

# Integrating IBM MQSeries Message Queuing with Transactional and Bulk Data Movement

**DataMirror**<sup>®</sup>  
The experience of now.<sup>™</sup>

## Contents

Contents .....	2
Introduction .....	2
Overview of IBM MQSeries .....	4
MQSeries Integrator.....	5
Constellar Hub MQSeries Connectivity .....	5
Non-invasive architecture .....	5
Capabilities of Constellar Hub MQSeries Connectivity.....	6
Developing MQSeries Interfaces with Constellar Hub.....	6
Why use Constellar Hub and MQSeries together? .....	7
Message Enrichment.....	7
Step-up/step-down Transformation.....	7
Message Consolidation/Break-up .....	8
Heterogeneous Join and Replicate .....	8
Message Pass-through .....	8
Purchase Order reconciliation .....	8
Address cleansing .....	8
Integration of operational systems with customer management system .....	9
Summary .....	9
About Constellar® Hub.....	10

## Introduction

Recent trends in the enterprise software industry - such as Enterprise Resource Planning, Customer Relationship Management and Data Warehouses - have made many organizations acutely aware of the benefits (and costs) of formalizing how applications share information and data with each other.

Beginning with specific business areas, many organizations have begun to construct an Enterprise Application Integration (EAI) infrastructure in an effort to ensure a standard, repeatable approach to application integration. The hope is that it will take a lot less time to implement new or modified applications with such an architecture.

In general terms, there are two main ways that data moves between applications - via individual *transactions*, or via *data sets*.

TP Monitors, Object Request Brokers (ORBs) and Message-oriented middleware are examples of transaction-oriented technologies. Database gateways, Extract-Transform-Transport and FTP tools are examples of set-oriented technologies.

Transaction-oriented integration is fast becoming the preferred method for integrating newly developed or bought applications. However, a major consideration for any IS department is that transaction-oriented technologies must be retrofitted into any and all applications that are not natively designed to support them.

Set-oriented integration arguably accounts for over 90% of all inter-application data sharing. The fact is, for many applications, such as legacy and stove-piped client-server, a non-invasive approach is demanded. There may be no expertise in-house to modify the application or it is simply not permitted due to the critical nature of the application.

When it comes to choosing an integration approach, many times the decision is a cultural one. When looking at the available resources for integrating applications the old adage "If all you have is a hammer, then every problem looks like a nail" holds true.

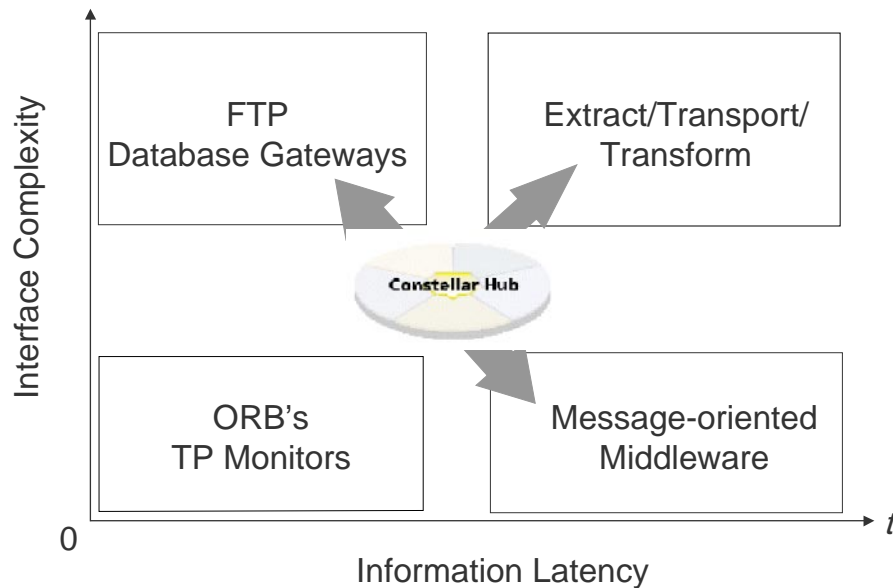
If the decision is left to the Data Processing Center, who are still immersed in keeping the mainframe running and running batch jobs for users, the chances are that they will opt for a set-oriented approach.

If it's up to department manager who has client-server and Windows expertise, it may well be a decision in favor of a transaction-oriented approach.

Whatever technologies are used to solve these integration problems, there is inevitably a bias towards one approach or another.

The architecture often breaks down whenever the integration problems is just too tricky for chosen approach. For example, if the preference is for message-oriented middleware, what happens when the problem calls for huge data volumes at regular intervals? The answer is usually hand-coding around the technology to make up for its limitations.

For true *Enterprise* Application Integration, a mixture of set- and transaction-oriented technologies is required. Unfortunately, these technologies are designed for different problems, and an "EAI gap" exists that must be filled by hand. Until now.



**Figure 1 - Constellar Hub 3.0 bridges "EAI gap"**

With the release of Constellar Hub 3.0 comes a new connectivity option for IBM MQSeries enabling Constellar to offer the first product line to offer both set- and transaction-oriented EAI.

Constellar Hub has traditionally only supported application-to-application interfaces through connectivity to databases and flat files. This new MQSeries connectivity option enables Constellar Hub customers to integrate with tMQSeries-enabled applications with those that are not.

The purpose of this white paper is to describe the combined functionality that Constellar Hub and IBM MQSeries brings to Enterprise Application Integration projects and how this combination addresses the rapidly expanding scope of today's EAI needs.h

## Overview of IBM MQSeries

IBM MQSeries is designed to send *messages* (discrete packets of information) around a network of loosely coupled applications. Each message consists of a header (specifying the originator and recipient and message structure) and payload (the message data).

These messages are placed (or *queued*) onto a 'message queue' where they can then be read (or *de-queued*) by the intended recipient application.

For any given message, the recipient may not be able to accept the message, as it becomes available. IBM MQSeries implements a technique known as *store and forward* that is the basis for the term 'queue' that ensures that the message is kept in the queue until the recipient has de-queued the message and acknowledged its receipt.

Fundamentally, IBM MQSeries is an efficient and straightforward transport mechanism for sharing data between applications. Within the context of an overall EAI approach, MQSeries solves the transport problem.

However, in order for two applications to share information via MQSeries the first application must be modified to enable it to create messages that it then puts on the queue. The second application must also be modified to de-queue the message and then interpret its contents.

As the number of applications to be linked by MQSeries increases, additional complicating factors are introduced. One of the most significant is dealing with the differences in how data is structured between source and target systems. There is usually no consistency in data format where it is stored from system to system. Consequently, special care must be taken when messages are created that their contents can be understood and interpreted by the applications that receive them.

And one message may need to be delivered to multiple applications that each use different parts of the content.

Consider the following examples of how two different systems treat the same set of business information.

<u>Information</u>	<u>Application 1</u>	<u>Application 2</u>
Customer Name	"John W Smith" stored in one field	"John", "W" and "Smith" stored in three separate fields
Status codes	Values 1, 2, 3 and 4 used	Values "A", "E", "S" and "T" used
Customer record	Stores name, two address lines, state, zip code and customer number	Stores first name, middle, initial, last name, single address line, zip code, country, telephone number, fax number and company name

The *Customer Name* information must be split into three distinct data items in order for it to be compatible with the target application's data structure. And the *Status Codes* must be mapped to the equivalent for the target system.

Apart from basic data type conversions or calls to external programs, MQSeries is not designed to reformat the data between the two applications. So either the source or target, or both, applications must do this work.

The *Customer record* is a more complicated matter. The second application requires data that the first application simply does not have access to. So where does it come from?

It could be that this data has to be delivered to Application 2 from a third application that *does* hold this data. But this complicates the extra program logic that Application 2 has to

have when reading messages. Application 2 must hold messages from one application until a message from the other application is available to provide the missing data.

The reverse problem also exists, what if the data from Application 1 is required at multiple other applications? Application 1 could generate a large message containing a super-set of all data needed by the receiving applications. But this would require Application 1 to be continually modified when additional applications join the queue. Or, Application 1 could generate multiple messages - one for each receiving application. But again, the onus is on Application 1 to change each time a new application comes on-line.

## ***MQSeries Integrator***

In early 1999, IBM announced *MQSeries Integrator*, a message broker that complements MQSeries. This provides message re-formatting and intelligent message routing, which directly addresses the *Customer Name* and *Status codes* problem described above.

Still, MQSeries Integrator does not provide a solution for the more complex data transformation that is required for the *Customer Record* scenario. It is not designed to merge/join data from many sources. Data enrichment is limited to the setting of default values for new fields in the target record. Data can be repaired, but usually requires calls to external programs. Nor can *MQSeries Integrator* aggregate/summarize multiple messages into one, or break down one message into multiple others.

Why is this important? This functionality is always required when application integration extends beyond a handful of applications. It simply becomes too expensive to modify the internals of an application whenever a new application has to be integrated or business processes change.

## **Constellar Hub MQSeries Connectivity**

To augment the MQSeries architecture, Constellar Hub has the *MQSeries Connectivity* option.

Constellar Hub is a tool for complex data transformation and interface management that provides a complete environment for developing and deploying interfaces between heterogeneous applications.

Constellar Hub integrates applications that are MQSeries-enabled with those that are not. It reads and writes data to and from flat files, databases and MQSeries irrespective of physical location, proprietary format, data volume or frequency of data movement.

Constellar Hub provides the facility to convert, clean, enhance and audit the transfer of data as well as manage and schedule the flow of data.

The MQSeries Connectivity option enables Constellar Hub to read MQSeries messages and transform or repair the message contents in a way that MQSeries is not designed to do. It can also create MQSeries messages once the data has been processed.

Above all, Constellar Hub represents a common set of tools and a standard development language for integrating heterogeneous applications whether they communicate via messages, database gateways or via flat files.

## ***Non-intrusive architecture***

Constellar Hub has a non-invasive approach that does not require the modification of any application that you are integrating with. This ensures that all business rules, data

transformation and repair code is managed under a central repository controlled by the Constellar Hub.

This does not only reduce development times; it dramatically reduces maintenance costs.

### ***Capabilities of Constellar Hub MQSeries Connectivity***

The combination of Constellar Hub with MQSeries is a powerful one. There is an array of extra capabilities that can be realized:

- "Open" a message and process its contents whilst in transit between applications
- Pack multiple source messages into one, larger, output message. The reverse is also true
- Convert messages into various flat file formats or load into relational databases. The reverse is also true
- Audit all data that passes through Constellar Hub. This includes tracking the number of records (All, Successful, Failed) and all errors that are encountered as well as the data that was at fault
- From a single source of data, generate multiple output messages of different formats and content.
- Keep track of primary keys as they are changed between source and target systems
- Validate and repair data in transit based on user-defined business rules
- Reject data that fails quality control. Rejected data can be stored in Constellar Hub for later manual repair, or it can be delivered to a 'Rejected data' message queue, file or database
- Track and analyze job statistics captured by Constellar Hub

This combination of batch/bulk data processing with message queuing technology will enable many organizations to consolidate their application integration efforts into fewer tools than before.

### ***Developing MQSeries Interfaces with Constellar Hub***

Constellar Hub has a development environment, called the *Metadata Manager*, which is used for building interfaces between heterogeneous applications.

Metadata Manager holds metadata describing every database, file and message queue across the company. It creates a normalized metadata definition for these disparate data sources and targets so that mapping, comparisons and heterogeneous joins can be performed.

For any application, Metadata Manager captures the following definitions:

Database - a description of the source or target application

Entity - a logical representation of a data "container" - the structure of a message, a file, or a relational database table.

Attribute - a logical representation of a data item within the Entity. This could be a field in a file or message or column in a table. It also can describe complex hierarchical structures such as those found in COBOL Copybooks or some messages

Domain - a list of valid values or ranges for an associated attribute

Transaction - a set of mappings and data transformation rules between source and target databases, entities and attributes.

Interface - a logical set of transaction that define data movement and transformation between sources and targets.

Using these definitions, developers can create complex data mappings and sophisticated business rules that will enable data to be shared between almost any application in the organization.

The most critical part of the interface is the data transformation logic. This logic ensures that data read from one application is able to be loaded into another.

Data transformation rules are specified using a simple, yet powerful language called the *Transformation Definition Language* (TDL). TDL is a function-rich language with over 100 standard functions. Developers can also define their own functions - in TDL, Oracle PL/SQL or even C - that can be reused across many interfaces.

TDL rules can change, repair or reject data based on a variety of user-defined constraints. As part of a rule, TDL can query data from external systems, for example, to get an up-to-the-minute stock price, and use this to augment the data read from a source application.

## **Why use Constellar Hub and MQSeries together?**

It is a fact that IBM MQSeries is the leading message-oriented middleware product on the market.

It is a fact that nearly 90% of automated information sharing between applications is done using batch/bulk data movement techniques.

It is also a fact that to enable an existing application to use MQSeries requires that the application be retrofitted to read and write messages.

As organizations strive to link and automate more business processes the 'batch vs. message' dilemma is gaining more attention due to the time and expense that it takes to reverse-engineer multiple applications to use a standard message protocol.

Using Constellar Hub and MQSeries together will enable organizations to implement a fully-fledged EAI infrastructure in a fraction of the time. It will enable applications that "speak" messages and those that "speak" databases and files to integrate together in a fraction of the time it would have taken to convert them all to "speak" message.

There are essentially five ways in which Constellar Hub and MQSeries can work together to create a robust EAI infrastructure.

### ***Message Enrichment***

Constellar Hub can be introduced into an existing MQSeries infrastructure to provide sophisticated data enrichment or cleansing services.

Constellar Hub can read messages directly from a message queue. Business rules in the Constellar hub can add supplemental data to the message from, say, a third system. Even though this third system may not use MQSeries, Constellar Hub can obtain data from it via a database query, flat file access or any API.

Once the data has been obtained, Constellar Hub can change the original message by adding the supplementary data to it and placing on to another message queue.

### ***Step-up/step-down Transformation***

Constellar Hub can be used to load the data contained in MQSeries messages directly into a database or a flat file. The reverse is also true, a large flat file or data set can be converted into a series of MQSeries messages.

This capability removes the need to modify existing applications in order that they can take advantage of MQSeries. Not only can this reduce the time to integrate applications, but it minimizes the long term maintenance costs of the interface itself.

### ***Message Consolidation/Break-up***

Constellar Hub has an integral staging area that can store a stream of MQSeries messages before delivering them all in one go to their destination.

If necessary, Constellar Hub can create a new outbound message containing the contents of multiple inbound messages.

Conversely, a single inbound message can be broken up into multiple outbound messages of potentially differing structure and destination.

### ***Heterogeneous Join and Replicate***

Constellar Hub stores all incoming data in a canonical form that makes it possible to join and process data that originated from MQSeries, databases or files. Constellar Hub also has the ability to perform very complex data transformation that allows completely unrelated data structures to be passed between systems.

This capability allows organizations to quickly add applications to their EAI infrastructure without retrofitting them for use with MQSeries.

When data leaves Constellar Hub for its destination, the same information can be delivered in multiple different formats. For example, a customer record can be delivered at once as a message, database insert statement or addendum to a flat file.

### ***Message Pass-through***

Constellar Hub also has the ability to "listen in" on MQSeries traffic. A pass-through capability ensures all messages go straight from source to target *unless* you have decided that it must be processed by Constellar Hub.

The following are just a few examples of some complex business processes and data flows that Constellar Hub and MQSeries can handle when working together -

### ***Purchase Order reconciliation***

As a purchase order is raised, it is placed as a message onto MQSeries. Constellar Hub then reads the message from the queue. However, the message itself does not contain enough information for complete reconciliation, so Constellar Hub stores the message and then retrieves invoices from another, independent system.

As invoices are received, Constellar Hub reconciles them with the purchase order data. If reconciliation fails then, depending on the severity, Constellar Hub can repair the purchase order or retain it in the hub for manual reconciliation later.

### ***Address cleansing***

Constellar Hub reads messages from MQSeries that contain customer data. The message is read and the customer address and zip code data is verified for accuracy by business rules in the interface. Any errors are corrected and a new message is generated containing the corrected data.

The message is placed onto another message queue for delivery to its destination.

### ***Integration of operational systems with customer management system***

Constellar Hub is responsible for loading a customer relationship management (CRM) system from data found in multiple, independent operational systems. The loading occurs every few hours.

During the load, data must be pulled together from a legacy application, relational database and an MQSeries message queue.

Constellar Hub extracts all the data from these different sources and transforms, cleans and merges the data into a format that can be loaded into the CRM system.

## **Summary**

IBM MQSeries is the market-leading message-oriented middleware for transaction-oriented communication between heterogeneous applications.

Constellar Hub is the market-leading EAI transformation engine that enables data from MQSeries to be transformed for loading into virtually any other application as well enriching and repairing that data.

Constellar Hub does not require modification to existing production applications in any way.

This enables existing MQSeries networks to be extended to include bulk/batch-oriented applications that either cannot or should be modified to read and write messages.

By combining Constellar Hub with IBM MQSeries, organizations can develop an EAI infrastructure much more quickly and at less cost than either technology can do on its own.

## About Constellar Hub

The Constellar Hub is an EAI engine that provides a maintainable solution for data transformation, data movement and interface management across enterprise application networks.

The Constellar Hub utilizes an Oracle database to provide a transient data cache for the staging of data from multiple applications as well as leveraging the security, scalability and performance of the Oracle environment.

In accordance with the EAI criteria outlined earlier in this paper, the Constellar Hub has centralized hub architecture for the development and maintenance of all inter-application interfaces in the network.

Using the Constellar Hub's graphical user interface and powerful Transformation Definition Language, sophisticated rules-based transformations and set-based manipulations are possible as data is moved around the application network.

To maximize the available batch windows on your operational systems, the integrated scheduler automates the extraction of data from the source system. As all transformation is undertaken within the Constellar Hub, once the data has left the source system, no further processing is required on the source. Processing of data from many sources can then take place within the hub permitting merging and aggregation of many data streams. Loading of target systems can then be timed to coincide with the batch windows on the targets.

Other components of the Constellar product family include Constellar WarehouseBuilder, Constellar Interfaces for Oracle Applications and Constellar MQSeries Connectivity.

Constellar WarehouseBuilder provides sophisticated pre-aggregation of data for data warehouses, marts and other OLAP environments including Oracle Express and Hyperion Essbase. Constellar WarehouseBuilder uses a unique approach, called "the Dimensional Framework", to construct multi-dimensional aggregates in a single pass of the source data.

Constellar Interfaces for Oracle Applications is a set of pre-built conversion templates and interfaces that are certified for Oracle Applications. This certification is part of Oracle's Cooperative Applications Initiative so you know that the Constellar solution has been examined and certified by Oracle for use with Oracle Applications.

Constellar MQSeries Connectivity provides Constellar Hub with the ability to read and write messages from IBM's MQSeries, the market-leading messaging middleware. This connectivity enables Constellar Hub to integrate batch/bulk and near real-time data movement in one product.

Many Global 2000 customers utilize Constellar Hub to support the interfacing of a wide variety of operational systems.

Constellar Hub is the leading product specifically designed for EAI and Constellar Corporation is committed to ensuring it evolves to become *the* platform for a strategic EAI infrastructure.

To that end, Constellar Corporation is partnering with leading technology providers - through OEM, acquisition and reseller arrangements - to provide the full platform solution described in this white paper.

## About DataMirror Corporation

DataMirror (Nasdaq: DMCX; TSE: DMC) delivers solutions that let customers integrate their data across their enterprises. DataMirror's comprehensive family of products includes advanced real-time capture, transform and flow (CTF) technology that gives customers the instant access, integration and availability they demand today across all computers in their business.

Over 1,400 companies use DataMirror to integrate their data. Real-time data drives all business. DataMirror is headquartered in Toronto, Canada, and has offices worldwide. DataMirror has been ranked in the Deloitte and Touche Fast 500 as one of the fastest growing technology companies in North America.

***DataMirror***<sup>®</sup>

[www.datamirror.com](http://www.datamirror.com)

1 800 362-5955