IBM Software Group

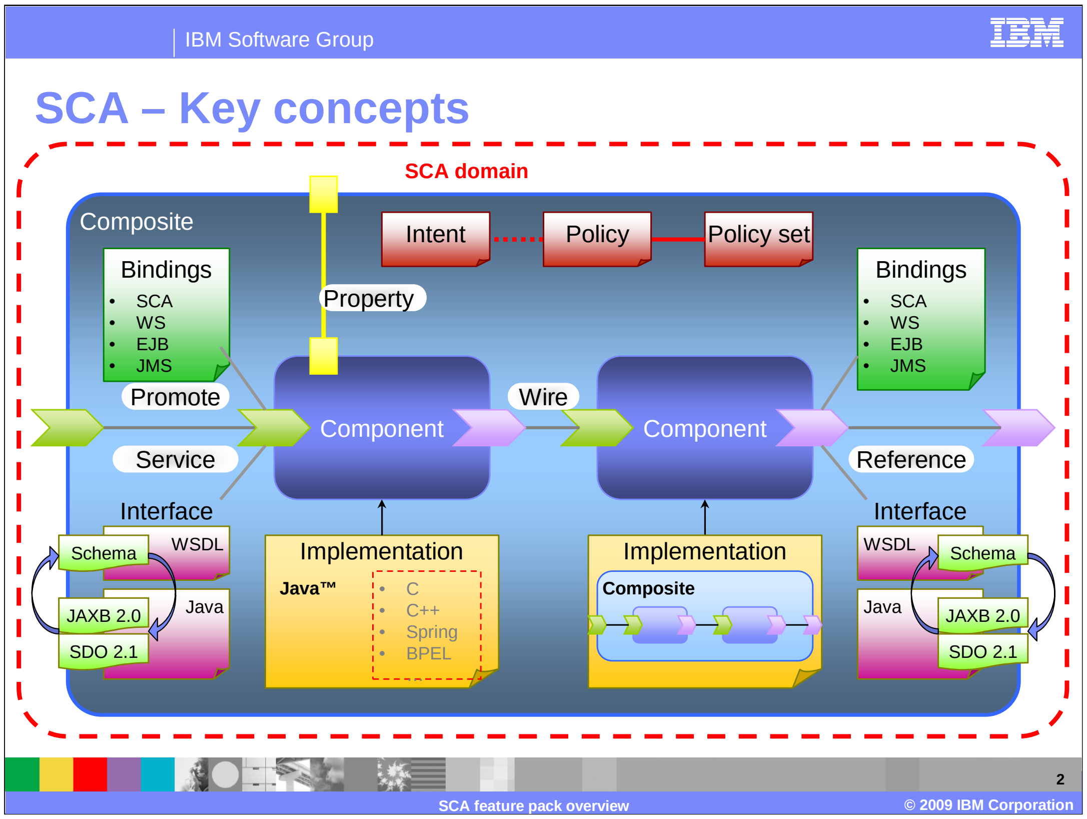
IBM WebSphere Application Server V7.0 Feature Pack for Service Component Architecture V1.0.1

SCA feature pack overview

business on demand.

Updated November 10, 2009

This presentation will discuss an overview of the WebSphere® Application Server V7.0 feature pack for Service Component Architecture (SCA). The presentation will highlight the concepts of SCA and specifications.



IBM Software Group

SCA – Key concepts

2

SCA feature pack overview © 2009 IBM Corporation

An essential characteristic of SOA is the ability to assemble new and existing services to create brand new applications that can consist of different technologies.

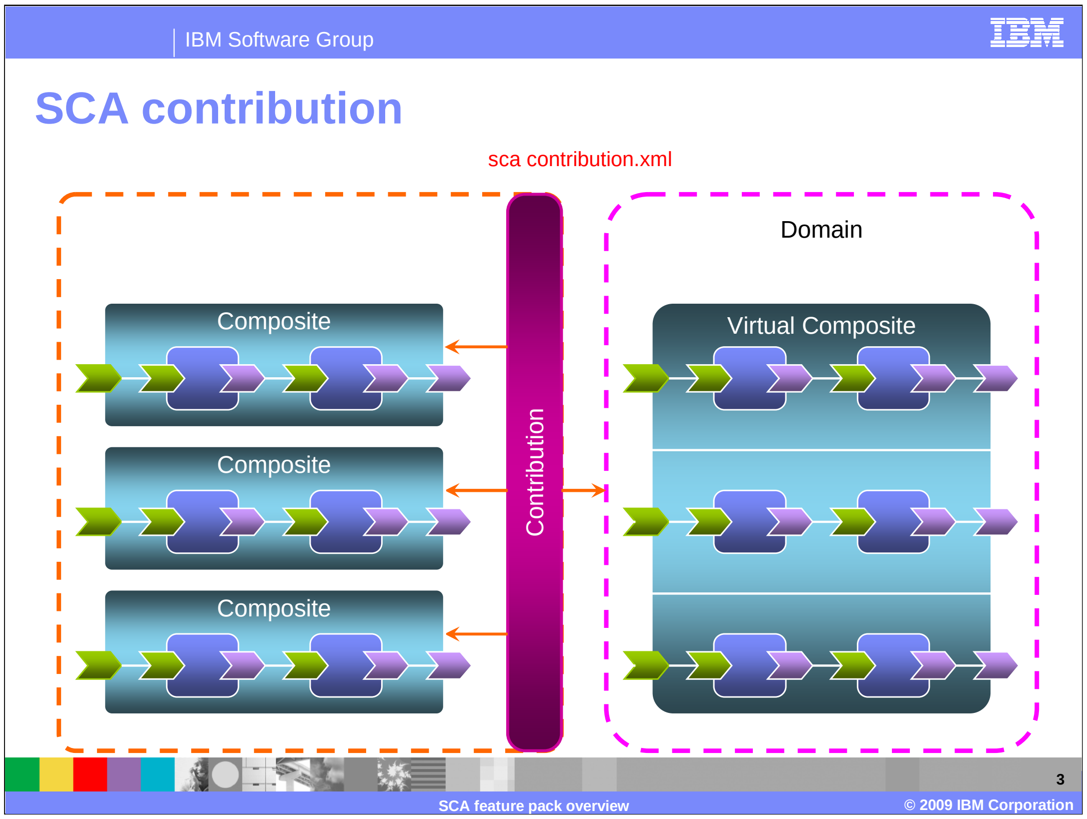
Service Component Architecture defines a simple, service-based model for construction, assembly and deployment of a network of services (both existing and new ones.) These services are defined in a language-neutral way. Tuscany implements the SCA Version 1.0. This picture covers the overview of SCA. It shows the basic concepts of what SCA is made up of. There's a lot to talk about on this chart.

As the name implies, Service Component Architecture is a component model. Service-oriented coarse-grained building blocks are represented as descriptions of components, shown here in blue. Components describe the services they provide, shown here by the green chevrons and the services they depend on or reference, shown here by the purple chevrons. Components also point at the chunk of code which provides the implementation, shown here in yellow, of the service it provides. Components are then connected together through wires, also using metadata. Components can tailor implementations through the usage of properties. Policy and quality of service intents can decorate services or references, called interaction intents, and can decorate components, called implementation intents.

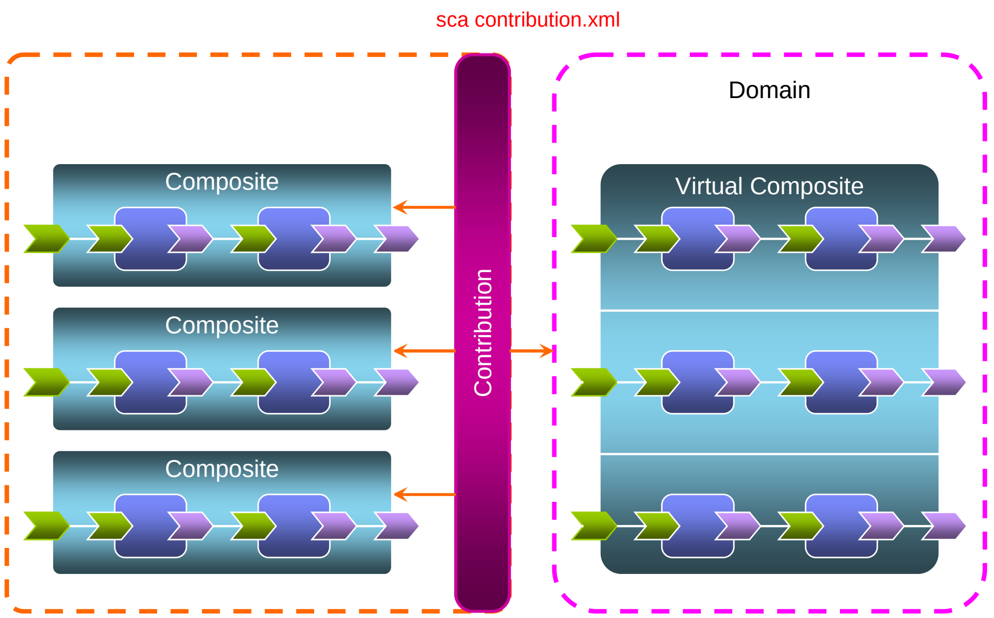
Assemblies of components are formally composed into ***composites***. A composite is the set of components and wires – the assembly of services. The composite provides a scoping mechanism which defines a local boundary for components, but further can hide services provided in components which are not intended for other SOA applications. Once defined, a composite can be reused to provide the implementation for other components in a nested fashion. The components, assemblies, internal wires and service and reference definitions are written in an open XML language called Service Component Definition Language (SCDL). Services and references in a composite are bound to specific protocols (such as Web services) through the usage of ***bindings,*** *shown in green*. The bindings are part of the SCDL definition and the business logic (implementation) does not need to be polluted with this detail.

Note that SCA is a language-neutral set of specifications which enable its usage in a variety of application environments. Each application environment will offer the set of implementation, quality of service, and policy features that make sense in that environment. For example, the SCA Feature Pack for WebSphere offers support for components whose implementations are Java or other SCA composites; and specifically does not support the deployment of C or C++ implementations.

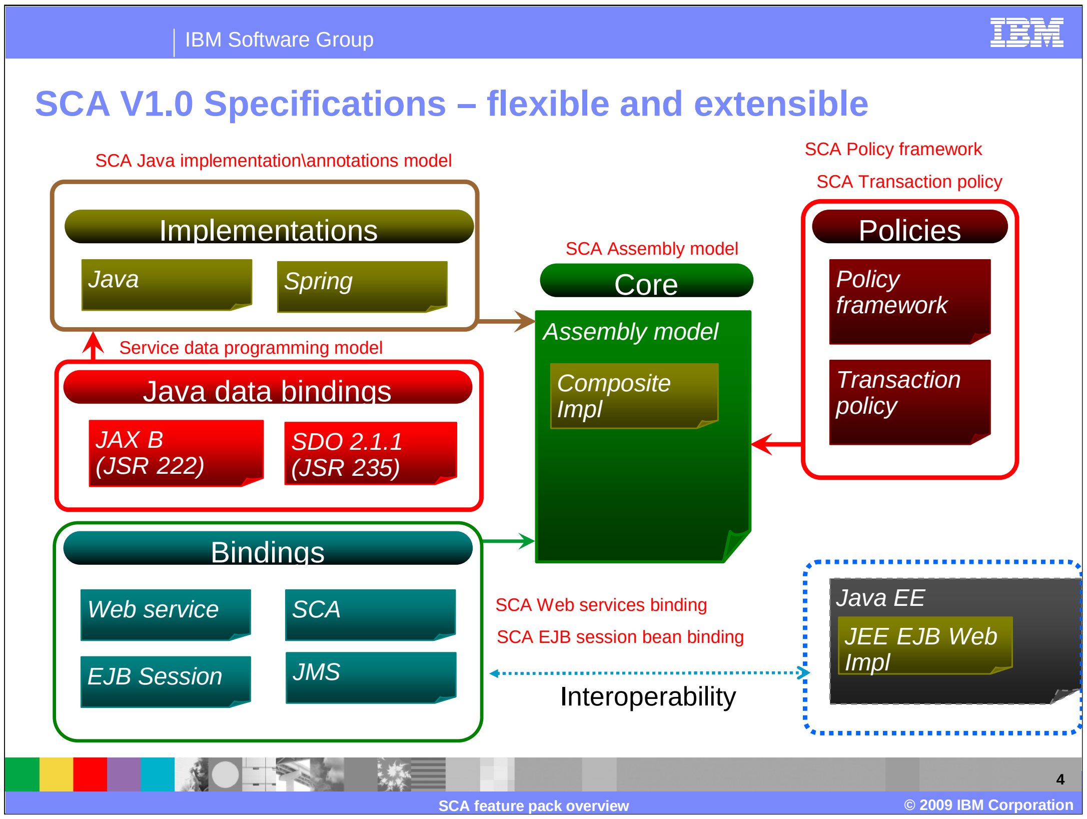
There is also the concept of **promote.** Just as components expose services, a composite can also expose one or more services. These services are actually implemented by components within the composite. To make them visible to the outside world, the composite's creator can *promote* those services. Last but not least, an **SCA Domain** represents a complete runtime configuration, potentially distributed over a series of interconnected runtime nodes. Components sit inside Composites which sit inside a domain. Note that in the SCA feature pack for example the SCA domain is the cell.



SCA contribution

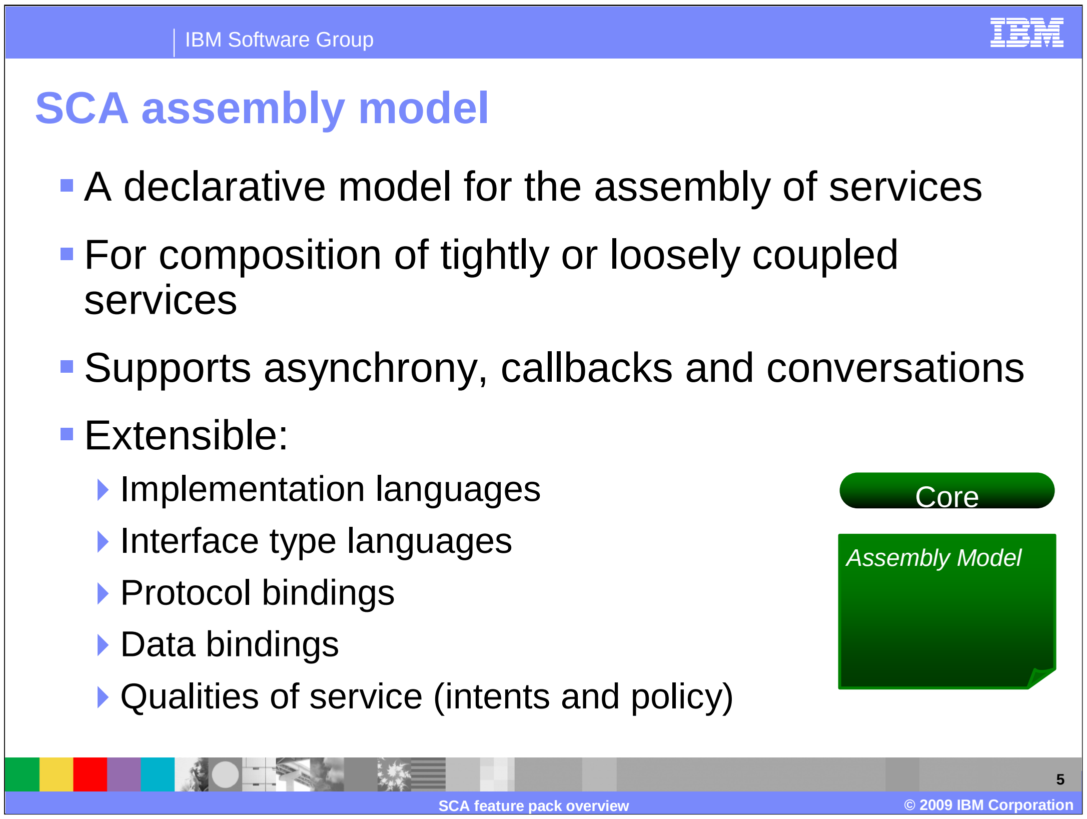


Once the composites are defined, they need to be deployed into a runtime. A special xml document called a *contribution* document (sca contribution.xml) describes how composites should be deployed. It contains information regarding which composites in the deployable jar are executable, what their namespaces are, and which can be used by other contributions. Contributions are deployed into an SCA *domain*. The SCA domain is an administration scoping mechanism, but it also provides a service catalog which can be used by the SCA runtime to simplify the wiring of SCA composites. Services deployed in the SCA domain are available over the *SCA default binding.* This allows wires to be specified by its logical domain name without having to specify the particulars of the endpoint configuration as is the case with other bindings. For the purposes of the SCA feature pack for WebSphere, the SCA domain and cell have the same scope (as mentioned in previous slide). Services deployed in the SCA domain are available throughout the WebSphere cell over the default binding.



SCA V1.0 Specifications – flexible and extensible

Service component architecture V1.0 has a couple of specifications which allow for flexibility and extensibility when developing SCA applications. Assembly model specification is a specification that defines structure of composite applications. Policies group has two specifications namely policy framework and transaction policy. The two define how to add infrastructure services to solutions, security, transactions, reliable messaging and so on. Implementations have the two Java specifications that define how to write business services in particular languages. Bindings has specifications for each binding example Web service binding specification, JMS binding specification and so on. Integrations group has the JEE specification. The next few slides go into more detail about these specifications.

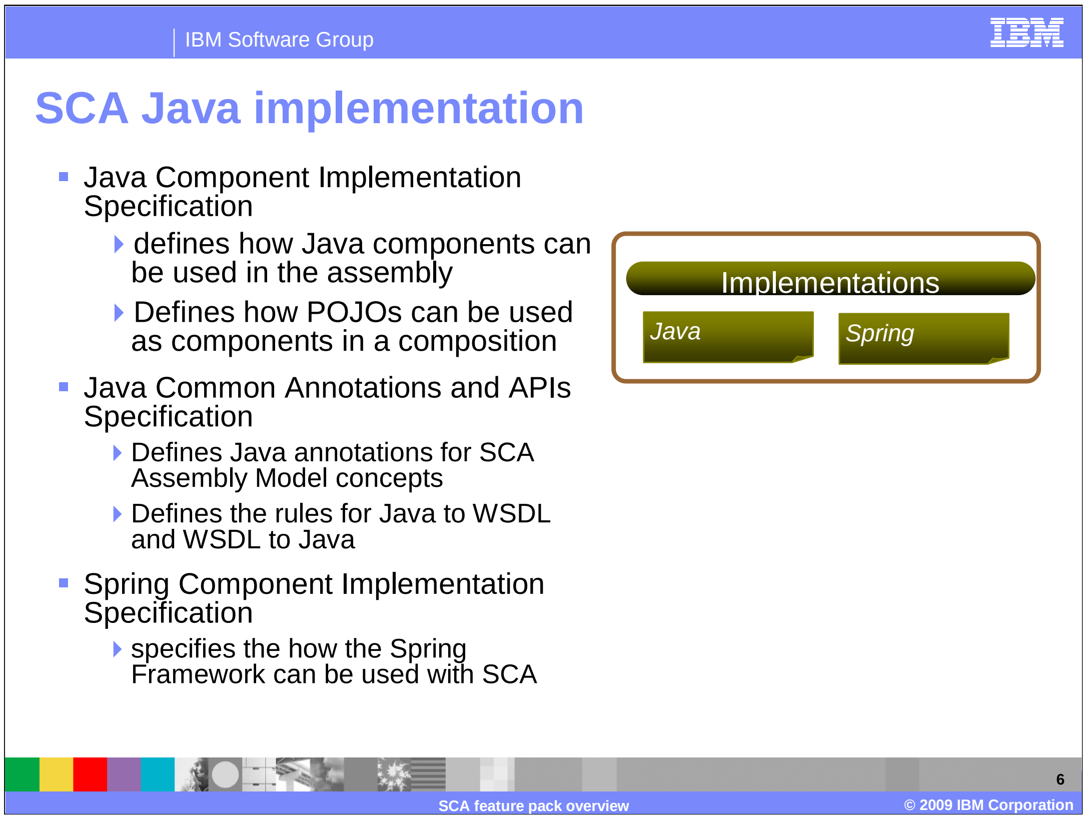


SCA assembly model

- A declarative model for the assembly of services
- For composition of tightly or loosely coupled services
- Supports asynchrony, callbacks and conversations
- Extensible:
  - Implementation languages
  - Interface type languages
  - Protocol bindings
  - Data bindings
  - Qualities of service (intents and policy)

Core
Assembly Model

The SCA assembly model provides specifications on how to define structure of composite applications. It is a model for the assembly of services, both tightly coupled and loosely coupled. It is also a model for applying infrastructure capabilities to services and to service interactions, including security and transactions. The SCA ***Assembly Model*** consists of a series of artifacts which define the configuration of an SCA domain in terms of composites. These contain assemblies of service components and the connections and related artifacts which describe how they are linked together.



IBM Software Group

SCA Java implementation

- Java Component Implementation Specification
  - defines how Java components can be used in the assembly
  - Defines how POJOs can be used as components in a composition
- Java Common Annotations and APIs Specification
  - Defines Java annotations for SCA Assembly Model concepts
  - Defines the rules for Java to WSDL and WSDL to Java
- Spring Component Implementation Specification
  - specifies the how the Spring Framework can be used with SCA

Implementations
Java
Spring

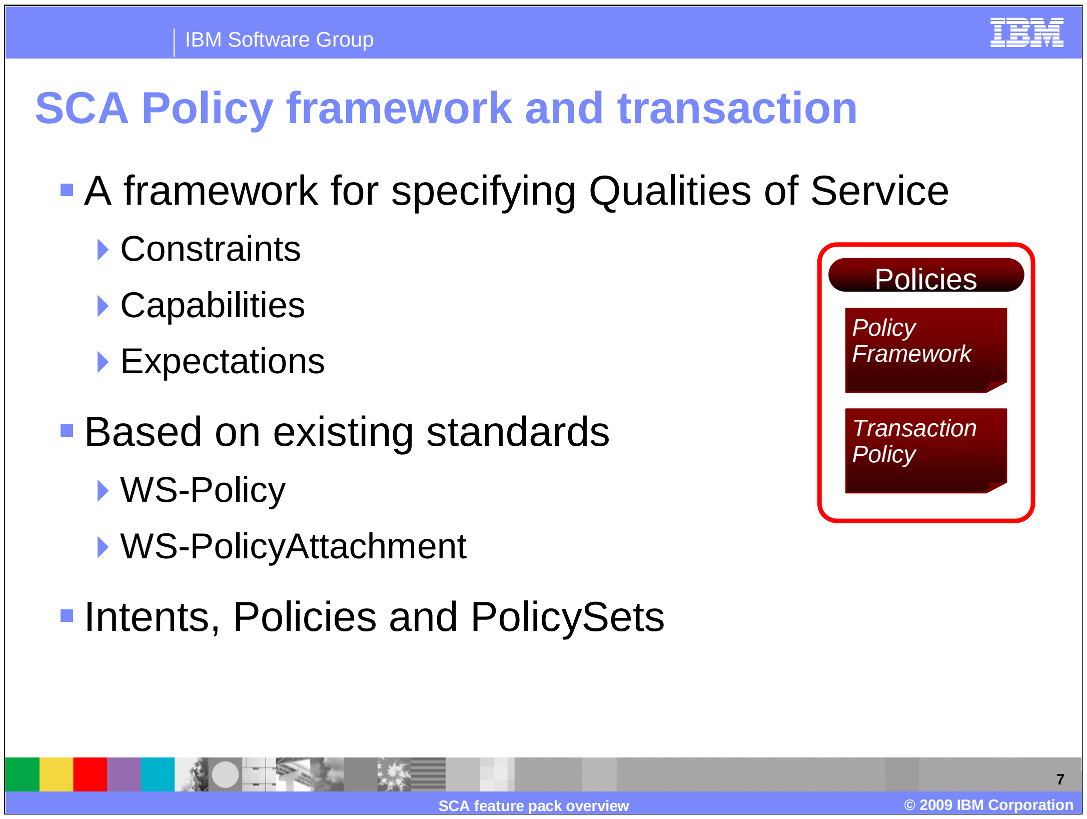
6

SCA feature pack overview

© 2009 IBM Corporation

SCA adopts the ease of use improvements provided by JEE5 annotations and modern container design supporting inversion of control or dependency injection. SCA Java implementations have the benefit of declaring service definitions using annotations; having SCA service references, policy and properties injected into their code by the SCA container. Implementations that use this pattern are not only insulated from the specifics of endpoint configuration, but they are also insulated from specific SCA framework APIs. The SCA architecture provides for enterprises to override annotations provided in the code with formal metadata, allowing you to keep things simple during development but giving the enterprise deployment the stringent control over service policy and configuration. There are two Java specifications: Java component implementation specification which extends the SCA assembly model and java common annotations and APIs specification. There is also spring component implementation specification. The SCA Java client and implementation model for spring specifies how the spring framework can be used with SCA.

Note that you can incorporate Java components into assembly model without adding any annotation. Together, these models provide the specifications for writing business services in particular languages like java, c++, BPEL, PHP, spring and so on. Note that for SCA feature pack, the implementation is in java. Also note that this time around spring is supported.

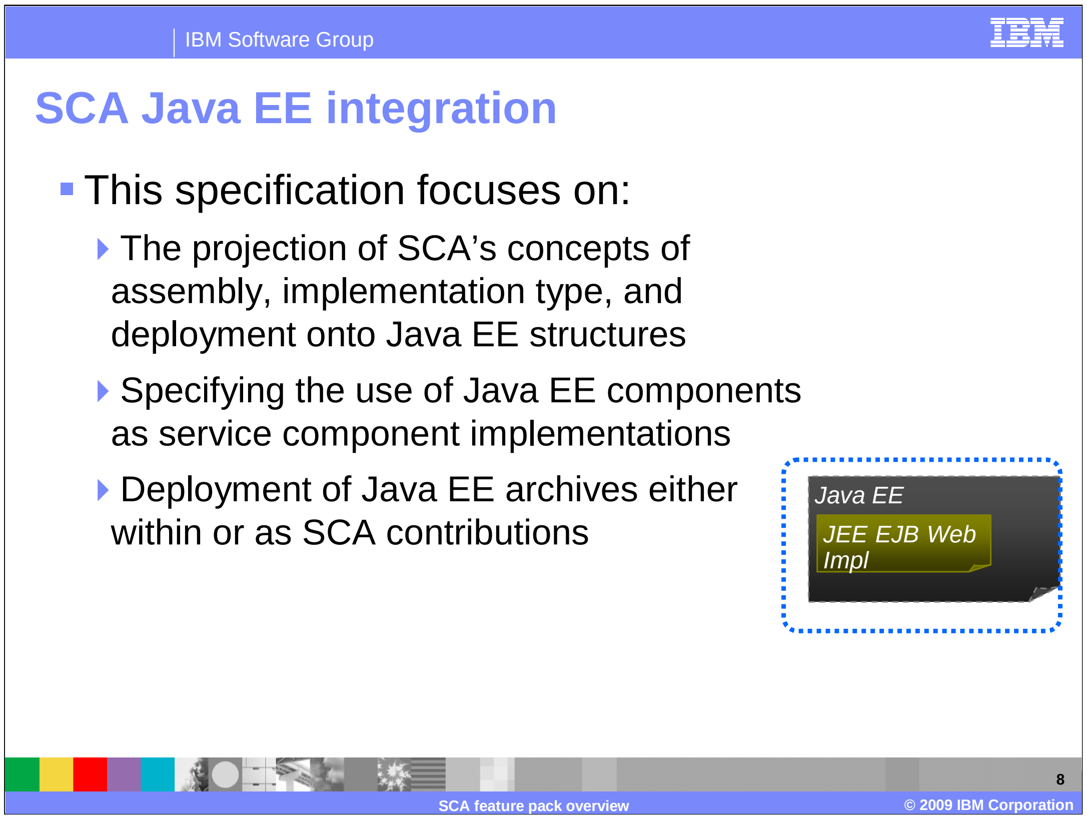


SCA Policy framework and transaction

- A framework for specifying Qualities of Service
  - Constraints
  - Capabilities
  - Expectations
- Based on existing standards
  - WS-Policy
  - WS-PolicyAttachment
- Intents, Policies and PolicySets

Policies
Policy Framework
Transaction Policy

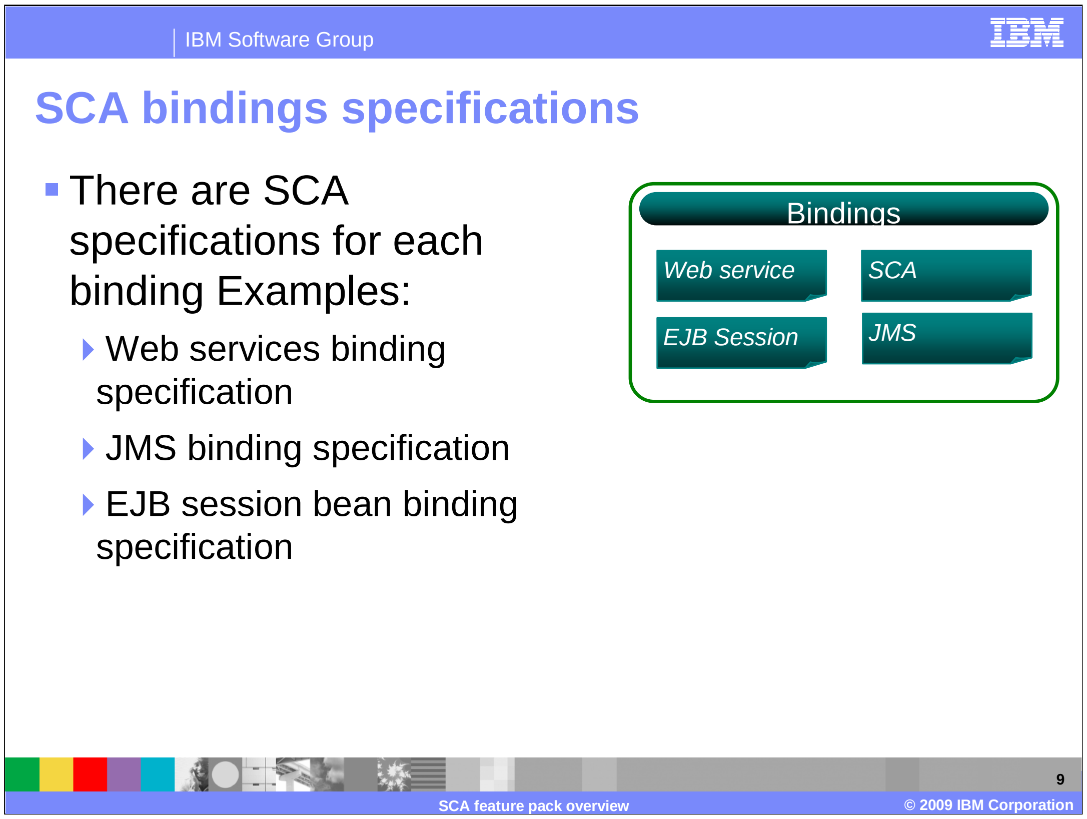
SCA provides a framework to support specification of constraints, capabilities and QoS expectations from component design through to concrete deployment. This specification describes the framework and its usage. It covers policies and policy languages, to be associated with SCA components. Basically it's a specification that provides the requirements on how to add infrastructure services to solutions security, transactions, reliable messaging, and so on.



SCA Java EE integration

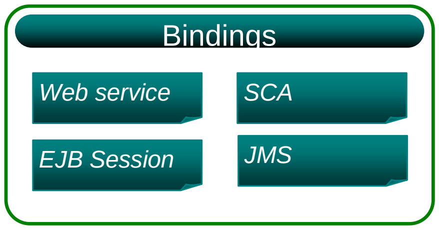
- This specification focuses on:
  - The projection of SCA's concepts of assembly, implementation type, and deployment onto Java EE structures
  - Specifying the use of Java EE components as service component implementations
  - Deployment of Java EE archives either within or as SCA contributions

The Java EE client and implementation specification focuses on the projection of SCA's concepts of assembly, implementation type, and deployment onto Java EE structures. This specification defines the integration of SCA and Java EE within the context of a Java EE application. It defines the use of Java EE components as service component implementations, and the deployment of Java EE archives either within or as SCA contributions. It is also possible to use bindings to achieve some level of integration between SCA and Java EE. These bindings are addressed in separate specifications.

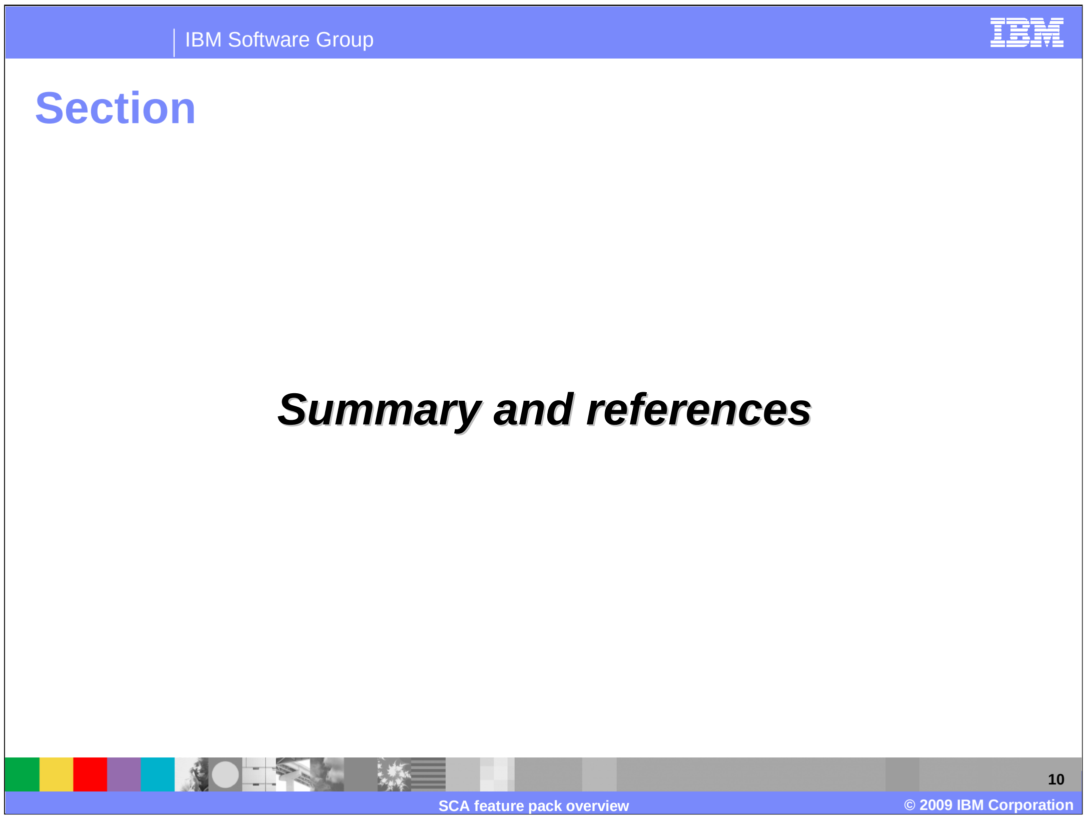


SCA bindings specifications

- There are SCA specifications for each binding Examples:
  - Web services binding specification
  - JMS binding specification
  - EJB session bean binding specification



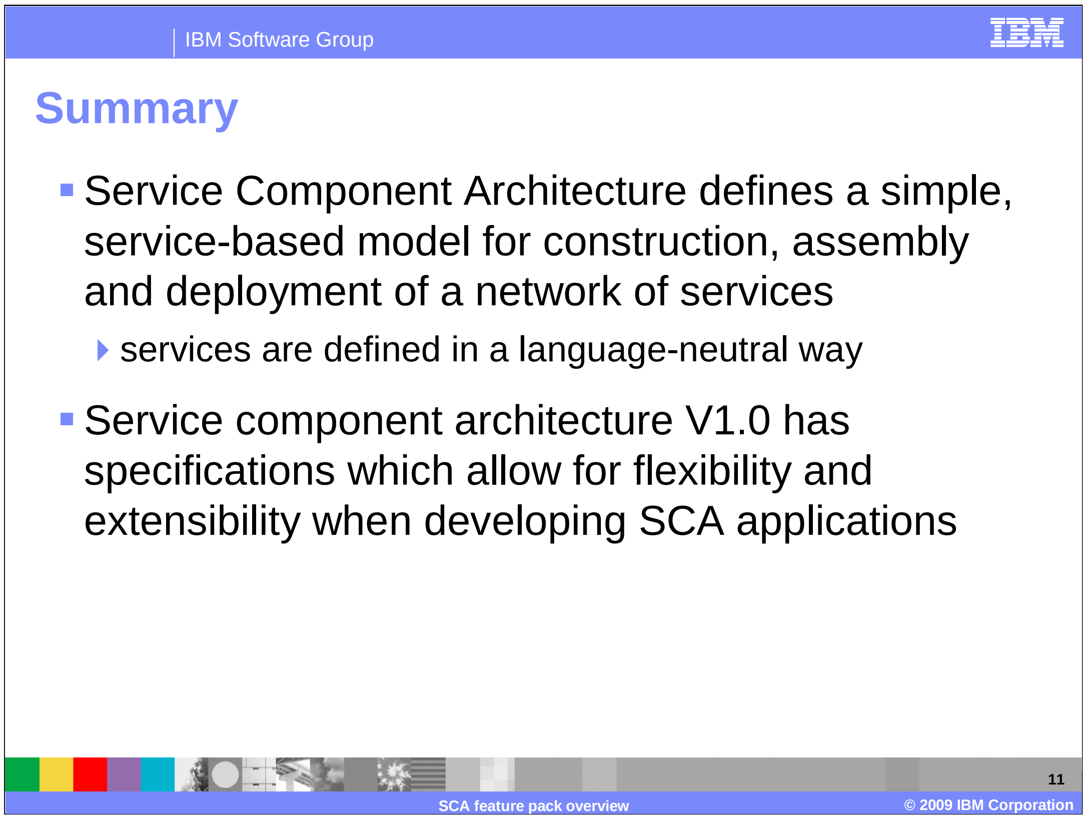
There are specifications for the different bindings involved namely Web service binding, EJB binding and JMS binding. Each of these provide specifications for each of these bindings. As an example, the Web service binding specification defines the manner in which a service can be made available as a Web service, and in which a reference can invoke a Web service. The JMS binding specification provides JMS-specific details of the connection to the required JMS resources. It supports the use of Queue and Topic type destinations. The EJB binding specification describes how to integrate a previously deployed session bean into an SCA assembly, and how to expose SCA services to clients which use the EJB programming model. Note that the Web service binding is important because it's INTEROPERABLE (WS-I).



Section

Summary and references

The next section provides a summary and references for this presentation.

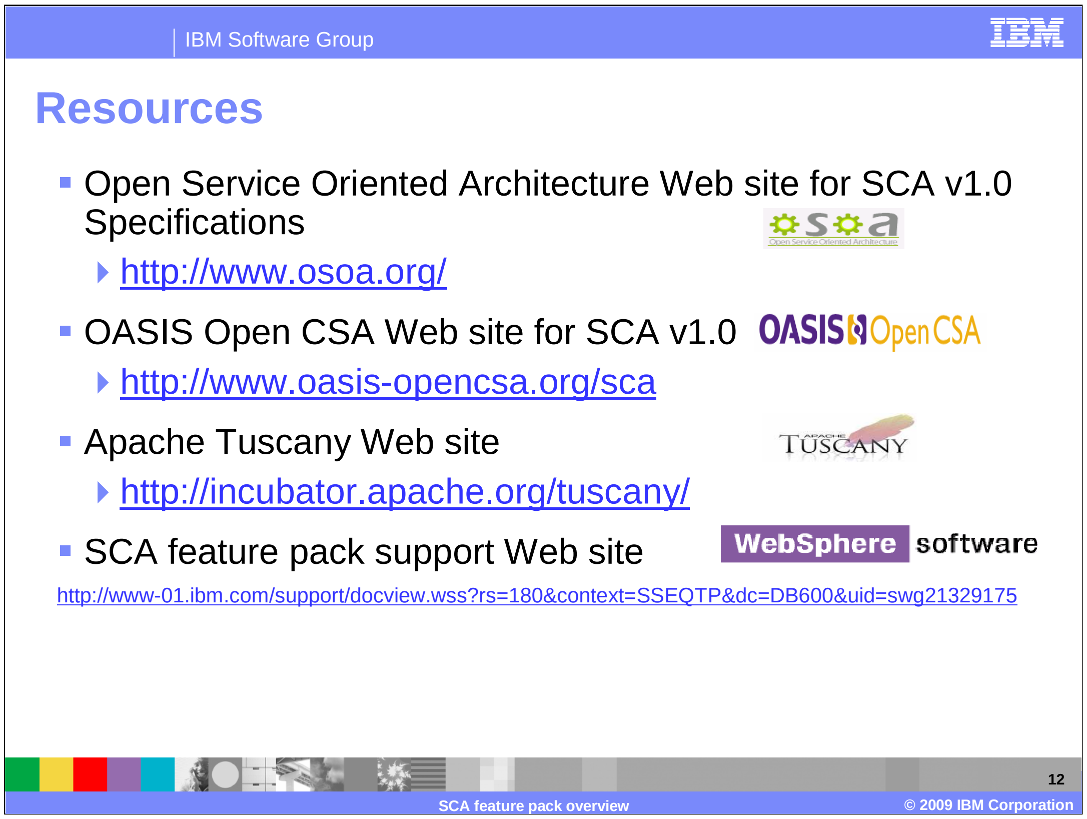


Summary

- Service Component Architecture defines a simple, service-based model for construction, assembly and deployment of a network of services
  - services are defined in a language-neutral way
- Service component architecture V1.0 has specifications which allow for flexibility and extensibility when developing SCA applications

An essential characteristic of SOA is the ability to assemble new and existing services to create brand new applications that might consist of different technologies.

Service Component Architecture defines a simple, service-based model for construction, assembly and deployment of a network of services (both existing and new ones). These services are defined in a language-neutral way. Tuscany implements the SCA Version 1.0. Service component architecture V1.0 has specifications which allow for flexibility and extensibility when developing SCA applications.

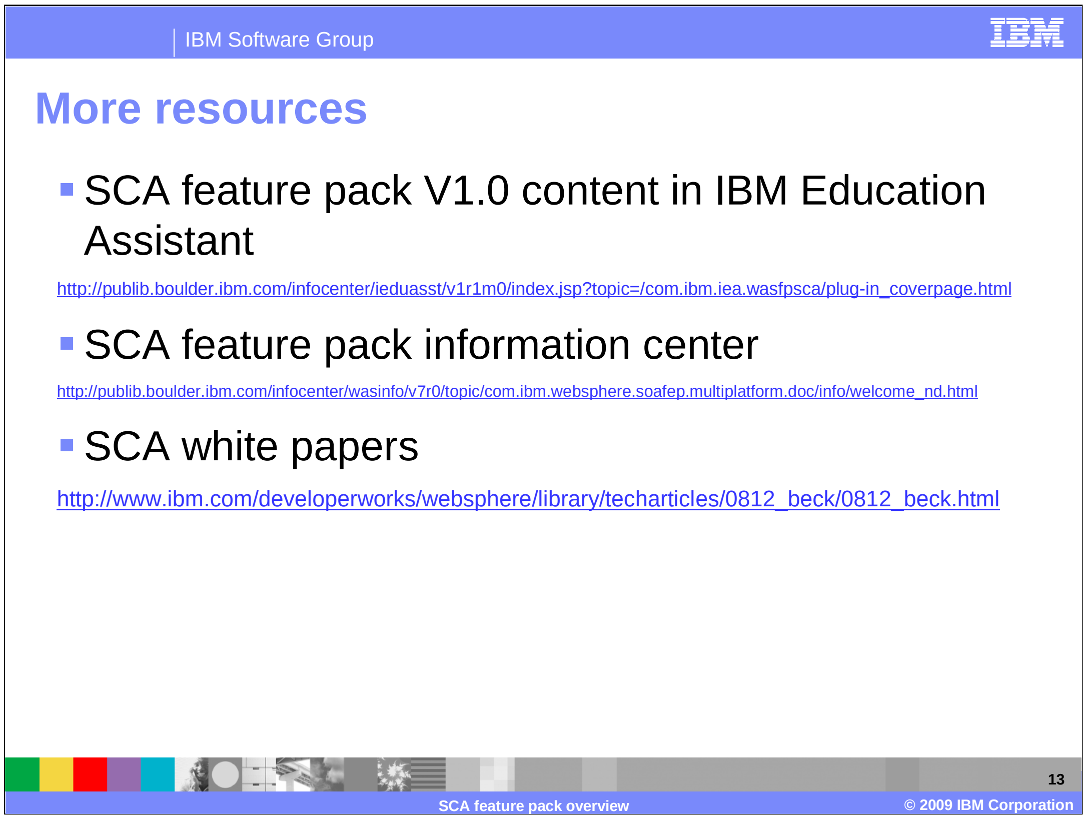


Resources

- Open Service Oriented Architecture Web site for SCA v1.0 Specifications
  - http://www.osoa.org/
- OASIS Open CSA Web site for SCA v1.0
  - http://www.oasis-opencsa.org/sca
- Apache Tuscany Web site
  - http://incubator.apache.org/tuscany/
- SCA feature pack support Web site

http://www-01.ibm.com/support/docview.wss?rs=180&context=SSEQTP&dc=DB600&uid=swg21329175

Listed here are some resources that you might find valuable.



More resources

- SCA feature pack V1.0 content in IBM Education Assistant

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wasfpsca/plug-in_coverpage.html

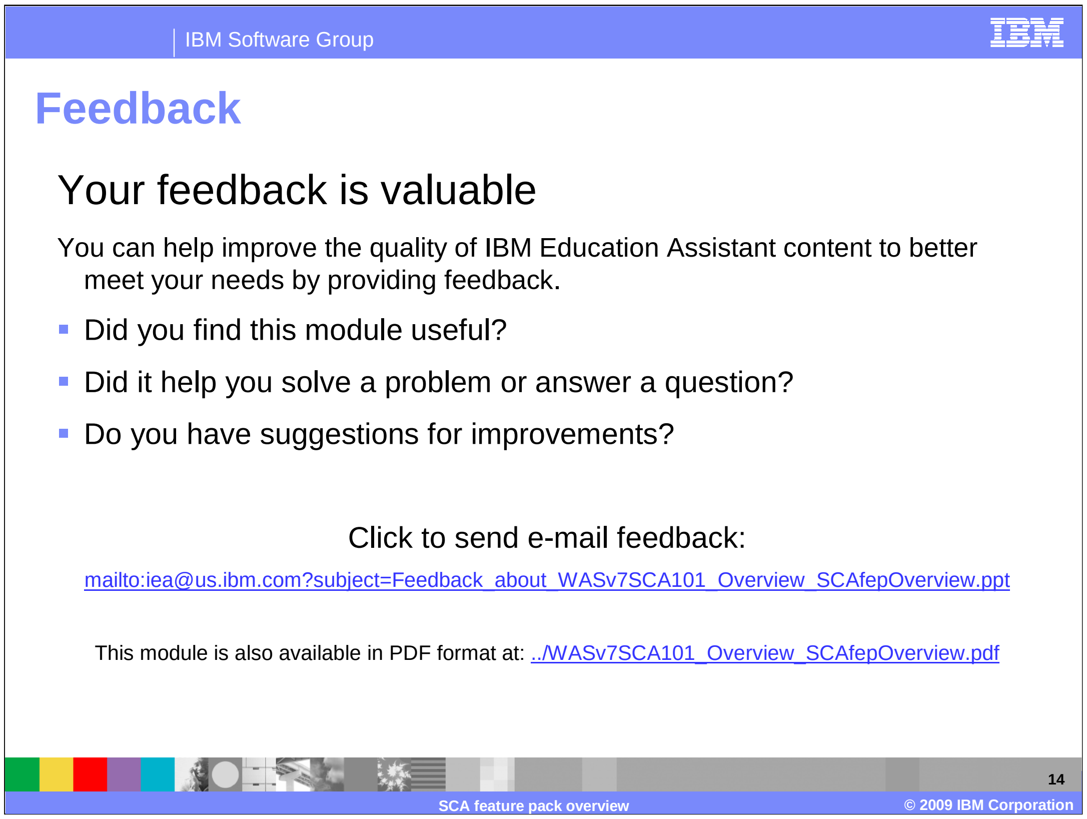
- SCA feature pack information center

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.soafep.multiplatform.doc/info/welcome_nd.html

- SCA white papers

http://www.ibm.com/developerworks/websphere/library/techarticles/0812_beck/0812_beck.html

More resources



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7SCA101_Overview_SCAfepOverview.ppt

This module is also available in PDF format at: ../WASv7SCA101_Overview_SCAfepOverview.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers