Sidecar Network Adapters: Open Architecture for ISR Data Sources

Craig McNally, Chuck Yee, Larry Barbieri
{cmcnally, yee, lbarbieri} @ll.mit.edu
MIT Lincoln Laboratory, Lexington, MA 02420

Introduction

With an ever increasing number of available ISR data feeds and sensors, consumers of these resources fight an uphill battle of integration. In order to make use of these resources they must develop software that understands numerous data formats, protocols, and interfaces. Such integration can be unnecessarily complicated and time consuming. The ISR Sidecar Network Adapter (SNA), being developed by MIT Lincoln Laboratory, is one approach to mitigating this problem.

SNAs adapt to the data source and expose their data to a service bus through a set of common, net-centric interfaces. This means data consumers need only integrate with a single set of interfaces and will be compatible with many feeds and sensors. Since the SNAs are designed to be part of a Service Oriented Architecture (SOA), they can utilize the service bus' core and domain Services to register their services and feeds. Consumers can then discover data sources as they become available and can begin utilizing them immediately. This is a vast improvement over having to develop software for each data source.

The high level design in Figure 1 shows the two main components of the SNA, the Network Adapter (NA) and the Sensor Adapter (SA). They each adapt to the particular protocols and formats of the networks/systems they're connected to. These protocols and formats are often different, meaning the SA and NA may need to understand different sets of technologies. However, there is always some overlap to allow communication between the two components.

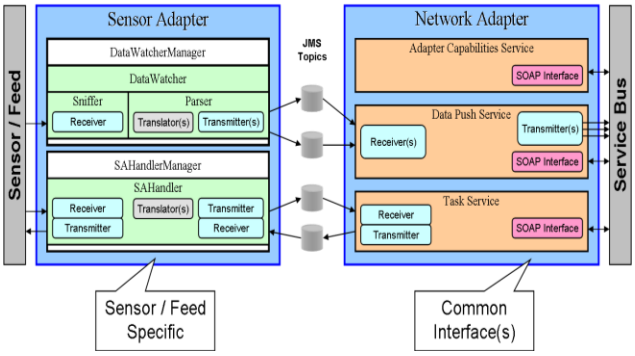

Figure 1: High level Sidecar Network Adapter design

Network Adapter

The Network Adapter provides a common interface to the service bus and exposes data via the service bus' supported protocols. Open standards such as Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) and eXtensible Markup Language (XML) are used to accomplish this.

The exposed Network Adapter's interfaces are made up of SOAP-based web services. The most commonly used services are the Adapter Capabilities Service, and the Data Push Service. Data consumers use these services to query the SNA for a list of available data feeds and subscribe to those feeds. For adaptations that require two way interactions, such as a database or taskable sensor, the Task Service, Search Service, Data Request Service and the Status Service are available.

Sensor Adapter

The Sensor Adapter interfaces with the data source via the source's native protocols and is usually responsible for data format translation. The types of translations can vary greatly, and depend on the data source. In some cases you might be parsing metadata from images or a video feed. In other cases you may be translating between binary and XML messages formats.

Although SNAs can produce feeds of any data format, XML is the most common. XML has been widely accepted as the standard for exchanging data among disparate systems. For example, MITRE's Cursor on Target (CoT) is an XML schema that has been widely adopted within the DoD [1]. Many of the adapters created to date use CoT as an output format. This allows applications that already understand CoT to immediately consume the messages produced by the adapter.

SA / NA Communication

By splitting the SA and NA components, a level of flexibility is attained that allows each of the components to reside where it makes the most sense. They could even be running on different computers, using different operating systems, in different geographic locations. In one experiment, an SA connected to a simulated Net-Centric Collaborative Targeting (NCCT) feed was running in Texas and communicating to an NA running at MIT Lincoln Laboratory in Massachusetts.

Although the SA and NA can be connected in many ways, a publish/subscribe protocol such as Java Messaging Service (JMS) is typically used for its reliability and its asynchronous design. In fact, this is how the SA and NA in the previously mentioned experiment communicated with each other. However, other means of connecting the SA and NA can and have been used. For example, adapters that need to handle streaming data such as video might use a much faster, but less reliable protocol such as the User Datagram Protocol/Internet Protocol (UDP/IP). Adapters that need to handle image files might use a shared file system or the File Transfer Protocol (FTP) rather than a messaging protocol.

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Architecture

SNAs are designed to be part of a larger Service Oriented Architecture (SOA). By being part of a SOA, SNAs can be used by, and can use core and domain Services that allow brokering and tasking of resources, data archival, federated search, and many more capabilities. In the past, SNAs have used these types of services to secure their web service interfaces, restricting access to the data being provided. They have also registered their data feeds in searchable registries that allowed consumers to discover and consume them. Since all SNAs implement common interfaces, consumers can immediately begin using these discovered services and feeds as they become available. This is a big advancement from the lengthy integration process of the past.

Software Development Kit

In order to facilitate the development of SNAs, a Software Development Kit (SDK) has been created. The SNA SDK is a Java based development aid comprised of a Framework and Toolbox. The Framework is a collection of abstract classes and interfaces that provide the basic functionality of the SNA. The Toolbox is a collection of concrete classes that implement and build upon the Framework's interfaces and base classes. Through modularity, the SDK is extensible, and promotes code reuse, allowing developers to create SNAs while writing little to no code. Instead, the bulk of SNA development consists of writing a set of configuration files that define which components of the Framework and Toolbox to use. Figure 2 illustrates this development process.

The ability to change an SNA's internal design through configuration files becomes extremely helpful when developing an SNA that will be deployed in a classified environment. It allows the developer to make changes and corrections to the translation/transformation of data without having to leave the classified environment to make code changes. It also allows the configuration to be classified, while the code of the SNA can stay unclassified and easily reusable.

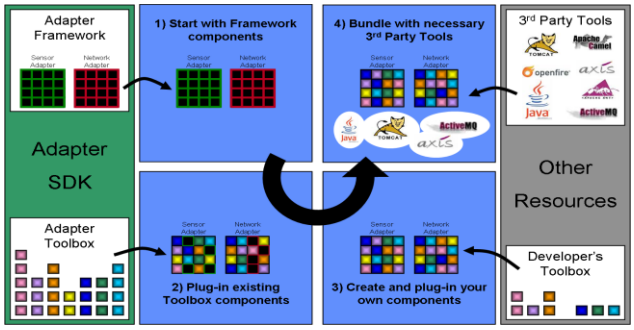

Figure 2 - The SNA Development Process

Examples

The Real-time Enhanced Situation Awareness (RESA) program at MIT Lincoln Laboratory utilizes several SNAs and is a good example of how they can be used to cut down on the amount of integration needed to consume data feeds. RESA's goal is to create a "Real-Time" SOA for rapid operational exposure of ISR data in Tactical Systems [2]. One of the components in this SOA, called an aggregator, consumes data from many sources and presents a control panel to the user allowing them to select a set of data sources to display. Rather than having the aggregator understand all of the different data formats and protocols of the many sources, SNAs were created. This allows the aggregator to consume all of the feeds while only having to speak to a single set of interfaces, and understand a single data format. Other components on the service bus, such as the SIGINT Alert Service, Air Track Correlator, and SIGINT Correlator also reap the benefits of using SNAs. These components consume some of the same data feeds as the aggregator, and provide additional, value-added feeds. Without the SNAs, they too would have needed to integrate with each data source, many of which use different protocols and data formats.

The SNA SDK was an essential tool for expediting the implementation of these adapters. Through code reuse, the SNA developers were able to implement new SNAs while writing little code. One example of this is the Near Real Time Intelligence (NRTI) adapter. NRTI is a data feed that encompasses many different types of information. The feed's content ranges from SIGINT to maritime, air, and ground position data. RESA consumes multiple NRTI feeds, each containing multiple types of messages with their own format. To handle this, each adapter has its own set of configuration files that specifies how to parse and map a given message type to the CoT format. However, the code used for each of the NRTI adapter instantiations is identical. In fact, the code used by the NRTI adapters to parse and map is generic enough that it is used by RESA in two non-NRTI adapters as well. One of them interfaces an Image Product Library (IPL), a DoD image & metadata repository. The second interfaces a COTS Automatic Dependent Surveillance-Broadcast (ADS-B) receiver.

Summary

Through the use of SNAs, the amount of integration with ISR data sources is greatly reduced. Data consumers can integrate with a common set of services and will be compatible with any data source interfaced with an SNA. Since SNAs are part of a SOA, core and domain services can be utilized for service registration and discovery. This enables data consumers to find, access and use data feeds/sources as they become available with minimal integration. An SDK has been developed that provides the building blocks necessary to implement new SNAs, alleviating the need to repeatedly write copious amounts of software. The SDK is easily extensible, and flexible enough to create SNAs capable of interfacing many types of data sources including feeds, sensors, and databases.

References

[1] E. Harding, L. Obrst, A. Rosenthal, "Creating Standards for Multiway Data Sharing", *The Edge: MITRE's Advanced Technology Newsletter*. Summer 2004 vol.8 No.1 pp.16-17

[2] H.E.M Viggh, "Real-time Enhanced Situational Awareness (RESA)", ISR Systems and Technology Workshop, 2009, Lexington, MA, MIT Lincoln Laboratory

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.