Study and Analysis on Web Methods Broker

Chen Chen
Dalian Institute of Science and Technology, Dalian, China
35765801@qq.com

Keywords: WebMethods Broker, Analysis, performance

Abstract: This paper describes the information about how to administer and manage webMethods Broker. It describes how to create and manage Brokers on a broker server, set up access permissions, and monitor document traffic, and points out that the system administrator who is responsible for configuring and monitoring webMethods Broker, assumes basic concepts of webMethods Broker architecture, and the systems on which you are running the webMethods Broker software meet or exceed the recommended system requirements outlined in the webMethods System Requirements spreadsheet on the webMethods Advantage Web site. The paper provides Broker Security Model, and describes how webMethods Broker security works and its performance.

1. Introduction

WebMethods Broker is the primary component in what is referred to as the “message backbone” in a webMethods integration environment. Along with other webMethods components, webMethods Broker facilitates asynchronous, message-based integration using the publish-and-subscribe model.The publish-and-subscribe model is a specific type of message-based solution in which applications exchange messages (called documents in webMethods) through a third entity called a broker.

2. Broker Monitor

Broker Monitor (awbrokermon) is a separate process that runs on the machine that hosts Broker Server. It has three functions:

It starts Broker Servers.

It monitors the state of the Broker Servers running on the host machine and automatically attempts to restart them if they fail.

It logs status messages about the Broker Servers.

Broker Monitor listens for requests from administrative clients (such as the Broker user interface in My webMethods) on port 6850. This port assignment is fixed and cannot be reassigned. Although a computer can host multiple instances of Broker Server, it can host only one instance of Broker Monitor. This one instance monitors all Broker Servers that run on the machine.

2.1. How Broker Monitor Starts Broker Servers

The Broker Monitor configuration file points to the Broker Servers that reside on the host machine. When you start Broker Monitor, it reads this configuration file and automatically starts all of the Broker Servers that are identified in the file.The Broker Monitor’s configuration file is updated automatically when you install Broker Server on the host machine and when you define additional instances of Broker Server using the server_config utility. You do not edit this file manually.

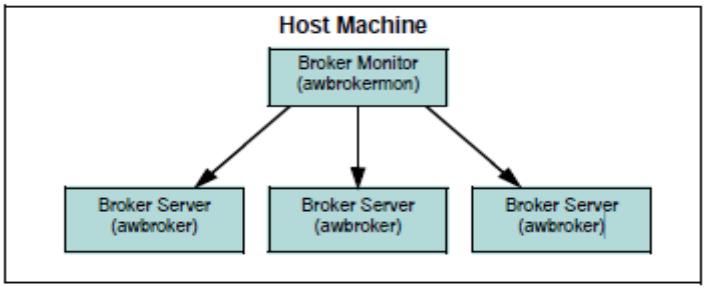

Figure 1. Broker Monitor monitors all Broker Servers on the host machine.

2.2. How Broker Monitor Monitors the State of Broker Servers

When Broker Monitor starts a Broker Server, it internally captures the process ID that the operating system assigns to the server process. Using this ID, Broker Monitor continually monitors the server's run state. If the server exits unexpectedly (i.e., it is not stopped in a controlled way through the Broker user interface or the server_config utility), Broker Monitor automatically attempts to restart it. To avoid the situation where Broker Monitor keeps restarting a Broker Server that is unable to stay up and running, Broker Monitor will not restart a Broker Server that has experienced three unexpected exits within a five minute period.

2.3. Status Messages Logged by Broker Monitor

Broker Monitor maintains a log file in which it records the following events for the Broker Servers that it monitors:

Broker Server is launched (by Broker Monitor).

Broker Server exits unexpectedly.

Broker Server is stopped by an administrative action.

On a Windows machine, this information is also written to the Windows event log and can be viewed through the Windows Event Viewer (e.g., Start > Settings > Control Panel > Administrative Tools > Event Viewer). On a UNIX machine, this information is written to the system log, /etc/syslog.conf.

3. Managing webMethods Broker Servers

The Broker Server is a container-like process that hosts one or more Brokers. It manages the communication, memory management, and queue storage functions for all the Brokers that it hosts.

3.1. The Data Directory

Every Broker Server has it own data directory, which holds the Broker Server's configuration file and its log files. Frequently, the data directory also holds the Broker Server's queue storage files, but often these files are placed on a separate (usually faster) storage device.

3.2. The Configuration File (awbroker.cfg)

The Broker Server configuration file (awbroker.cfg) contains parameters that define a single Broker Server instance. The configuration file resides in the Broker Server's data directory and supplies information such as the Broker Server's license key, base port, and the location of its queue storage files. Most parameters in the configuration file can be displayed and modified using the Broker user interface in My webMethods. Certain parameters, however, can only be modified using a command-line utility or by editing the awbroker.cfg file directly.

3.3. Broker Server Communication Ports

When you install Broker Server or create a new Broker Server with the server_config utility, you specify the server's base port. Broker Server uses the base port for non-SSLcommunications. It uses the two ports immediately below the base port for SSL-based communications. By default, port 6849 is the Broker Server base port. If you do not explicitly assign a base port when you install or

create a Broker Server, it uses the default port 6849 for non-SSL requests and ports 6848 and 6847 for SSL requests.

4. Managing Brokers

A Broker is an entity that resides on a Broker Server. When a client connects to BrokerServer, it specifies the Broker with which it wants to interact.A Broker contains three key types of objects:

Document types, which define the kinds of documents that clients of the Broker can exchange.

Client groups, which are objects that represent a set of client properties and permissions. Any client that connects to a Broker must declare the client group to which it belongs. If the Broker Server on which the Broker is running is SSL enabled, an access control list (ACL) can be assigned to the client group. By restricting access to a client group, you restrict access to the Broker.

Clients, which are state objects that represent the client programs that connect tothe Broker. The state object maintains a client's list of subscriptions and its queue. If a client program connects to the Broker as an "explicit destroy" client, the Broker maintains the state object even if the client program is physically disconnected from the Broker.

4.1. Creating a Broker

When you install Broker Server using the webMethods Installer, the Installer creates one Broker, called "default," on the Broker Server. You can use the following procedure to create additional Brokers on the Broker Server if necessary. You can also use this procedure to add a Broker to a Broker Server that you have created using the server_config utility. Unlike the installation process, the server_config utility does not install a default Broker on a new Broker Server. You must explicitly create the Broker yourself.

4.2. Deleting a Broker

Use the following procedure to delete a Broker from a Broker Server. When you delete a Broker, you immediately disconnect and delete all clients that exist on it. You also permanently delete all of its client groups and document types.

4.3. Managing Transactions

Transaction processing allows a Broker client to group the events it publishes as a single unit of work called a transaction. A transaction either completes successfully, is rolled back to some known earlier state, or it fails. Once all of the documents that make up a transaction have been published, delivered, or received, the Broker client ends the transaction. A transaction can be ended by committing the transaction or aborting the transaction. The Broker's transaction manager coordinates and controls the transactions that are initiated by Broker clients. Transactions that run under the Broker transaction manager include transactions initiated by regular Broker transactional clients, as well as regular and XA transactions that are initiated by the Broker JMS API. You use the transaction controls in My webMethods to monitor and manage transactions that are running under the transaction manager on the Broker. These controls allow you to monitor the activity of transactions as they execute and take action against transactions that do not appear to be running correctly.

4.3.1 Viewing Running Transactions

If a client has started a transaction on the Broker, that transaction appears on the Transactions tab. The transaction disappears from this tab when the client explicitly ends the transaction. In many cases, transactions complete too quickly to be viewed in the Transactions tab. However, the list is useful for monitoring the state of long-running transactions and for spotting transactions that have become hung in the system. To view transactions running on Broker:

In My webMethods: Messaging > Broker Servers > Brokers.

In the Brokers List, click the Broker whose transactions you want to view. If the Broker does not appear in the list, use the Search tab to locate it.

On the Broker Details page, click the Transactions tab. The transaction list at the bottom of this tab displays transactions that are currently running on the Broker.

4.3.2 Configuring the Transaction Timeout Options

For transaction processing, you can configure the Broker to monitor the length of time between stages of a transaction and to take a prescribed action if a transaction exceeds a specified period of time. When a transaction exceeds the specified time limit , the Broker automatically performs a commit or roll back for the transaction. Whether the Broker performs a commit or roll back depends on how you have configured the transaction timeout options. Transactions that expire and are completed by the Broker are considered to be heuristically completed, meaning that the decision to perform a commit or roll back did not come from the client. As required by the XA Specification, the Broker maintains a record of heuristically completed transactions in a log.

5. Managing Queues

This managing queues describes how to view the list of documents in a queue and examine the content of the documents themselves, also explains how to delete documents from a queue, move documents from one queue to another, and insert documents into a queue. A queue contains the published documents to which a client subscribes. A document remains in the queue until the client retrieves it (and acknowledges that it has retrieved the document successfully) or until the document expires. Each client has one queue, which is part of its client-state object. Each queue has a storage type which determines whether documents in the queue are saved in local memory (volatile storage) or are saved to disk (guaranteed storage). The storage type for a queue also dictates the lifecycle of the client. The lifecycle determines whether the Broker maintains state information for a client, including queue contents, after the client disconnects.My webMethods maintains statistics for a queue, including:

The total number of documents in the queue, including the number of documents available for delivery and the number of unacknowledged documents in the queue

The last time a document was placed in the queue

The last time a client retrieved a document from the queue

The size of the queue

You might want to take corrective or preventive action based on the queue statistics. For example, if the statistics indicate that the queue contains a large number of documents and it is been a long time since a client program retrieved a document, you might want to verify that the client program is running properly.If you want to view the contents of a client’s queue, you can browse the queue. When you browse a queue, you retrieve a list of the documents in the queue. For each document, My webMethods specifies the acknowledgement status of the document, the document size, and the document’s sequence number.

6. Conclusion

Based on the above content, this paper proposes Broker Security, which describes how webMethods Broker security works. It describes the Broker security model, and explains how to configure SSL for Broker. The usage and set up of keystore files and Access Control Lists, which are key elements of the Broker security model, are covered in detail. The chapter also covers the Broker utilities used to implement security, and the step-by-step procedures for implementing all aspects of Broker SSL security, so the paper proposes the Broker Security Model.

6.1 Broker Security Model

Broker installation as well as Broker data. The security model uses Secure Sockets Layer (SSL) to:

Authenticate Broker Servers and clients. Authentication is the process of validating the identity of an entity attempting to establish a connection. The Broker security model provides reference

integrity through SSL, guaranteeing the identity of a Broker Server to a requesting client, and optionally, that of a client to a Broker Server.

Authorize administrative access to a Broker Server and client access to Broker objects and data.

Authorization is the process of granting (or denying) access permissions. The Broker security model authorizes access permissions by comparing a client's SSL identity against a list of identities contained in an Access Control List (ACL). If the client's SSL identity matches one of those listed, it is granted permission; if not, it is denied permission.

Allow secure sharing of data between Brokers in Broker territories or connected by a Brokergateway. In the Broker security model, you can also set access permissions to control which Brokers can join a territory, and whether a Broker in another territory connected by a gateway is allowed to access data.

6.2 Securing Broker System Components

- Broker Server
- Clients, including the Broker user interface and Broker command-line utilities
- Remote Broker gateway
- Remote Brokers within territories
- Securing the Broker Server
- Securing the Broker User Interface
- Managing Territories
- Managing Gateways
- Configuring the Forwarding Mode
- Using My webMethods with JMS

My webMethods provides a broad range of capabilities for Broker metadata export and import, allowing you to save entire hierarchies of Broker objects and their relationships in XML files. You can import that data in whole or part to other locations in your Broker network. webMethods Broker consists of two main components: Broker Server, the run-time component with which publishers and subscribers interact, and the Broker user interface, the administrative component that runs on My webMethods Server.Broker Monitor monitors all of the Broker Servers running on the machine where it is installed. It will automatically attempt to restart any Broker Server that stops running. In solutions based on this model, applications that produce information send the information to the broker entity and applications that require the information connect to the broker and retrieve the information they need.

References

[1] Gregor Hohpe' Bobby Woolf. Enterprise Integration Patterns Designing Buildings and Deploying Messaging Solutions, America: Addison Wesley. 2003.

[2] Dai Lusi, Liao Wen?het Tian hong elai.Research of B2B E-commerce System Model Based on Web Service.Transactions of Nanjing University of Aeronautics& Astronautics, 2003' (7):

118-122.

[3] Web Methods Modeler User's Guide. Web Methods, Inc, 2006.