Composite Application Deployment with SCA Domain

Composite Application Deployment with SCA Domain

The sample scenarios

A retail application is built using SCA. The application is developed and packaged as 3 contributions:

- Asset: The common interfaces such as Catalog, Cart, Total
- Store: The java implementation of Catalog, ShoppingCart and CurrencyConverter
- Store-Client: The store client implementation

Both Store and Store-Client have dependency (importing the interfaces) on Asset.

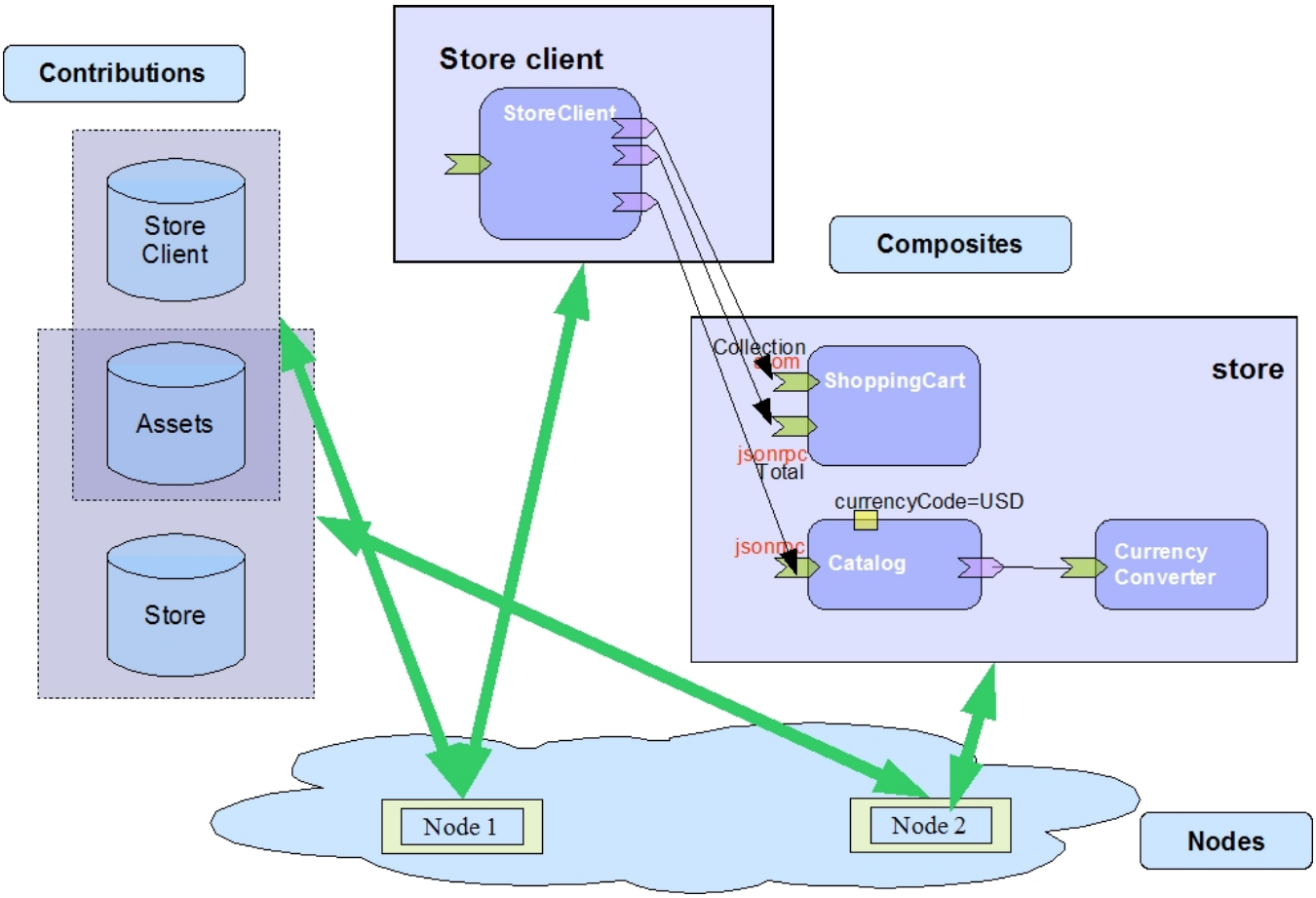
There are two deployable composites:

- StoreClient: It contains the StoreClient component with references to Catalog, Cart and Total
- Store: It contains the ShoppingCart, Catalog and CurrencyConverter components

The application will be deployed to different machines or JVMs:

- StoreClient: Run the store client
- Store: Run the store services

Illustration of an SCA domain



Deployment Steps

Steps	Domain Services	SPIs	Tools	Note

Add Assets, Store, and StoreClient contributions to the domain	Install /Uninstall contributions	Workspace	Static: • Add contribution URLs to the SCA domain manager • Create a workspace.xml to list the contributions • Add contributions to a repository Dynamic: • Discover contributions from the network • Watch a "contributions" folder	We can build different ways to make contributions available to the SCA domain.
Parse the contributions	Contribution Processing	ContributionScanner ArtifactProcessor		
Resolve dependencies across contributions	Import/Export resolution	ContributionDependencyBuilder	• Automatically calculated using the import/export • Assembler can use the admin tool to select a list of contributions for a given composite application	We need to find out a collection of contributions to support a composite application based on the import/export statements
Find/Load/Resolve the composites from the contributions: Store composite StoreClient composite	The assembly builders	ArtifactProcessor CompositeBuilder	The endpoints using SCA addresses or relative URIs should be resolved against the physical base URIs for the bindings based on the node configuration. The resolution can be deferred to runtime over a service registry (which I treat it as a way to form the SCA domain dynamically)	Deployable composites can be designated by the contributions. It's also possible that the assembler to define a deployment composite on the fly.
Configure two nodes to the SCA domain, one to run the StoreClient and the other for Store. The Store one requires WS, ATOM and JSONRPC	implementation.node	NodeImplementation	• Node can be predefined to the SCA domain using the admin tools. • Node can also be provisioned from the cloud based on the requirements by the composite. • Node can also be discovered on the network. There are types of nodes too: • Standalone • Webapp • JEE (JSR-88 based deployment)	The node represents the computing capabilities in the SCA domain.
Assign a deployment composite to a node: StoreClient --> node1 Store --> node2	Run the composite application by a node		• Node can connect to the domain manager to get the composite application. • The composite application can also be pushed to nodes (running in daemon). • The image of the composite application can be saved into a configuration file so that the node can run offline without connecting to the domain manager. • Nodes can run p2p to consistute the SCA domain too.	The deployable image of an SCA composite application to a node is the composite and a list of contributions to support the composite application.
Monitor and control the services running on a node	See what's going on, start /stop the services			