# Apache Tuscany 2.x and SCA 1.1

*...Model, compose, deploy and manage service based applications*

**Specs - OASIS OpenCSA (v1.1)**
- Get the v1.1 SCA specifications
  www.oasis-opensca.org

**Apache Tuscany**
- Read about the Tuscany open source SCA runtime
  tuscany.apache.org
- Download Tuscany SCA 2.x releases from
  tuscany.apache.org/sca-java-2x-releases.html

# Getting Started

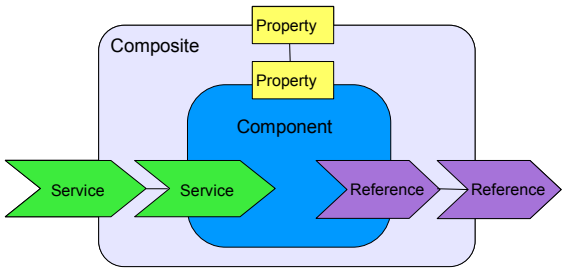
If you want to build SCA applications first get a Tuscany binary release (see link above) and try out one of the samples, for example,

```
cd tuscany_release_dir/samples/calculator
ant run
```

To build your own composite applications you start by creating an SCA contribution. You use the Tuscany runtime to load and run the contribution. There are guides with more details here

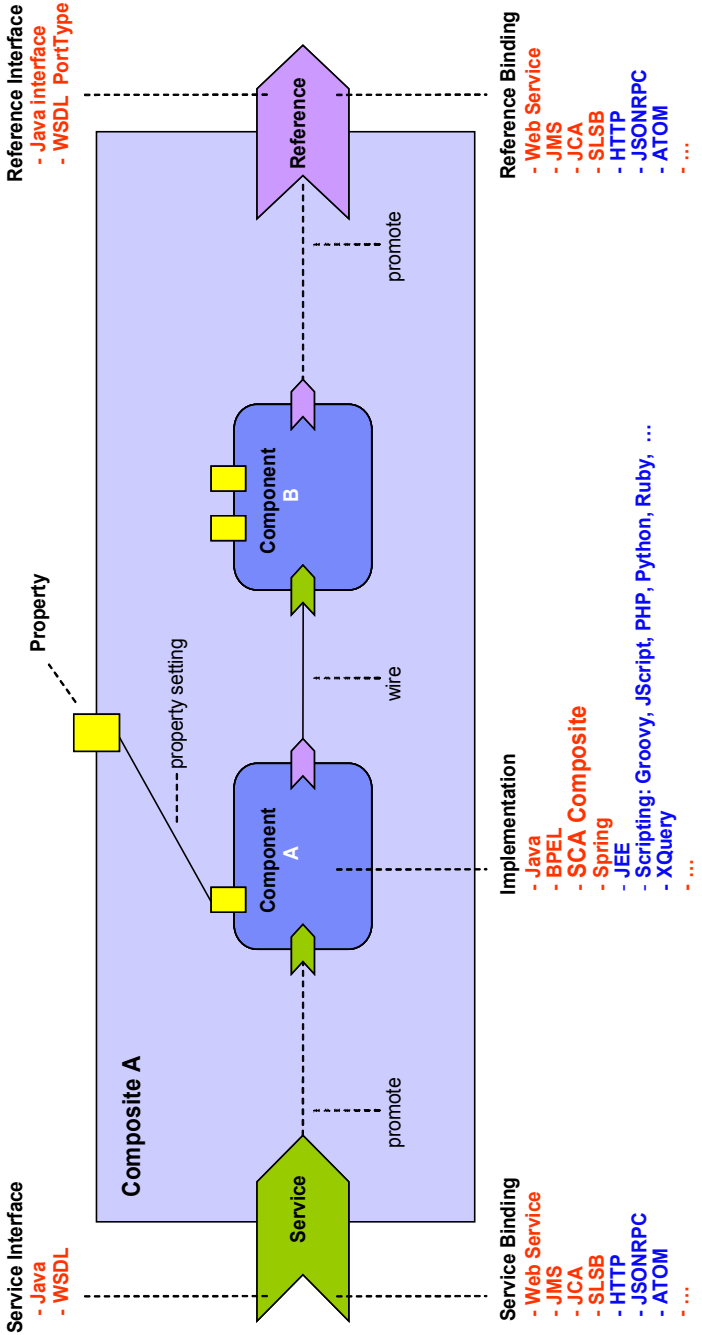
tuscany.apache.org/documentation-2x

# Composites

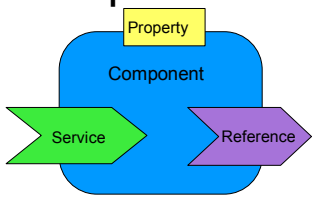



```
<composite
      xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
      targetNamespace="xs:anyURI"
      name="xs:NCName"
      local="xs:boolean"?
      autowire="xs:boolean"?
      requires="list of xs:QName"?
      policySets="list of xs:QName"?>
    <include … />*
    <requires/>*
    <policySetAttachment/>*
    <service … />*
    <reference … />*
    <property … />*
    <component … />*
    <wire … />*
</composite>
```

# Terminology Overview




# Components




```
<component name="xs:NCName"
           autowire="xs:boolean"?
           requires="list of xs:QName"?
           policySets="list of xs:QName"?>*
      <implementation … />?
      <service … />*
      <reference … />*
      <property … />*
      <requires/>*
      <policySetAttachment/>*
</component>
```

## Component Services

```
<service name="xs:NCName"
         requires="list of xs:QName"?
         policySets="list of xs:QName"?>*
    <interface … />?
    <binding … />*
    <callback>?
        <binding … />+
    </callback>
    <requires/>*
    <policySetAttachment/>*
</service>
```

## Component References

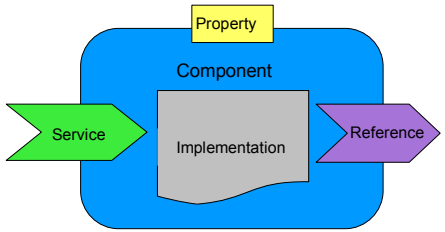
```
<reference name="xs:NCName"
           target="list of xs:anyURI"?
           autowire="xs:boolean"?
           multiplicity="0..1 or 1..1 or 0..n or 1..n"?
           nonOverridable="xs:boolean"
           wiredByImpl="xs:boolean"?
           requires="list of xs:QName"?
           policySets="list of xs:QName"?>*
    <interface … />?
    <binding uri="xs:anyURI"?
             requires="list of xs:QName"?
             policySets="list of xs:QName"?/>*
    <callback>?
        <binding … />+
    </callback>
    <requires/>*
    <policySetAttachment/>*
</reference>
```

## Component Properties

```
<property name="xs:NCName"
          (type="xs:QName" | element="xs:QName")?
          many="xs:boolean"?
          source="xs:string"? file="xs:anyURI"?
          value="xs:string"?>*
     [<value>+ | xs:any+ ]?
</property>
```

# Tuscany 2.x Extensions

## Component Implementations




```
<implementation.java class="helloworld.HelloWorldImpl"/> |
<implementation.composite name="sample:InnerComposite"/> |
<implementation.spring location="service-context.xml"/> |
<implementation.bpel process="hns:HelloWorld"/> |
<tuscany:implementation.script script="ServiceImpl.js |
                                       ServiceImpl.rb |
                                       ServiceImpl.py |
                                       ServiceImpl.groovy"/>|
<tuscany:implementation.widget location="ui.html"/>
```

## Service/Reference Interfaces

```
<interface.java interface="some.ServiceIface"
  callbackInterface="some.ServiceCallbackIface"/> |

<interface.wsdl
 interface="http://xyy/svc#wsdl.interface(Service)"
 callbackInterface="http://xyy/svc#wsdl.interface(Clbk)"/>
```

## Service/Reference Bindings

```
<binding.sca/>
<binding.ws/>
<binding.jms/>
<binding.ejb/>
<binding.rmi/>
<binding.corba/>
<binding.http/>
<binding.jsonrpc/>
<binding.atom/>
<binding.rss/>
<binding.dwr/>
```

# Tuscany Runtime Development

If you want to work with the Tuscany runtime source code itself then the following should help you.

## Development Environment

- Win/Lin/Mac are all OK.

- Java JDK (preferably 1.6)
  java.sun.com/javase/downloads/index.jsp

- Maven (2.2.1)
  maven.apache.org/download.html

- Subversion (1.6.6)
  subversion.apache.org/packages.html (if not already on your system)

- Eclipse IDE for Java (or other IDE)
  www.eclipse.org/downloads

## Development Instructions

There are some development instructions at:

tuscany.apache.org/documentation-2x/development-guides.html

Basically install Java, Subversion, and Maven, then:

```
svn co https://svn.apache.org/repos/asf/tuscany/sca-java-
2.x/trunk trunk
cd trunk
Set MVN_OPTS="-Xmx1024m -Xms512m -XX:MaxPermSize=512m"
mvn clean install -fae
```

The first mvn build can take a long time as it downloads all the required dependencies and may intermittently fail so just retry a few times.

The build compiles all the source and runs all the tests. You can run individual tests by changing directory to the module you're interested in an running maven.

```
cd samples/calculator
mvn
```

To build eclipse project files for all of the module do the following

```
cd trunk
mvn -o -fae eclipse:eclipse
```

You can then import these projects into Eclipse and develop there. You will need to configure the workspace so it knows where to find the Tuscany dependencies in the maven repo.

```
mvn -Declipse.workspace=[path-to-eclipse-workspace]
eclipse:add-maven-repo
```

The effect of this is to create a M2_REPO build path variable pointing at your local maven rspository.

# Tuscany SCA Calculator

## Composite File

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

 <component name="CalculatorServiceComponent">
   <implementation.java class="calculator.CalculatorServiceImpl"/>
   <reference name="addService" target="AddServiceComponent" />
   <reference name="subtractService" target="SubtractServiceComponent"/>
   <reference name="multiplyService" target="MultiplyServiceComponent"/>
   <reference name="divideService" target="DivideServiceComponent" />
 </component>

 <component name="AddServiceComponent">
   <implementation.java class="calculator.AddServiceImpl"/>
 </component>

 <component name="SubtractServiceComponent">
   <implementation.java class="calculator.SubtractServiceImpl"/>
 </component>

 <component name="MultiplyServiceComponent">
   <implementation.java class="calculator.MultiplyServiceImpl"/>
 </component>

 <component name="DivideServiceComponent">
   <implementation.java class="calculator.DivideServiceImpl"/>
 </component>

</composite>
```

## CalculatorServiceImpl.java

```
public class CalculatorServiceImpl implements CalculatorService {
    private AddService addService;
    private SubtractService subtractService;
    private MultiplyService multiplyService;
    private DivideService divideService;

    @Reference
    public void setAddService(AddService addService) {
        this.addService = addService;
    }

    @Reference
    public void setSubtractService(SubtractService subtractService) {
        this.subtractService = subtractService;
    }

    @Reference
    public void setDivideService(DivideService divideService) {
        this.divideService = divideService;
    }

    @Reference
    public void setMultiplyService(MultiplyService multiplyService) {
        this.multiplyService = multiplyService;
    }

    public double add(double n1, double n2) {
        return addService.add(n1, n2);
    }

    public double subtract(double n1, double n2) {
        return subtractService.subtract(n1, n2);
    }

    public double multiply(double n1, double n2) {
        return multiplyService.multiply(n1, n2);
    }

    public double divide(double n1, double n2) {
        return divideService.divide(n1, n2);
    }
}
```