Component Definition Document (CDD) for the Signal Processing Data Model (SPDM) Example Component Assembly

Rev. -

February 9, 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 Signal Processing Data Model (SPDM) 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

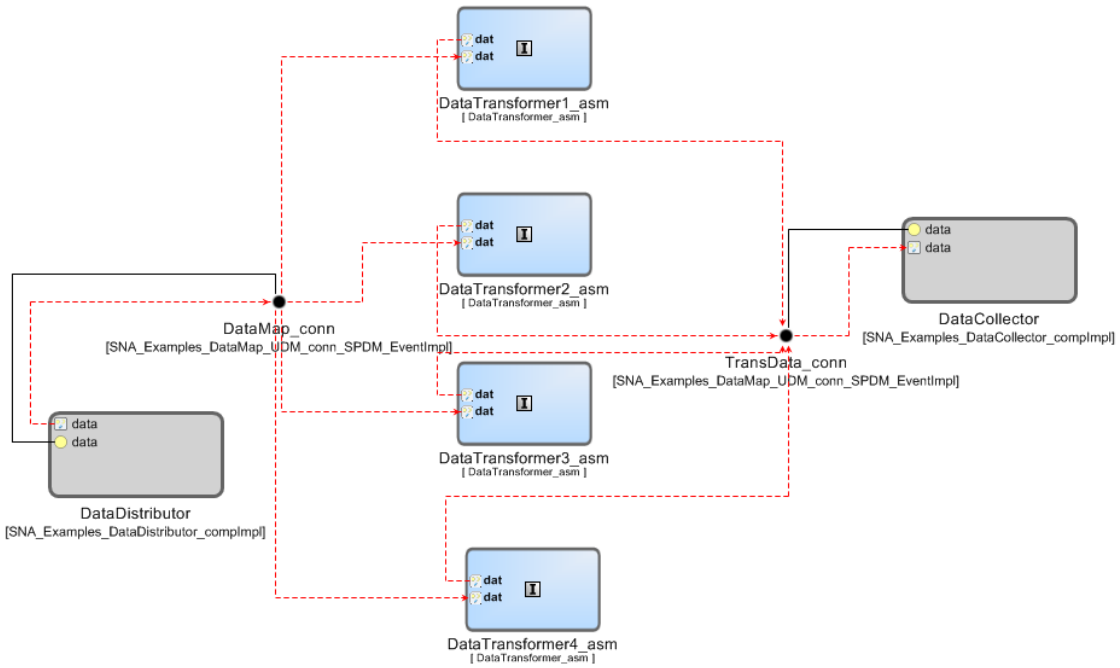
The SPDM Example component assembly is one of the component source examples included in the SNA SDK for reference, testing and experimentation. The SPDM example demonstrates the following:

- Data scatter/gather using the SPDM connector.
  - Writing non-contiguous data using PSAT.
  - Sub-sample aggregation.
- How to use and fill out the SPDM header using the DataViewTranslator helper functions.
- Creating a Signal Processing Orchestration Component (SPOC).

- Creating a Math Kernel Component (MKC).
- Using the VSIPL++ API.
- IDL syntax for Publish/Subscribe Attachment Transfer (PSAT) connector.
- Using the PSAT API from a derived connector.
- How to creating a content filtering via CoSMIC.
- How to update content filter parameters at run-time.
- Using the SNA logging system.
- How to schedule and register a handler for a timer.

The component assembly containing the SPDM example is shown in Figure 3-1 below.

Figure 3-1 – SPDM Example Component Deployment Assembly



The example begins with a component called the **DataDistributor_comp.** This component schedules a timer in **ccm_activate** with a handler which creates a buffer, fills this buffer with test data, and demonstrates how to ***scatter*** 4 different sections of the source buffer to 4 instances of the **DataTransformer_SPOC_comp** component contained in the **DataTransformer_asm** assembly (shown above).

Four instances of a SPOC component will receive their individual “chips” or “pieces” of data scattered from the **DataDistrutor_comp.** In order to receive the assigned PSAT message from the distributor, each SPOC component will update parameters from a content filter. By using content filtering, each SPOC will orchestrate a separate piece of the data published by the **DataDistribtor_comp**.

Once the SPOC components have received their individual chips of data, they will invoke a computation function from a Math Kernel Component (MKC) **DataTransformer_MKC_comp (**also contained in the **DataTransformer_asm).** The MKC component will scale the data by some amount and then return to the calling SPOC component to republish the transformed data chip.

The transformed data chips are then gathered by the **DataCollector_comp** to be reconstructed into the original data map.

3.2 Operational Context

The assembly is completely self-contained, with no external connections. The driving event is a periodic timer internal to the DataDistributor_comp component. 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 SDK 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 a number of internal interfaces and no external interfaces. The internal ports and component design functionality are described below.

4.1 Service Ports

4.1.1 CCM_DDS::ConnectorStatusListener Service (internal)

This service is provided by the DataDistributor_comp, DataTransformer_comp, and DataCollector_comp. It will be invoked by the SPDM connector derived from the PSAT connector derived from the DDS4CCM connector. It is intended to be used by the connector to callback on the application if/when there are new status events (i.e., a sample is rejected, there is incompatible qos, etc.,).

4.1.2 DataTransformerMath_obj Service (internal)

This service is provided by the DataTransformer_MKC_comp and invoked by the DataTransformer_SPOC_comp. This interface provides a few of the methods required to implement the signal processing pattern which separates the signal processing orchestration component (SPOC) from the math kernel component (MKC). This interface contains a "compute()" method, which this example will scale the input data by some amount before returning.

4.2 Client Ports

With the exception of the CCM_DDS::ConnectorStatusListener port, a matching client port for the service described in the prior section is reflectively available on the opposite monolithic component.

4.2.1 DataTransformerMath_obj Client (internal)

The client side of the DataTransformerMath_obj interface will be invoked by a SPOC component as described in section 4.1.2.

4.3 Publisher Ports

4.3.1 DataMap_UDM_conn - PSAT_Write (internal)

The DataMap_UDM_conn is an SPDM connector which is a derived PSAT connector. The DataDistributor_comp and DataTransformer_SPOC_comp use the PSAT_Write port to publish user defined meta-data (UDM) messages to apply the pub/sub attachment transfer pattern. This port enables the DataDistributor_comp to scatter data, and the DataTransformer_SPOC_comp to forward the transformed data to the DataCollector_comp.

4.4 Subscriber Ports

4.4.1 DataMap_UDM_conn - PSAT_Listen (internal)

The PSAT_Listen port given by the DataMap_UDM_conn is used by the DataTransformer_SPOC_comp component to received distributed data, and also by the DataCollector_comp to aggregate transformed pieces of distributed data published by the DataTransformer_SPOC_comp component.

5 Component Functionality

Reference the component assembly diagram in Figure 3-1 above. The critical functionality of each monolithic component designs in this assembly will be described separately.

5.1 DataDistributor_comp Component Functionality

As described in previous sections, the DataDistributor_comp will schedule a repeating timer task to allow the distributor to distribute some pre-defined test data via the PSAT_Write port implemented by the SPDM connector.

5.2 DataTransformer_SPOC_comp Component Functionality

In this assembly, there are four instances of this component type. The “on_one_data()” method will be invoked upon the receipt of a SPDM message published by the DataDistributor_comp.

In addition, this component has the responsibility of configuring a content filter via the deployment plan whose parameters are updated in “ccm_activate()”. The content filter will reject samples which do not match the correct “message_number” attribute configured by the deployment plan. In this way, each component instance will receive only 1 distributed message.

The SPOC component invoke a “compute()” method which is implemented by the MKC component to transform the received data and prepare the output buffer. The SPOC component will then publish the data to be received by the DataCollector_comp.

5.3 DataTransformer_MKC_comp Component Functionality

The DataTransformer_MKC_comp will implement the “compute()” method which will scale input data by some amount and prepare an output buffer with the newly scaled data.

5.4 DataCollector_comp Component Functionality

The DataCollector_comp component will aggregate each transformed data sample back into a matrix which will match the size of the matrix created by the DataDistributor_comp. In addition, this component will assert that the data has been received as expected and print a log4cxx error message if it is incorrect.

6 Configurable Parameters

A **SPDM_Example.cfg** configuration file allows configuration of several parameters:

Name	Component	Default Value	Description
initialDelay	DataDistributor	5 secs	For the send data timer that is used to trigger the publication of a set of PSAT sub-samples, this is the delay before that timer triggers for the first time. *Specifiable in seconds and microseconds*
repeat	DataDistributor	5 secs	For the send data timer, this is the time between all subsequent (2 .. N) timer triggers. *Specifiable in seconds and microseconds*
sendDataLoggingModulo	DataDistributor	1	When publishing data rapidly, logging every set of PSAT sub-samples may be too much output. This parameter allows only every “nth” set published to be logged where “n” is this parameter value.
receiveMsgLoggingModulo	DataCollector	1	This parameter allows only every “nth” set of PSAT sub-samples received to be logged where “n” is this parameter value.
receiveMsgLoggingModulo	DataTransformer *(MKC and SPOC)*	1	This parameter allows only every “nth” sub-sample received to be logged where “n” is this parameter value.

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. This example assembly will implement all PSAT_Write and PSAT_Listen ports using a SPDM connector.
2. The SPDM example will utilize VSIPL++ where possible to access and modify example data.
3. The SPDM example will be driven by a periodic event defined by an internal timer.
4. The SPDM example will illustrate how to apply scatter/gather operations using the capabilities of the SPDM connector.
5. The SPDM example will generate SNA compatible log messages to allow a user to track the progress of the application in order to indicate its overall success or failure.
6. 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.
7. This example will utilize the standard SNA APIs to perform all functions.

8 Component Test

This example 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 multi-host target environment if desired.

9 Component Dependencies

The SPDM 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.