Component Definition Document (CDD) for the Publish-Subscribe Client-Server Example Component Assembly

Rev. -

February 1, 2011

Prepared By:
Northrop Grumman Corporation
Electronic Systems
Baltimore, MD

Table of Contents

1 Introduction

1.1 Scope

This document captures the specification and design for the Publish-Subscribe Client-Server Example software component assembly. This example assembly is targeted for deployment on the Scalable Node Architecture (SNA) real-time component framework. As such, it must be compliant with SNA Component Based Architecture (CBA) design guidelines.

This specification defines the component assembly's functional, interface and performance requirements, the context in which it must operate, and any design constraints it must adhere to. It provides criteria for verifying compliance, but it does not state methods for achieving results.

This is intended to be a relatively informal living document, to be included in same CM repository and package as the component source code. This CDD will initially be populated by a system engineer or software architect/lead to define component design constraints & guidelines. Over time, it will transition to enhance the "to be built" specification sections with "as built" design information documenting the final component product.

2 Applicable Documents

2.1 Applicable Government Documents

Document No.	Title

2.2 Other Applicable Documents

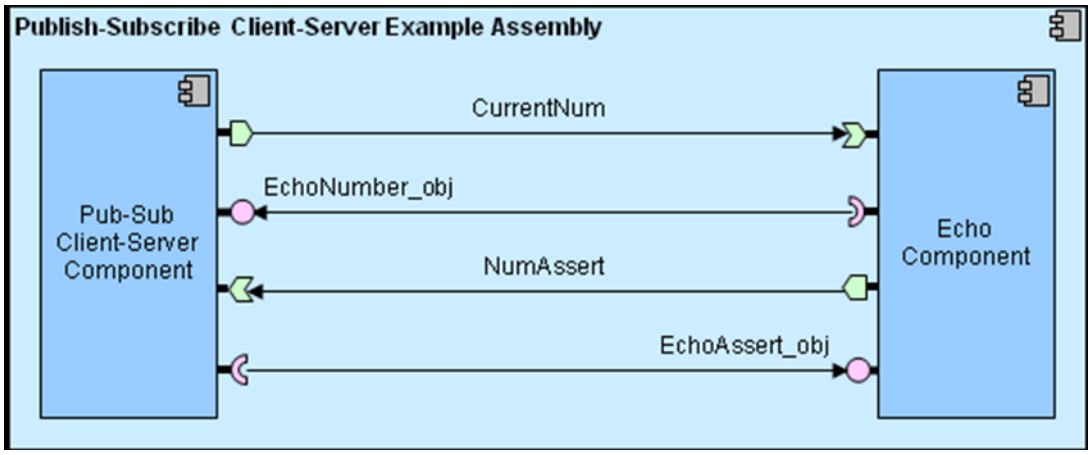
Document No.	Title

3 Component Description

3.1 Overview

The Publish-Subscribe Client-Server Example component assembly is one of the component source examples included in the SNA SDK for reference, testing and experimentation. It illustrates a repetitive sequence of four pub-sub and client-server message exchanges between two monolithic components, each of which implement all four of these basic port types. The assembly containing these two components is shown in Figure 3-1 below.

Figure 3-1 –Pub-Sub Client-Server Example Component Assembly



This example assembly illustrates component executor composition showing code that uses all basic port types and associated event handlers.

3.2 Operational Context

The assembly is completely self-contained, with no external connections. It is designed to operate solely within the constraints of the SNA platform's development environment to allow a new user/developer to step through the SNA component based development (CBD) process of loading a component assembly into the SNA IDE, building/compiling it, and then executing it.

The example is provided with an appropriate set of SNA configuration files and a deployment plan to support its execution within a single-host SNA SDK "localhost" Virtual Machine (VM). Alternative variations on the default supplied design and deployment are possible via experimentation by a software developer.

4 Component Interfaces

The example assembly defines two internal message struct types & topic connections between matching pairs of publisher and subscriber ports, and two internal interface definitions & client-server connections between matching client "uses" and server "provides" ports. The component assembly has no external interfaces. The 8 internal ports on the two child monolithic component designs are described below.

4.1 Service Ports

4.1.1 EchoNumber_obj Service (internal)

This service port is provided by the Pub-Sub Client-Server component. The service interface defines a single method called getCurrentNum() that can be called to return the current sequence number used in each iterative 4-message exchange loop. This service call is used to verify that the sequence number sent in a separate pub-sub message matches.

4.1.2 EchoAssert_obj Service (internal)

This service port is provided by the Echo component. The service interface defines a single method called getNumAssert() that can be called to return a boolean assertion variable indicating success or failure of each messaging loop. The returned assertion value indicates whether the sequence number received in prior, independent pub-sub and client-server exchanges was the same.

4.2 Client Ports

Matching client ports for the two service ports described in the prior section are reflectively available on the opposite monolithic components.

4.3 Publisher Ports

4.3.1 CurrentNum Publisher (internal)

This port is used by the Pub-Sub Client-Server component to publish a sequence number to kick off the messaging loop within the assembly. For each 4-message iterative loop, the sequence number in the CurrentNum message is incremented.

4.3.2 NumAssert Publisher (internal)

This port is used by the Echo component to publish a boolean assertion variable that indicates whether the sequence number received in a prior CurrentNum message matches the sequence number returned from a call to getCurrentNum() on the EchoNumber_obj service.

4.4 Subscriber Ports

Matching subscriber ports for the two publisher ports described in the prior section are reflectively available on the opposite monolithic components.

5 Component Functionality

Reference the component assembly diagram in Figure 3-1 above. The two monolithic components loop through a repetitive sequence of 4 messages transfers, drawn in sequential order from top to bottom in the diagram.

Upon assembly activation after launch, the Pub-Sub Client-Server component starts the repetitive message sequence, which begins with the Pub-Sub Client-Server component publishing a CurrentNum message containing an initial numeric value. The Echo component receives the subscribed message through use of the HistoricalSamples helper class in the event that the Pub-Sub Client-Server component publishes the message before the Echo component is up and running. The Echo component then makes a call to the EchoNumber_obj service provided by the Pub-Sub Client-Server component to retrieve the same sequence number via its client port, and then sets a boolean variable to true (sequence numbers matched) or false (numbers didn't match).

The Echo component then publishes a NumAssert message containing the true/false boolean variable. The Pub-Sub Client-Server component receives the subscribed message, calls the EchoAssert_obj service provided by the Echo component to retrieve the same boolean variable via its client port, and then generates a log message to indicate whether the boolean assertion

values received via both pub-sub and client-server exchanges matched, or didn't match. Finally, after logging the results of the final boolean compare, the Pub-Sub Client-Server component sleeps for 2 seconds, and then begins the sequence all over again by sending a CurrentNum message containing a new numeric sequence number value, incremented by 1.

6 Configurable Parameters

An InitNum.cfg configuration file allows configuration of an "InitNumber" parameter that defines the initial sequence number for the messaging loops, which is set to 0 by default.

Since this component assembly is intended to support run-time experimentation, it is also packaged with a full set of SNA compliant run-time execution configuration and deployment files.

7 Design Constraints

1. The example assembly will implement all 4 basic port types in a looping message exchange.
2. A full set of SNA compliant configuration and deployment files will be provided with this example in order to support run-time execution in a default single-host target deployment.
3. This example will utilize the standard SNA APIs to perform all functions.

8 Component Test

This 2-component assembly must be executable on a single "localhost" development computer, to include the SNA SDK x86-64 VM at a minimum. Users can experiment with the deployment plan to redeploy to an alternative 2-host target environment if desired.

9 Component Dependencies

The Publish-Subscribe Client-Server Example assembly is self contained and has no dependencies on any other application components. Its only dependency is on the SNA SDK's run time execution environment.