MQSeries Adapters

Introduction

This whitepaper describes the MQSeries Adapter Offering (MQAO) and how it fits into the business integration solution. It begins by highlighting some of the problems that impede effective integration and then describes how the MQSeries family of products addresses these, with particular emphasis on connecting applications into the integration infrastructure.

You could compare the MQSeries family to a national network of highways, which are integrated and connected, but currently have a limited number of on- and off-ramps. Today we have a number of bridges and links, including CICS, IMS, Lotus Notes and SAP R/3. To be a useful infrastructure the national network of highways needs on- and off-ramps to connect major towns and cities. In the same way an enterprise needs adapters to connect applications into the integration infrastructure.

The MQSeries Adapter Offering provides a tool to make building adapters more productive with patterns for common packaged applications and the ability to easily build additional adapters when required.

The MQSeries Adapter Offering provides solutions for customers and an exciting business opportunity for our Business Partners (Systems Integrator's, ISV's and education providers).

Terminology

Most vendors with offerings in the integration market have products that provide connectivity between the integration infrastructure and the applications or environments that are to be integrated. Consequently, there are probably as many terms used to describe the offerings as there are products. This section defines the terminology that will be used consistently throughout this paper, to improve clarification of the terms used.

Adapter - A piece of software that moves data between a message on a queue and an application or environment. Adapters handle data inbound-to and outbound-from the application or environment.
(Also known as bridges, links and connectors).

Format - A definition of the structure and encoding of the data within a message. Formats are significant because they describe the messages produced and consumed by adapters. Formats are utilised by the transformation engines within message brokers to parse and construct messages.
(Also known as structures, templates or schemas).

Library - A set of message formats.

The problem

Business integration is all about enabling information to flow the way that the organisation actually operates. This often involves building new connections between IT systems that were previously isolated (often called “islands of information”) so that the information within them is made available to any and all systems that need it.

The problems that make integration difficult typically fall into one of the following categories:

A constant state of flux

In today’s commercial world change is constant. The term “web year” has come to signify the rapid pace of change that new technologies are bringing to our lives. An integration solution that cannot respond to change will cease to be a benefit and becomes instead an inhibitor to adapting to new requirements.

An integration solution should protect the organisation from the negative effects of change by isolating the constituent parts of a connected system to contain change and minimise the effort required to accommodate it.

Myriad data structures and meanings

Each system that is to be connected typically uses its own proprietary data syntax (ie structures) and semantics (ie meanings). For example, the SAP R/3 system supports a data structure called IDOCs for passing information in and out of R/3. The IDOC structure defines the way data will look when the R/3 system provides some information and the way the data must look for the R/3 system to understand it. The semantics or meaning of data is specific to each system too. For example, a quantity to one system may represent the number of widgets to be manufactured, while another may understand quantity to mean the number of pallets of boxes of widgets to be shipped to a distributor. Clearly, quantity is the same *type* of information, but the meaning can be significantly different to each system. Imagine the consternation of the retailer who orders a replenishment of 1000 widgets expecting a number of 10-widget boxes to arrive but instead receives 1000 pallets!

Platform and application dependencies

Related to this variety of data formats and meanings is the wide range of applications and platforms that are typically involved in an organisation’s IT infrastructure. The problem comes about because there are several standards for the way that data is represented - string data may be ASCII or EBCDIC and then in language-specific code pages, for example Latin-1, Kanjii or Katakana. Binary data may be big-endian, little-endian or packed-decimal, etc. Usually the representation is related to the platform or operating system.

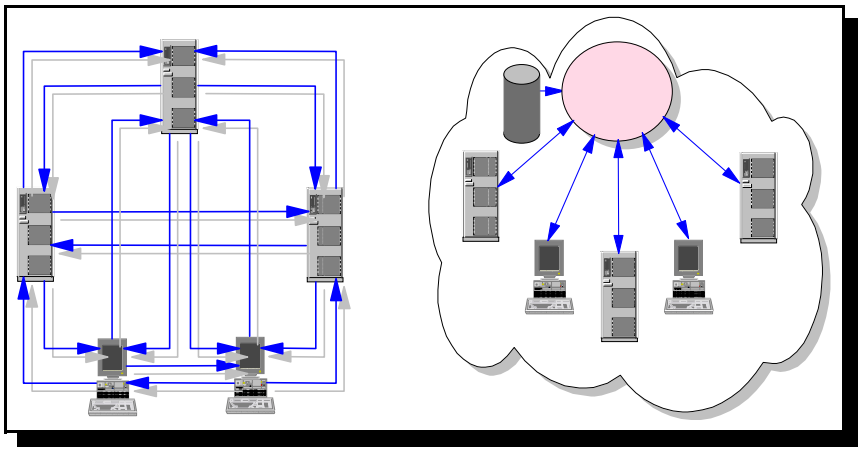
Similarly, even the version of an application may affect the data syntax, semantics and representation and so even connecting two applications of the same type together may be problematic.

The solution

Standardised data formats

When it comes to architecting large and complex integration infrastructures analysts and vendors have been promoting the hub-and-spoke design for a number of years. By routing all application messages through a centralised broker you avoid the proliferation of point-to-point connections between pairs of systems that become difficult to manage and expensive to maintain as their number and complexity grows.

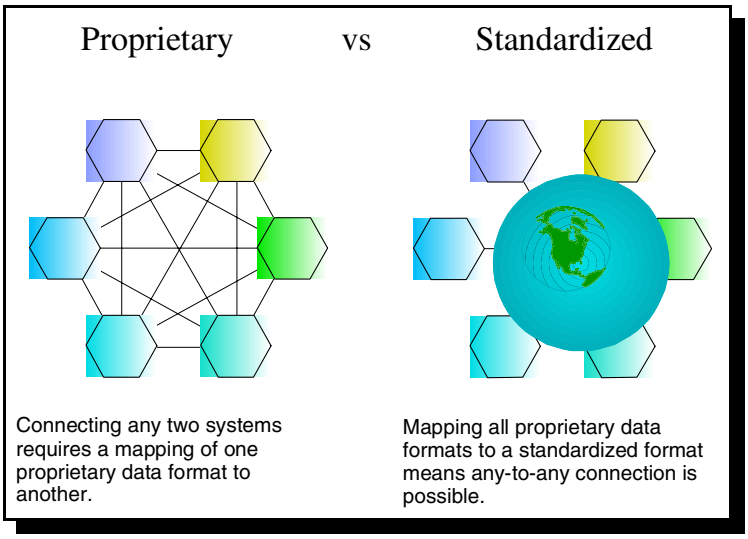
Spaghetti vs hub-and-spoke architecture



In the same way, adopting a standardised structure and representation for the application data exchanged between applications makes any-to-any connection much simpler because it eliminates the need for specific format-to-format transformations.

If the data emitted by one application is structured in a standard way then any application that can understand the standard representation can process the data. This is obviously an area where adapters can contribute to the integration solution by converting between proprietary and standard data structure and representation.

Proprietary vs standardised data formats



The question we are then faced with is how to define a standardised structure for message data? One source of a standard for data formats is the Open Applications Group, Inc. (OAG) - an industry consortium formed by many of the leading vendors of integration technology and services as well as customers actively engaged in integration projects. See Appendix A for the list of OAG members as of October 1999. The OAG has defined a wide range of Business Object Documents (BODs) which represent standard business processes that occur within an organisation. For example, Add Purchase Order, Show Product Availability, Add Sales Order, etc. BODs are defined in eXtensible Markup Language (XML), which is based on Unicode and therefore avoids the problems of platform-dependent data representations.

The MQSeries Adapter Offering

The MQSeries Adapter Offering consists of two product components:

- MQSeries Adapter Builder
- MQSeries Adapter Kernel

And two support pac components:

- MQSeries Adapter Sets
- MQSeries Integrator Library

The MQSeries Adapter Builder

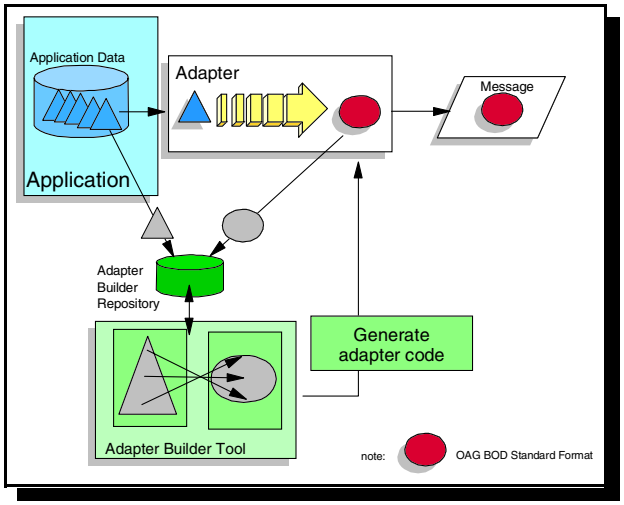
The adapter builder is a tool that allows a developer to import an application's interface into a repository by processing C header files containing function prototypes and structure definitions. It also allows a message format definition to be imported into the repository by processing XML Data Type Documents (DTDs). The OAG provide DTDs for all of their BODs on their website[1]. The repository used by the tool is based on the same technology used by MQSeries Integrator v2, and therefore the message format definitions can be shared between the adapters and the message broker.

[1] See the OAG web site at http://www.openapplications.org for more information.

A developer with knowledge of the application can visually map the data from the message structure into the data structures expected by the application's interfaces and vice versa, map the data from the application's interface into the desired message structure. The mapping definitions include basic data manipulation such as padding, truncation, alignment, etc. If more complex behaviour is required then custom code can be written and stored in the repository. The tool then allows the developer to sequence the calls to the application which are necessary to process all of the data within a message.

Once the logic required in the adapter to transform data and interact with the application has been defined, the tool generates C code to implement the adapter, together with make files to build the code on a number of different platforms. Any custom written code is included in the generated code and the generated code includes comments and tracing calls to aid debugging.

Adapter builder tool



The MQSeries Adapter Kernel

The MQSeries adapter kernel provides standard runtime functionality that's required by any MQSeries adapter created by the adapter builder tool, including interaction with MQSeries queuing, configuration, and handling log and debug information. The adapter kernel is the same for all adapters and the same code is therefore built for each of the supported platforms.

MQSeries Adapter Sets

Initially, there will be sets of standard adapters for three ERP applications: SAP R/3, Baan IVb and JD Edwards OneWorld. These adapters, the BODs that they handle and the functional areas in which they fall are shown in the following table. (The terms "Send" and "Receive" mean, respectively, that the adapter allows the application to send transactions to or receive transactions from another application.)

		Manufacturing Execution Systems (MES) Adapter Sets		Customer Order Servicing (COS) Adapter Sets
Business function	OAG BOD	SAP R/3 3.1h	Baan IVb	JDE OneWorld
Update item master	Sync Item	Send	Send	Send
Update inventory	Sync Inventory		Send	
Update inventory	Update Inventory	Send		
Create new sales order	Add Salesorder			Receive
Update sales order	Sync Salesorder	Send	Send	Send
Update production order	Sync Prodorder			
Update routing	Sync Routing			
Update bill of materials.	Sync BOM			
Update customer	Sync Customer	Receive	Send	
Receive production order	Receive Prodorder	Receive		
Issue material against order	Confirm Issue	Receive		
Report operation complete	Update WIPConfirm			
Change sales order	Change Salesorder	Receive	Receive	
Show item availability	Show Prodavail			Send

These adapter sets are not complementary; e.g. the “Sends” and “Receives” of two sets do not correspond. The adapters in the MES sets, for instance, are intended to support connections to MES applications. Matching adapters are available from some MES vendors, or can be developed with the Adapter Builder.

The adapter sets will be available to download from the web as 'patterns' to be loaded into the adapter builder tool and customised to meet each customer's specific requirements. The patterns will define predetermined mapping between the fields in the OAG BOD messages and the application's data which can be augmented with extra non-standard data defined in the customer's installation of the application.

MQSeries Integrator Libraries
For those customers who use the NEON components of MQSeries Integrator, IBM will provide format libraries for the BOD messages handled by the adapter sets. These BOD definitions will be distributed in the form of import files; each of these will contain all the BOD definitions used by one of the functional areas (e.g. MES) corresponding to an adapter set.

The format libraries will also provide useful definitions for customers who want to define their own XML messages in the NEON formatter.

Conclusion

Standardising the structure of data passed between applications brings benefits in several respects: It makes it possible to connect two applications together using adapters to interface to each application and MQSeries messaging to provide the reliable asynchronous connection between the applications and their platforms.

This solution is well suited to solving integration problems where there are a few applications to integrate and the data distribution or message routing requirements are straightforward. Because of its simplicity, this solution has predictable costs for implementation and management.

For more sophisticated integration requirements, using standard messages simplifies the configuration of the broker. A broker requires several types of information to process messages: Firstly, it needs information (known as meta-data) about the structure of messages to enable it to decompose them into their constituent fields. By standardising all messages passing through the broker you no longer need meta-data for every application-specific data structure, only the standard messages.

Secondly, the broker needs information about how to transform an input message into a new format to route on to a receiving application. Using standard message formats means that all applications expect to receive messages in the same format and therefore there are no transformations required in the broker.

Finally, the broker needs information about how messages should be processed. With MQSeries Integrator this information is defined as message flows and rules specific to each input message type. Therefore, by reducing the number of input message types you reduce the number of message processing definitions in the broker.

When you consider the long-term costs of maintaining an integration infrastructure, using standard messages is advantageous because it provides an extra layer of isolation that prevents changes to one application affecting the rest of the system. For example, if an application is

upgraded or replaced by a similar product from a different vendor (providing the information produced and consumed by that application is not reduced) then all of the change can be accommodated in a new adapter and not exposed to any other application or even the broker.

The future is never certain, but there are a number of areas where we see MQSeries Adapters evolving and maturing to help you manage change in your business:

- Closer integration within the MQSeries family:
  - Consistent use of tools for configuration and management.
  - Use of the new MQSeries interfaces (AMI and CMI).
- More powerful models for defining adapter behaviour.
- Wider platform coverage.
- Education and services packages provided by IBM and partners around the tool, adapters, quick-start offerings, etc.
- Application vendors offering pre-built adapters following the MQSeries Adapter approach.

Appendix A

Members of the Open Applications Group, inc.

Application Vendors
Great Plains
I2 Technologies
J.D. Edwards & Company
Marcam Solutions, Inc.
Oracle Corporation
PeopleSoft, Inc.
QAD, Inc.
SAP AG
Trilogy

Customers
AT&T
Ford Motor Company
Lockheed-Martin
Lucent Technologies
NEC Corporation

Technology Vendors
Compaq
Computer Associates
CrossWorlds
Extricity Software
IBM
Microsoft
TSI
Vitria
WebMethods

Systems Integrators
PricewaterhouseCoopers