Data Distribution Service

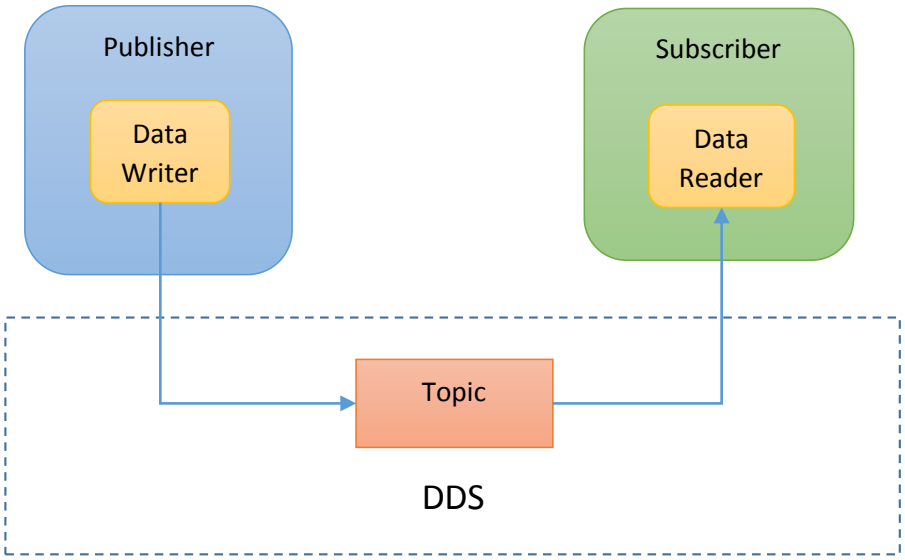
Overview of the Publish/Subscribe Architecture

Introduction

The Data Distribution Service (DDS), created by the Object Management Group (OMG), is a specification for a data-centric, publish/subscribe communication model. The purpose of DDS is to address the challenges associated with developing real-time, distributed applications. At least ten implementations are available, and many organizations use DDS to develop applications or a variety of military and industrial applications. This document provides a high-level description of the publish/subscribe architecture and gives an example of how it can be used.

Publish/Subscribe Architecture

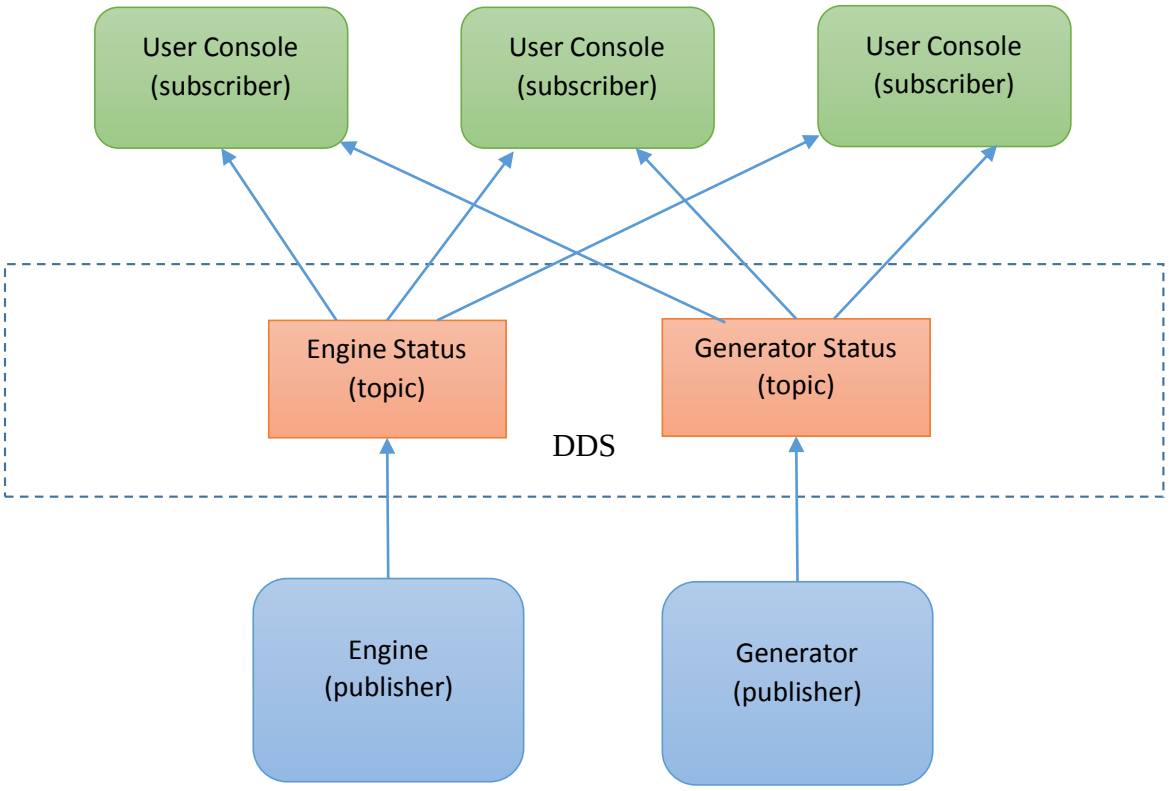
The primary DDS entities are *Publishers*, which product data, and *Subscribers*, which consume data. DDS implements a bus architecture to transmit data from publishers to subscribers. Publishers use one or more *Data Writers* to publish *Topics* (messages), which are typed data structures defined by an Interface Definition Language (IDL). *Subscribers* use one or more *Data Readers* to asynchronously receive published topics. The following diagram illustrates the publish/subscribe architecture.



As an example, consider a ship systems monitoring application that involves multiple sensors attached to multiple pieces of equipment. Several user interface consoles distributed throughout the ship receive data from the sensors and provide an aggregated display of data.

The equipment sensors are publishers and the user consoles are subscribers. The various sensors publish different topics containing data appropriate to their function. The engine sensors publish “Engine Status” topics, which include speed, temperature, vibration, and oil level. The generator publishes “Generator Status” topics, which include voltage and amperage. The user consoles subscribe to “Engine

Status" and "Generator Status" topics and asynchronously receive the data as it is published, as illustrated in the following diagram.



When building a distributed application, software engineers can expend a large amount of time developing the software components that handle communication among the distributed components. DSS reduces this effort by abstracting away platform and network protocols, allowing developers to focus on application-level data and logic.

Another challenge arises when it is necessary to integrate disparate systems and sub-systems. The DDS publish/subscribe architecture decouples data producers from data consumers and facilitates developing heterogeneous applications comprised of components from different vendors using different platforms and programming languages.

Summary

DDS provides a robust, data-centric publish/subscribe data model that can reduce the effort in developing real-time distributed systems across many application domains. For in-depth information about the DDS, who is using it, and vendors providing DDS implementations, visit the OMG website at http://www.omg.org/.