiceImpl.java	1 service with interface Service3Operations, but declared through an @WebService annotation

	0 references
Service3OperationsWSDLImpl.java	1 service with interface Service3OperationsWSDL 0 references
service4Impl1.java	1 service with interface Service4, where the implementation method modifies the input parameter 0 references - service is marked with @AllowsPassByReference
service4Impl.java	1 service with interface Service4, where the implementation method modifies the input parameter 0 references
service5Impl2.java	1 service with interface Service1, 1 reference (1..1) with interface Service5Intents reference gets called when service1 operation1 is invoked
service5Impl.java	1 service with interface Service5Intents 0 references
service6Impl2.java	1 service with interface Service1, 1 reference (1..1) with interface Service6PolicySets reference gets called when service1 operation1 is invoked
service6Impl.java	1 service with interface Service6PolicySets 0 references
serviceJAXBImpl.java	1 service with interface ServiceJAXB 0 references invocation of operation1 causes the implementation to read data from the supplied JAXB_Class1 parameter and return data in all the fields of the returned JAXB_Class2 parameter
serviceOneWayImpl.java	1 service with interface ServiceOneWay 0 references The implementation is COMPOSITE scope and is stateful - it remembers whether its OneWay method inonly1 was called - it reports this when its request/Response method is called

TestException.java	Exception thrown by SCA Test services
JAXB.JAXB_Class1.java	JAXB annotated complex class that is used in a service interface
JAXB.JAXB_Class2.java	JAXB annotated complex class that is used in a service interface

4.4 WSDL Interface Files

Name	Description
Service1ImplWSService.wsdl	WSDL mapped from service1ImplWS class
Service3OperationsWSDL.wsdl	WSDL version of the Service3OperationsWSDL.java interface, but containing only operations "operation1" and "operation2" - "operation3" intentionally missing
TestClient.wsdl	WSDL version of the TestInvocation.java interface
TestInvocation.wsdl	WSDL version of the client TestInvocation.java interface

5 Conformance

There are no conformance statements relating to the TestCases.

Appendix A. Acknowledgments

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

Participants:

Participant Name	Affiliation
Bryan Aupperle	IBM
Vladislav Bezrukov	SAP AG*
David Booz	IBM
Martin Chapman	Oracle Corporation
Vamsavardhana Reddy Chillakuru	IBM
Mark Combellack	Avaya, Inc.
Mike Edwards	IBM
Anish Karmarkar	Oracle Corporation
Ashok Malhotra	Oracle Corporation
Plamen Pavlov	SAP AG*
Eric Wells	Hitachi, Ltd.

Appendix B. Revision History

Revision	Date	Editor	Changes Made
1	25/09/09	Mike Edwards	Initial version
4	02/10/09	Dave Booz	Section 3 testcases
5	06/10/09	Dave Booz	Section 8, 8001-8005
7	07/10/09	Dave Booz	Complete Section 8
11	09/10/09	Mike Edwards	All sections complete
12	21/06/10	Mike Edwards	Testcases added for CD04 of Java CAA specification: JCA_4008 JCA_7001 - JCA_7003 JCA_9006 - JCA_9016
13	28/06/10	Mike Edwards	Further Testcases added for CD04: JCA_3013 - JCA_3014 JCA_7004 - JCA_7006 JCA11009 - JCA_11017
cd01	19/07/10	Mike Edwards	All changes accepted Frontmatter adjusted to OASIS requirements