[MS-MQBR]:
Message Queuing (MSMQ):
Binary Reliable Messaging Algorithm

Revision Summary

Date	Revision History	Revision Class	Comments
07/20/2007	0.1	Major	MCPP Milestone 5 Initial Availability
09/28/2007	1.0	Major	Updated and revised the technical content.
10/23/2007	1.0.1	Editorial	Revised and edited the technical content.
11/30/2007	2.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
01/25/2008	2.0.1	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction 4
1.1 Glossary 4
1.2 References 5
1.2.1 Normative References 5
1.2.2 Informative References 5
1.3 Protocol Overview (Synopsis) 5
1.3.1 Direct Connection 6
1.3.2 Intra-Site Routing 6
1.3.3 Inter-Site Routing 6
1.4 Relationship to Other Protocols 7
1.5 Prerequisites/Preconditions 8
1.6 Applicability Statement 8
1.7 Versioning and Capability Negotiation 8
1.8 Vendor-Extensible Fields 8
1.9 Standards Assignments 8

2 Messages 9
2.1 Message Syntax 9

3 Protocol Details 10
3.1 Algorithm Details 10
3.1.1 Abstract Data Model 10
3.1.1.1 MachineRecordTable 10
3.1.1.2 SiteRecordTable 11
3.1.1.3 RoutingLinkRecordTable 11
3.1.1.4 Routing Table 11
3.1.2 Timers 12
3.1.2.1 MSMQ Directory Query Timer 12
3.1.3 Initialization 12
3.1.3.1 RoutingTable Initialization 12
3.1.4 Higher-Layer Triggered Events 13
3.1.5 Message Processing Events and Sequencing Rules 13
3.1.5.1 GetNextHop 13
3.1.5.2 GetNextHopForRouter 14
3.1.5.3 GetNextHopForSiteGate 14
3.1.5.4 GetMachineRecord 15
3.1.5.5 GetRoutingServer 16
3.1.5.6 GetSiteGate 16
3.1.5.7 GetSiteGateForSite 17
3.1.5.8 IsSiteGate 17
3.1.6 Timer Events 18
3.1.6.1 MSMQ Directory Query Timer Event 18
3.1.7 Other Local Events 18

4 Protocol Examples 19

5 Security 20
5.1 Security Considerations for Implementers 20
5.2 Index of Security Parameters 20

6 Appendix A: Windows Behavior 21

7 Index 23

1 Introduction

The Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm is used by **Message Queuing** (also known as MSMQ) to communicate across both **connected networks** and heterogeneous networks.

This document specifies how the **message** flow occurs within an **enterprise** to guarantee message delivery and efficient routing. The document describes **intra-site routing**, **inter-site routing** and direct connection between source and destination **nodes** where it is possible, as specified in section 1.3.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Active Directory (AD)
Administrator
Directory Service (DS)
Globally Unique Identifier (GUID)

The following terms are defined in [MS-MQMQ]:

Connected Network
Direct Format Name
Enterprise
Message
Message Queue
Microsoft Message Queuing (MSMQ)
Message Queuing Information Store (MQIS)
MSMQ Directory Service
MSMQ Routing Server
MSMQ Site
MSMQ Site Gate
MSMQ Site Link
Queue
Queue Manager

The following terms are specific to this document:

In-Routing Server: An **MSMQ Routing Server** that receives all **messages** on behalf of a particular client and forwards those **messages** to that client.

Inter-site Routing: The process of routing a **message** between different **MSMQ Sites** within an **Enterprise**.

Intra-Site Routing: The process of routing a **message** within a single **MSMQ Site**.

Node: A computer listed in the **MSMQ Directory Service** that has a **Queue Manager** running.

Out-Routing Server: An **MSMQ Routing Server** that receives all **messages** sent by a particular client and routes those **messages** on behalf of that client.

Routing Link: An **MSMQ Site Link**.

Routing Link Cost: A value that models the relative cost of direct communication between two **MSMQ Sites**.

Routing Table: A table maintained by each **MSMQ Site Gate** for **Inter-site Routing**. For each **MSMQ Site** in an **Enterprise**, the table specifies the **MSMQ Site** to which a **Message** should be forwarded in order to minimize the total **Routing Link Cost** for that **Message**.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification", June 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-MQDS] Microsoft Corporation, "Message Queuing (MSMQ): Directory Service Protocol Specification", July 2007.

[MS-MQMQ] Microsoft Corporation, "Message Queuing (MSMQ): Data Structures", August 2007.

[MS-MQQB] Microsoft Corporation, "Message Queuing (MSMQ): Message Queuing Binary Protocol Specification", August 2007.

1.2.2 Informative References

[LDAP] Microsoft Corporation, "About Lightweight Directory Access Protocol", http://msdn2.microsoft.com/en-us/library/aa366075.aspx

If you have any trouble finding [LDAP], please check here.

[MSDN-MSMQ] Microsoft Corporation, "Message Queuing (MSMQ)", http://msdn2.microsoft.com/en-us/library/ms711472.aspx

[RFC1] Cormen, T. H., Leiserson, C. E., and Rivest, R.L., "Introduction to Algorithms".

1.3 Protocol Overview (Synopsis)

The Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm describes message routing within an enterprise <1> network.

Message queuing clients transfer messages either by direct connection with the destination or by sending to an **MSMQ Routing Server**.<2> If a direct connection is not possible or the client is configured to use a routing service, MSMQ routing servers can temporarily store messages and subsequently forward them to the destination node or to another MSMQ routing server.

Message routing occurs when one or more of the following conditions exist:

- The nodes belong to different **sites**.
- The nodes do not share a connected network.

- The source node is configured to use an **out-routing server**.
- The destination node is configured to use an **in-routing server**.
- An attempt to send a message via direct connection fails, for example, if the destination node is offline.

More information on queuing messages is specified in [MS-MQQB].

1.3.1 Direct Connection

A direct connection between two nodes that share a connected network is possible when the source node is not configured to use one or more out_routing_server(s), and the destination node is not configured to use one or more in-routing server(s). A node may belong to more than one connected network.

1.3.2 Intra-Site Routing

If a source node is configured to use an out-routing server, every outgoing message is routed through that out-routing server. Similarly, if a destination node is configured to use an in-routing server, every incoming message is routed through that in-routing server. Using in-routing and out-routing servers to route messages within an MSMQ site may reduce network bandwidth consumption by providing session concentration.

An MSMQ routing server may also be used to exchange messages between two nodes within an MSMQ site that do not share a common connected network.

An MSMQ routing server may also be used to exchange messages between two nodes within an MSMQ site when direct connection between those nodes fails.

1.3.3 Inter-Site Routing

The **MSMQ Directory Service** <3> enables administrators to model some aspects of the physical topology of an enterprise. The Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm uses this model to make inter-site routing decisions.

MSMQ Sites represent a grouping of nodes in the enterprise network according to physical location. Nodes in one MSMQ site use **MSMQ site gates** within the same MSMQ site to route messages to nodes in other MSMQ sites. An MSMQ site gate can route a message to another MSMQ site by sending that message to another MSMQ site gate.

Routing inter-site traffic only through MSMQ site gates often results in session concentration, which can reduce network bandwidth consumption between physically distant nodes.

If an enterprise network has more than one MSMQ site, an administrator creates **routing links** to allow messages to be routed between those MSMQ sites. Routing links identify neighboring MSMQ sites whose MSMQ site gates can communicate directly. Each routing link includes a **cost** that represents how expensive it is to transfer messages directly between the two sites.

A message may be transferred through multiple MSMQ sites on the way to the destination MSMQ site. Each MSMQ site gate along the way uses a **routing table** to find the next hop in a least-cost path to the destination MSMQ site.

To build the routing table, MSMQ site gates consider the enterprise as a graph with vertices as MSMQ sites and bi-directional nonnegative edge weights as the routing link costs. An MSMQ site

gate builds a least-cost spanning tree using its MSMQ site as the root and uses this tree to populate its routing table.

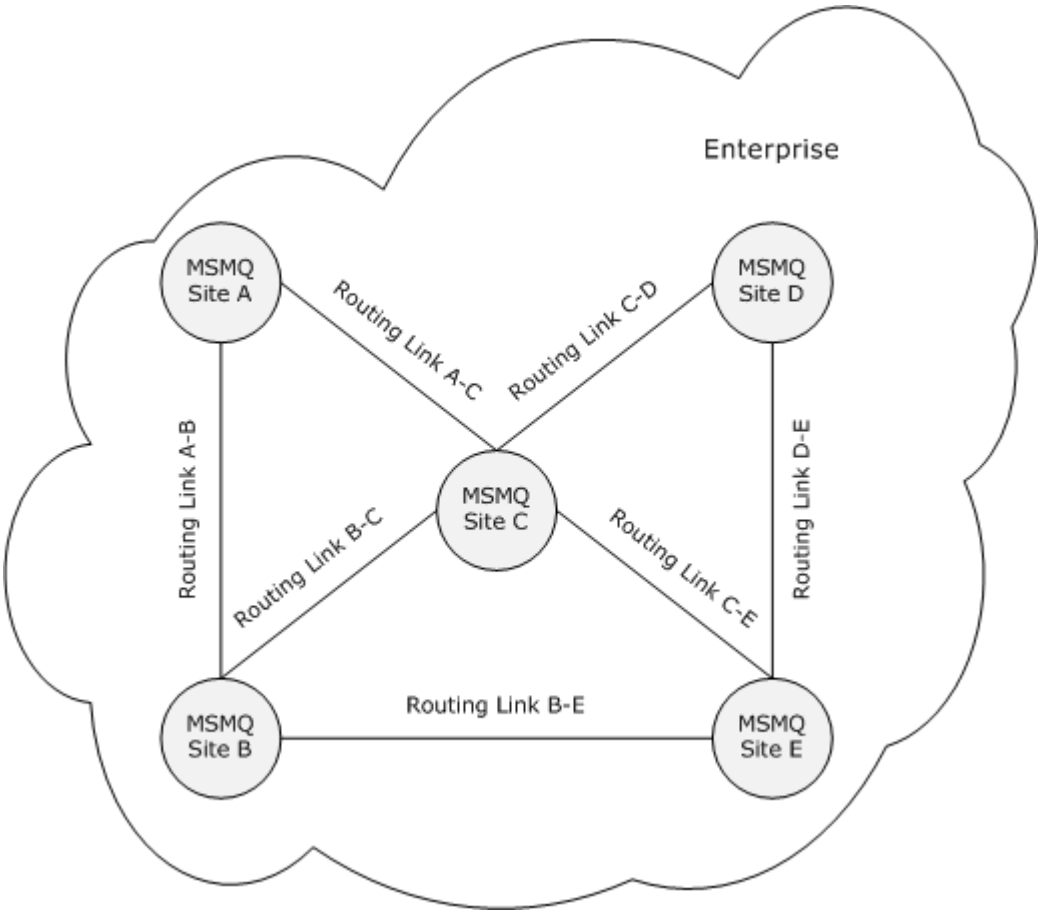

Figure 1: The enterprise as a set of MSMQ sites and routing links

Routing link cost provides a mechanism for administrators to enforce one route over another in cases where multiple routes exist.

As described in section 3.1.6.1, each computer that runs MSMQ within an enterprise periodically queries the MSMQ Directory Service to determine if it should act as an MSMQ site gate, and to build a routing table if the querying computer is an MSMQ site gate. More information on MSMQ Directory Service objects and their properties is specified in [MS-MQMQ].

1.4 Relationship to Other Protocols

The Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm MAY rely upon [LDAP] (as specified in [MS-ADTS]) or on the Directory Service protocol (as specified in [MS-MQDS]) to look up persistently stored entries in the MSMQ Directory Service storage.<4>

The Message Queuing (MSMQ): Binary Reliable Messaging Protocol, as specified in [MS-MQQB], uses the Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm in order to determine the next hop for a message on its way to the destination **message queue**.

1.5 Prerequisites/Preconditions

The MachineRecordTable, SiteRecordTable, and RoutingLinkRecordTable MUST be initialized by an administrator. For more details, see section 3.1.1.

If an enterprise has more than one MSMQ site, each MSMQ site has one or more MSMQ site gates assigned to it.

The following are the requirements for a node to be an MSMQ site gate:

- The node must belong to the MSMQ site for which it is an MSMQ site gate.
- The node must be able to connect directly to each MSMQ site gate in each neighboring MSMQ site.

MSMQ routing servers within an MSMQ site must be able to communicate on all connected networks used by the nodes within that MSMQ site.

1.6 Applicability Statement

The Message Queuing (MSMQ): Binary Reliable Messaging Algorithm is applicable when an enterprise includes more than one MSMQ site.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Message Syntax

None.

3 Protocol Details

3.1 Algorithm Details

3.1.1 Abstract Data Model

This section specifies a data model that can be used to implement the Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm. The specified organization is provided to facilitate the explanation of how the algorithm behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that specified in this document.

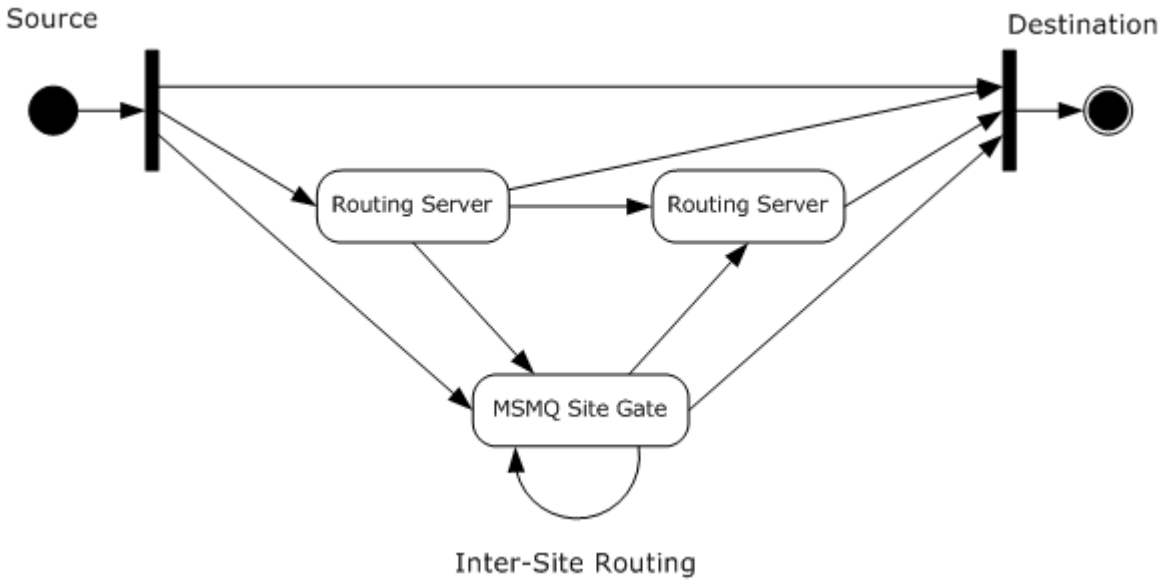

Figure 2: Message Routing

This diagram represents all possible paths that a message may take through an MSMQ enterprise. Each box represents a class of machines participating in message transfer. Each arrow represents a transfer of a message from a machine in the class at the tail of the arrow to a machine in the class at the head of the arrow. In this diagram, the only arrow that represents a message transfer between machines in different MSMQ sites is the 'Inter-Site Routing' arrow between two MSMQ site gates. All other arrows represent message transfer between machines within the same MSMQ site.

To perform message routing, the MSMQ: Binary Reliable Messaging Algorithm uses information from the MSMQ Directory Service. This information describes the enterprise network where MSMQ operates.

3.1.1.1 MachineRecordTable

The MachineRecordTable represents all nodes running MSMQ in the enterprise. An entry in the MachineRecordTable is called a MachineRecord. The following table lists the properties of the MachineRecord.

Name	Meaning
QM_SITE_ID	A **GUID** that identifies an MSMQ site in which the machine is located.
QM_MACHINE_ID	A GUID that identifies the machine.
QM_CNS	A vector of GUIDs that identifies the connected networks for a given machine.
QM_OUTFRS	A vector of GUIDs that identifies the out-routing server of the machine.
QM_INFRS	A vector of GUIDs that identifies the in-routing server of the machine.
QM_SERVICE_ROUTING	An integer that indicates if the machine is or is not configured as an MSMQ routing server. 0 - the machine is not an MSMQ routing server; 1 - the machine is an MSMQ routing server.

3.1.1.2 SiteRecordTable

The SiteRecordTable represents all MSMQ sites in the enterprise.

An entry in the SiteRecordTable is called a SiteRecord. The following table lists the properties of the SiteRecord.

Name	Meaning
S_SITEID	A GUID that identifies the MSMQ site.
S_GATES	A vector of GUIDs that identifies the MSMQ site gates of the MSMQ site identified by S_SITEID.

3.1.1.3 RoutingLinkRecordTable

The RoutingLinkRecordTable represents all **MSMQ site links** in the enterprise.

An entry in the RoutingLinkRecordTable is called a RoutingLinkRecord. The following table lists the properties of the RoutingLinkRecord.

Value	Meaning
L_NEIGHBOR1	A GUID that identifies the first of two neighboring MSMQ sites.
L_NEIGHBOR2	A GUID that identifies the second of two neighboring MSMQ sites.
L_COST	An integer that indicates the routing link cost between the two neighboring MSMQ sites.

3.1.1.4 Routing Table

The RoutingTable contains a mapping from the unique identifier of the destination MSMQ site to the unique identifier of the next hop MSMQ site on the least-cost path to the destination MSMQ site.

Name	Description
DestinationSiteID	A GUID that identifies the destination MSMQ site.

Name	Description
NextHopSiteID	A GUID that identifies the next hop MSMQ site.

3.1.2 Timers

The Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm SHOULD maintain the following timer.

3.1.2.1 MSMQ Directory Query Timer

This timer fires the MSMQ Directory Query Timer Event, as described in section 3.1.6.1.

3.1.3 Initialization

The MachineRecordTable<5>, SiteRecordTable<6>, and RoutingLinkRecordTable<7> are initialized by the **administrator**.

The MSMQ Directory Query Timer SHOULD<8> be started.

3.1.3.1 RoutingTable Initialization

If IsSiteGate returns 0 when called with the GUID that identifies the node initializing its RoutingTable, the RoutingTable is initialized to be empty.

Otherwise, if the RoutingLinkRecordTable or the SiteRecordTable is empty when initializing the RoutingTable, the RoutingTable is initialized to be empty.

Otherwise, to populate the RoutingTable, consider an enterprise as a connected, nonnegative, weighted, directed graph E with vertices S and directed edges L, as follows:

E = (S, L).

Each vertex in S represents an MSMQ site. Each edge in L represents one direction of a RoutingLinkRecord between two MSMQ sites. That is, for any two MSMQ sites x and y, the directed edge (x, y) exists if and only if a RoutingLinkRecord exists where x is equal to the L_NEIGHBOR1 property and y is equal to the L_NEIGHBOR2 property, or where x is equal to the L_NEIGHBOR2 property and y is equal to the L_NEIGHBOR1 property.

Associated with each edge (x, y) is a weight that is equal to the L_COST value of the corresponding RoutingLinkRecord.

The cost of a path between two vertices from S in graph E is a sum of costs of all of the edges in that path. The least-cost path between any two vertices from S in graph E is the path with the lowest cost.

Given such a mapping of an enterprise to a directed graph, discovering the least-cost path between two MSMQ sites corresponds exactly to finding the least-cost path through a directed graph. Dijkstra's algorithm<9> can be used to find least-cost paths to each destination site by finding a spanning tree that covers the graph. The Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm populates the RoutingTable with destination MSMQ sites in DestinationSiteID and the corresponding next hop MSMQ site in NextHopSideID from the spanning tree.

When for a given pair of source and destination vertices there are two or more routes with equal cost, the algorithm selects one of them.

3.1.4 Higher-Layer Triggered Events

There are no higher-layer triggered events.

3.1.5 Message Processing Events and Sequencing Rules

This section describes how a node uses information in the MachineRecordTable, SiteRecordTable, RoutingLinkRecordTable, and RoutingTable to determine the next hop for a message.

3.1.5.1 GetNextHop

GetNextHop defines the algorithm that a node uses to determine the next hop for a message to get from the current node (identified by SourceID) to the ultimate destination (identified by DestinationID). The return value is a GUID that identifies the node that is to be used as the next hop.

```
GetNextHop(SourceID of type GUID, DestinationID of type GUID)

;SourceID      - GUID that identifies the source machine
;DestinationID - GUID that identifies the destination machine

INIT NextHop of type GUID                ;Next hop identifier
INIT SourceMachine of type MachineRecord
INIT DestinationMachine of type MachineRecord

SET NextHop to Nothing
SET SourceMachine to result of CALL GetMachineRecord(SourceID)
SET DestinationMachine to result of
    CALL GetMachineRecord(DestinationID)

IF SourceMachine <> Nothing AND DestinationMachine <> Nothing THEN
   IF SourceMachine.QM_SERVICE_ROUTING = 1 THEN
      SET NextHop to result of GetNextHopForRouter(SourceMachine,
                                                   DestinationMachine)
   ELSE
      IF SourceMachine.QM_OUTFRS is not empty THEN
         SET NextHop to one of the SourceMachine.QM_OUTFRS
      ELSE
         IF intersection of SourceMachine.QM_CNS AND
            DestinationMachine.QM_CNS is not empty THEN
            IF DestinationMachine.QM_INFRS is not empty THEN
               SET NextHop to one of the Destination.QM_INFRS
            ELSE
               SET NextHop to DestinationID
            ENDIF
         ELSE
            SET NextHop to result of CALL GetRoutingServer(SourceID)
         ENDIF
      ENDIF
   ENDIF
ENDIF

IF NextHop = Nothing
   Raise Exception   ; Routing Attempt Fails
ENDIF
```

```
RETURN with NextHop
```

3.1.5.2 GetNextHopForRouter

GetNextHopForRouter returns a GUID that identifies the node that is to be used as the next hop on success, or Nothing on failure.

```
GetNextHopForRouter(SourceMachine of type MachineRecord,
                    DestinationMachine of type MachineRecord)

INIT NextHop of type GUID         ;Next hop identifier
INIT IsSourceSiteGate of type Boolean

SET IsSourceSiteGate to result of
    CALL IsSiteGate(SourceMachine.QM_MACHINE_ID)

IF IsSourceSiteGate = 1 THEN
     SET NextHop to result of
         CALL GetNextHopForSiteGate(SourceMachine, DestinationMachine)
ELSE IF SourceMachine.QM_MACHINE_ID is one of
     DestinationMachine.QM_INRFS THEN
     SET NextHop to DestinationMachine.QM_MACHINE_ID
ELSE IF SourceMachine.QM_SITE_ID = DestinationMachine.QM_SITE_ID THEN
        IF DestinationMachine.QM_INRFS is not empty THEN
           SET NextHop to one of DestinationMachine.QM_INFRS
        ELSE
           SET NextHop to DestinationMachine.QM_MACHINE_ID
        ENDIF
ELSE
   SET NextHop to return of
       CALL GetSiteGate(SourceMachine.QM_MACHINE_ID)
ENDIF

RETURN with NextHop
```

3.1.5.3 GetNextHopForSiteGate

GetNextHopForSiteGate returns a GUID that identifies the node that is to be used as the next hop on success, or Nothing on failure.

```
GetNextHopForSiteGate(SourceMachine of type MachineRecord,
                      DestinationMachine of type MachineRecord)

INIT NextHop of type GUID         ;Next hop identifier

IF SourceMachine.QM_MACHINE_ID
   is one of DestinationMachine.QM_INFRS
THEN
   SET NextHop to DestinationMachine.QM_MACHINE_ID
```

```
ELSE IF SourceMachine.QM_SITE_ID = DestinationMachine.QM_SITE_ID THEN
   IF DestinationMachine.QM_INRFS is not empty THEN
      SET NextHop to one of DestinationMachine.QM_INFRS
   ELSE
      SET NextHop to DestinationMachine.QM_MACHINE_ID
   ENDIF
ELSE
   INIT NextSite of type GUID
   INIT Entry of type RECORD of RoutingTable

   SET NextSite to Nothing

   FOREACH Entry FROM RoutingTable DO
      IF Entry.DestinationSiteID = DestinationMachine.QM_SITE_ID THEN
         SET NextSite = Entry.NextHopSiteID
      ENDIF
   END FOREACH

   IF NextSite <> Nothing THEN
      SET NextHop to return of CALL GetSiteGateForSite(NextSite)
   ENDIF
ENDIF

RETURN with NextHop
```

3.1.5.4 GetMachineRecord

GetMachineRecord returns the MachineRecord from MachineRecordTable, where MachineID identifies the requested machine if successful; otherwise, GetMachineRecord returns Nothing.

```
GetMachineRecord(MachineID of type GUID)

;MachineID - GUID that identifies the machine
;corresponding to the requested MachineRecord

INIT Machine of type MachineRecord
INIT MachineTemp of type MachineRecord

SET Machine to Nothing

FOREACH MachineTemp FROM MachineRecordTable DO
   IF MachineTemp.QM_MACHINE_ID = MachineID THEN
        Machine = MachineTemp;
        BREAK FOREACH
   ENDIF
END FOREACH

RETURN with Machine
```

3.1.5.5 GetRoutingServer

Given a GUID identifying a machine, GetRoutingServer returns the GUID identifying an MSMQ routing server within the MSMQ site to which that machine belongs, or Nothing if there is none.

```
GetRoutingServer(MachineID of type GUID)
;MachineID - GUID that identifies the machine that requested
;an MSMQ routing server

INIT SourceMachine of type MachineRecord
INIT MachineTemp of type MachineRecord
INIT Router type of GUID

SET SourceMachine to result of CALL GetMachineRecord(MachineID)
SET Router to Nothing

FOREACH MachineTemp FROM MachineRecordTable DO
   IF MachineTemp.QM_SITE_ID = SourceMachine.QM_SITE_ID  AND
      MachineTemp.QM_SERVICE_ROUTING = 1 THEN
        SET Router to MachineTemp.QM_MACHINE_ID
        BREAK FOREACH
   ENDIF
END FOREACH

RETURN with Router
```

3.1.5.6 GetSiteGate

Given a GUID identifying a machine, GetSiteGate returns the GUID identifying an MSMQ site gate within the MSMQ site to which that machine belongs, or Nothing if there is none.

```
GetSiteGate(MachineID of type GUID)

;MachineID – GUID that identifies the machine

INIT SourceMachine of type MachineRecord
INIT SiteGate of type GUID
INIT SiteTemp of type SiteRecord

SET SourceMachine to result of CALL GetMachineRecord(MachineID)
SET SiteGate to Nothing

FOREACH SiteTemp FROM SiteRecordTable DO
   IF SiteTemp.S_SITEID = SourceMachine.QM_SITE_ID THEN
        SET SiteGate to one of SiteTemp.S_GATES
        BREAK FOREACH
   ENDIF
END FOREACH

RETURN with SiteGate
```

3.1.5.7 GetSiteGateForSite

Given a GUID identifying an MSMQ site, GetSiteGateForSite returns the GUID identifying an MSMQ site gate within that MSMQ site, or Nothing if there is none.

```
GetSiteGateForSite(SiteID of type GUID)

;SiteID  - GUID that identifies MSMQ site

INIT SiteGate of type GUID
INIT SiteTemp of type SiteRecord

SET SiteGate to Nothing

FOREACH SiteTemp FROM SiteRecordTable DO
   IF SiteTemp.S_SITEID = SiteID THEN
        SET SiteGate to one of SiteTemp.S_GATES
        BREAK FOREACH
   ENDIF
END FOREACH

RETURN with SiteGate
```

3.1.5.8 IsSiteGate

Given a GUID identifying a node, IsSiteGate returns 1 if the node acts as an MSMQ site gate; otherwise, it returns 0.

```
IsSiteGate(MachineID of type GUID)

;MachineID – GUID that identifies the machine

INIT Machine of type MachineRecord
INIT Site of type SiteRecord

SET Machine to result of CALL GetMachineRecord(MachineID)

IF Machine.QM_SERVICE_ROUTING = 0 THEN
   RETURN with 0
ENDIF

FOREACH Site FROM SiteRecordTable DO
   IF Site.S_SITEID = Machine.QM_SITE_ID  AND
      Site.S_GATES has MachineID  THEN
        RETURN with 1
   ENDIF
END FOREACH

RETURN with 0
```

3.1.6 Timer Events

3.1.6.1 MSMQ Directory Query Timer Event

When this timer fires, a **queue manager** reinitializes its RoutingTable as specified in section 3.1.3.1.

3.1.7 Other Local Events

There are no local events.

4 Protocol Examples

The Message Queuing (MSMQ): Binary Reliable Message Routing Algorithm calculates inter-site routing based on the routing link costs.

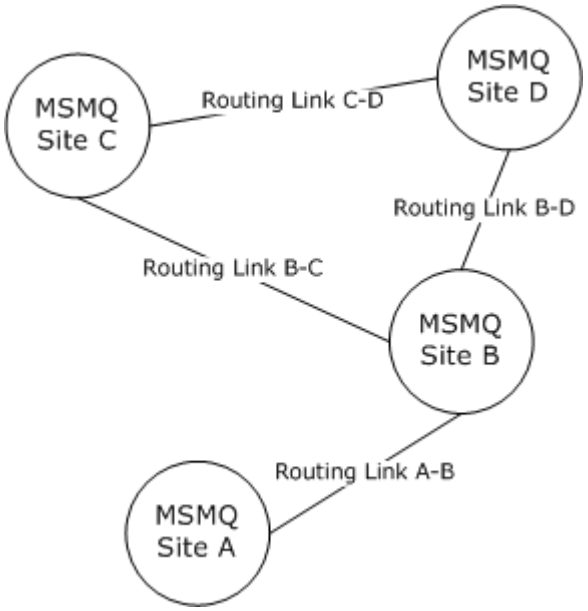

Figure 3: An Enterprise Example

If the cost associated with the routing link A-B is 3, and the cost associated with the other three routing links B-C, B-D, and C-D is 1, then messages routed from A to C always travel from A to B and then from B to C. However, if the cost associated with routing links A-B and B-C is 3, and the cost associated with routing links C-D and B-D is 1, then messages routed from A to C always travel from A to B, from B to D, and then from D to C.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows NT
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 1.3: Windows only supports cross-enterprise message routing by use of **direct format names**. Messages whose destinations are specified by direct format names are never routed, but are instead sent directly from the source to the destination. For more information on cross-enterprise message routing, see [MSDN-MSMQ].

<2> Section 1.3: Only Windows NT, Windows 2000, Windows Server 2003, and Windows Server 2008 can be configured as MSMQ routing servers.

<3> Section 1.3.3: Windows NT employs the **Message Queuing Information Store (MQIS)** database, which runs on top of SQL Server as an MSMQ Directory Service. In Windows 2000, Windows XP, Windows Server 2003, and Windows Vista, the MSMQ Directory Service is exposed by **Active Directory (AD)**.

<4> Section 1.4: Windows NT Server contains a directory service that implements the server side of the [MS-MQDS] protocol. All other versions of Windows Server use Active Directory and implement the server side of this protocol as a pass-through to Active Directory.

All versions of Windows, except Windows Vista, implement the client side of the [MS-MQDS] protocol. Windows XP and Windows Server 2003 use [LDAP] to Active Directory when it is available; otherwise, they use [MS-MQDS]. Windows Vista always uses [LDAP].

<5> Section 3.1.3: Windows uses the MSMQ Directory Service to implement the MachineRecordTable. The identifiers and schema of the data in the MSMQ Directory Service are specified in [MS-MQMQ]. The mapping of a MachineRecord to the corresponding machine property of the MSMQ Directory Service is as follows:

MachineRecord	Machine property identifier in the MSMQ Directory Service
QM_SITE_ID	PROPID_QM_SITE_ID ([MS-MQMQ] section 2.3.2.1)
QM_MACHINE_ID	PROPID_QM_MACHINE_ID ([MS-MQMQ] section 2.3.2.2)
QM_CNS	PROPID_QM_CNS ([MS-MQMQ] section 2.3.2.5)
QM_OUTFRS	PROPID_QM_OUTFRS ([MS-MQMQ] section 2.3.2.6)

MachineRecord	Machine property identifier in the MSMQ Directory Service
QM_INFRS	PROPID_QM_INFRS ([MS-MQMQ] section 2.3.2.7)
QM_SERVICE_ROUTING	Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008: PROPID_QM_SERVICE_ROUTING ([MS-MQMQ] section 2.3.2.20). Windows NT: PROPID_QM_SERVICE ([MS-MQMQ] section 2.3.2.8), and the following values: 0: The computer is not configured as an MSMQ Routing Server; 1: The computer is configured as an MSMQ Routing Server.

The MachineRecordTable is considered populated if and only if the MSMQ Directory Service query succeeds; otherwise, the MachineRecordTable is considered empty.

<6> Section 3.1.3: Windows uses the MSMQ Directory Service to implement the SiteRecordTable. The identifiers and schema of the data in the MSMQ Directory Service are specified in [MS-MQMQ]. The mapping of a SiteRecord to the corresponding site property of the MSMQ Directory Service is as follows:

SiteRecord	Site property identifier in the MSMQ Directory Service
S_SITEID	PROPID_S_SITEID ([MS-MQMQ] section 2.3.3.2)
S_GATES	PROPID_S_GATES ([MS-MQMQ] section 2.3.3.3)

The SiteRecordTable is considered populated if and only if the MSMQ Directory Service query succeeds; otherwise, the SiteRecordTable is considered empty.

<7> Section 3.1.3: Windows uses the MSMQ Directory Service to implement the RoutingLinkRecordTable. The identifiers and schema of the data in the MSMQ Directory Service are described in [MS-MQMQ]. The mapping of a RoutingLinkRecord to the corresponding site link property of the MSMQ Directory Service is as follows:

RoutingLinkRecord	Site Link property identifier in the MSMQ Directory Service
L_NEIGHBOR1	PROPID_ L_NEIGHBOR1 ([MS-MQMQ] section 2.3.7.1)
L_NEIGHBOR2	PROPID_ L_NEIGHBOR2 ([MS-MQMQ] section 2.3.7.2)
L_COST	PROPID_L_COST ([MS-MQMQ] section 2.3.7.3)

The RoutingLinkRecordTable is considered populated if and only if the MSMQ Directory Service query succeeds; otherwise, the RoutingLinkRecordTable is considered empty.

<8> Section 3.1.3: Windows sets the timer to 3600 seconds.

<9> Section 3.1.3.1: Windows uses Dijkstra's algorithm to find the least-cost path from a single source vertex to any other vertex in a nonnegative, weighted, directed graph, as specified in [RFC1].

7 Index

A

Abstract data model - algorithm
Algorithm
  abstract data model
  higher-layer triggered events
  initialization (section 3.1.3, section 3.1.3.1)
  local events
  message processing
  overview
  sequencing rules
  timer events
  timers
Applicability

C

Capability negotiation

D

Data model - abstract -algorithm
Direct connection

E

Examples - overview

F

Fields - vendor-extensible

G

GetMachineRecord
GetNextHop
GetNextHopForRouter
GetNextHopForSiteGate
GetRoutingServer
GetSiteGate
GetSiteGateForSite
Glossary

H

Higher-layer triggered events - algorithm

I

Implementer - security considerations
Index of security parameters
Informative references
Initialization - algorithm (section 3.1.3, section 3.1.3.1)
Inter-site routing
Intra-site routing
Introduction
IsSiteGate

L

Local events - algorithm

M

MachineRecordTable
Message processing - algorithm
Messages
  overview
  syntax

N

Normative references

O

Overview (synopsis)

P

Parameters - security index
Preconditions
Prerequisites

R

References
  informative
  normative
  overview
Refresh timer - routing table
Refresh timer event - routing table
Relationship to other protocols
Routing table
  overview
  refresh timer
  refresh timer event
RoutingLinkRecordTable

S

Security
  implementer considerations
  overview
  parameter index
Sequencing rules - algorithm
SiteRecordTable
Standards assignments
Syntax

T

Timer events - algorithm
Timers - algorithm
Triggered events - higher-layer - algorithm

V

Vendor-extensible fields
Versioning

W

Windows behavior