Abstract

This is a proposal for an application that demonstrates the propagation of a WS-CAF Context to a group of services that implement specific interfaces to operate on the
The goals of the demo application will be as follows:

1) To show the implementation of a use case for the WS-Context specification.

2) To demonstrate interoperability between services that implement the WS-Context interfaces as specified in the WS-Context wsdl and schema files.

3) To provide a basis for developing functional tests for WS-Context.

4) To provide the first step in demonstrating the entire suite of WS-CAF specifications (WS-CTX, WS-CF and WS-TXM). Although WS-CF and WS-TXM demos will be based on this demo, the details of such applications are out of scope of this document and will not be discussed.

**Overview**

The demo application will be loosely based on the SCM Application currently implemented by the WS-I Sample Application working group. Since the demo is only focused on WS-Context, it will not be necessary to implement all of the interfaces in an SCM application. The WS-I SCM application, which this demo is modeled after, currently has four service interfaces: Retailer, LoggingFacility, Warehouse, and Manufacturer. Of these four, only the Retailer service is required by the demo. It is important to note that the proposed demo application will only reference the WS-I Sample Application as an acknowledgment of the existing architecture and artifacts that inspired its development. In no way is it intended to be an implementation of the Sample Application.

The demo will illustrate the scenario of a user purchasing items from \( n \) number of related online stores, where \( n \) is the number of WS-CAF implementers providing a demo implementation. These participating stores will propagate the Context in the form of a
shopping cart that the user uses to collect the items he wishes to purchase. At the end of shopping a user can 'check out' using a check out service from any one of the participating stores. A receipt will be printed out at this point showing the items purchased and where (which store) the items were procured.

To implement this scenario each implementer will need to provide implementations of the following services:

i) Retailer Service: The Retailer service will be based on the WS-I Retailer service\(^1\).

   This service provides the ability to get a catalog of items and the quantity on hand, offered by the implementing application, and also to submit an order of items (check out).

ii) WS-CTX Context Service: This service is an entry point for creating new Contexts. When the `begin` operation is called, a new Context will be sent to the designated recipient. Prior to returning the Context, the service will invoke an ALS that will augment the Context with necessary information for the shopping cart activity.

iii) WS-CTX Activity Lifecycle Service (ALS): The WS-CTX Activity Lifecycle service will be used to initiate a new shopping cart activity. This service is used by the WS-Context Service for Context creation and termination.

iv) WS-CTX Manager: This service will manage Contexts that are passed by reference. It will facilitate the storing and retrieval of Context objects and their associated IDs.

In addition to these services, implementers will also need to provide a simple web based UI that provides the following functions:

---
\(^1\)WS-I Retailer WSDL [http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Retailer.wsdl]
- Display participating stores: since the application will be focused on interoperability, each implementer will need to provide a UI to facilitate selection of participating stores. Each participating store is a Retailer endpoint. This list can be formed dynamically (e.g., by querying a UDDI registry), or statically (hard coded). A possible display could show the stores as checkboxes. When a user selects all the check boxes and continues using the demo, he can link only to the sites he selected and they will be able to share his context. The list of stores selected should be displayed as HTTP links throughout the demo. For example, while shopping at store X, a user can click on a link to store Y (that he selected from the list of participating stores) and continue shopping there with the same shopping cart.

- Display the catalog of items returned by each store: The UI must be able to accurately display the items returned by a Retailer Service and allow users to select a quantity of each item to be added to their shopping cart.

- A confirmation and checkout view: After selecting the items for purchase a user should be able to confirm his order and commit a sale at any of the participating stores.

- A shopping cart view: Users should be allowed to add, remove and edit quantity in their shopping carts.

- Display a receipt: After checking out a receipt detailing all the items purchased and their point of procurement should be displayed.

Although there may be other UI screens, the above functionality is the minimum that must be provided by implementers.
Use Cases

The following are 3 common use cases for the demo application:

UC-1) A user purchases a single item from a single store -Company X- using the UI provided by Company Y:
   i. He selects the Company X store from the list of stores.
   ii. He views a list of items offered by the store.
   iii. He selects 1 Qty. of an item and adds it to his cart.
   iv. Because he only selected one store, he must checkout from Company X only.
   v. He confirms his order and checks out.
   vi. A receipt is displayed showing all the items as purchased from Company X.

UC-2) A user purchases a quantity of five of the same item from two stores – Company X and Company Y – using the UI provided by Company X:
   i. He selects both Company X and Company Y from the list of stores.
   ii. He views a list of items offered by Company X.
   iii. He selects 2 Qty. (Company X only offers a maximum of 2) of an item and adds it to his cart.
   iv. He clicks on the Company Y store link.
   v. The list of items from Company Y are displayed.
   vi. He selects 3 Qty. of the same item.
   vii. He decides to confirm and check out from the Company Y store site.
   viii. A receipt is displayed showing 2 items purchased from Company X and 3 items
purchased from Company Y.

UC-3) A user selects a quantity of 5 of a single item from two stores – Company X and Company Y – and a quantity of 5 from a third store – Company X, using Company X's UI:

i. He selects the three stores.

ii. He performs steps ii to iv in UC-2

iii. He clicks on Company X's link.

iv. He selects 5 qty. of a different item from Company X's store site.

v. He decides to checkout from Company X's store site.

vi. A receipt is printed showing all items purchased and their point of procurement.

Collaboration Architecture

1. Adding an item to the cart.

N.B. operation names used in the diagram do not represent actual method names in the application and may not correlate directly to any existing APIs. These sequence diagrams demonstrate synchronous calls only for the purpose of concise illustration. The use of asynchronous or synchronous calls is an implementation detail out of scope of these diagrams.

The sequence diagram above shows the objects and operations involved in adding an
item to the shopping cart:

1) A user adds an item from Store X to his shopping cart.

2) Store X checks the request for the existence of a Context. Essentially this will involve checking for the existence of the Context with the Context Manager (if no header exists, implementations may simply return null without checking with the Context Manager).

3) A return of null here simply means that no shopping cart (Context) exists and thus a new one needs to be created.

4) A new shopping cart is created by sending a begin request to Store X's Context Service.

5) The Context Service creates the Context and propagates the request to the list of registered ALSes which need to be notified for begin and complete actions. Since only one ALS is registered, it is invoked and augments the Context with the items from the user. (This is an implementation details... some implementations may not use an ALS but instead will add this information directly from the Context service).

6) The ALS returns the augmented Context.

7) The Context Service returns the shopping cart in the form of a WS-CTX Context. This Context can now be shared with other services.

8) The item is shown as added to the user's shopping cart.

2. Adding another item from another store into the cart.
The sequence diagram above shows the objects and operations involved in adding additional items from another store to an already existent shopping cart:

1) A user adds an item from store Y into his cart.

2) Store Y checks the header for the existence of a Context and finds the reference to one. The header element also provides the location of a ContextManager where the Context has been stored.

3) The Context is retrieved from the Context Manager.

4) The Context is augmented with the necessary data (items). This step may be performed by a separate service.

5) The ContextManager is updated with the new Context.

6) The ContextManager confirms the update.

7) The new item has been added to the existing cart.

3. Checking out
The sequence diagram above shows the objects and operations involved in a checkout operation:

1) A user completes his sale by checking out from Store Y. (Assume that the user has purchased items from Stores X and Y).

2) Store Y checks the header of the request for a Context reference. The ContextManager is then asked for the Context based on this reference URI.

3) The full Context is returned.

4) Store Y sends the complete operation to the Context Service. Note: this is the same Context Service that began the shopping cart activity and therefore may be an implementation provided by Store X.

5) The Context Service propagates the complete request to registered ALSes. In this case a single ALS is registered and it is responsible for completing the sale and augmenting the Context with the receipt.

6) The augmented Context (with the receipt) is returned.

7) The activity is completed. Future calls to getStatus should return Completed on this Context. The shopping cart is no longer valid for adding items.

8) The checkout operation completes.