Technical Overview

Version 1

Stahura Brenner

The Stahura-Brenner Group, Inc.
5405 Alton Parkway, 5-A #359
Irvine, CA 92604
(949) 733-8526

Copyright

Table of Contents

1 Technical Summary

The first release of the message broker was in 1999 as a customized solution for a very large health care company. At that time, the software was known as The Stahura-Brenner MQSeries Gateway. As more clients came on board, new interfaces and functionality were added to support many technologies in addition to the IBM WebSphere® MQ. The Stahura-Brenner Group is now re-releasing the software under the name of Plexus Message Broker.

1.1 Beginnings

The first iteration of the software was for a very large health care company. The crux of the problem was that the legacy back-end mainframes were incapable of processing more than 10 transactions per second yet a much higher throughput was needed. The strategy was to move all IBM MQSeries® (as it was known at the time) communication logic, message routing logic, message transformation logic, and error handling logic to a Windows® environment(s) where multiple, extensible Windows® systems could be configured to achieve the performance characteristics needed.

1.2 Components

In time other interfaces were added for other customers: a NetBIOS interface was added for communication with NetWare, an SNA interface was added to access older IBM systems, and so on. Not only were communication interfaces added, but the fundamental architecture was changed to include the concept of a Filter where Filters were stackable modules that could be placed between the two communication endpoints to perform various message handling functions. A Filter could aggregate multiple messages into a single message, spawn new messages in addition to letting the original message go through, ignore certain messages, change the format of a message from a data stream to an XML document and back again, and add function calls to other services to either validate the contents of the message or to add additional information to the message.

1.3 Threading

The Plexus Message Broker's threading architecture consists of multiple threading models. TCP and Microsoft® MSMQ, which are thread agnostic, utilize one kind of threading model. WebSphere® MQ, which requires each session to be on a separate thread, utilizes a different kind of threading model. If a Message Server is interfacing between TCP and WebSphere® MQ, the Message Server also bridges the two threading models. This architecture allows for over 1,000 TCP sessions to be marshalled into a few dozen WebSphere® MQ Integrator (WMQI) MQ threads for optimal resource utilization.

1.4 Unmanaged and Managed Code

Around 2007 there was a client requirement to allow the messaging software to interface .Net applications. This required the unmanaged C++ code to interface to managed .NET code. While the implementation of this was "interesting," the benefit was huge. Not only did it allow message filters to call Microsoft® services such as SOAP and advanced encryption services, but it also allowed the Message Servers of a Plexus Message Broker instance to run in the application domain space of the Plexus Message Broker instance. Thus all global STATIC variables are

really local to the individual Plexus Message Broker instance, making running multiple instances of the Plexus Message Broker on the same Windows system infinitely doable.

The integration of the .NET environment also meant that a communication endpoint of a Message Server could be a .NET application written in any .NET language such as C#, Visual Basic, or Java.

2 Architecture Summary

The Plexus Message Broker consists of the following four layers of software that allow it to be easily reused and adapted to new solutions:

Kernel	The Plexus Kernel provides the framework for a set of common services that can be accessed by Message Servers, Adapters and Filters.
Message Servers	A combination of Adapters and Filters that take messages from one source, manipulate the message(s), and send the resulting message to one or more destinations
Adapters	A communication interface component
Filters	Message manipulation components

2.1 Kernel

The Plexus Message Broker Kernel is the software framework that supports all other Plexus Message Broker services. It provides the common services required by the other components of the Plexus Message Broker. These include but are not limited to: the external user interface, tracing facilities, threading models, and alerting services.

2.2 Message Servers

This type of Plexus Message Broker component receives messages, manipulates the messages, and sends the messages on their way. A simple Plexus Message Broker implementation might have two Message Servers: one for inbound messages and one for outbound messages. A complex Plexus Message Broker might have six or more Message Servers. Depending on the performance requirements, there may be multiple instances of each these Message Servers.

A Message Server can take input from a source external to the Plexus Message Broker or from a source that is internal to the Plexus Message Broker. Similarly, a Message Server can output messages to either an internal Plexus Message Broker destination or to a destination that is external to the Plexus Message Broker.

A Message Server can accept messages from one communication source, such as TCP, and output the resultant message to a different communication destination such as WMQI MQ. It is equally possible that either the input source or the output destination, or both, are .NET applications.

A Message Server can also manipulate the message, although this is not a requirement, as when simply transitioning message from WMQI MQ to Microsoft® MSMQ. Message manipulation can range from simple editing of the message to more complex operations such as consolidating multiple messages into a single message or calling an external service provider to acquire additional information that is added to the message before it is sent on its way.

A Message Server may support a single sessions or many sessions.

2.3 Adapters

An Adapter is the component of a Message Server that provides the communication interface to services such as TCP, Microsoft® MSMQ, WMQI MQ, or SNA. There are usually two Adapters defined for each Message Server. Typically, one Adapter might act as a service provider (receiving a message), while the other acts as a service user (sending a message). The two Adapters can provide access to two different communication interfaces. That is, the source might be TCP and the destination might be WebSphere® MQ.

2.4 Filters

Filters reside on the data path between the two Adapters and provide the ability to tailor the Plexus Message Broker to each unique environment and task without requiring modification to the Plexus Kernel or Plexus Adapters.

A Filter is a programmable component that can be assigned to an Adapter. It is typically used to manipulate the content of a message, intercept a message, or route a message. They can also be used to manipulate Adapter interfaces. Filters can be written in: C#; Visual Basic .NET; C++; or JAVA.

Since a Message Server may have two Adapters, a Message Server may have up to two Filters as well.

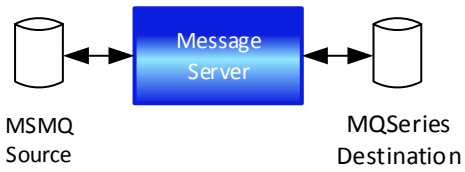
3 Kernel Details

The Plexus Message Broker Kernel provides the following functionality:

- Configuration User Interface
- Run-time Display Interface
- Character Code Translation Services (ASCII, EBCDIC, etc.)
- Performance Metric Management
- SNMP Alerting Services
- Heartbeat Services
- Threading Services
- Unmanaged to Managed Code Transition Services
- Unmanaged to Java Services
- Auditing Services
- Tracing Services
- Printing Services
- Thread Management
- Automatic Initiation
- Automatic Recovery

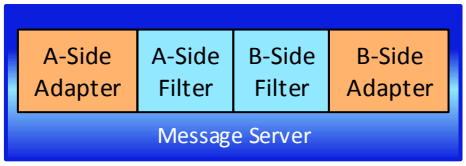
4 Message Server Details

A Message Server is a logical component of the Plexus Message Broker that takes input from one source, possibly manipulates the message(s), and sends the output to a destination. It can take input from any of the supported communication protocols and send the message out through any of the supported protocols. In the example below, the Message Server takes a message from a Microsoft® MSMQ queue and sends the message to an IBM MQSeries® queue.

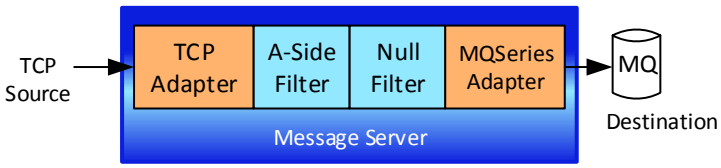


Message Servers are comprised of two Adapters and zero to two Filters. Adapters are responsible for providing messages to a network interface, such as TCP or IBM MQSeries®. Filters are responsible for message transformation and routing rules

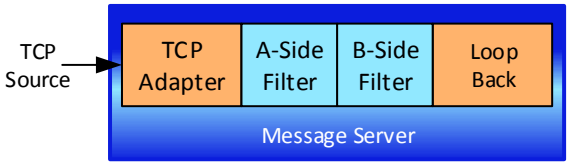
Each Message Server has an A Side and a B Side as illustrated below. The A Side consists of an A-Side Adapter and an A-Side Filter. The B Side consists of a B-Side Filter and a B-Side Adapter.



There are cases where a B-Side filter is not assigned, as demonstrated by the Null Filter below.



There are also cases when the B-Side Adapter is not needed. In these cases the Loop Back Adapter is configured, as illustrated below.

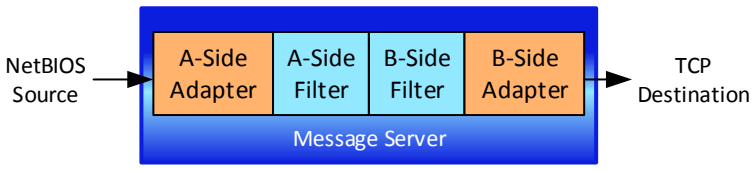


Message Server functionality is defined by the combination of the Adapters, Filters and their configuration. If the A-Side and B-Side Adapters were both TCP Adapters, their behaviors could be very different based on their configuration.

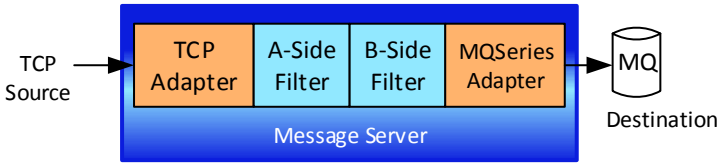
A single instance of a Plexus Message Broker can support many Message Servers. The number of Message Servers supported depends on hardware configuration, message volume, message flow control issues, complexity of message transformation, and Message Server redundancy requirements.

5 Adapter Details

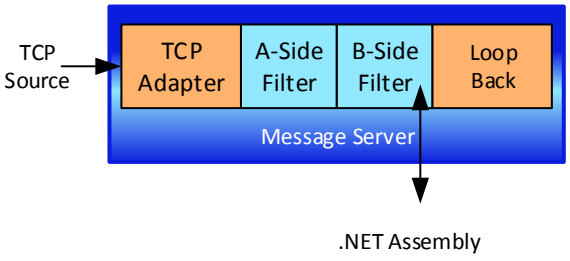
An Adapter is a software component that is a configurable part of a Message Server. There are different types of Adapters. As illustrated below, each type communicates with a unique type of external interface such as TCP sockets, IBM MQSeries® queues, or Microsoft® MSMQ queues. A typical Adapter is capable of both receiving messages from an external communications interface and sending messages to the same type of external communications interface.



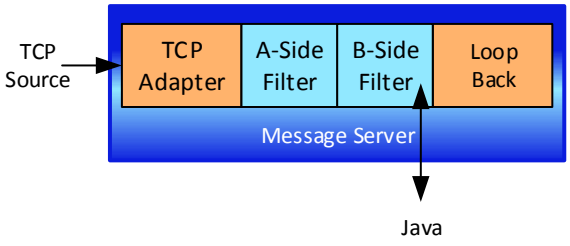
Each Message Server can have up to two Adapters: an A-Side Adapter and a B-Side Adapter. The Adapters operate independently and may be of different types or of the same type. In the Message Server illustrated below, the protocol employed to transport the message is being converted from TCP to an IBM MQSeries® connection. A message may be received by the TCP Adapter and forwarded to a queue using the MQSeries® Adapter. The associated response may be received by the MQSeries® Adapter and forwarded by the TCP Adapter.



A Loopback can also be used for the B-Side Adapter. As illustrated below, the A-Side Adapter interfaces with TCP ports; the A-Side Filter provides some messaging transformation; and the B-Side Filter provides the interface to a .Net Assembly.



Similarly, it is possible that the B-Side Filter provide the interface to Java bytecode as illustrated below.



The following Adapters are supported in the Plexus Message Broker:

IBM WebSphere® MQ	Provides connection to IBM MQSeries® queues. Queues may be either local or remote queues.
Microsoft® MSMQ	Provides connection to Microsoft® MSMQ queues. Queues may be local or remote private queues.
TCP/IP	Provides connection to TCP sockets. Message framing of the TCP data stream is specified in the configuration file.
NetBIOS	Provides connection to NetBIOS sessions
SNA	Provides connection to LU6.2 sessions
LoopBack	A stub adapter used as a placeholder but provides no functionality. Typically, in these cases, the Filter is either the application or an interface to an application.

Note: Some communication interfaces have built-in message framing technology such as IBM MQSeries®, Microsoft® MSMQ, NetBIOS, or LU6.2. TCP does not contain such a technology. Thus the TCP Adapter provides support for a number of message framing protocols so that the TCP adapter can identify a complete message.

5.1 IBM MQSeries® Adapter

The MQSeries® Adapter is a high performance messaging adapter that can be coupled with other adapters to provide a very wide array of messaging technology solutions. Examples of these pairings are listed below.

Example Adapter Pairing	Pairing Purpose
MQ to MQ	Used whenever the messages in an MQ message flow need to be modified, duplicated, eliminated, rerouted or simply audited.
MQ to MSMQ	Allows for transitioning IBM messaging technology to/from Microsoft® MSMQ. Also allows for messages to be modified, duplicated, eliminated, rerouted or simply audited.
MQ to TCP	Allows for MQ to TCP sockets transitions. The choice of filters additionally provides for MQ to/from Web Services, MQ to HTTP, MQ to .Net applications, and MQ to Java applications.
MQ to NetBIOS	Provides a gateway that allows messages to be transitioned between MQ and the NetBIOS protocol. Filters allow the messages to be modified, duplicated, eliminated, rerouted or simply audited.
MQ to LU6.2	Provides an MQSeries® interface to older IBM systems that support LU6.2 but not MQSeries®.
MQ to LoopBack	In this scenario, the LoopBack Adapter doesn't provide any meaningful functionality. However, the Filter in the LoopBack adapter can provide a data path to either .Net applications or Java applications. This thus creates an MQ to local application messaging interface.

5.1.1 Features

- Supports both local and remote queue managers
- Provides ability to commit/abort messages
- Can commit messages to multiple queues
- Can commit messages across local and remote Queue Managers
- Supports Correlation IDs
- Can wait for messages with specific correlations IDs
- Supports message expiration
- Supports over 125 messages per second on a single quad processor Windows® server utilizing approximately 10% of the processor
- Supports over 2000 send and receives queues
- Services each input queue by an individual thread
- Can service over 200 output queues per individual thread
- Can have multiple queue servers in the same Plexus Message Broker listening to the same input queue
- Can have multiple instances of the Plexus Message Broker on the same server listening to the same input queue
- Can have multiple Plexus Message Brokers on multiple physical servers listening to the same queue

- Provides auditing of received and sent messages
- For every session, can view last message received and sent, as well as all messages received and sent

5.1.2 Configuration Attributes

Configuration Attribute	Acting as Client	Acting as Server
Application ID	Yes	Yes
Channel	Yes	Yes
Character Translation	Yes	Yes
Default Output Queue		Yes
Default ReplyTo Queue		Yes
Dynamic Routing through Cluster	Yes	Yes
Expiration	Yes	
Host Name of IP Address	Yes	
Limit Number of Sessions		Yes
Number of Sessions	Yes	
Number of Threads	Yes	
Output Only	Yes	
Output Queue	Yes	
Persistence	Yes	Yes
Queue Manager Identification	Yes	Yes
Receive Queue		Yes
Remote or Local Queue	Yes	Yes
Response Queue	Yes	
Transactional		Yes
UserID	Yes	

5.2 Microsoft® MSMQ Adapter

The Microsoft® MSMQ Adapter is a high performance messaging adapter that can be coupled with other adapters to provide a very wide array of messaging technology solutions. Examples of these pairings are listed below.

Example Adapter Pairing	Pairing Purpose
MSMQ to MQ	Allows for transitioning of Microsoft® MSMQ messaging technology to/from IBM messaging technology. Also allows for messages to be modified, duplicated, eliminated, rerouted or simply audited.
MSMQ to MSMQ	Used whenever the messages in an MSMQ message flow need to be modified, duplicated, eliminated, rerouted or simply audited.
MSMQ to TCP	Allows for MSMQ to TCP sockets transitions. The choice of Filters additionally provides for MSMQ to/from Web Services, MSMQ to HTTP, MSMQ to .Net applications, and MSMQ to Java applications.
MSMQ to NetBIOS	Provides a gateway that allows messages to be transitioned between MSMQ and the NetBIOS protocol. Filters allow the messages to be modified, duplicated, eliminated, rerouted or simply audited.
MSMQ to LU6.2	Provides an MSMQ interface to older IBM systems that support LU6.2.
MSMQ to LoopBack	In this scenario, the LoopBack Adapter doesn't provide any meaningful functionality. However, the Filter in the LoopBack adapter can provide a data path to either .Net applications or Java applications. This thus creates an MSMQ to local application messaging interface.

5.2.1 Features

- Supports local private queues
- Provides ability to commit/abort messages
- Provides support for Correlation IDs
- Can wait for messages with specific correlations IDs
- Supports thread pools for receive queue listeners
- Can have multiple queue servers in the same Plexus Message Broker listening to the same input queue
- Can have multiple instances of the Plexus Message Broker on the same server listening to the same input queue
- Provides auditing of received and sent messages
- For every session, can view last message received and sent, as well as all messages received and sent

5.2.2 Configuration Attributes

Configuration Attribute	Acting as Client	Acting as Server
Application ID	Yes	
Character Translation	Yes	Yes
Default Response Queue		Yes
Expiration	Yes	
Output Queue	Yes	
Output Only	Yes	
Persistence	Yes	Yes
Receive Queue		Yes
Response Queue	Yes	
Transactional		Yes

5.3 TCP Adapter

The TCP Adapter is a high performance messaging adapter that can be coupled with other adapters to provide a very wide array of messaging technology solutions. Examples of these pairings are listed below.

Example Adapter Pairing	Pairing Purpose
TCP to MQ	Allows for transitioning TCP messaging technology to/from IBM MQSeries® messaging technology. Also allows for messages to be modified, duplicated, eliminated, rerouted or simply audited.
TCP to MSMQ	Allows for transitioning TCP messaging technology to/from Microsoft® MSMQ. Also allows for messages to be modified, duplicated, eliminated, rerouted or simply audited.
TCP to TCP	Used whenever the messages in a TCP stream need to be modified, duplicated, eliminated, rerouted or simply audited. The choice of filters additionally provides for TCP to/from Web Services, TCP to HTTP, TCP to .Net applications, and TCP to Java applications.
TCP to NetBIOS	Provides a gateway that allows messages to be transitioned between TCP and the NetBIOS protocol. Filters allow the messages to be modified, duplicated, eliminated, rerouted or simply audited.
TCP to LU6.2	Provides a TCP interface to older IBM system applications that support LU6.2.
TCP to LoopBack	In this scenario, the LoopBack Adapter doesn't provide any meaningful functionality. However, the Filter in the LoopBack adapter can provide a data path to either .Net applications or Java applications. This thus creates a TCP to local application messaging interface.

5.3.1 Features

- Supports up to 2000 connections per single listener
- Supports up to 125 transactions per second
- Includes various message framing protocols
- Provides message delivery confirmation partner Adapter
- Supports multiple TCP sessions per thread
- Supports filtering of inbound IP addresses
- Allows the identification of the network card in multi network card systems
- Provides auditing of received and sent messages
- For every session, can view last message received and sent, as well as all messages received and sent

5.3.2 Configuration Attributes

Configuration Attribute	Acting as Client	Acting as Server
Character Translation	Yes	Yes
Encrypted Remote Login	Yes	Yes
Idle Timeout		Yes
IP Address Filtering		Yes
Local IP Address	Yes	Yes
Persistence of Session	Yes	Yes
Port	Yes	Yes
Protocols	Yes	Yes
CCF	Yes	Yes
HTTP	Yes	Yes
NCIC	Yes	Yes
Nlets	Yes	Yes
NySpin	Yes	Yes
RCF	Yes	Yes
Record Separator	Yes	Yes
Remote Host Name or IP address	Yes	
Response Timeout		Yes
Transactional		Yes

5.4 NetBIOS Adapter

The NetBIOS Adapter is a high performance messaging adapter that can be coupled with other adapters to provide a very wide array of messaging technology solutions. Examples of these pairings are listed below.

Example Adapter Pairing	Pairing Purpose
NetBIOS to MQ	Allows for transitioning of NetBIOS messaging technology to/from IBM MQSeries® messaging technology. Also allows for messages to be modified, duplicated, eliminated, rerouted or simply audited.
NetBIOS to MSMQ	Allows for transitioning of NetBIOS messaging technology to/from Microsoft® MSMQ messaging technology. Also allows for messages to be modified, duplicated, eliminated, rerouted or simply audited.
NetBIOS to TCP	Allows for NetBIOS to TCP sockets transitions. The choice of filters additionally provides for NetBIOS to/from Web Services, NetBIOS to/from HTTP, NetBIOS to/from .Net applications, and NetBIOS to/from Java applications.
NetBIOS to NetBIOS	Used whenever the messages in a NetBIOS message flow need to be modified, duplicated, eliminated, rerouted or simply audited.
NetBIOS to LU6.2	Provides a NetBIOS interface to older IBM systems that support LU6.2.
NetBIOS to LoopBack	In this scenario, the LoopBack Adapter doesn't provide any meaningful functionality. However, the Filter in the LoopBack adapter can provide a data path to either .Net applications or Java applications. This thus creates a NetBIOS to local application messaging interface.

5.4.1 Features

- Supports 254 connections
- Supports specific or non-specific NetBIOS name client and server pairs
- Supports delivery confirmation
- Supports selection of NetBIOS adapter
- Has the ability to wait for a particular response before proceeding
- Can have multiple instances of the Plexus Message Broker on the same server providing more than 254 connections
- Provides auditing of received and sent messages
- For every session, can view last message received and sent, as well as all messages received and sent

5.4.2 Configuration Attributes

Configuration Attribute	Acting as Client	Acting as Server
Character Translation	Yes	Yes
LANA Number	Yes	Yes
Listen Name	Yes	
Listen Peer Name	Yes	
MyName		Yes
Peer Name		Yes
Transactional	Yes	Yes

5.5 LU6.2 Adapter

The LU6.2 Adapter is a high performance messaging adapter that can be coupled with other adapters to provide a very wide array of messaging technology solutions. Examples of these pairings are listed below.

Example Adapter Pairing	Pairing Purpose
LU6.2 to MQ	Allows for transitioning IBM LU6.2 messaging technology to/from IBM MQSeries® messaging technology. Also allows for messages to be modified, duplicated, eliminated, rerouted or simply audited.
LU6.2 to MSMQ	Allows for transitioning IBM LU6.2 messaging technology to/from Microsoft® MSMQ. Also allows for messages to be modified, duplicated, eliminated, rerouted or simply audited.
LU6.2 to TCP	Allows for LU6.2 to TCP sockets transitions. The choice of filters additionally provides for LU6.2 to/from Web Services, LU6.2 to HTTP, LU6.2 to .Net applications, and LU6.2 to Java applications.
LU6.2 to NetBIOS	Provides a gateway that allows messages to be transitioned between IBM LU6.2 message technology and the NetBIOS protocol. Filters allow the messages to be modified, duplicated, eliminated, rerouted or simply audited.
LU6.2 to LU6.2	Used whenever the messages in a LU6.2 message flow need to be modified, duplicated, eliminated, rerouted or simply audited.
LU6.2 to LoopBack	In this scenario, the LoopBack Adapter doesn't provide any meaningful functionality. However, the Filter in the LoopBack adapter can provide a data path to either .Net applications or Java applications. This thus creates a LU6.2 to local application messaging interface.

5.5.1 Features

- Supports LU Actions confirmation
- Supports delivery confirmation
- Supports connect on demand when data exists
- For every session, can view last message received and sent, as well as all messages received and sent

5.5.2 Configuration Attributes

Configuration Attribute	Acting as Client	Acting as Server
Character Translation	Yes	Yes
Confirmation Requirement	Yes	Yes
Connect on Demand	Yes	
Local LU Alias	Yes	
Local TP Name	Yes	Yes
Mode Name	Yes	
Password	Yes	
Persistence of Connection	Yes	
Remote LU Alias	Yes	
Remote TP Name	Yes	
Transactional		Yes
UserID	Yes	

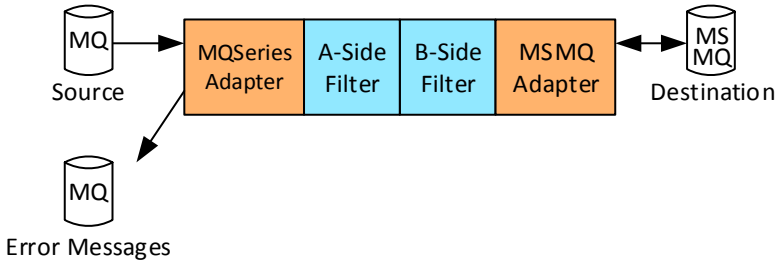
6 Filter Details

Filters are programmable software components that customize the processing of data passing through a Message Server. Each Filter is contained within a separate code file. They may be written in C#, Visual Basic .NET, C++, or Java. Up to two Filters may be assigned to a Message Server: one for the A Side and one for the B Side.

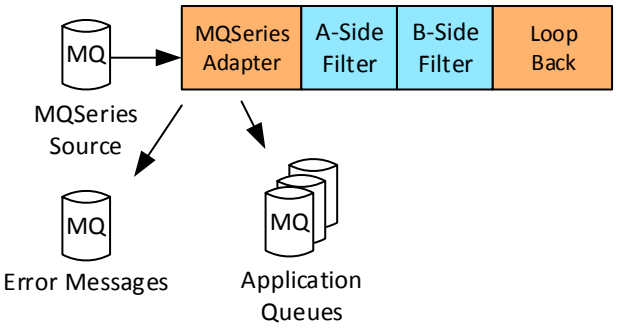
6.1 Inbound and Outbound Messages

In the simplest case, the A-Side Filter manages inbound messages while the B-Side Filter manages the outbound messages. In more complex scenarios, the A-Side Filter can handle inbound messages as well as outbound messages.

Often the A-Side Filter is capable of routing messages that don't conform to business rules to an alternate queue. This is illustrated in the diagram below in which the A-Side Filter is responsible for validating the MQSeries® messages based on business rules. If the A-Side Filter determines that a message does not conform to these rules, it inserts the message along with the appropriate error code into an MQSeries® error queue for later processing. In this scenario, only the A-Side Filter understands the business rules and therefore only the A-Side Filter can identify the error.



It is equally possible that the message is inserted into one of possibly many MQSeries® queues either for error processing or routing to an application. The routing of the messages is dependent on the business rules engine and/or the severity of the error. In this case all the logic is in the A Side with the B Side simply being a placeholder.



6.2 Messaging

The transformation of a message is the responsibility of a Filter. This transformation can be as simple as reformatting a message. It can also be as complex as aggregating multiple inbound messages into a single outbound message or generating multiple outbound messages to many different destinations based on a single inbound message. It is equally possible for the Filter to "drop the message on the floor" based on the business rules engine.

6.3 Filter Functionality

Filters are flexible enough to support just about anything that can be done to a message via code. They can be written in either managed or unmanaged code. Some examples of Filter operations include:

- Translate character sets
- Modify an existing message
- Encrypt messages
- Change one application-level protocol to another application-level protocol
- Accumulate multiple messages and combine the contents into a single message, then send the single message on its way
- Take a single message and generate multiple copies of that message, then send each on its way
- Take a single message and generate many different messages from it, then send each message to a different destination
- Take a message, manipulate the message, send the message to a user-written .Net or Java library for processing, wait for the response, manipulate the response, and send the response to a destination
- Take a single message and send a copy of that message to two different destinations; for example, to a primary site and a secondary site
- Enforce security
- Retain some state of the message in persistent storage for later reassembly with an associated response message

Over the years, a large number of Filters have been written. To write a new Filter, an existing Filter is typically modified to include the new feature set.

6.4 Current Filter Descriptions

The currently available filters are described herein, organized by *General* or solution.

6.4.1 General Filters

Filter	Description
HealthCheck	Data path viability health check requests. Used both with an F5 load balancer and system software.
HTTPAdmin	Provides Web Service interface to Plexus Message Broker Message Server status and configuration
HttpClientMq	Take a message and encapsulates in a SOAP wrapper
HttpServerMq	Removes a SOAP wrapper from message
NETRequestor	Provides a general interface for writing .NET-based client transaction modules
NetSoap	Provides a general interface for writing .NET-based server transaction modules

6.4.2 Health Care Filters

Filter	Description
HCPcm	Provides an interface between large numbers of Linux servers and IBM mainframe using the Health Care TCP Protocol
HCWMQI	Provides an interface between WebSphere® Message Broker (WMB) and IBM mainframe using MQ with Health Care Protocol

6.4.3 Emergency Services Filters

Filter	Description
AsapCsIn	Manages inbound Central Station Connections, identifies source, and performs authorization
AsapNletsIn	Receives messages from PSAPs, performs schema validation, authorization, and routes message to specific central station delivery queue
AsapWafIn	Receives messages from Web Application Firewall appliance, performs schema validation, routes Address Verification requests to metering queue, and routes Alarm requests to Nlets delivery queue
AsapWafOut	Submits Central Station ASAP message to Web Application Firewall (WAF) appliance, and manages any rejected (virus infected) messages
ASAPWebClient	Wraps ASAP message in ASAP SOAP envelope and delivers using HTTP protocol
ASAPWebServer	Receives ASAP SOAP message using HTTP protocol and extracts the encased ASAP message. Handles both SendASAPMessage and ReceiveASAPMessage.
NletsMeter	Provides metering of ASAP address verification messages from Central Stations
Psap	Provides simulation of Public Safety Access Point (PSAP)

6.4.4 Justice / Law Enforcement Filters

Filter	Description
FromNcicAsyncMq	Receives NCIC unsolicited messages and determines destination based on business rules and message content
IIIFlow	Component of Law Enforcement Gateway Service (LEGS) that delivers III messages to NCIC
Ncic	Part of LEGS that provides synchronous interface to NCIC using MQ
NcicProxy	Provides a simulation of NCIC 2000 asynchronous protocol. Used to convert NCIC 2000 TCP protocol to MQ.
NcicProxyAsyncResp	Provides a simulation of NCIC 2000 synchronous protocol. Used to convert NCIC 2000 TCP protocol to MQ.
NcicSoapServerMq	Part of LEGS that provides synchronous interface to NCIC using MQ

6.4.5 Unemployment Benefits Filters

Filter	Description
IconWebClient	Delivers message to the Unemployment Information Interstate Connection Network(UI-ICON) via Web Services
IconWebServer	Receives Web Services inbound messages from the Unemployment Information Interstate Connection Network(UI-ICON)
JsndIdb	Provides IDB protocol for communications between UI-ICON and the MCP Server
HLCNTS	Converts the Unisys HLCN terminal services protocol to a Unisys COMS station Protocol
HLCNTSClient	Connects as a client to Unisys MCP Server using HLCN terminal services protocol

6.4.6 Legacy Terminals Filters

Filter	Description
NyspinAdmin	Provides Web Service interface to Broker operational data by TGExplorer (NYSPIN Transitional Adapter Administrative Console)
NySpinTa	Transitional adapter that provides conversion between deprecated NYSPIN protocol interface and transitional broker

6.4.7 Public Safety Filters

Filter	Description
NetWebClient	Provides interface between MQ queue and LEXS system
NletsGwyMq	Provides interface to Nlets NJIN
ReceiveNLETSMsg	Receives an Nlets Web Service message and unwraps the payload
RmvControl	Provides an interface to the Registry of Motor Vehicles using the TCP protocol
RmvMq	Provides an interface that translates a Criminal Justice Information Service message to a Registry of Motor Vehicles message
SendNLETSMsg	Wraps an Nlets message in an Nlets SOAP enveloper and delivers it using HTTP

7 Redundancy

Within a corporation, there are two forms of redundancy:

- Local redundancy that addresses single component failures
- Geographic redundancy that addresses data center failures

Most corporations first focus on local site or data center redundancy. This ensures that if any hardware components that support mission critical applications fail, there are additional hardware components to take their place. "Take their place" means having two such hardware components configured in an Active/Active mode or having a hardware component in Hot Standby mode. Most corporations focus on local redundancy because the probability of the data center failing is less than the probability of a single hardware component failing.

Once local redundancy is achieved, geographic redundancy is sought mainly for purposes of disaster recovery. In this scenario, a second data center is acquired that contains comparable hardware. All critical applications and its data are backed up to this second data center. As with the hardware, the primary data center and secondary data center can be run in Active/Active mode (preferred) or in Active/Passive mode.

7.1 Local Redundancy

The Plexus Message Broker supports many different types of local redundancy including:

- More than one Message Server in the same instance on the same physical server
- More than one instance of the Plexus Message Broker on the same physical server
- Plexus Message Broker instances on multiple physical servers

In each of the above cases, the creation of redundant instances of the Plexus Message Broker not only provides resiliency in the service but also can provide massive throughput gain when high transaction volumes measured in hundreds of transactions per second is desired.

7.2 Geographic Redundancy

The Plexus Message Broker supports two types of geographic redundancy including:

- Plexus Message Brokers at the Primary and Secondary data center in Active/Passive mode
- Plexus Message Brokers at the Primary and Secondary data centers in Active/Active mode

8 System Requirements

System requirements for the Plexus Message Broker are:

Windows Environment:	32-bit[1]
Windows Server:	Windows® Server 2008 R2
	Windows® 7
Processor:	Intel Dual Core or higher
	AMD Dual Core or Higher
Memory:	1 GB Minimum
Disk space:	500 MB
Management Console:	Built-in Client
	Web Services
.Net Framework	4.0

1. Can also run as a 32 bit application in a 64 bit environment

9 Licensing

9.1 Production and Active Disaster Recovery Systems

- Cost per production Plexus Message Broker instance is a one-time fee.
- Business Hours support is a yearly fee.
- Extended Hours support is an additional fee per year, billed on the same schedule as the Business Hours support.
- Customer is responsible for all addition taxes and fees.
- Business Hours support contract is required 90 days after Production go-live.
- Extended Hours support contract is optional.

9.2 Development and Quality Assurance (QA) Systems

- Cost per development or QA Plexus Message Broker instance is significantly reduced.
- Customer is responsible for all addition taxes and fees.
- Business Hours support contract is required 90 days after Production go-live.
- Extended Hours support contract is optional.

9.3 Passive Disaster Recovery Systems

- There is no charge for passive Disaster Recovery systems.

9.4 Custom Programming

- Pricing is determined based on the extent of the job.