TestCases for the SCA_J Common Annotations and APIs Version 1.1 Specification

Working Draft 10

09 October 2009

Specification URIs:

This Version:

http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-testcases-wd10.html

http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-testcases-wd10.odt

http://docs.oasis-open.org/opencsa/sca-j/sca-j-caa-1.1-testcases-wd10.pdf(Authoritative)

Previous Version:

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

Chair(s):

David BoozMartin Chapman, IBMOracle
Mark CombellackMike Edwards, AvayaIBM, 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

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", [insert specific trademarked names, abbreviations, etc. here] 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] SCA_J Common Annotations and APIs Test Assertions

http://docs.oasis-open.org/sca-j/sca-j-caa-1.1-test-assertions_WD07.pdf

1.5 Non-normative References

[TBD] [TBD]

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 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”

JCA_3004_TestCase

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”

JCA_3005_TestCase

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"

207

208 JCA_3011_TestCase

209

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"

210

211 JCA_3012_TestCase

212

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

215 2.3 Section 4

216 JCA_4001_TestCase

217

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"

218

219 JCA_4002_TestCase

220

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"

221

222 JCA_4003_TestCase

223

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"

JCA_4004_TestCase

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"

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"

230

231 ## JCA_4006_TestCase

232

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"

233

JCA_4007_TestCase

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"

2.4 Section 5

JCA_7001_TestCase

Testcase ID	JCA_7001_TestCase
Test Assertion	JCA-TA-7001
Description	Tests that
Artifacts	JCA_7001_TestCase.java Test_JCA_7001.composite TestInvocation.wsdl TestClient_0002.composite

	Test_JCA_7001.constraint TestComposite1.composite Service1.wsdl
Expected output	

241

242

243 2.5 Section 8

244 JCA_8001_TestCase

245

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"

246

247 JCA_8002_TestCase

248

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"

249

250 JCA_8003_TestCase

251

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"

252

253 JCA_8004_TestCase

254

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"

255

256 JCA_8005_TestCase

257

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"

258

259 JCA_8006_TestCase

260

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"

261

262 JCA_8007_TestCase

263

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"

264

265 JCA_8008_TestCase

266

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"

267

268 JCA_8009_TestCase

269

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"

270

271

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"

272

273 JCA_8011_TestCase

274

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"

275

276

277 2.6 Section 9

278 JCA_9001_TestCase

279

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"

280

281 JCA_9002_TestCase

282

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"

283

284 JCA_9003_TestCase

285

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"

286

287 JCA_9004_TestCase

288

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"

289

290 ## JCA_9005_TestCase

291

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"

292

293

294 2.7 Section 10

295 JCA_10001_TestCase

296

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	

297

298 JCA_10002_TestCase

299

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	

JCA_10003_TestCase

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	

JCA_10004_TestCase

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	

306

307 JCA_10005_TestCase

308

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:

309

310 JCA_10006_TestCase

311

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	

312

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	

JCA_10009_TestCase

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	

321

322 JCA_10010_TestCase

323

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	

324

325 JCA_10011_TestCase

326

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	

JCA_10012_TestCase

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:

JCA_10013_TestCase

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:

JCA_10014_TestCase

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:

JCA_10015_TestCase

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:

JCA_10016_TestCase

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:

JCA_10044_TestCase

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"

JCA_10045_TestCase

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"

JCA_10046_TestCase

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"

JCA_10047_TestCase

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"

355

356 JCA_10048_TestCase

357

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"

358

359 JCA_10049_TestCase

360

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"

361

362 ### JCA_10050_TestCase

363

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"

364

365 ## JCA_10051_TestCase

366

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"

367

368 JCA_10052_TestCase

369

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"

370

371

372 2.8 Section 11

373 JCA_11001_TestCase

374

Testcase ID	JCA_11001_TestCase
Test Assertion	JCA-TA-11001
Description	Tests that
Artifacts	JCA_11001_TestCase.java Test_JCA_11001.composite TestInvocation.wsdl TestClient_0002.composite TestComposite1.composite

	Service1.wsdl
Expected output	

375

376

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

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

381

382

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

383

384

Test Assertion	Test Cases
JCA-TA-9001	JCA_9001_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

385

386

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

387

388

Test Assertion	Test Cases
JCA-TA-11001	JCA_11001_TestCase

389

390

391

392

393

394

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*
TestComposite10.composite	**Y**	1 service 1 reference with binding and target

398

399 4.2 Java Interfaces

400

Name	Description
test.Service1Overload.java (JCA_2001 contribution)	2 operations: - overloaded, both named "operation1" #1 - 1 string input, 1 string output parameter #2 - 1 int input, 1 string output
test.Service2.java	Remotable service interface that is not compatible with Service1 interface

4.3 Java Implementation Classes

Name	Description
	Services and references use interface Service1 unless otherwise described
test.service1OverloadImpl.java (JCA_2001 contribution)	1 service implementing Service1Overload 0 references
test.service1Impl2.java	1 service 1 reference (1..1)

403 4.4 WSDL Interface Files

404

Name	Description

405

406 # 5 Conformance

407 There are no conformance statements relating to the TestCases.

408

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. Non-Normative Text

Appendix C. Revision History

Revision	Date	Editor	Changes Made
1	09/25/09	Mike Edwards	Initial version
4	10/02/09	Dave Booz	Section 3 testcases
5	10/06/09	Dave Booz	Section 8, 8001-8005
7	10/07/09	Dave Booz	Complete Section 8