SYSCOM Message Handling Framework

Introduction

Many enterprise-tier server-side applications follow a similar implementation pattern; requests are received, dispatched to coarse-grained business services, and then responses from the services are sent. Those server applications use various communication protocols with the client applications such as RMI, RMI/IIOP, message-based communication, sockets, or many other types of communication mediums such as file of FTP based file transfer, email, etc…

Through the integration of many solutions using IBM MQSeries Workflow, WebSphere MQ Integrator, and WebSphere MQ, we have concluded that a generic message handling framework is often required to provide for the implementation of automated workflow activities and other services based on the above mentioned pattern. Many systems are WebSphere MQ enabled, however, integration with MQSeries Workflow requires complex business logic that goes beyond data transformation (i.e., WebSphere MQ Integrator) and simple data exchange (i.e., WebSphere MQ). Large-scale integration requires that non-message-enabled applications be included in the workflow process.

The Message Handling Framework (MHF) is a set of Java interfaces and classes that facilitate the portable development of the style of the above server applications, either from scratch or by using a feature-rich, modular based implementation. The MHF, in itself, is not a message brokering technology or an application server platform. It simply provides a framework made to facilitate and accelerate the development of request/dispatch/service/response types of applications. The MHF doesn't rely on a Web application server or architecture to function, especially because server applications built on the MHF are often located in the enterprise tier. Web applications can, however, access the services of the MHF-based server application through whatever protocol is most convenient (RMI, MQ Series, etc…) to invoke enterprise tier-based business functions.

Beyond facilitating the creation of business services and providing flexible sets of receiver and sender types, the MHF makes available a configuration, logging and error handling infrastructure that can be used by all the various modules (receivers, dispatchers, services, senders, logging services, command modules) loaded in the MHF for the specific implementation needs of the application.

Lastly, the MHF provides the ability to control the server application, through a flexible command interface to query, and administer it.

Several "enterprise tier" server applications have been built by SYSCOM using the MHF, including:

- The SYSCOM Error Handling Framework (EHF)
- The SYSCOM Workflow UPES Activity Implementation Framework (UPES Framework)
- The SYSCOM User Administration Facility (UAF)

This document presents the architecture of the MHF and shows how the framework can be extended or built upon to create a server application that fits within the request/dispatch/service(s)/response(s) paradigm.

Functional overview

The various modules that exist in an MHF-based server application include:

- Receivers
- Dispatchers (usually one per server application)
- Services
- Senders

In addition to those basic modules, the MHF also includes:

- Logging services
- Command modules

In MHF terminology, all the module types enumerated above are referred to as nodes. In the MHF, messages are retrieved by the receiver nodes, dispatched by a dispatcher node, processed by one or more message handling service nodes and responses are sent by sender nodes.

Each of the nodes described above must comply with a specific interface in order to be loaded by and participate in the framework. The following scenario is representative of the processing of a message through the framework:

- A receiver node receives a message.
- The message (or "request") is forwarded to the dispatcher node.
- The dispatcher node examines the request parameters (and content if required) according to a set of rules – defined by the specific dispatcher implementation – and forwards it to the appropriate message Handling service(s).
- The message handling service node processes the request and (if required) responds through "response" message.
- The response is filtered through the dispatcher node which examines response parameters (and content if required) according to a set of rules and forwards it to the appropriate sender node(s).

Since MHF facilitates the implementation of server applications, MHF-based solutions must support high transaction volumes (i.e. a high number of requests processed simultaneously). The ability to safely "multithread" requests (and their associated processing) is neither imposed nor prevented by the basic architecture of the framework. The retrieval of a message as well as its dispatching, processing, and response can be performed within one thread, or across several threads.

The following list summarizes the major features available from the current MHF implementation. Since MHF is often used in the context of workflow/UPES activity implementations, an emphasis is made on the features of the MHF in the context of UPES activity implementations:

- Multithreaded framework capable of handling concurrent message invocations.
- Configurable processing volume capabilities.
- Configurable XA compliance (failed processing can cause invocation messages to be rolled back - XA compliant resources used to process a message can participate in the transaction).
- Extensible multi-point transactional integrity support before and after messages are processed by an MHF service. As long as the message serialization mechanism isn't compromised (e.g. file system full), messages cannot be lost during the end-to-end processing cycle (from message receipt to service processing to response sending).
- Flexible/extensible administrative interface to control and query the running framework (through files, MQ queue command messages, console based).
- Flexible/extensible logging capabilities (File logs, daily rotating file logs, console logging). Log buffering capabilities to minimize impact of logging on performance.
- Integration with UPES activity performance analysis facility. Allows tracking of average, minimum, maximum, and standard deviation for the duration of each UPES activity.
- Full integration with the SYSCOM Error Handling Framework to report errors.
- Support for synchronously implemented UPES activities through workflow/UPES specific interface using highly workflow-specific objects to facilitate rapid UPES activity implementation.
- Rule-based message dispatching to allow UPES activities to be processed by UPES services based on queue of origin and (optionally) workflow activity name.
- Support for asynchronously implemented UPES activities – this feature allows synchronous UPES activities to be completed through any kind of external event (MQ messages, files in directories, Web Service invocation, etc…). When an external event occurs it is retrieved by the MHF and correlated with the appropriate waiting UPES activity. The content of the activity completion message can include data from the original invocation message and/or from the external event message. The acceptable order in which messages should be received (UPES message first, or external message first, or any message first) is configurable.

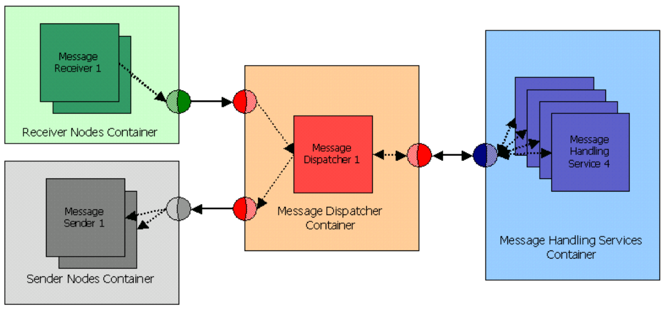
High level framework design

All nodes in the framework are grouped inside containers according to their type, namely:

- **The Message Receiver container:** Repository of Message Receiver nodes. The receiver container typically loads "daemon" nodes, which continuously check for incoming messages. When any receiver node receives a message, it signals the event to the container which relays the message to the dispatcher container.
- **The Dispatcher container:** Repository of the Dispatcher node. Typically, only one dispatcher node is be loaded at any time. The dispatcher container receives a "standardized" message from the

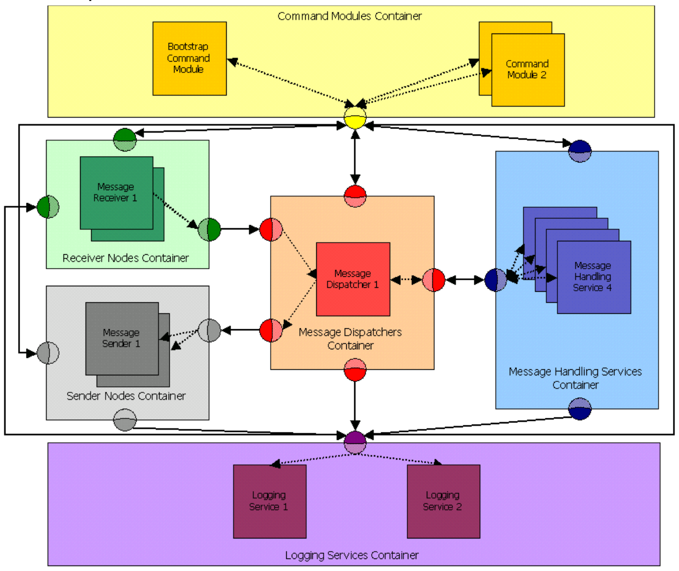
message nodes container and routes it to the dispatcher node. The dispatcher container also routes the message handling service invocation from the dispatcher to the message handling services container. When the appropriate service or services have been invoked the dispatcher routes the response to zero or more sender nodes.

- **The Message Handling Services container:** Repository of the Message Handling Service nodes. The container is a message dispatch listener, which routes messages to one or more services as specified by the message dispatcher.
- **The Message Sender container:** Repository of Message Sender nodes. The sender nodes in the container are instantiated and wait for the dispatcher signal that a response must be sent. The container routes the message response to the appropriate node as specified by the dispatcher.
- **The Command Modules container:** Repository of the Command Module nodes. The command modules provide full framework operational and configuration control. The interface presented to command module nodes through their container allows the dynamic loading, unloading and configuring of nodes, shutdown of the framework, and allows the invocation actions specific to each node in the framework.
- **The Logging Services container:** Repository of Logging Service nodes. All nodes in the framework can generate logging events. Those logging events ultimately arrive in the Logging Services container. The container then controls how logging events are relayed to log services.



The diagram above presents the simplified high-level architecture of the MHF, detailing receivers, dispatcher, services, and senders within their respective containers.

The bi-colored spheres on the diagram represent container connections. The light-colored side of each connector represents the internal interface between a container and its nodes. The dark-colored side of the connectors represents the external interface between two or more containers. The dotted arrows represent intra-container communication while solid arrows show container-to-container communication. Those conventions also apply to the diagram below, which presents the comprehensive high-level architecture of the MHF, including the nodes detailed above and the command modules and logging services within their respective containers.



For more information…

If you would like additional information about the ***SYSCOM Message Handling Framework***, or, if you would like to engage SYSCOM in your project, please contact us at Sales@SYSCOM.com or:

Vickie Wysokinski
VP of Sales and Marketing
410.539.3737 x 1300
VWysokinski@SYSCOM.com

Rick Marcuson
Director of Sales – West
913.897.3304
RMarcuson@SYSCOM.com

Brian McConnell
Director of Sales – East
410.539.3737 x 1310
BMcConnell@SYSCOM.com