Test Suite Adaptation for SCA Assembly Model Version 1.1 Specification

Committee Draft 01 / Public Review 01

20 July 2010

Specification URIs:

This Version:

http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-testsuite-adaptation-cd01.html

http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-testsuite-adaptation-cd01.odt

http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-testsuite-adaptation-cd01.pdf (Authoritative)

Previous Version:

N/A

Latest Version:

http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-testsuite-adaptation.html

http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-testsuite-adaptation.odt

http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-testsuite-adaptation.pdf (Authoritative)

Technical Committee:

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

Chair(s):

Martin Chapman, Oracle
Mike Edwards, IBM

Editor(s):

Bryan Aupperle
Dave Booz
Mike Edwards
Jeff Estefan

Related Work:

This document is related to:

- Service Component Architecture Assembly Specification Version 1.1
  http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-cd05.pdf

Declared XML Namespace(s):

none

Abstract:

This document defines the requirements for adaptation of the SCA Assembly Test Suite to use a new SCA implementation type that is provided by a conforming SCA Runtime. The SCA Runtime needs to pass the SCA Assembly Test Suite without failures. Where the SCA Runtime supports an implementation type that is not currently supported by the SCA Assembly Test Suite, it is necessary for the test suite to be adapted to use that implementation type, before the test suite can be run successfully against that SCA Runtime.

Status:

This document was last revised or approved by the OASIS Service Component Architecture / Assembly (SCA-Assembly) 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-assembly/.

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-assembly/ipr.php.

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

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

Notices

Table of Contents

1 Introduction

[All text is normative unless otherwise indicated.]

This document defines the requirements for adaptation of the SCA Assembly Test Suite to use a new SCA implementation type that is provided by a conforming SCA Runtime. The SCA Runtime needs to pass the SCA Assembly Test Suite without failures. Where the SCA Runtime supports an implementation type that is not currently supported by the SCA Assembly Test Suite, it is necessary for the test suite to be adapted to use that implementation type, before the test suite can be run successfully against that SCA Runtime.

The SCA Assembly Test Suite is designed for adaptation to new implementation types. The test suite is divided into two groups of artifacts, handled by means of separate SCA contributions:

- Implementation type-independent artifacts, largely consisting of SCA composites, with associated supporting artifacts such as interfaces provided as WSDL declarations.
- Implementation type-dependent artifacts, which consist of implementations in the relevant implementation type, plus SCA composites which directly wrap those implementations, and other associated artifacts which can include interfaces provided in a language which is natural for the implementation type (eg. Java implementation classes have their interfaces declared as Java interfaces)

The SCA Assembly Test Suite is described through two documents:

- The TestCases document [SCA-TestCases], which describes the detail of the testcases themselves and their related artifacts.
- The Test Assertions document [SCA-Assertions], which describes a set of assertions which must be validated by the test suite. The test assertions are derived directly from the normative statements of the SCA Assembly specification. The test assertions effectively render the requirements of the specification as a series of assertions that can be cast into the form of one or more testcases which then validate conformance with the specification.

This document largely concerns itself with the TestCases document and the associated sets of artifacts that make up the test suite itself. The Test Assertions document can be used as a guide to the intention of each testcase and is the way in which each testcase can be linked back to appropriate normative statements in the SCA Assembly specification [SCA-Assembly].

1.1 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].

1.2 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.
[SCA-Assembly]	OASIS Committee Draft 05, Service Component Architecture Assembly Model Specification Version 1.1, January 2010. http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-spec-cd05.pdf
[SCA-TestCases]	OASIS Committee Draft 01, TestCases for the SCA Assembly Model V1.1 Specification, June 2009.

http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-testcases-cd01.pdf

[SCA-Assertions] OASIS Committee Draft 02, Test Assertions for the SCA Assembly Model Version 1.1 Specification, June 2010. http://docs.oasis-open.org/opencsa/sca-assembly/sca-assembly-1.1-test-assertions-cd02.pdf

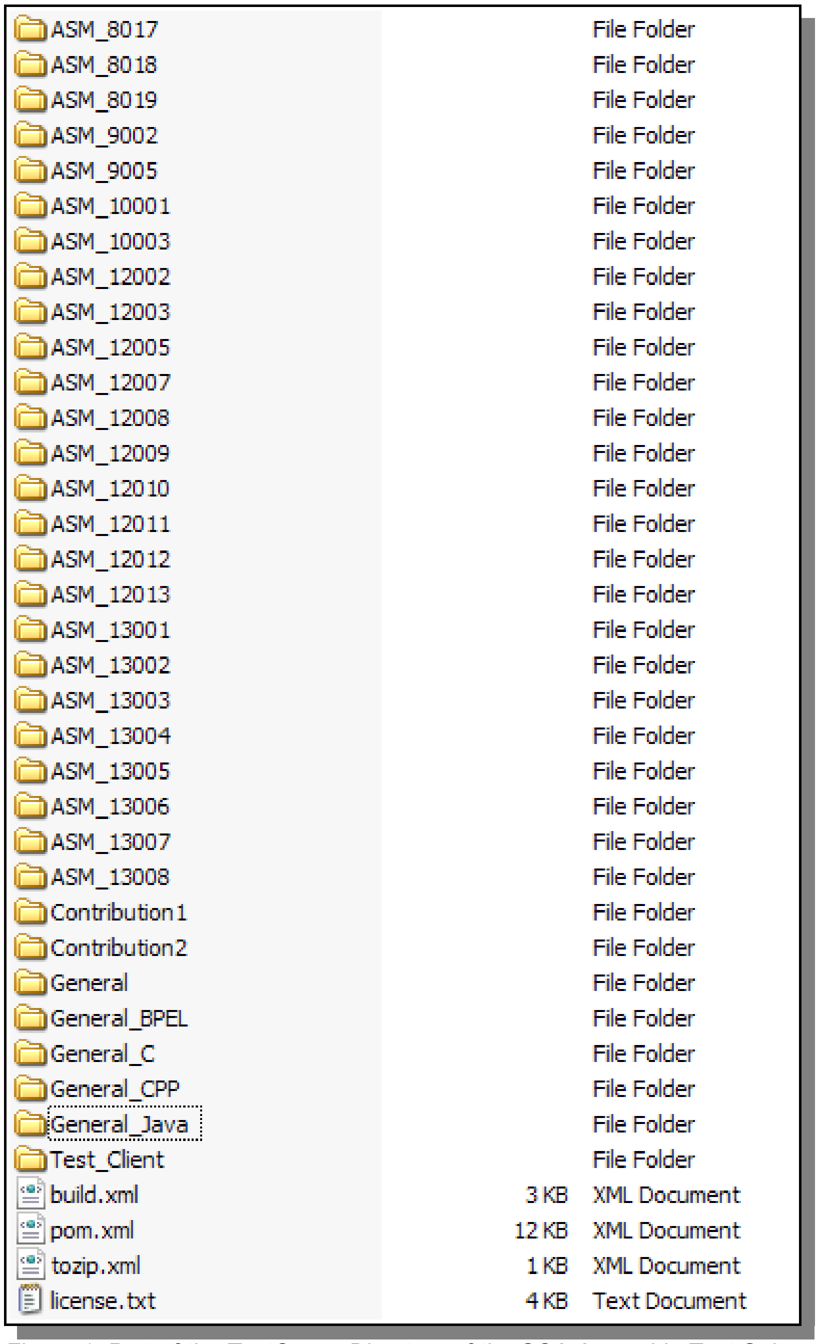
1.3 Non-normative References

None

2 SCA Assembly Test Suite Organization

The SCA Assembly Test Suite is fully described in the TestCases for the SCA Assembly Model specification [SCA-TestCases]. This section describes the aspects of the test suite organization which are relevant when producing a new version of the test suite for a new implementation type.

Figure 1 displays part of the main TestCases directory of the SCA Assembly Test Suite.



ASM_8017		File Folder
ASM_8018		File Folder
ASM_8019		File Folder
ASM_9002		File Folder
ASM_9005		File Folder
ASM_10001		File Folder
ASM_10003		File Folder
ASM_12002		File Folder
ASM_12003		File Folder
ASM_12005		File Folder
ASM_12007		File Folder
ASM_12008		File Folder
ASM_12009		File Folder
ASM_12010		File Folder
ASM_12011		File Folder
ASM_12012		File Folder
ASM_12013		File Folder
ASM_13001		File Folder
ASM_13002		File Folder
ASM_13003		File Folder
ASM_13004		File Folder
ASM_13005		File Folder
ASM_13006		File Folder
ASM_13007		File Folder
ASM_13008		File Folder
Contribution1		File Folder
Contribution2		File Folder
General		File Folder
General_BPEL		File Folder
General_C		File Folder
General_CPP		File Folder
General_Java		File Folder
Test_Client		File Folder
build.xml	3 KB	XML Document
pom.xml	12 KB	XML Document
tozip.xml	1 KB	XML Document
license.txt	4 KB	Text Document

Figure 1: Part of the TestCases Directory of the SCA Assembly Test Suite

The contents of the directory are mainly a set of SCA contributions contained in subdirectories such as ASM_8017, ASM_13008 and General. The test client runner application is contained in the Test_Client subdirectory.

The General contribution contains a series of artifacts such as composites and WSDL files which are used by a range of testcases.

Directories which have names like ASM_8017 contain an SCA contribution with artifacts which are specific to the testcase with the same name as the directory. Typically, testcase-specific contributions are provided for testcases that have artifacts that contain static errors - such contributions are not necessary for testcases which use artifacts that contain no errors.

The directories with names like General_BPEL and General_Java are the ones of interest when adapting the test suite to a new implementation type. These directories represent contributions which contain implementation-type dependent artifacts. Their names are stylized so that they have a common stem ("General") followed by an underscore ("_") and then a name which characterizes the implementation type. So "BPEL" is used for the WS-BPEL implementation type, "Java" is used for the Java POJO implementation type and so on.

In general, when adapting the test suite to a new implementation type, it is necessary to produce a new contribution similar to General_BPEL and General_Java, containing all the appropriate artifacts for the new implementation type. The contribution needs to have a name which reflects the new implementation type, such as "General_Foo" - the exact name is not important, but it should be unique. The name suffix (e.g. “Foo”) is used as part of the configuration of the Test Client.

The adapted test suite MUST contain a directory which contains an SCA contribution, termed the "implementation type contribution", holding the implementation-type dependent artifacts, [TST10001]

The implementation type contribution must have a name that starts with the stem "General_" followed by a suffix which is unique to the implementation type. [TST10002]

2.1 Implementation Type Dependent Artifacts

To produce the contributions containing implementation type dependent artifacts, it is necessary to produce:

- A specific set of composite files which contain components that use the implementation type
- An associated set of artifacts required to satisfy the set of composite files. This set of artifacts will always contain a set of implementation artifacts in the form used by the implementation type, each supplying a specific componentType. The set of artifacts can also contain other artifacts if required by the implementation type, for example interface definition files.
- An sca-contribution.xml file which is used to declare SCA exports from this contribution, which are then available for imports to satisfy references to the artifacts made from other contributions

It is useful to use the General_Java contribution as the canonical example of what is required for any new contribution that supports a new implementation type.

First, the set of implementation-type dependent SCA composite files, which are held in the \src\main\resources directory are shown in Listing 1:

```
TestClient_0002.composite
TestClient_0003.composite
TestClient_0004.composite
TestComposite1.composite
TestComposite4.composite
TestComposite5.composite
```

```
TestComposite6.composite
TestComposite7.composite
TestComposite8.composite
TestComposite9.composite
TestComposite52.composite
TestComposite53.composite
TestComposite54.composite
TestComposite55.composite
TestComposite60.composite
TestComposite61.composite
TestComposite65.composite
TestComposite66.composite
TestComposite70.composite
TestComposite71.composite
TestComposite73.composite
```

Listing 1: List of Implementation-Type Dependent Composite files

For Java POJOs, the implementation artifacts consist of:

- Implementations, which are simple Java classes
- Java interfaces, which are Java versions of the set of service interfaces used by the SCA Assembly Test Suite
- Exceptions, which are present as a set of Java classes

The Java POJO implementations are shown in Listing 2:

```
ASM_0002_Client.java
ASM_0003_Client.java
Service1Callback5Impl.java
Service1Callback9Impl.java
Service1Impl.java
Service1Impl2.java
Service1Impl3.java
Service1Impl4.java
Service1Impl5.java
Service1Impl6.java
Service1Impl7.java
Service1Impl8.java
Service1Impl9.java
Service1SupersetImpl.java
Service2Impl.java
Service4Impl.java
Service5Impl.java
Service5Impl2.java
Service9Impl.java
```

Listing 2: List of Implementations

The Java interfaces used in conjunction with the Java POJO implementations are shown in Listing 3:

```
Service1.java
Service1_Intent.java
Service1Superset.java
Service2.java
```

```
Service4.java
Service5.java
Service5Callback.java
Service8Callback.java
Service9.java
Service9Callback.java
TestInvocation.java
```

Listing 3: List of Implementation-Type Dependent Interface files

The Java exceptions used in conjunction with the Java POJO implementations are shown in Listing 4:

```
TestException.java
InvalidSCAConfigurationException.java
```

Listing 4: List of Java Exception Classes

2.1.1 Implementation-Dependent Composite Files

It is necessary to produce a version of each of the implementation-type dependent composite files shown in Listing 1. Most of these composite files are basically wrappers for a single implementation artifact, and their role is to provide an implementation with a consistent componentType that can be used within higher level composites that are implementation type independent. Typically, the implementation used within each of these composites has the same componentType as the composite itself.

As an example of what needs to be done, examine the Java POJO version of TestComposite4, shown in Listing 5:

```xml
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           targetNamespace=
              "http://docs.oasis-open.org/ns/opencsa/scatests/200903"
           name="TestComposite4">

   <service name="Service1" promote="Composite4Component1/Service1">
      <interface.java interface="org.oasisopen.sca.test.Service1"/>
   </service>

    <property name="serviceName" type="string"/>

    <component name="Composite4Component1">
       <implementation.java class="org.oasisopen.sca.test.Service1Impl2"/>
         <service name="Service1">
           <interface.java interface="org.oasisopen.sca.test.Service1"/>
         </service>
         <property name="serviceName" source="$serviceName"/>
         <reference name="reference1"/>
    </component>

    <reference name="Reference1" promote="Composite4Component1/reference1"
       multiplicity="1..1">
       <interface.java interface="org.oasisopen.sca.test.Service1"/>
    </reference>

</composite>
```

Listing 5: TestComposite4.composite from General_Java

This corresponds to a componentType with:

- 1 service with the name "Service1" with the interface Service1

- 1 reference with the name "Reference1" with the interface "Service1" and multiplicity "1..1"
- 1 property with the name "serviceName" with the type "xsd:string"

A rendering of the componentType of TestComposite4 is shown in Listing 6:

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
              targetNamespace=
                 "http://docs.oasis-open.org/ns/opencsa/scatests/200903"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              name="Service1Impl2">
   <service name="Service1">
      <interface.java interface="org.oasisopen.sca.test.Service1"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

    <reference name="reference1">
       <interface.java interface="org.oasisopen.sca.test.Service1"/>
    </reference>

</componentType>
```

Listing 6: ComponentType of TestComposite4.composite

Note that here, the interface type used is interface.java - this can be mapped to interface.wsdl, where the equivalent interface declaration is shown in Listing 7:

```
<interface.wsdl
   interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
```

Listing 7: Service1 Interface declaration in terms of interface.wsdl

A version of TestComposite4 for the C++ implementation type is shown in Listing 8 with the corresponding componentType in Listing 9.

```
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           targetNamespace=
              "http://docs.oasis-open.org/ns/opencsa/scatests/200903"
           name="TestComposite4">

    <service name="Service1" promote="Composite4Component1/Service1">
       <interface.cpp header="test\Service1.h" remotable="true"/>
    </service>

    <property name="serviceName" type="string"/>

    <component name="Composite4Component1">
       <implementation.cpp library="general" path="bin\"
                          class="service1Impl2"/>
         <service name="Service1">
           <interface.cpp header="test\Service1.h" remotable="true"/>
         </service>
         <property name="serviceName" source="$serviceName"/>
         <reference name="reference1"/>
    </component>

    <reference name="Reference1" promote="Composite4Component1/reference1"
               multiplicity="1..1">
       <interface.cpp header="test\Service1.h" remotable="true"/>
    </reference>

</composite>
```

Listing 8: TestComposite4.composite from General_CPP

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
              targetNamespace=
                 "http://docs.oasis-open.org/ns/opencsa/scatests/200903"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              name="Service1Impl2">
   <service name="Service1">
      <interface.cpp header="test\Service1.h" remotable="true" />
   </service>

   <property name="serviceName" type="xsd:string"/>

    <reference name="reference1">
       <interface.cpp header="test\Service1.h" remotable="true" />
    </reference>

</componentType>
```

Listing 9: Service1Impl2.componentType from General_CPP

The C++ version in Listing 8 illustrates what is necessary to produce a version of TestComposite4 for a new implementation type. Listing 8 has a <implementation.cpp/> element instead of the <implementaiton.java/> element in Listing 5. To create the corresponding TestComposite4 for "implementation.foo", the <implementation.java/> element shown in Listing 5 has to be replaced by an appropriate <implementation.foo/> element, which needs to reference an implementation artifact that provides a componentType as shown in Listing 6, either implicitly or explicitly as appropriate for the implementation type.

If the new implementation type requires the use of an interface type other than WSDL, e.g. "interface.foo", then replace the interface elements in the implementation type-dependent composite files with interface.foo. For example, notice how the <interface.java/> elements in Listing 5 and Listing 6 are replaced with <interface.cpp/> elements in Listing 8 and 9. Implementation type-dependent composites can also use <interface.wsdl/> elements to declare the interfaces of services and references. Note that all the WSDL files declaring the various interfaces can be found in the "General" contribution.

This procedure would create a new version of the TestComposite4.composite file as shown in Listing 10:

```
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           targetNamespace=
              "http://docs.oasis-open.org/ns/opencsa/scatests/200903"
           name="TestComposite4">

   <service name="Service1" promote="Composite4Component1/Service1">
      <interface.foo.../>
   </service>

    <property name="serviceName" type="string"/>

    <component name="Composite4Component1">
       <implementation.foo.../>
         <service name="Service1">
           <interface.foo.../>
         </service>
         <property name="serviceName" source="$serviceName"/>
         <reference name="reference1"/>
    </component>

    <reference name="Reference1" promote="Composite4Component1/reference1"
       multiplicity="1..1">
       <interface.foo.../>
    </reference>

</composite>
```

Listing 10: TestComposite4.composite for implementation type "foo"

The "implementation type contribution" MUST contain a set of composite files as defined in section "Composite Files in the Implementation Type Contribution", each file having the composite name and having the componentType defined there, and where the components in those composite files use implementations of the type supported by the adapted test suite. [TST10003]

The composites contained in the "implementation type contribution" MAY use an implementation-type specific form of interface declaration rather than using the interface.wsdl form of interface declaration. [TST10004]

2.1.2 Implementations

It is necessary to produce an implementation type specific version of each of the implementation artifacts contained in Listing 2. How this is done depends on the implementation type itself. These implementation artifacts are referenced by the implementation-type dependent composites discussed in the section "Implementation-Dependent Composite Files".

Each implementation artifact needs to have a componentType as required by the composite(s) that use it.

2.1.3 Interfaces

If an implementation type requires the use of a new interface definition type, the adapted test suite MUST contain an interface definition type version of each of the interface artifacts contained in Listing 3. [TST10005] Each interface artifact in the adapted test suite MUST map to equivalent WSDL file of the same name contained in the General contribution. [TST10006]

2.1.4 sca-contribution.xml file

The General contribution for a new implementation type needs to have an sca-contribution file containing appropriate imports and exports for resolving implementation type specific artifacts.

The sca-contribution.xml file from the General_Java contribution is shown in Listing 11:

```
<!-- sca-contribution for General_Java contribution -->
<contribution xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912">

   <export namespace="http://docs.oasis-open.org/ns/opencsa/scatests/200903"/>
    <!-- Composites namespace 1 -->
   <export namespace=
      "http://docs.oasis-open.org/ns/opencsa/scatests/2009032"/>
    <!-- Composites namespace 2 -->
   <export.java package="org.oasisopen.sca.test"/>
    <!-- Java package -->

</contribution>
```

Listing 11: sca-contribution.xml file from General_Java contribution

It is necessary to export the two composites namespaces shown in Listing 11 – elements defined in these namespaces are used by the composites in the contribution and are therefore necessary if composite references used by the testcases are to resolve correctly to the composites in the contribution.

The final export shown in Listing 11 is specific to the Java POJO implementation type - it enables the re-use of Java implementations and Java interface artifacts by other contributions in the test suite. Whether this is relevant to a new implementation type depends first on whether that implementation type has an implementation type specific artifact sharing mechanism. It also depends on whether it makes sense to share artifacts in the implementation type General contribution with other language-specific contributions.

2.2 The Test Client

The SCA Assembly Test Suite contains a Test Client that is used to start and run the testcases. The test client is responsible for:

- Starting the SCA Runtime that is under test
- Loading the set of contributions relevant to the specific testcase into the SCA Runtime
- Invoking the test
- Stopping/unloading the contributions related to the testcase

The SCA Assembly Test Suite supplies a Test Client that is written using Java - it depends only on the capabilities of the publicly available JavaSE runtime. Facilities are provided in the test client for a piece of SCA Runtime-dependent code to be provided by the SCA Runtime provider, which has the task of starting the SCA Runtime and of passing to it the contributions and artifacts used for testcases. This permits the test client to be adapted to any particular SCA Runtime.

The test client invokes the testcase using Web services. The testcases are all built to expose a Web service endpoint that can be used to invoke the test and which feeds back the results of executing the test. This separates the test client from the SCA Runtime and it means that the SCA Runtime can in principle be implemented using any technology, independent of the test client. Thus it is possible to use the test client supplied with the SCA Assembly Test Suite for a wide variety of SCA runtimes.

The test client contains two sets of configuration information:

- General configuration of the test client itself that applies to all testcases
- Testcase-specific configuration that describes the important aspects of each testcase

2.2.1 Configuring the Test Client

The Java test clent has some general configuration that can be used to influence the way that the test client operates. This general configuration is contained in the properties file "oasis-sca-tests.properties". An example of this properties file is shown in Listing 12:

```
#
# Copyright(C) OASIS(R) 2005,2009. All Rights Reserved.
# OASIS trademark, IPR and other policies apply.
#

# OASIS SCA Assembly test properties
#
# 1) The implementation type to use for Assembly test suite
# Examples: "Java" "BPEL" "CPP" "C"
org.oasis.sca.tests.assembly.lang=Java

# 2) The class to use as the Runtime Bridge for the SCA runtime under test
org.oasis.sca.tests.assembly.runtime_bridge=client.TuscanyRuntimeBridge

# 3) The location of the contributions for the test suite
# %1 represents the placement of the name of each contribution into the
# location URI
# Following uses ZIP files for the contributions
org.oasis.sca.tests.assembly.contribution.location=file:/C:/OASIS_TESTS/SCA-
Assembly/TestCases/%1/target/%1.zip
```

Listing 12: Example of the oassi-sca-tests.properties file

The oasis-sca-tests.properties file contains 3 pieces of configuration:

1. The name of the implementation type to use (e.g. "Java")
2. The name of the runtime bridge to use (i.e. configures the client to use a particular SCA Runtime)
3. The location to use for the contributions at runtime, supplied as a URL with a substitution string for the short name of the contribution:
   file:/C:/OASIS_TESTS/SCA-Assembly/TestCases/%1/target/%1.zip
   where %1 is substituted at runtime with the name of the contribution
   This provides flexibility in the location of the contributions.

The runtime bridge is a piece of code that connects the test client to the SCA Runtime that is used to run the testcases. It is described in detail in the TestCases documentation for SCA Assembly [SCA-TestCases]. The configuration needs to contain the fully qualified name of the class for the runtime bridge. So, for a new implementation type, it is necessary to have a runtime bridge for the SCA Runtime that provides that implementation type and for the name of the runtime bridge to be entered into the properties file.

To use a different runtime bridge, the Java classes for the bridge are added to a (sub)directory of the test client code and the name of the main class for the bridge is put into the properties file.

The name of the implementation type, such as "Java" needs to match the name used for the suffix of the implementation-type dependent contributions described in the section "Implementation Type Dependent Artifacts". This name is used to influence the detailed configuration for each testcase, described in the next section.

2.2.2 Configuration of Each TestCase

For the supplied test client, the configuration of each testcase is contained in the Test_Client subdirectory of the test suite, specifically in the ASM_nnnn_TestCase.java files contained in the src/main/javaclient subdirectory. Listing 10 shown the contents of a typical client file, ASM_5004_TestCase.java:

```
    protected TestConfiguration getTestConfiguration() {
       TestConfiguration config = new TestConfiguration();
       config.testName     = this.getClass().getSimpleName().substring(0, 8);
       config.input        = "request";
       config.output[0]        = "exception";
       config.composite        = "Test_" + config.testName + ".composite";
       config.testServiceName  = "TestClient";

       config.contributionNames   =
            new String[] { "ASM_5004", "General", "General" + _Lang };
       config.serviceInterface = TestInvocation.class;
       return config;
    }
```

Listing 10: Contents of Test Client configuration for ASM_5004_TestCase

The significant aspects of the configuration are:

1. The test name = "ASM_5004"
2. Input string = "request"
3. Expected output string = "exception"
   Note - the string can be some string like "ASM_5002 request service1 operation1 invoked" if the testcase is expected to succeed (i.e., it is a positive test), but the string "exception" is used where it is expected that the test will fail (i.e., it is a negative test) and the SCA Runtime is expected to raise an exception.

4. Composite to run = "Test_ASM_5004.composite"
   Every testcase involves running one specific composite, which is present in one of the contributions - this name is supplied in the configuration.

5. ContributionNames = "ASM_5004", "General", "General" + _Lang
   This is a list or array of the names of the Contributions to use for this testcase. The test client MUST load all the contributions declared in the testcase configuration into the SCA Runtime. [TS T10007] Note that the final one in this list is a contriibution which depends on the name of the implementation type that is under test - "Lang" gets replaced with the name of the implementation type, to result in a contribution name like "General_Java" or "General_BPEL", based on the implementation type name contained in the general configuration of the test client.

6. ServiceInterface = "TestInvocation"
   In principle the test client allows for different Web service interfaces to be used to communicate from the test client to the SCA application. This piece of the configuration names the interface to use. However, in practice, for the SCA Assembly Test Suite, only a single interface is used for all of the testcases - "TestInvocation".

2.2.3 Creating an Alternative Test Client

If it is impossible or difficult to use the Java test client supplied with the test suite, it is possible to create a new test client written using some alternative technology. The main requirements are that:

- The test client SHOULD execute the testcases through Web services.[TST10008] , The Java test client is a good example of this approach. This is to ensure separation of the client from the SCA Runtime that is being tested.
- The test client MUST have a means of starting the SCA Runtime, a means of passing the test artifacts to the SCA Runtime (contributions and test composite) and a means of starting the SCA application (based on the test composite) [TST10009]
- The test client MUST have a mechanism for holding the configuration of each testcase defined in the test suite and of using that configuration when called on to execute a particular testcase. [TS T10010]
- The test client MUST be able to compare the expected output of the test with the actual output, including those testcases that are expected to fail with an exception. The test client MUST report the success or failure of the testcase dependant on whether the expected output matches the actual output. [TST10011]
  (Note that in the case of testcases which are expected to raise an exception, SCA makes no statement concerning what exception is raised - only that some exception is raised)

3 Conformance

For an adapted version of the SCA Assembly Test Suite that claims to conform to the requirements of this specification MUST meet the following conditions

1. The adapted version of the test suite MUST comply with all the mandatory statements listed in in the table Mandatory Items in the appendix "Conformance Items"

A. Required Artifacts

A.1. Composite Files in the Implementation Type Contribution

The General_xxx implementation type contribution contains composite file with the following names:

- TestClient_0002
- TestClient_0003
- TestClient_0004
- TestComposite1
- TestComposite4
- TestComposite5
- TestComposite6
- TestComposite7
- TestComposite8
- TestComposite9
- TestComposite52
- TestComposite53
- TestComposite54
- TestComposite55
- TestComposite60
- TestComposite61
- TestComposite65
- TestComposite66
- TestComposite70
- TestComposite71
- TestComposite73

The componentType of the composites is as shown in the following listings:

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="TestInvocation">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/
                    #wsdl.porttype(TestInvocation)"/>
   </service>

   <property name="testName" type="xsd:string"/>

   <reference name="reference1" multiplicity="1..1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </reference>

</componentType>
```

Listing A1: ComponentType of TestClient_0002 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="TestInvocation">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/
                    #wsdl.porttype(TestInvocation)"/>
   </service>

   <property name="testName" type="xsd:string"/>

   <reference name="reference1" multiplicity="1..n">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </reference>

</componentType>
```

Listing A2: ComponentType of TestClient_0003 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="TestInvocation">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/
                    #wsdl.porttype(TestInvocation)"/>
   </service>

   <property name="testName" type="xsd:string"/>

   <reference name="reference1" multiplicity="0..1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </reference>

</componentType>
```

Listing A3: ComponentType of TestClient_0004 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

</componentType>
```

Listing A4: ComponentType of TestComposite1 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

   <reference name="Reference1" multiplicity="1..1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </reference>

</componentType>
```

Listing A5: ComponentType of TestComposite4 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service2)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

</componentType>
```

Listing A6: ComponentType of TestComposite5 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

   <reference name="Reference1" multiplicity="1..n">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </reference>

</componentType>
```

Listing A7: ComponentType of TestComposite6 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

   <reference name="Reference1" multiplicity="0..n">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </reference>

</componentType>
```

Listing A8: ComponentType of TestComposite7 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

   <reference name="Reference1" multiplicity="0..1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </reference>

</componentType>
```

Listing A9: ComponentType of TestComposite8 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1Superset">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/
                    #wsdl.porttype(Service1Superset)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

</componentType>
```

Listing A10: ComponentType of TestComposite9 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service4)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

</componentType>
```

Listing A11 ComponentType of TestComposite52 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

   <reference name="Reference1" multiplicity="1..1">
      <interface.wsdl
```

```
            interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service4)"/>
      </reference>

   </componentType>
```

Listing A12: ComponentType of TestComposite53 composite

```
   <componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <service name="Service1">
         <interface.wsdl
            interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
      </service>

      <property name="serviceName" type="xsd:string"/>

      <reference name="Reference1" multiplicity="1..1">
         <interface.wsdl
            interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service5)"
            callbackInterface="http://test.sca.oasisopen.org/
                               #wsdl.porttype(Service5Callback)"/>
      </reference>

   </componentType>
```

Listing A13: ComponentType of TestComposite54 composite

```
   <componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <service name="Service1">
         <interface.wsdl
            interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service5)"
            callbackInterface="http://test.sca.oasisopen.org/
                               #wsdl.porttype(Service5Callback)"/>
      </service>

      <property name="serviceName" type="xsd:string"/>

   </componentType>
```

Listing A14 ComponentType of TestComposite55 composite

```
   <componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <service name="Service1">
         <interface.wsdl
            interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
      </service>

      <property name="serviceName" type="xsd:string"/>

      <reference name="Reference1" multiplicity="1..1">
         <interface.wsdl
            interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service9)"/>
      </reference>

   </componentType>
```

Listing A15: ComponentType of TestComposite60 composite

```
   <componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <service name="Service1">
         <interface.wsdl
            interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service9)"/>
      </service>

      <property name="serviceName" type="xsd:string"/>

   </componentType>
```

Listing A16 ComponentType of TestComposite61 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service5)"
         callbackInterface="http://test.sca.oasisopen.org/
                           #wsdl.porttype(Service5Callback)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

</componentType>
```

Listing A17 ComponentType of TestComposite65 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/
                    #wsdl.porttype(Service1_Intent)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

</componentType>
```

Listing A18 ComponentType of TestComposite66 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>

   <reference name="Reference1" multiplicity="1..1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/
                    #wsdl.porttype(Service1Superset)"/>
   </reference>

</componentType>
```

Listing A19: ComponentType of TestComposite70 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
```

```
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>
   <property name="serviceData1" type="xsd:string"/>
   <property name="serviceData2" type="xsd:string"/>

</componentType>
```

Listing A20 ComponentType of TestComposite71 composite

```
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <service name="Service1">
      <interface.wsdl
         interface="http://test.sca.oasisopen.org/#wsdl.porttype(Service1)"/>
   </service>

   <property name="serviceName" type="xsd:string"/>
   <property name="serviceData1" type="xsd:float"/>

</componentType>
```

Listing A21 ComponentType of TestComposite73 composite

B. Conformance Items

This section contains a list of conformance items for the SCA Assembly Test Suite Adaptation specification.

B.1. Mandatory Items

Conformance ID	Description
[TST10001]	The adapted test suite MUST contain a directory which contains an SCA contribution, termed the "implementation type contribution", holding the implementation-type dependent artifacts,
[TST10002]	The implementation type contribution must have a name that starts with the stem "General_" followed by a suffix which is unique to the implementation type.
[TST10003]	The "implementation type contribution" MUST contain a set of composite files as defined in section "Composite Files in the Implementation Type Contribution", each file having the composite name and having the componentType defined there, and where the components in those composite files use implementations of the type supported by the adapted test suite.
[TST10004]	The composites contained in the "implementation type contribution" MAY use an implementation-type specific form of interface declaration rather than using the interface.wsdl form of interface declaration.
[TST10005]	If an implementation type requires the use of a new interface definition type, the adapted test suite MUST contain an interface definition type version of each of the interface artifacts contained in Listing 3.
[TST10006]	Each interface artifact in the adapted test suite MUST map to equivalent WSDL file of the same name contained in the General contribution.
[TST10007]	The test client MUST load all the contributions declared in the testcase configuration into the SCA Runtime.
[TST10008]	The test client SHOULD execute the testcases through Web services
[TST10009]	The test client MUST have a means of starting the SCA Runtime, a means of passing the test artifacts to the SCA Runtime (contributions and test composite) and a means of starting the SCA application (based on the test composite)
[TST10010]	The test client MUST have a mechanism for holding the configuration of each testcase defined in the test suite and of using that configuration when called on to execute a particular testcase.
[TST10011]	The test client MUST be able to compare the expected output of the test with the actual output, including those testcases that are expected to fail with an exception. The test client MUST report the success or failure of the testcase dependant on whether the expected output matches the actual output.

C. Acknowledgments

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

Participants:

- Mike Edwards, IBM
- Dave Booz, IBM
- Bryan Aupperle, IBM
- Jeff Estefan, Jet Propulsion Laboratory

D. Revision History

Revision	Date	Editor	Changes Made
1	04/19/10	Mike Edwards	Initial version created
2	04/22/10	Mike Edwards	Updated and extended based on feedback
3	05/12/10	Jeff Estefan	Clean-up and formatting consistency
4	05/13/10	Mike Edwards	Initial set of normative statements Appendix for normative statements added Appendix for General_xxx contents added
5	06/15/10	Mike Edwards	Added line numbering
cd01	06/20/10	Mike Edwards	All changes accepted.