[MC-MQDSRP]: Message Queuing (MSMQ): Directory Service Replication Protocol

Revision Summary

Date	Revision History	Revision Class	Comments
02/27/2009	0.01	Major	First Release.
04/10/2009	0.01.1	Editorial	Revised and edited the technical content.
05/22/2009	0.01.2	Editorial	Revised and edited the technical content.
07/02/2009	0.2	Minor	Updated the technical content.
08/14/2009	0.2.1	Editorial	Revised and edited the technical content.
09/25/2009	0.2.2	Editorial	Revised and edited the technical content.
11/06/2009	0.2.3	Editorial	Revised and edited the technical content.
12/18/2009	0.2.4	Editorial	Revised and edited the technical content.
01/29/2010	1.0	Major	Updated and revised the technical content.
03/12/2010	1.0.1	Editorial	Revised and edited the technical content.
04/23/2010	1.0.2	Editorial	Revised and edited the technical content.
06/04/2010	1.0.3	Editorial	Revised and edited the technical content.
07/16/2010	1.0.4	Editorial	Changed language and formatting in the technical content.
08/27/2010	1.0.4	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	2.0	Major	Significantly changed the technical content.
11/19/2010	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	3.0	Major	Significantly changed the technical content.
03/25/2011	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	3.0	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	3.1	Minor	Clarified the meaning of the technical content.
09/23/2011	3.1	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	4.0	Major	Significantly changed the technical content.

Date	Revision History	Revision Class	Comments
03/30/2012	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	5.0	Major	Significantly changed the technical content.
01/31/2013	5.1	Minor	Clarified the meaning of the technical content.

Contents

1 Introduction

This specification describes the Message Queuing (MSMQ): Directory Service Replication Protocol. **MSMQ** stores the network topology and objects in a **directory service**. When the directory service is maintained by MSMQ, the Message Queuing (MSMQ): Directory Service Replication Protocol is used by MSMQ **directory servers** to maintain a distributed and consistent data storage. The MSMQ: Directory Service Replication Protocol is a queued protocol that uses MSMQ as its transport infrastructure to send replications that are wrapped within MSMQ **messages**.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

Active Directory
Coordinated Universal Time (UTC)
directory service
globally unique identifier (GUID)
little-endian

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

Backup Site Controller (BSC)
connected network
enterprise
message
Microsoft Message Queuing (MSMQ)
MSMQ Directory Service
MSMQ Directory Service server
MSMQ routing link
Primary Enterprise Controller (PEC)
Primary Site Controller (PSC)
property identifier
queue
queue manager

The following terms are specific to this document:

directory change: An operation and the associated data that can be applied to a directory object.

directory client: The client of the Message Queuing (MSMQ): Directory Service Protocol that communicates with a **directory server**.

directory neighbor: Another **directory server** to which the current **directory server** propagates **directory changes**. The **PEC** and all **PSCs** in the **enterprise** are **directory neighbors** to each other. The **directory neighbors** of a **PSC** also include its **BSCs**, and a **BSC** does not have any **directory neighbor**.

directory partition: A collection of enterprise-wide directory objects or a collection of site-wide directory objects that share the same **object authority**.

directory replica: A **directory partition** whose **object authority** is not the hosting **directory server**.

directory server: See **MSMQ Directory Service server**.

Message Queuing (MSMQ): See **Microsoft Message Queuing (MSMQ)**.

MQDSRP protocol client: An MSMQ **directory server** that initiates the MQDSRP protocol to generate a replication **message** or request that is sent to another MSMQ **directory server**.

MQDSRP protocol server: An MSMQ **directory server** that handles a request that is initiated by an **MQDSRP protocol client** and if appropriate, provides a reply to that request.

MSMQ directory: See **MSMQ Directory Service**.

object authority: A **directory server** that can change (create, update, and delete) an object, as requested by a **directory client**. Each directory object has only one **directory server** that is the object authority for that object.

purged sequence number: A **sequence number** that indicates that any deleted objects whose **sequence number** is less than the **purged sequence number** have been purged.

replication queue: A private **Microsoft Message Queuing (MSMQ) queue** to which replication **messages** are sent and from which replication **messages** are received.

replication state: A collection of attributes that describe the **replication state** of a **directory partition** that is maintained by a **directory server**.

sequence number: A data type that is used by this protocol to (1) identify **directory changes**, (2) track the change history of an object (the last change to an object), and (3) maintain the **replication state** of a **directory partition**.

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

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

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".

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-MQDMPR] Microsoft Corporation, "Message Queuing (MSMQ): Common Data Model and Processing Rules".

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

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

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

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, http://www.ietf.org/rfc/rfc1321.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, http://www.ietf.org/rfc/rfc4234.txt

1.2.2 Informative References

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

[MS-MQOD] Microsoft Corporation, "Message Queuing Protocols Overview".

1.3 Overview

The MSMQ: Directory Service Replication Protocol (MQDSRP) defines a mechanism that is used by the **MSMQ Directory Service** to synchronize changes to directory objects between **MSMQ Directory Service servers**. To facilitate this functionality, MQDSRP uses the MSMQ: Message Queuing Binary Protocol as its transport protocol, employs MSMQ shared state and processing rules, and also uses MSMQ: Directory Service Protocol events to access directory data elements and functionalities.

Microsoft Message Queuing (MSMQ) is a communications service that provides asynchronous and reliable passing of messages between applications that run on different hosts. In MSMQ, applications send messages to **queues** and consume messages from queues. The queue provides persistence of the messages, which enables the messages to survive across application restarts. Queues also allow applications to send and receive messages asynchronously among each other.

Queues are typically hosted by a communications service that is called a **queue manager**. Because the queue manager is hosted on an MSMQ directory server as a separate service from applications, the applications can communicate—even if they do not execute at the same time—by exchanging messages via a queue that is hosted by the queue manager.

Because MSMQ involves messages passing between nodes, a directory service can be useful to MSMQ services in the following ways.

- A directory service can provide network topology information that the MSMQ services can use to route messages between nodes.
- A directory service can be used as a key distribution mechanism for security services that are used by MSMQ to secure messages and to authenticate message senders.
- A directory service can provide applications with discovery capabilities, which allow them to discover the queues that are available in the network.

- A directory service can contain collections of directory objects that represent **enterprises**, sites, **routing links**, machines, users, queues, **connected networks**, and deleted objects.

MSMQ can work with a directory service such as **Active Directory**, in which the directory service servers store the directory objects and manage object changes. MSMQ can also work with an MSMQ-specific directory service, referred to as an **MSMQ directory**, in which the MSMQ Directory Service servers store directory objects and also manage them by replicating **directory changes** to maintain consistency and correctness of the directory.

1.3.1 MSMQ Directory Overview

This section provides an overview of the MSMQ directory.

1.3.1.1 System Overview

The MSMQ directory contains the following types of directory servers, which fulfill an MSMQ Directory Service role that is played by an MSMQ queue manager:

- **Primary Enterprise Controller (PEC):** The **PEC** acts as the **object authority** for the enterprise configuration information that is stored in the directory. An enterprise has only one PEC. The PEC also acts in the role of **Primary Site Controller (PSC)** for the site to which it belongs.
- **Primary Site Controller (PSC):** The PSC acts as the object authority for the directory information of the site to which it belongs. The PSC may satisfy directory lookup requests and directory change requests for objects that belong to the same site. Each site has only one PSC.
- **Backup Site Controller (BSC):** A **BSC** contains a read-only copy of the directory. A BSC may satisfy directory lookup requests but cannot satisfy directory change requests. A site may have zero or more BSCs.

The PEC and the PSCs have direct connection with each other, and a PSC has direct connection with its BSCs. These servers serve directory requests from **directory clients** and replicate directory changes in the enterprise.

The following figure illustrates a typical topology for an MSMQ enterprise.

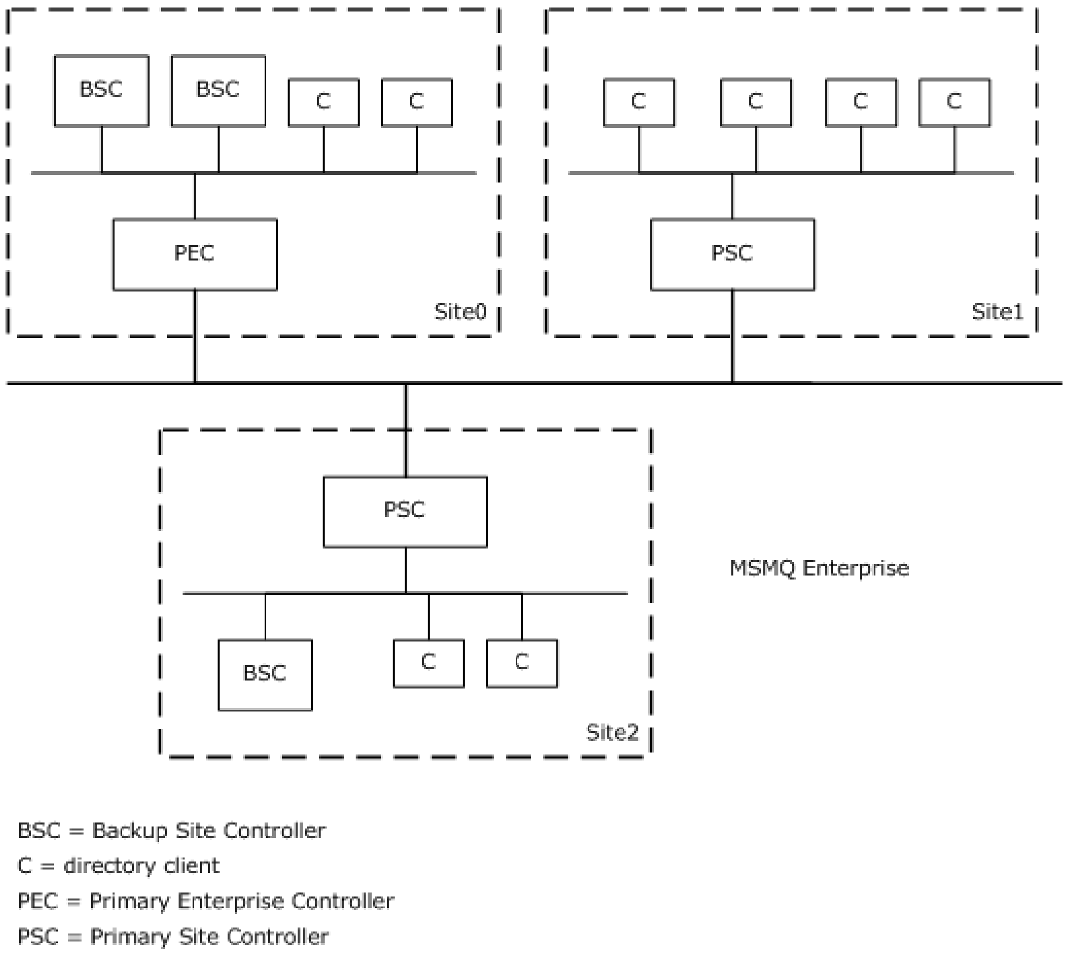
BSC
BSC
C
C
PEC
Site0
C
C
C
C
PSC
Site1
PSC
MSMQ Enterprise
BSC
C
C
Site2
BSC = Backup Site Controller
C = directory client
PEC = Primary Enterprise Controller
PSC = Primary Site Controller

Figure 1: Example of an MSMQ enterprise topology

Note The PEC is also the PSC of the initial site that is created in a typical MSMQ enterprise (see Site0 in the preceding figure). Therefore, in this specification, when a statement is made about a PSC, it also applies to the PEC. If a statement only applies to the PEC, the term PEC is explicitly called out.

The figure shows an enterprise that has three sites. Each site has one PSC (the PSC for Site0 is the PEC shown), and zero or more BSCs. Directory lookup and change requests from directory clients (marked "C" in the preceding figure) can be served by any directory server (the PSC or BSCs) of that site. However, a directory server cannot satisfy a directory change request if the server is not the authority for the object that is associated with the change. The server must forward the change request to the object authority, which applies the change to the specified object, and then the authority replies to the requester with the operation result.

Each directory server maintains a copy of all objects in the entire enterprise. When the object authority changes its objects, it propagates directory changes to its **directory neighbors**. When a neighbor receives the directory changes, it applies them to its **directory replica**. When an expected directory change is not received, a neighbor can ask the object authority for directory changes by sending a synchronization request.

1.3.1.2 Data Model

The MSMQ directory uses a single-authority data model: the lifetime of an object is managed by one and only one directory server. This server is referred to as the object authority. Only the object authority can change (create, update, and delete) its objects. Other directory servers forward change requests to the object authority and update their directory replica when they receive directory changes from the object authority.

The object authority for enterprise-wide objects (Enterprise, Site, RoutingLink, ConnectedNetwork, and User) is the Primary Enterprise Controller (PEC). The object authority for site-wide objects (Machine and Queue) is the PSC of that site. For the default (initial) site, the object authority is the PEC.

To offer efficient directory service lookup requests, each directory server maintains a local copy of the directory. The local data is partitioned according to the object authorities. Site-wide objects that share the same PSC as their authority form a partition, referred to as a site partition. The enterprise-wide objects form one specific partition, referred to as the enterprise partition, with the PEC as its authority.

A directory server maintains the **replication state** of each **directory partition** in its local directory data. For the MSMQ enterprise topology in the preceding topic, the following diagram shows how the data is partitioned and how the replication state is maintained for each partition by the PEC and the PSC of site 2.

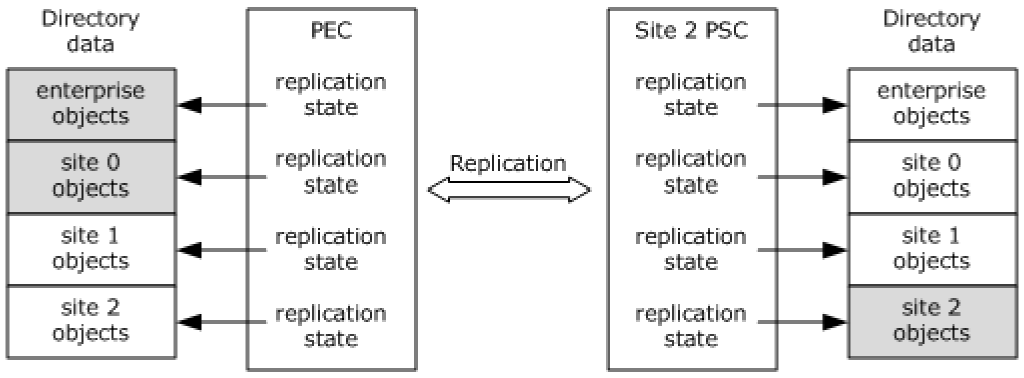

Figure 2: MSMQ directory data partitioning and replication

Using the site 2 PSC as an example, the queue manager is the authority of the site 2 partition (grayed box). It handles directory change requests to these objects, either directly from a directory client or indirectly from another directory server. After change operations are performed, the queue manager propagates the changes to its neighbors (the PEC in site 0, the PSC in site 1, and the BSC in site 2). The queue manager also receives directory changes from the PEC and the site 1 PEC for objects in the replica partitions (white boxes), and it updates the replica with these changes accordingly.

The replication state of each directory partition that is maintained by the queue manager is used for data replication among directory servers. When the replication state of each partition is the same across all directory servers, the directory is fully replicated.

1.3.2 MSMQ Directory Replication

The MSMQ: Directory Service Replication Protocol uses certain mechanisms to keep data consistent in the directory, including a **replication queue** and different types of MSMQ messages for directory data replication.

1.3.2.1 Directory Change Request

When a directory client requests a directory server to change an object, if the server is not the object authority, it forwards the request to the object authority. The object authority then executes the change operation on the object.

1.3.2.2 Directory Change Propagation

An object authority periodically propagates the executed directory changes to its neighbors so that the neighbors can update their directory replica. When a PSC receives a directory change from the object authority, it sends the change to its BSCs so that the BSCs can also update their replica.

1.3.2.3 Synchronization Request

As each directory change occurs at the object authority, it is stamped with a **sequence number** to maintain the order of directory changes. If a failure occurs when a directory server is propagating a change, that directory server sends a synchronization request to the object authority (a BSC sends a synchronization request to its PSC) and asks for missing directory changes. The object authority or the PSC prepares the requested changes and sends them to the requester.

1.3.2.4 Deleted Object

To handle object deletions, the MQDSRP protocol keeps information about the deleted objects. When the object authority deletes an object, it creates a deleted object data element and replicates the associated directory change to other servers through directory change propagation or through synchronization. Acknowledgments are sent back from BSCs to their PSC and from PSCs to the object authority for permanently purging the deleted objects.

1.3.2.5 Full-Partition Synchronization

The deletion acknowledgment mechanism of the MQDSRP protocol is intended to ensure that no other directory servers miss a deletion change before deleted objects are permanently purged. However, because successful sending, delivery, and processing of acknowledgment messages cannot be guaranteed, the protocol makes a tradeoff by purging deleted objects, even if not all directory neighbors have acknowledged them. This tradeoff can lead to a situation where a directory neighbor requests synchronization, but the object authority no longer has the deleted object.

A directory server may lose replication state for maintained partitions after having acknowledged a directory change for a deletion. In this case, the object authority may have already purged the deleted objects and is unable to provide a directory change during synchronization.

To resolve this issue, the protocol implements full-partition synchronization. When a synchronization request cannot be satisfied because the requested objects are not available, the **MQDSRP protocol server** informs the **MQDSRP protocol client** to perform a full-partition synchronization. The MQDSRP protocol client recreates the affected partition by deleting all the existing objects in that partition, resetting the replication state of that partition, and asking the MQDSRP protocol server for a full copy of the partition. The MQDSRP protocol client remains in a special purge state, known as the **sync0** state, during the entire process.

1.4 Relationship to Other Protocols

The MSMQ: Directory Service Replication Protocol (MQDSRP) is a queued protocol that uses the Message Queuing (MSMQ): Message Queuing Binary Protocol (MQQB) as its transport protocol. MQDSRP also uses the shared state and processing rules that are defined in MSMQ: Common Data Model and Processing Rules (MQDMPR). In addition, MQDSRP uses events from the MSMQ: Directory Service Protocol (MQDS) to access directory data elements and functionalities.

The following figure illustrates the relationship of the MQDSRP protocol to other protocols.

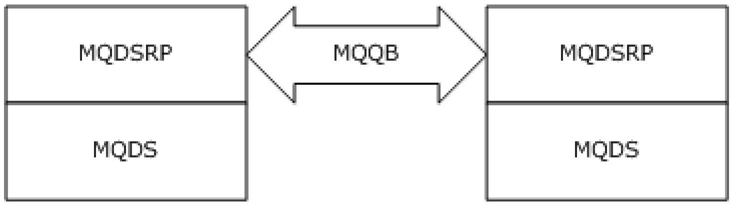

Figure 3: Relationship of this protocol to other protocols

1.5 Prerequisites/Preconditions

Before an MQDSRP protocol client invokes the MQDSRP protocol, the MQDSRP protocol client must obtain the name of a server computer that supports the MQDSRP protocol and also obtain the name of the replication queue that is hosted on that server. Implementers can determine how the MQDSRP protocol client obtains this information.

1.6 Applicability Statement

The MSMQ: Directory Service Replication Protocol (MQDSRP) provides directory service replication functionality. Both the MQDSRP client and MQDSRP server sides of this protocol are applicable for implementation by a queue manager with support for a fixed and limited set of directory object types and MSMQ object properties. The MQDSRP protocol is not intended as a general-purpose directory service replication protocol.

The functionality of this protocol is superseded by Active Directory and the Active Directory Technical Specification [MS-ADTS]. Future development based on this protocol is strongly discouraged.<1>

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports**: The MQDSRP protocol is implemented on top of MSMQ, as specified in section 2.1. It relies on the sending and receiving mechanisms of MSMQ messages to send and receive replications as messages.
- **Capability Negotiation**: This protocol does not perform any capability negotiation; the protocol has only one version.

1.8 Vendor-Extensible Fields

This protocol does not define any vendor-extensible fields.

1.9 Standards Assignments

This section specifies standard parameters within the context of MSMQ.

Parameter	**Value**	**Reference**
REPLICATION_QUEUE_NAME	<Machine Name>\private$\mqis_queue$	None.
REPLICATION_QUEUE_PRIVATE	PRIVATE=<Queue Manager GUID>\1	None.
REPLICATION _QUEUE_DIRECT	DIRECT=OS:<Machine Name>\private$\mqis_queue$	None.
SCOPE_ENTERPRISE	1	None.
SCOPE_SITE	0	None.
PEC_PARTITION_ID	GUID_NULL: {00000000-0000-0000-0000-000000000000}	None.

2 Messages

The sections that follow specify the transport and the message formats used by this protocol.

2.1 Transport

The protocol uses the MSMQ binary transport for all replication messages, as specified in [MS-MQQB]. The replication data is described by a replication message, which is carried in the body of an MSMQ message. Both the MQDSRP server and MQDSRP client sides of the MQDSRP protocol MUST maintain a replication queue to receive these messages.

2.2 Message Syntax

The MSMQ: Directory Service Replication Protocol uses **little-endian** byte order.

2.2.1 Common Data Types

This protocol uses the following data types as defined in [MS-DTYP], [MS-MQMQ], and [MS-MQDS].

HRESULT: As specified in **[MS-DTYP] (section 2.2.18)**.

GUID: As specified in [MS-DTYP] (section 2.3.4).

PROPID: As specified in **[MS-MQMQ] (section 2.3)**.

PROPVARIANT: As specified in **[MS-MQMQ] (section 2.2.13.2)**.

Directory Object Types: As specified in [MS-MQDS] (section 2.2.8).

MSMQ Object Properties: As specified in [MS-MQDS] (section 2.2.10).

MQPROPERTYRESTRICTION: As specified in **[MS-MQDS] (section 2.2.11)**.

MQRESTRICTION: As specified in **[MS-MQDS] (section 2.2.12)**.

MQCOLUMNSET: As specified in **[MS-MQDS] (section 2.2.13)**.

UCHAR: As specified in **[MS-DTYP] (section 2.2.45)**.

WCHAR: As specified in **[MS-DTYP] (section 2.2.60)**.

This specification references the following PROPID constants that are defined in **[MS-MQMQ]** (section 2.3).

PROPID constant	Value
PROPID_Q_INSTANCE	101
PROPID_Q_TYPE	102
PROPID_Q_PATHNAME	103
PROPID_Q_JOURNAL	104
PROPID_Q_QUOTA	105

PROPID constant	Value
PROPID_Q_BASEPRIORITY	106
PROPID_Q_JOURNAL_QUOTA	107
PROPID_Q_LABEL	108
PROPID_Q_CREATE_TIME	109
PROPID_Q_MODIFY_TIME	110
PROPID_Q_AUTHENTICATE	111
PROPID_Q_PRIV_LEVEL	112
PROPID_Q_TRANSACTION	113
PROPID_Q_SCOPE	114
PROPID_Q_QMID	115
PROPID_Q_PARTITIONID	116
PROPID_Q_SEQNUM	117
PROPID_Q_HASHKEY	118
PROPID_Q_LABEL_HASHKEY	119
PROPID_QM_SITE_ID	201
PROPID_QM_MACHINE_ID	202
PROPID_QM_PATHNAME	203
PROPID_QM_ADDRESS	206
PROPID_QM_CNS	207
PROPID_QM_OUTFRS	208
PROPID_QM_INFRS	209
PROPID_QM_SERVICE	210
PROPID_QM_PARTITIONID	211
PROPID_QM_HASHKEY	212
PROPID_QM_SEQNUM	213
PROPID_QM_QUOTA	214
PROPID_QM_JOURNAL_QUOTA	215
PROPID_QM_MACHINE_TYPE	216
PROPID_QM_CREATE_TIME	217
PROPID_QM_MODIFY_TIME	218

PROPID constant	Value
PROPID_QM_FOREIGN	219
PROPID_QM_OS	220
PROPID_S_PATHNAME	301
PROPID_S_SITEID	302
PROPID_S_GATES	303
PROPID_S_PSC	304
PROPID_S_INTERVAL1	305
PROPID_S_INTERVAL2	306
PROPID_S_PARTITIONID	307
PROPID_S_SEQNUM	308
PROPID_CN_PROTOCOLID	501
PROPID_CN_NAME	502
PROPID_CN_GUID	503
PROPID_CN_PARTITIONID	504
PROPID_CN_SEQNUM	505
PROPID_E_NAME	601
PROPID_E_NAMESTYLE	602
PROPID_E_CSP_NAME	603
PROPID_E_PECNAME	604
PROPID_E_S_INTERVAL1	605
PROPID_E_S_INTERVAL2	606
PROPID_E_PARTITIONID	607
PROPID_E_SEQNUM	608
PROPID_E_ID	609
PROPID_E_CRL	610
PROPID_E_CSP_TYPE	611
PROPID_E_ENCRYPT_ALG	612
PROPID_E_SIGN_ALG	613
PROPID_E_HASH_ALG	614
PROPID_E_CIPHER_MODE	615

PROPID constant	Value
PROPID_E_LONG_LIVE	616
PROPID_E_VERSION	617
PROPID_U_SID	701
PROPID_U_SIGN_CERT	702
PROPID_U_PARTITIONID	703
PROPID_U_SEQNUM	704
PROPID_U_DIGEST	705
PROPID_U_ID	706
PROPID_L_NEIGHBOR1	801
PROPID_L_NEIGHBOR2	802
PROPID_L_COST	803
PROPID_L_PARTITIONID	804
PROPID_L_SEQNUM	805
PROPID_L_ID	806
PROPID_Q_SECURITY	1101
PROPID_QM_SECURITY	1201
PROPID_QM_SIGN_PK	1202
PROPID_QM_ENCRYPT_PK	1203
PROPID_S_SECURITY	1301
PROPID_S_PSC_SIGNPK	1302
PROPID_D_SEQNUM	1401
PROPID_D_PARTITIONID	1402
PROPID_D_SCOPE	1403
PROPID_D_OBJTYPE	1404
PROPID_D_IDENTIFIER	1405
PROPID_CN_SECURITY	1501
PROPID_E_SECURITY	1601

2.2.2 SEQUENCE_NUMBER

The SEQUENCE_NUMBER data type is an 8-byte array of type **UCHAR**. The protocol uses this structure, which is called a sequence number, to uniquely identify each directory change. Each object also carries a sequence number that indicates the last directory change that occurred to the object. The protocol also uses sequence numbers to maintain the replication state of a directory partition.

```
typedef UCHAR[8] SEQUENCE_NUMBER;
```

The following operations can be performed on this data type:

Increment: Assuming a sequence number sn, this operation increments the number by one as follows:

- For i :=8 to 1:
  - sn[i] := sn[i] + 1
  - If sn[i] is not 0:
    - Take no further action.

Decrement: Assuming a sequence number sn, this operation decrements the number by one as follows:

- For i :=8 to 1:
  - Set tempVal to sn[i].
  - sn[i] := sn[i] - 1
  - If tempVal is not 0:
    - Take no further action.

Addition: Add an unsigned integer n to a sequence number sn and return a new sequence number.

- Set rSeqNumber to sn.
- Increment rSeqNubmer by n times.

Subtraction: Subtract an unsigned integer n from a sequence number sn and return a new sequence number.

- Set rSeqNumber to sn.
- Decrement rSeqNubmer by n times.

Compare two sequence numbers: Assuming two sequence numbers sn1 and sn2, use the following rules to perform a comparison:

- Rule 1: sn1 is equal to sn2 when each element in sn1 is equal to the number in sn2 at the same position.
- If rule 1 fails, sn1 is not equal to sn2.
- Perform the following actions to further determine which element is greater:
  - For i :=1 to 8:

- If sn1[i] > sn2[i]:
  - sn1 is greater than sn2 regardless of the remaining elements.
- If sn1[i] < sn2[i]:
  - sn1 is less than sn2 regardless of the remaining elements.

This protocol uses the following pre-defined sequence numbers.

Constant	Value
MAX_SEQ_NUMBER	[0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF]
MIN_SEQ_NUMBER	[0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]

2.2.3 PARTITION_SEQ_NUMBERS

Object authorities identify each directory change with a sequence number. They also use sequence numbers to represent the replicate state of each directory partition. The PARTITION_SEQ_NUMBERS structure contains two sequence numbers of a directory partition.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
PartitionID																															
...																															
...																															
...																															
LastSeqNumber																															
...																															
PurgedSeqNumber																															
...																															

PartitionID (16 bytes): A GUID that identifies the directory partition with which the sequence numbers are associated.

LastSeqNumber (8 bytes): A SEQUENCE_NUMBER element (section 2.2.2) that contains the sequence number of the last directory change in the associated directory partition.

PurgedSeqNumber (8 bytes): A SEQUENCE_NUMBER element that contains the **purged sequence number** in the associated directory partition.

2.2.4 DirectoryChange

A DirectoryChange contains information about a directory change that can be applied to a directory object.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Command								UseGuid								PathName (variable)															
...																															
GuidIdentifier (optional)																															
...																															
...																															
...																															
PartitionID																															
...																															
...																															
...																															
PreviousSeqNumber																															
...																															
SeqNumber																															
...																															
PurgedSeqNumber																															
...																															
NumberOfProperties								PropertyID (variable)																							
...																															
PropertyValue (variable)																															
...																															

Command (1 byte): A **UCHAR** that indicates the directory change operation. It MUST be one of the following values.

Value	Meaning
0x00	A creation operation.
0x01	An update operation.
0x02	A deletion operation.
0x03	A synchronization operation.

UseGuid (1 byte): A **UCHAR** that indicates if a directory change uses the **PathName** or the **GuidIdentifier** field. This field MUST be set as follows.

Value	Meaning
0x00	The **PathName** is used.
0x01	The **GuidIdentifier** is used.

PathName (variable): A null-terminated **WCHAR** buffer that contains the object path name. This field MUST NOT be present if **UseGuid** is set to 0x01. Instead, **GuidIdentifier** MUST be set with a value.

GuidIdentifier (16 bytes): A **GUID** of the object to which the directory change applies. This field MUST be present if **UseGuid** is set to 0x01.

PartitionID (16 bytes): A **GUID** of the directory partition to which this directory change applies.

PreviousSeqNumber (8 bytes): A SEQUENCE_NUMBER element (section 2.2.2) that contains the sequence number of the previous directory change in the associated directory partition.

SeqNumber (8 bytes): A **SEQUENCE_NUMBER** element (section 2.2.2) that contains the sequence number of this directory change in the associated directory partition.

PurgedSeqNumber (8 bytes): A **SEQUENCE_NUMBER** element (section 2.2.2) that contains the purged sequence number of the associated directory partition.

NumberOfProperties (1 byte): A **UCHAR** that identifies the number of properties that are included as part of this directory change.

PropertyID (variable): An array of **PROPID** that contains a collection of **property identifiers** that are part of this directory change. The number of elements in the array MUST be equal to the value of **NumberOfProperties**.

PropertyValue (variable): An array of **PROPVARIANT** that contains a collection of property values that are part of this directory change. The number of property values in the array MUST be equal to the value of **NumberOfProperties**. The format of each property value depends on the variant type, which is determined by the corresponding property identifier in **PropertyID**.

2.2.5 Headers

The following sections specify the formats of the headers that are contained in replication messages.

2.2.5.1 BaseReplicationHeader

At the beginning of a replication message is a BaseReplicationHeader, which contains the basic information about a replication message, including the version, the site identifier of the sending queue manager, and the operation type.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Version								SiteID																							
...																															
...																															
...																															
...								Operation																							

Version (1 byte): A **UCHAR** that contains the version of the directory replication message. This field MUST be set to 0x00.

SiteID (16 bytes): A **GUID** that identifies the site to which the sending queue manager belongs.

Operation (1 byte): A **UCHAR** that contains the operation type of the replication message. This field MUST be set to one of the following values.

Value	Meaning
0x00	A change propagation message.
0x01	A change request message.
0x02	A synchronization request message.
0x03	A synchronization reply message.
0x04	A change reply message.
0x05	An already purged message.
0x06	A PSC acknowledgment message.
0x07	A BSC acknowledgment message.

2.2.5.2 ChangeRequestHeader

A ChangeRequestHeader contains information about a change request operation and is in the format that follows.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BaseReplicationHeader																															
...																															
...																															
...																															
...																PartitionID															
...																															
...																															
...																															
...																RequestIdentifier															
...																PSCNameOffset															
...																RequesterName (variable)															
...																															
PSCName (optional)																															
...																															

BaseReplicationHeader (18 bytes): The BaseReplicationHeader, as specified in section 2.2.5.1, where the **Operation** field MUST be set to 0x01.

PartitionID (16 bytes): A GUID that identifies the directory partition to which the change request applies.

RequestIdentifier (4 bytes): A 32-bit unsigned integer that uniquely identifies the Process Change Request event (section 3.3.7.1) that is associated with this change request message. It is used by the requester to correlate a reply with the original request.

PSCNameOffset (4 bytes): A 32-bit unsigned integer that specifies the offset, in **WCHARs**, of the null-terminated **WCHAR** buffer **PSCName**, relative to the beginning of the **RequesterName**.

RequesterName (variable): A null-terminated **WCHAR** buffer that contains the machine name of the original requester.

PSCName (8 bytes): A null-terminated **WCHAR** buffer that contains the machine name of the primary site controller (PSC) that shares the same site with the original requester. This field is only set when the requester is a Backup Site Controller (BSC); otherwise it MUST NOT be set and the **PSCNameOffset** field MUST be 0x00.

2.2.5.3 SyncReplyHeader

A SyncReplyHeader is part of a SyncReplyMessage (section 2.2.6.5) and contains information about a synchronization reply.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BaseReplicationHeader																															
...																															
...																															
...																															
...																PartitionID															
...																															
...																															
...																															
...																FromSeqNumber															
...																															
...																ToSeqNumber															
...																															
...																PurgedSeqNumber															
...																															
...																Count															
...																CompleteSync0															

...

BaseReplicationHeader (18 bytes): The BaseReplicationHeader, as specified in section 2.2.5.1, where the **Operation** field MUST be set to 0x03.

PartitionID (16 bytes): A GUID that identifies the directory partition to which the synchronization reply applies.

FromSeqNumber (8 bytes): A SEQUENCE_NUMBER element (section 2.2.2) that contains the sequence number where the synchronization starts.

ToSeqNumber (8 bytes): A **SEQUENCE_NUMBER** element (section 2.2.2) that contains the sequence number where the synchronization ends.

PurgedSeqNumber (8 bytes): A **SEQUENCE_NUMBER** element (section 2.2.2) that contains the purged sequence number of the associated directory partition.

Count (4 bytes): A 32-bit unsigned integer that indicates the number of directory changes that are contained in the SyncReplyMessage.

CompleteSync0 (4 bytes): A 32-bit unsigned integer that indicates the completion status of the synchronization on the object authority. It MUST be set to one of the following values.

Value	Meaning
0x00000000	The partition is not in the **sync0** state.
0x00000001	The **sync0** state is completing soon.
0x00000002	The **sync0** state is completed now.

2.2.5.4 SeqNumberHeader

A SeqNumberHeader contains the sequence numbers of directory partitions.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Count																MachineName (variable)															
...																															
PartitionSeqNumbers (variable)																															
...																															

Count (2 bytes): A 16-bit unsigned integer that indicates the number of elements in the array of **PartitionSeqNumbers**.

MachineName (variable): A null-terminated **WCHAR** buffer that contains the machine name of the sending queue manager. This field MUST NOT be present if **Count** is set to 0x0000.

PartitionSeqNumbers (variable): An array of PARTITION_SEQ_NUMBERS elements (section 2.2.3). This field MUST NOT be present if **Count** is set to 0x0000; otherwise, the number of elements in this array MUST be the same as **Count**.

2.2.6 Replication Messages

A replication message is contained in an MSMQ message body and is sent by using the Message Queuing Binary Protocol. All replication messages begin with the common structure, BaseReplicationHeader (section 2.2.5.1), followed by data that varies according to the replication message type.

For purposes of this protocol specification, an abstract replication message is introduced that is called a BaseReplicationMessage (section 2.2.6.1). It is inherited by the following types of replication messages:

- ChangeRequestMessage (section 2.2.6.2)
- ChangeReplyMessage (section 2.2.6.3)
- SyncRequestMessage (section 2.2.6.4)
- SyncReplyMessage (section 2.2.6.5)
- BSCAckMessage (section 2.2.6.6)
- PSCAckMessage (section 2.2.6.7)
- AlreadyPurgedMessage (section 2.2.6.8)
- ChangePropagationMessage (section 2.2.6.9)

The topics in the next section of this specification provide the format of these messages.

2.2.6.1 BaseReplicationMessage

A BaseReplicationMessage is an abstract data element of the MQDSRP protocol, from which all other replication messages inherit.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BaseReplicationHeader																															
...																															
...																															
...																															
...																Data (variable)															
...																															

BaseReplicationHeader (18 bytes): The BaseReplicationHeader, as specified in section 2.2.5.1.

Data (variable): A byte array that contains the data that is meaningful to the corresponding operation, as determined by a **BaseReplicationHeader.Operation** (section 2.2.5.1). The format of this field is specified in the sections that follow for each type of replication message.

2.2.6.2 ChangeRequestMessage

A ChangeRequestMessage is sent from a directory server to the object authority to change a directory object.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ChangeRequestHeader (variable)																															
...																															
DirectoryChange (variable)																															
...																															

ChangeRequestHeader (variable): A ChangeRequestHeader, as specified in section 2.2.5.2, where the **Operation** field MUST be set to 0x01.

DirectoryChange (variable): A DirectoryChange element, as specified in section 2.2.4.

2.2.6.3 ChangeReplyMessage

A ChangeReplyMessage is a response to a ChangeRequestMessage (section 2.2.6.2). It contains the results of the operation that occurred at the object authority.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BaseReplicationHeader																															
...																															
...																															
...																															
...																RequestIdentifier															
...																Result															
...																RequesterName (variable)															

...

BaseReplicationHeader (18 bytes): A BaseReplicationHeader element, as specified in section 2.2.5.1, where the **Operation** field MUST be set to 0x04.

RequestIdentifier (4 bytes): A 32-bit unsigned integer that uniquely identifies the Process Change Request event (section 3.3.7.1) that is associated with the ChangeRequestMessage. The requester uses this value to correlate a reply with the original request.

Result (4 bytes): An **HRESULT** value that indicates the result of the change operation.

RequesterName (variable): A null-terminated **WCHAR** buffer that contains the original requester name.

2.2.6.4 SyncRequestMessage

A SyncRequestMessage contains information for a synchronization request.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BaseReplicationHeader																															
...																															
...																															
...																															
...																PartitionID															
...																															
...																															
...																															
...																FromSeqNumber															
...																															
...																ToSeqNumber															
...																															
...																KnownPurgedSeqNumber															
...																															

...	IsSync0	Scope
RequesterName (variable)		
...		

BaseReplicationHeader (18 bytes): The BaseReplicationHeader element (section 2.2.5.1), where the **Operation** field MUST be set to 0x02.

PartitionID (16 bytes): A GUID that identifies the directory partition with which the synchronization request is associated.

FromSeqNumber (8 bytes): A SEQUENCE_NUMBER element (section 2.2.2) that contains the sequence number where the synchronization starts.

ToSeqNumber (8 bytes): A SEQUENCE_NUMBER element (section 2.2.2) that contains the sequence number where the synchronization ends.

KnownPurgedSeqNumber (8 bytes): A SEQUENCE_NUMBER element (section 2.2.2) that contains the purged sequence number of the associated partition that is known to the requester.

IsSync0 (1 byte): A **UCHAR** that indicates the sync0 state of the partition. It MUST be set to one of the following values.

Value	Meaning
0x00	The partition is not in the sync0 state.
0x01	The partition is in the sync0 state.

Scope (1 byte): A **UCHAR** that indicates the scope of the synchronization request. It MUST be set to one of the following values.

Value	Meaning
0x00	Indicates no scope flag is set.
0x01	Indicates an enterprise scope.

RequesterName (variable): A null-terminated **WCHAR** buffer that contains the machine name of the sending queue manager.

2.2.6.5 SyncReplyMessage

A SyncReplyMessage is a response to a SyncRequestMessage (section 2.2.6.4). It contains the directory changes that satisfy the request in the SyncRequestMessage.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
SyncReplyHeader																															

...	
...	
...	
...	
...	
...	
...	
(SyncReplyHeader cont'd for 8 rows)	
...	DirectoryChanges (variable)
...	

SyncReplyHeader (66 bytes): A SyncReplyHeader element, as specified in section 2.2.5.3, where the **Operation** field MUST be set to 0x03.

DirectoryChanges (variable): An array of DirectoryChange elements, as specified in 2.2.4. The number of elements in the array MUST be the same as SyncReplyHeader.Count (section 2.2.5.3).

2.2.6.6 BSCAckMessage

A BSCAckMessage is an acknowledgment that is sent from a Backup Site Controller (BSC) to its Primary Site Controller (PSC).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BaseReplicationHeader																															
...																															
...																															
...																															
...																BSCMachineID															
...																															

...	
...	
...	BSCName (variable)
...	

BaseReplicationHeader (18 bytes): A BaseReplicationHeader element, as specified in 2.2.5.1, where the **Operation** field MUST be set to 0x07.

BSCMachineID (16 bytes): A **GUID** that identifies the queue manager of the BSC.

BSCName (variable): A null-terminated **WCHAR** buffer that contains the machine name of the BSC.

2.2.6.7 PSCAckMessage

A PSCAckMessage is an acknowledgment from a PSC to the object authority that contains the acknowledged sequence number of the directory partition that is owned by the object authority.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BaseReplicationHeader																															
...																															
...																															
...																															
...																PSCSiteID															
...																															
...																															
...																															
...																AckedPartitionID															
...																															
...																															
...																															

...	AckedSeqNumber
...	
...	PSCName (variable)
...	

BaseReplicationHeader (18 bytes): A BaseReplicationHeader element, as specified in 2.2.5.1, where the **Operation** field MUST be set to 0x06.

PSCSiteID (16 bytes): A **GUID** that identifies the site to which the PSC belongs.

AckedPartitionID (16 bytes): A **GUID** that identifies the directory partition to which this message applies.

AckedSeqNumber (8 bytes): A SEQUENCE_NUMBER element (section 2.2.2) that contains the acknowledged sequence number.

PSCName (variable): A null-terminated **WCHAR** buffer that contains the machine name of the PSC.

2.2.6.8 AlreadyPurgedMessage

An AlreadyPurgedMessage is a reply to a SyncRequestMessage (section 2.2.6.4) that indicates the request cannot be fulfilled. It contains the purged sequence number of the directory partition that is owned by the object authority.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BaseReplicationHeader																															
...																															
...																															
...																															
...																PartitionID															
...																															
...																															
...																															
...																PurgedSeqNumber															

...	
...	

BaseReplicationHeader (18 bytes): A BaseReplicationHeader element, as specified in 2.2.5.1, where the **Operation** field MUST be set to 0x05.

PartitionID (16 bytes): A **GUID** that identifies the directory partition to which the message applies.

PurgedSeqNumber (8 bytes): A SEQUENCE_NUMBER element (section 2.2.2) that contains the purged sequence number of the directory partition.

2.2.6.9 ChangePropagationMessage

A ChangePropagationMessage is sent from an object authority to its directory neighbors to propagate the directory changes that have occurred to its objects. It is also sent from a Primary Site Controller (PSC) to one of its Backup Site Controllers (BSC) when the PSC receives a directory change from the object authority.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BaseReplicationHeader																															
...																															
...																															
...																															
...																Flush								Count							
...								DirectoryChanges (variable)																							
...																															
SeqNumberHeader (variable)																															
...																															

BaseReplicationHeader (18 bytes): A BaseReplicationHeader element, as specified in 2.2.5.1, where the **Operation** field MUST be set to 0x00.

Flush (1 byte): A **UCHAR** that indicates whether this is a flush message (from a PSC to its BSCs). It MUST be set to one of the following values.

Value	Meaning
0x00	The recipient directory server MUST check whether the change needs to be flushed to its BSCs.
0x01	This is a flush message. The recipient directory server MUST NOT flush this change to its BSCs.

Count (2 bytes): A 16-bit unsigned integer that indicates the number of directory changes that are included in this message.

DirectoryChanges (variable): An array of DirectoryChange elements, as specified in section 2.2.4. The number of elements in this array MUST be the same as **Count**.

SeqNumberHeader (variable): A SeqNumberHeader element, as specified in section 2.2.5.4.

3 Protocol Details

Replication messages are exchanged between two MSMQ directory servers. For purposes of this protocol, the initiator of a replication message, which is also called a request, is referred to as the MQDSRP protocol client. The request handler, which processes the request and returns a reply (as appropriate), is referred to as the MQDSRP protocol server. Under certain conditions, it is possible for the MQDSRP protocol server to initiate another instance of the MQDSRP protocol and act as an MQDSRP protocol client in a new instance of additional data exchange.

3.1 Common Details

Both the MQDSRP protocol client and the MQDSRP protocol server share common data elements.

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with the behavior described in this document.

The abstract data model for this protocol comprises elements that are private to this protocol and others that are shared between multiple MSMQ protocols that are co-located at a common queue manager. The shared abstract data model is defined in [MS-MQDMPR] section 3.1.1, and the relationship between this protocol, a queue manager, and other protocols that share a common queue manager is described in [MS-MQOD].

Section 3.1.1.1 references the elements from the shared data model that are manipulated by this protocol. Section 3.1.1.2 specifies the elements that are locally manipulated by this protocol.

3.1.1.1 Shared Data Elements

The MQDSRP server and MQDSRP client sides of the MQDSRP protocol manipulate the following abstract data model elements from the shared abstract data model that is defined in [MS-MQDMPR] section 3.1.1.

Enterprise: A data element as defined in [MS-MQDMPR] section 3.1.1.6.

Site: A data element as defined in [MS-MQDMPR] section 3.1.1.7.

RoutingLink: A data element as defined in [MS-MQDMPR] section 3.1.1.8.

ConnectedNetwork: A data element as defined in [MS-MQDMPR] section 3.1.1.9.

User: A data element as defined in [MS-MQDMPR] section 3.1.1.15.

QueueManager: A data element as defined in [MS-MQDMPR] section 3.1.1.1.

Queue: A data element as defined in [MS-MQDMPR] section 3.1.1.2.

Message: A data element as defined in [MS-MQDMPR] section 3.1.1.12.

OutgoingQueue: A data element as defined in [MS-MQDMPR] section 3.1.1.3.

OpenQueueDescriptor: A data element as defined in [MS-MQDMPR] section 3.1.1.16.

This protocol adds the following attributes for the **QueueManager** data element. These attributes are not persisted in the directory and MUST be computed during protocol initialization:

MyEnterprisePartition: A reference to a DirectoryPartition data element (section 3.1.1.2.3) that represents the replication state of the enterprise partition. This attribute is only populated when the queue manager is a Primary Enterprise Controller (PEC).

MySitePartition: A reference to a DirectoryPartition data element that represents the replication state of the site partition where the queue manager resides.

MyPSCName: A null-terminated **WCHAR** buffer that contains the machine name of the Primary Site Controller (PSC) of the site to which the queue manager belongs. This attribute is populated only if the queue manager is a Backup Site Controller (BSC).

3.1.1.2 Local Data Elements

In addition to the shared data elements, the MQDSRP server and MQDSRP client sides of the MQDSRP protocol manipulate the local abstract data elements that are specified in this section.

3.1.1.2.1 ReplicationQueue Data Element

The ReplicationQueue data element represents a local private queue for directory service replication. The name for this queue is in the format that is specified by the following ABNF [RFC4234] rule.

```
QueuePathName = (Computer "\private$\mqis_queue$")
Computer    = 1*256(VCHAR) ; computer where the queue manager resides.
```

3.1.1.2.2 DeletedObject Data Element

The DeletedObject data element represents a deleted object in the directory. When an object is deleted, a **DeletedObject** object is created to represent the deletion operation so that the directory change can be replicated to other directory servers. This data element contains the following directory attributes:

Identifier: A **GUID** that identifies the deleted object.

PartitionID: A GUID that identifies the directory partition that the deleted object belongs to.

SeqNumber: A SEQUENCE_NUMBER element (section 2.2.2) that contains the sequence number that is associated with the deletion operation.

ObjectType: An enumeration that contains the type of the deleted object. It MUST have one of the following values.

MQDS_ENTERPRISE: The deleted object is an **Enterprise** data element.

MQDS_SITE: The deleted object is a **Site** data element.

MQDS_ROUTINGLINK: The deleted object is a **RoutingLink** data element.

MQDS_CN: The deleted object is a **ConnectedNetwork** data element.

MQDS_MACHINE: The deleted object is a **QueueManager** data element.

MQDS_QUEUE: The deleted object is a **Queue** data element.

MQDS_USER: The deleted object is a **User** data element.

Scope: An enumeration that contains the scope of the **DeletedObject**. This enumeration MUST have one of the following values.

SCOPE_ENTERPRISE: The deleted object is in the enterprise scope.

SCOPE_SITE: The deleted object is in the site scope.

The following table specifies the mapping between property identifiers and the attributes of a **DeletedObject** data element.

Attribute	Property identifier
Identifier	PROPID_D_IDENTIFIER
PartitionID	PROPID_D_PARTITIONID
SeqNumber	PROPID_D_SEQNUM
Scope	PROPID_D_SCOPE
ObjectType	PROPID_D_OBJTYPE

3.1.1.2.3 DirectoryPartition Data Element

The DirectoryPartition data element represents the replication state of a directory partition that is maintained by a directory server. It has the following attributes:

PartitionID: A globally unique identifier (GUID) that identifies the directory partition. The identifier MUST be the same on all directory servers for the same directory partition. The identifier of the enterprise partition MUST be PEC_PARTITION_ID as defined in section 1.9. The identifier of a site partition MUST be the site identifier.

AuthorityName: A null-terminated **WCHAR** buffer that contains the machine name of the object authority for this partition.

LastSeqNumber: A SEQUENCE_NUMBER element (section 2.2.2) that contains the sequence number of the last directory change in this partition.

PurgedSeqNumber: A SEQUENCE_NUMBER element that contains the purged sequence number in this partition.

AllowedPurgeSeqNumber: A SEQUENCE_NUMBER element that contains a sequence number indicating that any object with a lesser sequence number is allowed to be purged in this partition.

PurgeState: A **UCHAR** that indicates the purge state of the partition. It MUST be one of the following values.

Purge state	Meaning
0x00	The partition is not in **sync0** state (PURGE_STATE_NORMAL).
0x01	The partition is starting the **sync0** state (PURGE_STATE_STARTSYNC0).
0x02	The partition is in the **sync0** state (PURGE_STATE_SYNC0).

Purge state	Meaning
0x03	The partition is completing the **sync0** state (PURGE_STATE_COMPLETESYNC0).

The following attributes are not persisted and are updated by the directory server during runtime:

PreviousPurgedSeqNumber: A SEQUENCE_NUMBER element that contains a sequence number at which the last **Purge Deleted Objects** event occurred. It is used to trigger the **Purge Deleted Objects** event at a certain interval. During protocol initialization or when a new **DirectoryPartition** object is created, this attribute MUST be set to MIN_SEQ_NUMBER as defined in section 2.2.2.

ChangeMissingWindow: A SEQUENCE_NUMBER element that contains a sequence number indicating the window of missing changes in this partition. During protocol initialization or when a new **DirectoryPartition** object is created, this attribute MUST be set to MAX_SEQ_NUMBER, as defined in section 2.2.2.

PendingChangeCollection: A collection of DirectoryChange objects that are waiting to be applied to this partition. After a directory server receives a directory change, it verifies the sequence number in the directory change against the partition replication state. If the directory change is out of order, it is appended to this collection and processed at a later time when the missing changes are received. During protocol initialization or when a new **DirectoryPartition** object is created, this attribute MUST be set to empty.

3.1.1.2.4 DirectoryNeighbor Data Element

The DirectoryNeighbor data element represents a directory neighbor of a directory server. It has the following attributes:

PartitionID: A GUID that identifies the site to which the directory neighbor belongs.

MachineName: A null-terminated **WCHAR** buffer that contains the machine name of the directory neighbor.

AckedSeqNumber: A SEQUENCE_NUMBER element (section 2.2.2) that contains the acknowledged purged sequence number from the neighbor for the directory partition that is owned by this queue manager (not the directory partition that is identified by **PartitionID**). This attribute is only populated when the current directory server is a PSC and the directory neighbor is a PSC.

AckedPECSeqNumber: A SEQUENCE_NUMBER element that contains the acknowledged purged sequence number from this neighbor for the enterprise partition that is owned by this queue manager. This attribute is only populated when the current directory server is the PEC and the directory neighbor is a PSC.

LastAckedTime: The **UTC** date and time at which the last acknowledgment was received from the directory neighbor. This attribute is only populated when the current directory server is a PSC and the directory neighbor is a BSC.

The following attribute is not persisted and is updated by the protocol during runtime:

AvailableChangeCollection: A collection of **DirectoryChange** objects that are to be propagated to the directory neighbor. During protocol initialization or when a new **DirectoryNeighbor** object is created, this attribute MUST be set to empty.

3.1.1.3 Protocol State Data Elements

The protocol MUST maintain the following data elements:

DirectoryPartitionCollection: A collection of DirectoryPartition objects (section 3.1.1.2.3), each of which represents the replication state of one directory partition that is maintained by this directory server. The protocol MUST maintain state for the attributes of each DirectoryPartition element. The state MUST be persisted, except for those attributes that are explicitly specified as not persisted.

PSCNeighborCollection: A collection of DirectoryNeighbor objects (section 3.1.1.2.4), each of which represents a PSC directory neighbor of the current queue manager. The protocol MUST maintain state for the attributes of each DirectoryNeighbor element. The state MUST be persisted, except for those attributes that are explicitly specified as not persisted.

BSCNeighborCollection: A collection of DirectoryNeighbor objects, each of which represents a BSC directory neighbor of the current queue manager. The protocol MUST maintain state for the attributes of each DirectoryNeighbor element. The state MUST be persisted, except for those attributes that are explicitly specified as not persisted.

DeletedObjectCollection: A collection of DeletedObject objects (section 3.1.1.2.2), each of which represents a deletion operation on a directory object. The protocol MUST maintain state for the attributes of each DeletedObject element. The state MUST be persisted, except for those attributes that are explicitly specified as not persisted.

3.1.1.4 PROPID and Object Type Mapping

Given a **PROPID** *propID*, the associated directory object type *rType* is determined as follows:

- Set *rType* to 0.
- If *propID* > 1000:
  - Set *rType* to ((*propID* - 1000) / 100).
- Else:
  - Set *rType* to (*propID* / 100).
- The directory object type is defined in the following table.

rType	Directory object type
0x01	MQDS_QUEUE
0x02	MQDS_MACHINE
0x03	MQDS_SITE
0x05	MQDS_CN
0x06	MQDS_ENTERPRISE
0x07	MQDS_USER
0x08	MQDS_ROUTINGLINK

A value of *rType* not present in the table indicates an unknown object type.

3.1.1.5 Protocol Sequence Diagrams

The sequence diagrams in this section describe the message flow in common scenarios of data replication in the MSMQ Directory Service.

3.1.1.5.1 Change Request and Reply

Assuming that a BSC receives a directory change request from a directory client, the following figure shows the message flow sequence among directory servers to handle the change request.

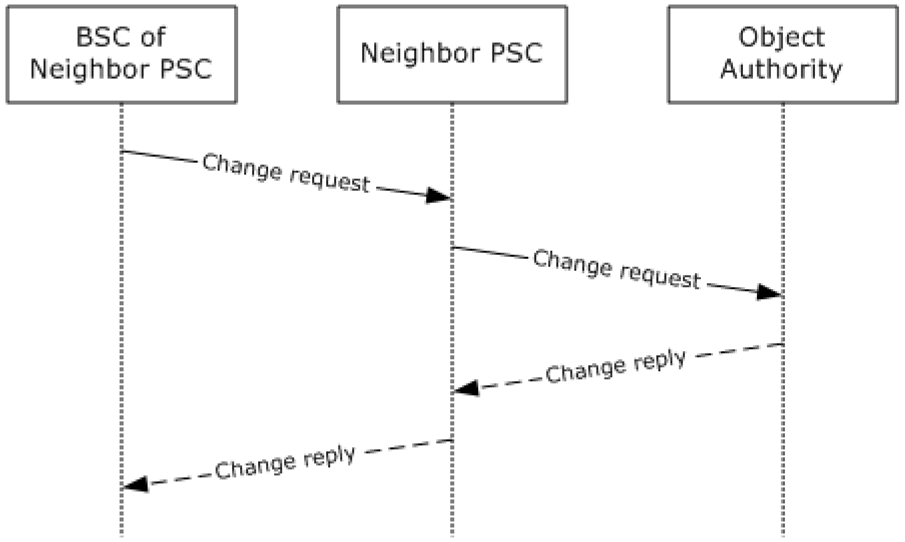

Figure 4: Change request and reply sequence

The BSC sends the change request to its PSC. The PSC checks the ownership of the object that is specified by the change request. If the PSC is not the object authority, it sends the change request to the object authority. The object authority executes the change on the object and sends the operation result in a change reply. The neighbor PSC forwards the change reply to the original requester, the BSC.

In this scenario, the intermediate PSC acts as both the MQDSRP protocol server (for the request from the BSC) and the MQDSRP protocol client (for forwarding the request to the object authority). Forwarding through the intermediate PSC is optional, and it is not needed in the following scenarios:

- A BSC receives a directory change request, and its PSC is the object authority.
- A PSC receives a directory change request. It communicates with the object authority directly.

3.1.1.5.2 Change Propagation

A ChangePropagationMessage (section 2.2.6.9) is sent as follows:

- From the object authority to its neighbor PSCs and to its neighbor BSCs when the authority changes its objects.
- From a PSC to its neighbor BSCs when it receives a directory change propagation from the object authority.

The following figure illustrates the message flow in propagating directory changes.

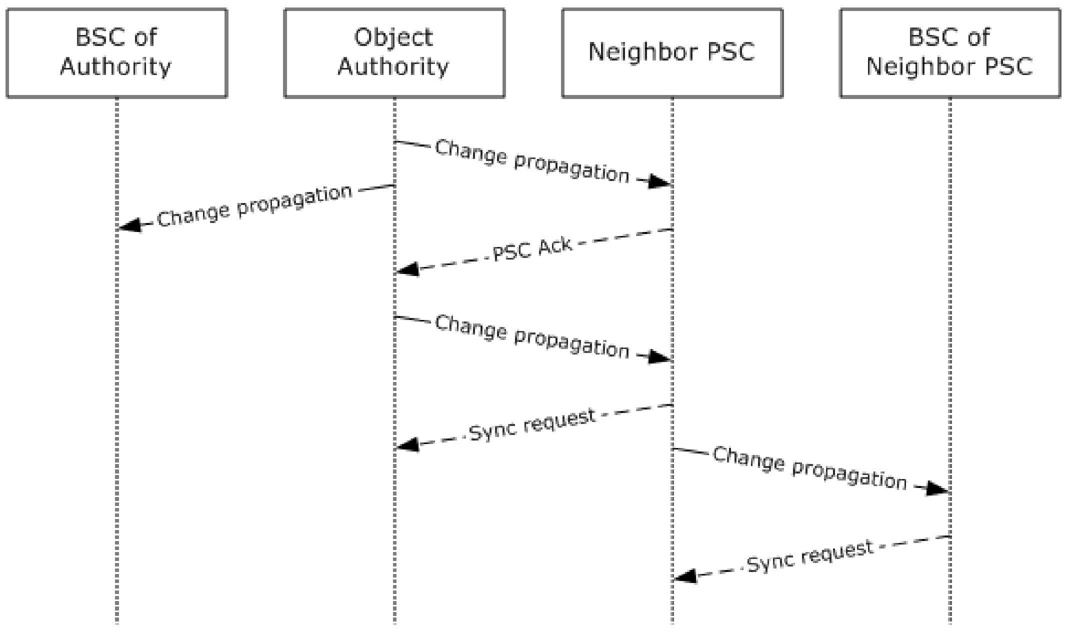

Figure 5: Change propagation sequence

When the object authority changes its directory objects, it periodically propagates the directory changes to its directory neighbors. These directory changes are generated either by the authority in response to a directory change request or through an event that is triggered by the directory service server co-located with the MQDSRP protocol client as specified in [MS-MQDS].

When receiving a ChangePropagationMessage from the object authority, a PSC synchronizes its local copy of that partition with the directory changes that are contained in the ChangePropagationMessage, and at a certain interval, as specified in section 3.1.7.2.3, it sends back an acknowledgment (PSC Ack) to the object authority with the state of that partition as maintained by the PSC. BSCs also send acknowledgments (BSC Ack) to its PSC; however, the acknowledgment is regulated by a timer.

When the change propagation is received out-of-sync, the receiver can ask for missing changes by sending a SyncRequestMessage (section 2.2.6.4), as specified in section 3.1.7.2.2. If the ChangePropagationMessage contains a SeqNumberHeader (section 2.2.5.4), it can also trigger sending a SyncRequestMessage, as specified in section 3.2.7.1.

3.1.1.5.3 Synchronization Request and Reply

A synchronization request is sent under the following conditions:

- A directory server initializes its neighbor PSCs as specified in section 3.3.3.
- A directory server receives out-of-sync directory changes as specified in sections 3.1.7.2.2 and 3.1.7.2.4.
- A directory server receives a ChangePropagationMessage (section 2.2.6.9) that contains a SeqNumberHeader, as specified in 3.2.7.1.
- A directory server receives a ChangePropagationMessage that triggers a new partition to be added in the directory, as specified in sections 3.1.7.2.5 and 3.1.7.2.11.

- A directory server receives an AlreadyPurgedMessage (section 2.2.6.8) in response to its previous SyncRequestMessage (section 2.2.6.4), as specified in section 3.3.7.10.
- A directory server receives a SyncReplyMessage (section 2.2.6.5) that has a field set to indicate that there are more changes from the sender, as specified in section 3.3.7.5.

Sending a synchronization request can be triggered by both the MQDSRP protocol client and the MQDSRP protocol server in the context of handling other events. After the request is triggered, the sender acts as the MQDSRP protocol client of a new protocol instance.

The following figure shows the message flow and sequence for handling synchronization requests and replies.

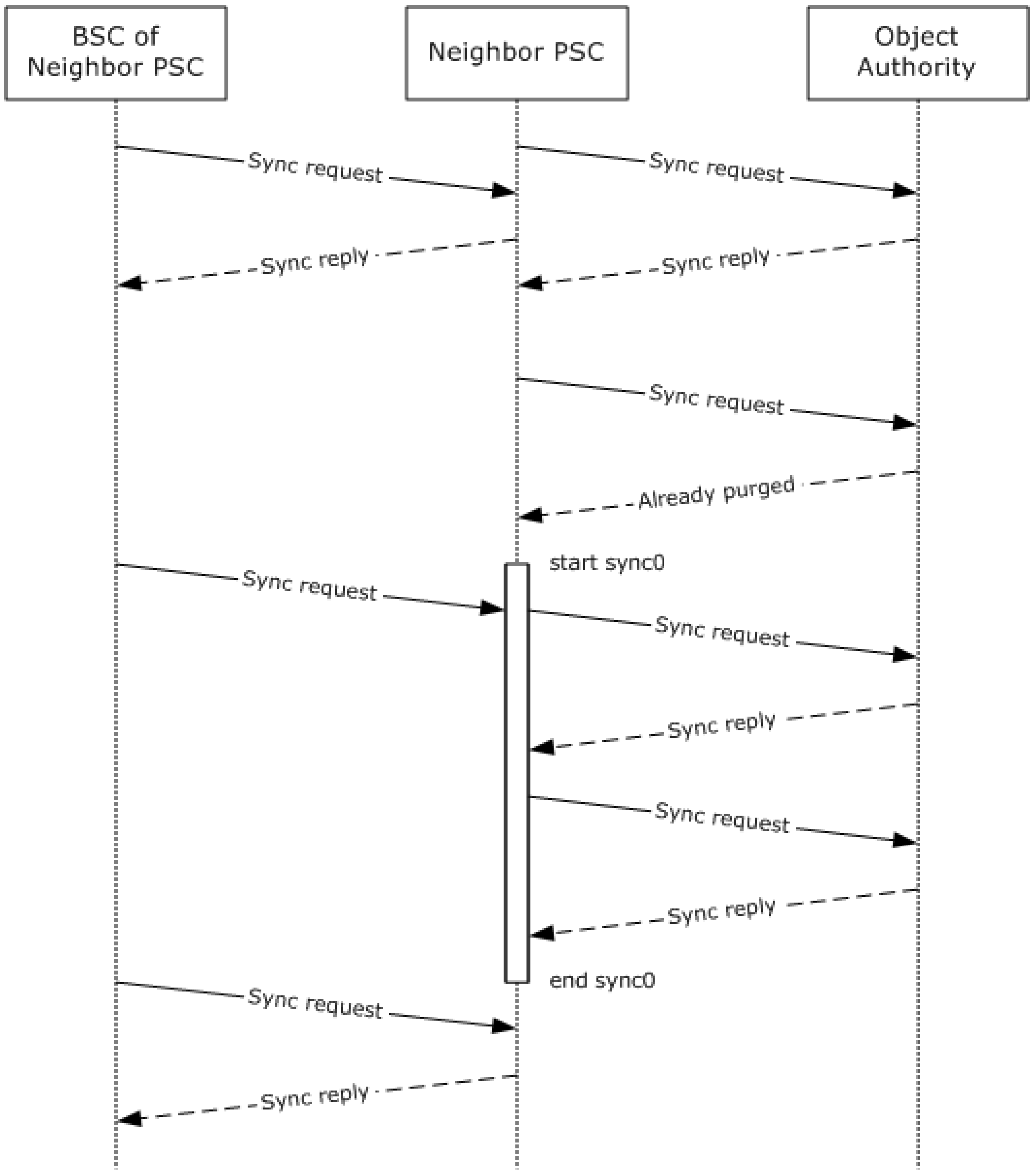

Figure 6: Synchronization request and reply sequence

The receiver of a SyncRequestMessage replies to the requester with either a SyncReplyMessage or an AlreadyPurgedMessage. A SyncReplyMessage contains the directory changes that satisfy the sender's request. An AlreadyPurgedMessage indicates a wrong state of the partition on the sender side that cannot be corrected without a full-partition synchronization.

After receiving an AlreadyPurgedMessage, the sender of the synchronization request starts a full-partition synchronization with the MQDSRP protocol server. During this time period, the partition that is maintained by the MQDSRP protocol client is marked as **sync0** state. The MQDSRP protocol client starts the **sync0** state by marking all objects in that partition as invalid and sending a SyncRequestMessage for all objects to the MQDSRP protocol server. The MQDSRP protocol client completes the **sync0** state when it receives all changes for that partition. Multiple synchronization requests may be needed if the MQDSRP protocol server cannot include all changes in one reply message. If a directory partition is marked as the **sync0** state, any synchronization request for that partition is discarded.

The receiver processes synchronization requests with its best efforts. If a request cannot be processed, no reply is sent. The sender is responsible for resending another request at a later time.

3.1.2 Timers

None.

3.1.3 Initialization

Both the MQDSRP server and MQDSRP client sides of the MQDSRP protocol MUST perform the following initializations:

- Replication state initialization
- Directory neighbor initialization
- Replication queue initialization

3.1.3.1 Replication State Initialization

The queue manager MUST initialize the replication state as follows:

- If **DirectoryPartitionCollection** is empty:
  - Create a new **DirectoryPartition** object, referred to as *entPartition*, with the following attributes:
    - PartitionID := PEC_PARTITION_ID, as defined in section 1.9.
    - AuthorityName: If the queue manager is a PEC, set it to **QueueManager.MachineName**; otherwise set it to the PEC machine name (typically provided by the administrator).
    - LastSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
    - PurgedSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
    - AllowedPurgeSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
    - PurgeState := 0 (PURGE_STATE_NORMAL)

  - Add *entPartition* to **DirectoryPartitionCollection**.
- Initialize the non-persisted attributes of each **DirectoryPartition** object (section 3.1.1.2.3), referred to as *dirPartition*, in **DirectoryPartitionCollection**, as follows:
  - Set *dirPartition.PreviousPurgedSeqNumber* to MIN_SEQ_NUMBER, as defined in section 2.2.2.
  - Set *dirPartition.ChangeMissingWindow* to MAX_SEQ_NUMBER, as defined in section 2.2.2.
  - Set *dirPartition.PendingChangeCollection* to empty.
- If the queue manager is a PEC:
  - Set **QueueManager.MyEnterprisePartition** to the element in **DirectoryPartitionCollection** whose PartitionID is **PEC_PARTITION_ID**, as defined in section 1.9.
- Find a **DirectoryPartition** object, referred to as *mySitePartition*, in **DirectoryPartitionCollection**, where:
  - *mySitePartition.PartitionID* is the same as **QueueManager.SiteIdentifier**.
- If *mySitePartition* is found:
  - Set **QueueManager.MySitePartition** to *mySitePartition*.
  - If QueueManager is BSC:
    - Set **QueueMananger.MyPSCName** to *mySitePartition.AuthorityName*.
- Else:
  - **QueueManager.MySitePartition** remains uninitialized. It will be initialized when a **ChangePropagationMessage** (section 2.2.6.9) or **SyncRequestMessage** (section 2.2.6.4) is received, as specified in sections 3.1.7.2.5 and 3.1.7.2.11.
  - If QueueManager is BSC:
    - Set **QueueManager.MyPSCName** to the machine name of the PSC of the site that the queue manager belongs to. How the PSC machine name is obtained is implementation-specific.

3.1.3.2 Directory Neighbor Initialization

The queue manager MUST initialize the directory neighbors as follows:

- Initialize the non-persisted attributes of each **DirectoryNeighbor** object (section 3.1.1.2.4), referred to as *dirPSCNeighbor*, in **PSCNeighborCollection** as follows:
  - Set *dirPSCNeighbor*.*AvailableChangeCollection* to empty.
- Initialize the non-persisted attributes of each DirectoryNeighbor object, referred to as *dirBSCNeighbor*, in **BSCNeighborCollection** as follows:
  - Set *dirBSCNeighbor*.*AvailableChangeCollection* to empty.

3.1.3.3 Replication Queue Initialization

The queue manager MUST initialize the replication queue as follows:

- Raise an **Open Queue** event ([MS-MQDMPR] section 3.1.7.1.5) with the following arguments:
  - *iFormatName* := REPLICATION_QUEUE_PRIVATE as defined in section 1.9, where <Queue Manager GUID> is set to **QueueManager.Identifier**.
  - *iRequiredAccess* := **ReceiveAccess**.
  - *iSharedMode* := **DenyNone**.
- if *rStatus* of the Open Queue event is not MQ_OK:
  - Take no further action. This protocol cannot work correctly.
- Set *replicationQueueOpenDescriptor* to *rOpenQueueDescriptor* of this event.
- Raise a **Dequeue Message** event ([MS-MQDMPR] section 3.1.7.1.10) with the following arguments:
  - *iQueueDesc* := *replicationQueueOpenDescriptor*.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Receiving Replication Messages

The queue manager MUST initialize the replication queue as specified in section 3.3.3. When the **Dequeue Message** event ([MS-MQDMPR] section 3.1.7.1.10) returns, the queue manager MUST process the returned *rMessage* as follows:

- Set *repMessage* to *rMessage.Body*.
- If *repMessage* is empty:
  - Take no further action.
- If *repMessage.BaseReplicationHeader.Version* is not 0x00:
  - Take no further action.
- If *rMessage.Class* is Normal:
  - Case *repMessage.BaseReplicationHeader.Operation* of:
    - 0x00: raise a **Receive Change Propagation** event (section 3.2.7.1) with the following arguments:
      - iChangePropagationMessage := *repMessage*
    - 0x01: raise a **Receive Change Request** event (section 3.2.7.2) with the following arguments:

            - iChangeRequestMessage := *repMessage*
        - 0x02: raise a **Receive Sync Request** event (section 3.2.7.3) with the following arguments:
            - iSyncRequestMessage := *repMessage*
        - 0x03: raise a **Receive Sync Reply** event (section 3.3.7.5) with the following arguments:
            - iSyncReplyMessage := *repMessage*
        - 0x04: raise a **Receive Change Reply** event (section 3.3.7.6) with the following arguments:
            - iChangeReplyMessage := *repMessage*
        - 0x05: raise a **Receive Already Purged** event (section 3.3.7.7) with the following arguments:
            - iAlreadyPurgedMessage := *repMessage*
        - 0x06: raise a **Receive PSC Ack** event (section 3.2.7.4) with the following arguments:
            - iPSCAckMessage := *repMessage*
        - 0x07: raise a **Receive BSC Ack** event (section 3.2.7.5) with the following arguments:
            - iBSCAckMessage := *repMessage*
        - Others: take no further action.
- Else if *rMessage.Class* is a Nack (a Nack class is a value from **NackBadDestQueue** to **NackReceiveTimeoutAtSender**, inclusive, defined in [MS-MQDMPR] section 3.1.1.12 under the **Message.Class** attribute):
    - Case *repMessage.BaseReplicationHeader.Operation* of:
        - 0x01: raise a **Receive Change Request Nack** event (section 3.3.7.8) with the following arguments:
            - iChangeRequestMessage := *repMessage*
        - 0x00, 0x02, 0x03, 0x04, 0x05, 0x06, or 0x07: raise a **Receive Request Nack** event (section 3.3.7.9) with the following arguments:
            - iNackMessage := *rMessage*
    - Others: take no further action.
- Else:
    - Take no further action.

After a replication message is processed, another Dequeue Message event MUST be generated as specified in section 3.1.3.3. This process is repeated during the lifetime of the queue manager.

3.1.5.2 Processing Directory Changes

Directory changes are processed by the queue manager during the handling of the following events:

- **Receive Change Propagation** (section 3.2.7.1) on the MQDSRP protocol server side.
- **Receive Sync Reply** (section 3.3.7.5) on the MQDSRP protocol client side (as a result of a previous **Send Sync Request** event (section 3.3.7.3)).

In either event, the replication message can carry multiple directory changes. For each directory change, referred to as *dirChange*, the queue manager MUST process it as follows:

- Find a **DirectoryPartition** object (section 3.1.1.2.3), referred to as *dirPartition*, in **DirectoryPartitionCollection**, where:
  - *dirPartition.PartitionID* is the same as *dirChange.PartitionID*.
- If *dirPartition* is not found:
  - Take no further action.
- Raise a **Handle Directory Change** event (section 3.1.7.2.2) with the following arguments:
  - iPartition := *dirPartition*
  - iChange := *dirChange*
  - iCheckFlush MUST be set as follows:
    - If it is a **Receive Change Propagation** event (section 3.2.7.1), set iCheckFlush to true when the **Flush** field in the **ChangePropagationMessage** (section 2.2.6.9) is 0x00, set iCheckFlush to false when the field is 0x01.
    - If it is a **Receive Sync Reply** event (section 3.3.7.5), set iCheckFlush to true.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

The events listed in this section are generated and subscribed by the protocol implementations.

3.1.7.1 Events Raised by Related Protocols

The events listed in this section are generated by the protocol implementations co-located with the queue manager for accessing various common abstract data model elements.

The queue manager MUST process the events listed in this section.

3.1.7.1.1 Change Remote Object

This event MUST be generated with the following arguments:

- *iOperation*: A **UCHAR** that indicates the directory operation.
- *iObjectType*: A value of **Directory Object Types** that identifies the directory object type.
- *iPathName*: A null-terminated **WCHAR** buffer that contains the path name of the directory object.

- *iGuidIdentifier*: A GUID that identifies the directory object.
- *iPartitionID*: A GUID that identifies the directory partition to which the change applies.
- *iNumberOfProperties*: A 32-bit unsigned integer that indicates the size (in elements) of the arrays *iPropertyID* and *iPropertyValue*. The arrays *iPropertyID* and *iPropertyValue* MUST have an identical number of elements.
- *iPropertyID*: An array of **PROPID** that identifies the object attributes that are included in the change.
- *iPropertyValue*: An array of **PROPVARIANT** that contains the property values associated with the property identifier in *iPropertyID*.

Return Values:

- *rStatus*: An **HRESULT** status code that indicates the result of this directory change operation.

The queue manager MUST perform the following actions to process this event:

- Create a **ChangeRequestMessage** object (section 2.2.6.2), referred to as *changeRequest*, with the following attributes:

    changeRequest.*ChangeRequestHeader*:

    - BaseReplicationHeader.Operation := 0x01.
    - PartitionID := *iPartitionID*.
    - RequestIdentifier: Must be set to an internal value (a 32-bit unsigned integer) that uniquely identifies the **Process Change Request** event (section 3.3.7.1) in the queue manager.
    - PSCNameOffset: If the queue manager is a PSC, set this field to 0x0. If the queue manager is a BSC, set this field to the length, in **WCHAR**, of **QueueManager.MachineName**, including the null-terminator.
    - RequesterName := **QueueManager.MachineName**.
    - PSCName: If the queue manager is a PSC, do not set this field. If the queue manager is a BSC, set it to **QueueManager.MyPSCName**.

    changeRequest.*DirectoryChange*:

    - Command := *iOperation*.
    - UseGuid := 0x00.
    - PathName := *iPathName*.
    - GuidIdentifier: Not set.
    - PartitionID := *iPartitionID*.
    - PreviousSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
    - SeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
    - PurgedSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.

        - NumberOfProperties := the number of elements in *iPropertyID*.
        - PropertyID:= *iPropertyID*.
        - PropertyValue:= *iPropertyValue*.
- Raise a **Process Change Request** event (section 3.3.7.1) with the following arguments:
    - iChangeRequestMessage := *changeRequest*
- Set *rStatus* to the return result of the Process Change Request event.

3.1.7.1.2 Propagate Directory Change

This event MUST be generated with the following arguments:

- *iOperation*: A **UCHAR** indicating the directory operation.
- *iObjectType*: A value of **Directory Object Types** identifying the directory object type.
- *iPathName*: A null-terminated **WCHAR** buffer that contains the path name of the directory object.
- *iGuidIdentifier*: A GUID that identifies the directory object.
- *iPartitionID*: A GUID that identifies the directory partition to which the change applies.
- *iNumberOfProperties*: A 32-bit unsigned integer indicating the size (in elements) of arrays *iPropertyID* and *iPropertyValue*. The arrays *iPropertyID* and *iPropertyValue* MUST have an identical number of elements.
- *iPropertyID*: An array of **PROPID** that identifies the object attributes included in the change.
- *iPropertyValue*: An array of **PROPVARIANT** that contains the property values associated with the property identifier in *iPropertyID*.
- *iSeqNumber*: A SEQUENCE_NUMBER element (section 2.2.2) that contains the sequence number of the directory change.

Return Values:

- None

The queue manager MUST perform the following actions to process this event:

- If the queue manager is a PEC and *iPartitionID* is equal to **QueueManager.MyEnterprisePartition.PartitionID**:
    - Set *dirPartition* to **QueueManager.MyEnterprisePartition**.
- Else if the queue manager is a PSC, **QueueManager.MySitePartition** is initialized, and *iPartitionID* is equal to **QueueManager.MySitePartition.PartitionID**:
    - Set *dirPartition* to **QueueManager.MySitePartition**.
- Else
    - Take no further action.

- Set *status* to MQ_OK. Handle replication state change, if any, as follows:
  - Case *iOperation* of:
    - 0x00 (create):
      - Raise a **Handle Directory Create** event (section 3.1.7.2.5) with the following arguments:
        - iPartition := *dirPartition*
        - iObjectType := *iObjectType*
        - iPathName := *iPathName*
        - iNumberOfProperties := *iNumberOfProperties*
        - iPropertyID := *iPropertyID*
        - iPropertyValue := *iPropertyValue*
        - iSeqNumber := *iSeqNumber*
        - iPostCreate := true
      - Set *status* to the return value *rStatus* of the Handle Directory Create event.
    - 0x01 (update):
      - Raise a **Handle Directory Update** event (section 3.1.7.2.6) with the following arguments:
        - iPartition := *dirPartition*
        - iObjectType := *iObjectType*
        - iPathName := *iPathName*
        - iGuidIdentifier := *iGuidIdentifier*
        - iNumberOfProperties := *iNumberOfProperties*
        - iPropertyID := *iPropertyID*
        - iPropertyValue := *iPropertyValue*
        - iSeqNumber := *iSeqNumber*
        - iPostUpdate := true
      - Set *rStatus* to the return value *rStatus* of the Handle Directory Update event.
    - 0x02 (delete):
      - Raise a **Handle Directory Delete** event (section 3.1.7.2.7) with the following arguments:
        - iPartition := *dirPartition*.
        - iObjectType := the property value of the second element in *iPropertyValue*.

                - iScope := the property value of the first element in *iPropertyValue*.
                - iPathName := *iPathName*.
                - iGuidIdentifier := *iGuidIdentifier*.
                - iSeqNumber := *iSeqNumber*.
                - iPostDelete := true.
            - Set *rStatus* to the return value *rStatus* of the Handle Directory Delete event.
        - 0x03 (synchronize):
            - Raise a Handle Directory Sync event (section 3.1.7.2.8) with the following arguments:
                - iPartition := *dirPartition*.
                - iObjectType := the property value of the second element in *iPropertyValue*.
                - iPathName := iPathName.
                - iGuidIdentifier := iGuidIdentifier.
                - iNumberOfProperties := *iNumberOfProperties*.
                - iPropertyID := *iPropertyID*.
                - iPropertyValue := *iPropertyValue*.
                - iSeqNumber := *iSeqNumber*.
            - Set *rStatus* to the return value *rStatus* of the Handle Directory Sync event.
- If *status* is not MQ_OK:
    - Take no further action.
- Create a **DirectoryChange** object (section 2.2.4), referred to as *dirChange*, with the following attributes:
    - Command := *iOperation*.
    - UseGuid: If *iGuidIdentifier* is NULL, set it to 0x00; otherwise set it to 0x01.
    - PathName: If *iPathName* is NULL, do not set this field; otherwise set it to *iPathName*.
    - GuidIdentifier: If *iGuidIdentifier* is NULL, do not set this field; otherwise set it to *iGuidIdentifier*.
    - PartitionID := *iPartitionID*.
    - PreviousSeqNumber := *dirPartition*.*LastSeqNumber*.
    - SeqNumber := *iSeqNumber*.
    - PurgedSeqNumber := *iSeqNumber*.
    - NumberOfProperties := *iNumberOfProperties*.

    - PropertyID := i*PropertyID*.
    - PropertyValue := *iPropertyValue*.
- For each **DirectoryNeighbor** object (section 3.1.1.2.4), referred to as *dirPSCNeighbor*, in **PSCNeighborCollection**:
    - Append *dirChange* to *dirPSCNeighbor.AvailableChangeCollection*.
- For each DirectoryNeighbor object, referred to as *dirBSCNeighbor*, in **BSCNeighborCollection**:
    - Append *dirChange* to *dirBSCNeighbor.AvailableChangeCollection*.
- Set *dirPartition.LastSeqNumber* to *iSeqNumber*.

3.1.7.1.3 Read Next Sequence Number

This event MUST be generated with the following arguments:

- *iPartitionID*: A GUID identifying the directory partition for which the next sequence number is returned.

Return Values:

- *rNextSeqNumber*: A **SEQUENCE_NUMBER** element (section 2.2.2) that contains the sequence number of the next directory change in the specified directory partition.

The queue manager MUST perform the following actions to process this event:

- If the queue manager is a PEC and *iPartitionID* is equal to **QueueManager.MyEnterprisePartition.PartitionID**:
    - Set *dirPartition* to **QueueManager.MyEnterprisePartition**.
- Else if the queue manager is a PSC, **QueueManager.MySitePartition** is initialized, and *iPartitionID* is equal to **QueueManager.MySitePartition.PartitionID**:
    - Set *dirPartition* to **QueueManager.MySitePartition**.
- Else
    - Set *rNextSeqNumber* to MIN_SEQ_NUMBER (section 2.2.2).
    - Take no further action.
- Set *rNextSeqNumber* to *dirPartition*.*LastSeqNumber*.
- Increment *rNextSeqNumber* as specified in section 2.2.2.

3.1.7.2 Shared Events

The events listed in this section are raised by both the MQDSRP protocol server and the MQDSRP protocol client to facilitate their event processing.

The queue manager MUST process the events listed in this section.

3.1.7.2.1 Send Replication Message

This event MUST be generated with the following arguments:

- *iDestination*: A null-terminated **WCHAR** buffer that contains the destination machine name.
- *iReplicationMessage*: A reference to a data element inherited from **BaseReplicationMessage** (section 2.2.6.1).
- *iTimeout*: The message timeout in seconds.
- *iPriority*: The message priority.
- *iAckRequested*: Acknowledgment requested for this message.
- *iAdminQueueFormatName*: The administration queue format name.
- *iResponseQueueFormatName*: The response queue format name.

Return Values:

- None

The queue manager MUST perform the following actions to process this event:

- Create a **Message** object, referred to as *outMessage*, with the following attributes:
  - Class :=**Normal**.
  - TimeToReachQueue:= *iTimeout*
  - TimeToBeReceived:= *iTimeout*
  - Priority:= *iPriority*
  - DeliveryGuarantee := **Express**
  - AcknowledgmentRequested:= *iAckRequested*
  - AdministrationQueueFormatName: If *iAdminQueueFormatName* is not NULL, set this field to *iAdminQueueFormatName*; otherwise do not set the field.
  - ResponseQueueFormatName: If *iResponseQueueFormatName* is not NULL, set this field to *iResponseQueueFormatName*; otherwise do not set the field.
  - Body := *iReplicationMessage*.
  - HashAlgorithm := CALG_MD5. The queue manager MUST sign the message with this algorithm. The message properties covered by the signature are specified in [MS-MQMQ] section 2.2.20.6. For more details on the MD5 hash algorithm, see [RFC1321].
  - SenderIdentifierType := 2
  - SenderIdentifier := **QueueManager.Identifier**

  For fields not specified here, values MUST be used as specified in [MS-MQMQ] and [MS-MQQB].
- Set *outFormatName* to REPLICATION_QUEUE_DIRECT where <Machine Name> := *iDestination*.

- Find an **OpenQueueDescriptor** object, referred to as *outOpenQueueDescriptor*, in the **OpenQueueDescriptorCollection** of each **Queue** object in **QueueManager.QueueCollection**, where:
  - *outOpenQueueDescriptor.FormatName* is the same as *outFormatName*.
- If *outOpenQueueDescriptor* is not found:
  - Raise an **Open Queue** event with the following arguments:
    - iFormatName := *outFormatName*
    - iRequiredAccess := **SendAccess**
    - iSharedMode := **DenyNone**
  - Set *outOpenQueueDescriptor* to *rOpenQueueDescriptor* of the **Open Queue** event.
- Generate an **Enqueue Message** event with the following arguments:
  - iQueue := *outOpenQueueDescriptor*
  - iMessage := *outMessage*

3.1.7.2.2 Handle Directory Change

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) representing the directory partition with which the directory change is associated.
- *iChange*: A reference to a **DirectoryChange** object (section 2.2.4).
- *iCheckFlush*: A Boolean value indicating whether the change should be flushed to the BSCs of the site to which the queue manager belongs.

Return Values:

- None

The queue manager MUST perform the following actions to process this event:

- If *iChange.SeqNumber* <= *iPartition.LastSeqNumber* or *iChange.PurgedSeqNumber* < *iPartition.PurgedSeqNumber*:
  - Take no further action.
- If *iChange.PreviousSeqNumber* <= *iPartition.LastSeqNumber*:
  - Raise a **Handle In-Sync Change** event (section 3.1.7.2.3) with the following arguments:
    - iPartition := *iPartition*
    - iChange := *iChange*
    - iCheckFlush := *iCheckFlush*
  - If *rStatus* of the Handle In-Sync Change event is not MQ_OK:

        - Take no further action.
    - Raise a **Check Pending Changes** event (section 3.1.7.2.4) with the following arguments:
        - iPartition := *iPartition*
- Else (The change is out of order. Queue and process it when the previous ones come):
    - If *iPartition.PendingChangeCollection* is empty:
        - Append *iChange* to *iPartition.PendingChangeCollection*.
        - If *iChange.SeqNumber* > *iPartition.ChangeMissingWindow*:
            - Set *iPartition.ChangeMissingWindow* to *iChange.SeqNumber*.
            - Raise a **Send Sync Request** event (section 3.3.7.3) with the following argument:
                - iPartition := *iPartition*
    - Else if the size of *iPartition.PendingChangeCollection* < 100:
        - If no element in *iPartition.PendingChangeCollection* whose **SeqNumber** field matches *iChange.SeqNumber*:
            - Append *iChange* to *iPartition.PendingChangeCollection*.
            - Sort *iPartition.PendingChangeCollection* by the **SeqNumber** field in ascending order.
    - Else:
        - Take no further action.

3.1.7.2.3 Handle In-Sync Change

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) representing the directory partition with which the directory change is associated.
- *iChange*: A reference to a **DirectoryChange** object (section 2.2.4).
- *iFlush*: A Boolean value indicating whether the change should be flushed to the BSCs of the site to which the queue manager belongs.

Return Values:

- *rStatus*: An **HRESULT** status code.

The queue manager MUST perform the following actions to process this event:

- Set *needFlush* to false.
- Use the first **PROPID** element in *iChange*.*PropID* to determine the object type, referred to as *objectType*, that is associated with this change, as specified in section 3.1.1.4.
- Set *pathName* to NULL.
- Set *guidIdentifier* to NULL.

- If *iChange*.*UseGuid* is 0x00:
  - Set *pathName* to *iChange*.*PathName*.
- Else
  - Set *guidIdentifier* to *iChange*.*GuidIdentifier*.
- Case *iChange.Command* of:
  - 0x00 (create):
    - Raise a **Handle Directory Create** event (section 3.1.7.2.5) with the following arguments:
      - iPartition := *iPartition*
      - iObjectType := *objectType*
      - iPathName := *pathName*
      - iNumberOfProperties := *iChange*.*NumberOfProperties*
      - iPropertyID := *iChange*.*PropertyID*
      - iPropertyValue := *iChange*.*PropertyValue*
      - iSeqNumber := *iChange*.*SeqNumber*
      - iPostCreate := false
    - Set *rStatus* to the return value *rStatus* of the Handle Directory Create event.
    - Set *needFlush* to the return value *rNeedFlush* of the Handle Directory Create event.
  - 0x01 (update):
    - Raise a **Handle Directory Update** event (section 3.1.7.2.6) with the following arguments:
      - iPartition := *iPartition*
      - iObjectType := *objectType*
      - iPathName := *pathName*
      - iGuidIdentifier := *guidIdentifier*
      - iNumberOfProperties := *iChange.NumberOfProperties*
      - iPropertyID := *iChange.PropertyID*
      - iPropertyValue := *iChange.PropertyValue*
      - iSeqNumber := *iChange.SeqNumber*
      - iPostUpdate := false
    - Set *rStatus* to the return value *rStatus* of the Handle Directory Update event.
    - Set *needFlush* to the return value *rNeedFlush* of the Handle Directory Update event.

- 0x02 (delete):
  - Raise a **Handle Directory Delete** event (section 3.1.7.2.7) with the following arguments:
    - iPartition := *iPartition*
    - iObjectType := the property value of the second element in *iChange.PropertyValue*
    - iScope := the property value of the first element in *iChange.PropertyValue*
    - iPathName := *pathName*
    - iGuidIdentifier := *guidIdentifier*
    - iSeqNumber := *iChange.SeqNumber*
    - iPostDelete := false
  - Set *rStatus* to the return value *rStatus* of the Handle Directory Delete event.
  - Set *needFlush* to the return value *rNeedFlush* of the Handle Directory Delete event.
- 0x03 (sync):
  - Raise a **Handle Directory Sync** event (section 3.1.7.2.8) with the following arguments:
    - iPartition := *iPartition*
    - iObjectType := *objectType*
    - iPathName := *pathName*
    - iGuidIdentifier := *guidIdentifier*
    - iNumberOfProperties := *iChange.NumberOfProperties*
    - iPropertyID := *iChange.PropertyID*
    - iPropertyValue := *iChange.PropertyValue*
    - iSeqNumber := *iChange.SeqNumber*
  - Set *rStatus* to the return value *rStatus* of the Handle Directory Sync event.
  - Set *needFlush* to the return value *rNeedFlush* of the Handle Directory Sync event.

- If *rStatus* is not MQ_OK:
  - Take no further action.
- Set *iPartition.LastSeqNumber* to *iChange.SeqNumber*.
- If *iPartition.PurgeState* is 0 (PURGE_STATE_NORMAL):
  - For each **DirectoryNeighbor** object (section 3.1.1.2.4), referred to as *dirBSCNeighbor*, in **BSCNeighborCollection**:
    - Append *iChange* to *dirBSCNeighbor.AvailableChangeCollection*.
    - If *iFlush* is true and *needFlush* is true:

            - Raise a **Send Change Propagation** event section (section 3.3.7.2) with the following arguments:
                - iNeighbor:= *dirBSCNeighbor*
                - iScope := **Intrasite**
        - If *iPartition.LastSeqNumber* > (*iPartition.PreviousPurgedSeqNumber* + 256):
            - Raise a **Purge Deleted Objects** event (section 3.1.7.2.9) with the following arguments:
                - iPartition := *iPartition*
            - Set *iPartition.PreviousPurgedSeqNumber* to *iPartition.LastSeqNumber*.
- If *iPartition.PurgeState* is 0 (PURGE_STATE_NORMAL) or *iPartition.LastSeqNumber* >= *iPartition.PurgedSeqNumber*:
    - If the queue manager is a PSC and the eighth element in *iPartition.LastSeqNumber* is 0x00:
        - Raise a **Send PSC Ack** event (section 3.3.7.4) with the following arguments:
            - iPartition := *iPartition*
            - iLastSeqNumber := *iPartition.LastSeqNumber*
- Set *rStatus* to MQ_OK.

3.1.7.2.4 Check Pending Changes

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) representing the directory partition whose pending directory changes will be checked.

Return Values:

- None

The queue manager MUST perform the following actions to process this event:

- For each **DirectoryChange** object (section 2.2.4), referred to as *dirPendingChange* in *iPartition.PendingChangeCollection*:
    - If *dirPendingChange.PurgedSeqNumber* < *iPartition.PurgedSeqNumber*:
        - Remove *dirPendingChange* from the collection.
        - Continue the loop.
    - If *dirPendingChange.SeqNumber* > *iPartition.LastSeqNumber*:
        - If *dirPendingChange.SeqNumber* > *iPartition.ChangeMissingWindow*:
            - Raise a **Send Sync Request** event (section 3.3.7.3) with the following argument:
                - iPartition := *iPartition*.
            - Take no further action.

    - Remove *dirPendingChange* from the collection.
    - If *dirPendingChange.PreviousSeqNumber* is the same as *iPartition.LastSeqNumber*:
        - Raise a **Handle In-Sync Change** event (section 3.1.7.2.3) with the following arguments:
            - iPartition := *iPartition*
            - iChange := *dirPendingChange*
            - iFlush := true

3.1.7.2.5 Handle Directory Create

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) representing the directory partition with which the directory change is associated.
- *iObjectType*: A value of **Directory Object Types** identifying the directory object type.
- *iPathName*: A null-terminated **WCHAR** buffer that contains the path name of the directory object.
- *iNumberOfProperties*: A 32-bit unsigned integer indicating the size (in elements) of arrays *iPropertyID* and *iPropertyValue*. The arrays *iPropertyID* and *iPropertyValue* MUST have identical numbers of elements.
- *iPropertyID*: An array of **PROPID** that identifies the attributes for creating the object.
- *iPropertyValue*: An array of **PROPVARIANT** that contains the property values associated with all property identifiers in *iPropertyID*.
- *iSeqNumber*: A sequence number associated with this directory change.
- *iPostCreate*: A Boolean value indicating whether this event is triggered after the creation operation is already performed. If the value is true, this event only handles the protocol state changes.

Return Values:

- *rStatus*: An **HRESULT** status code.
- *rNeedFlush*: A Boolean value indicating whether the change should be flushed to the BSCs of the site to which the queue manager belongs.

The queue manager MUST perform the following actions to process this event:

- Set *rNeedFlush* to false.
- If (queue manager is a PEC and *iPartition* is **QueueManager.MyEnterprisePartition**) or *iPartition* is **QueueManager.MySitePartition**:
    - Set *myPartition* to true.
- Else:
    - Set *myPartition* to false.

- If *iPostCreate* is false:
  - Raise an [MS-MQDS] **Create Directory Object** event (section 3.2.6.1) with the following arguments:
    - iObjectType := *iObjectType*
    - iPathName := *iPathName*
    - iNumberOfProperties := *iNumberOfProperties*
    - iPropertyID := *iPropertyID*
    - iPropertyValue := *iPropertyValue*
    - iPartitionID := *iPartition.PartitionID*
    - iSeqNumber := *iSeqNumber*
    - iReplicate := *myPartition*
  - Set *rStatus* to the return value *rStatus* of the Create Directory Object event.
  - If *rStatus* is not MQ_OK:
    - Take no further action.
- If *iObjectType* is MQDS_SITE:
  - Set *pscName* to the property value in *iPropertyValue* associated with the property identifier PROPID_S_PSC in *iPropertyID*.
  - Create a new DirectoryPartition object, referred to as *newPartition*, with the following attributes:
    - PartitionID := the property value in *iPropertyValue* associated with the property identifier PROPID_S_SITEID in *iPropertyID*.
    - AuthorityName := *pscName*.
    - LastSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
    - PurgedSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
    - AllowedPurgeSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
    - PurgeState := 0 (PURGE_STATE_NORMAL).
  - Add *newPartition* to **DirectoryPartitionCollection**.
  - If *pscName* is different from **QueueManager.MachineName**:
    - Raise a **Send Sync Request** event (section 3.3.7.3) with the following arguments:
      - iPartition := *newPartition*
    - Create a new **DirectoryNeighbor** object (section 3.1.1.2.4), referred to as *newPSCNeighbor*, with the following attributes:

        - PartitionID := *iPartition.PartitionID*
        - MachineName := *pscName*
        - AckedSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
        - AckedPECSeqNumber := MIN_SEQ_NUMBER, as defined in section 2.2.2.
      - Add *newPSCNeighbor* to **PSCNeighborCollection**
      - Start a **Change Propagation Timer** (section 3.3.2.1) with the following arguments:<2>
        - iNeighbor := *newPSCNeighbor*
        - iScope := **Intersite**
    - Else:
      - Set **QueueManager.MySitePartition** to *newPartition*.
    - Set *rNeedFlush* to true.
- If the queue manager is a PSC and *iObjectType* is MQDS_MACHINE:
  - Set *qmService* to the property value associated with the property identifier PROPID_QM_SERVICE in *iPropertyID*.
  - If *qmService* is SERVICE_BSC (0x2) and *iPartition* is the same as **QueueManager.MySitePartition**:
    - Create a new DirectoryNeighbor object, referred to as *newBSCNeighbor*, with the following attributes:
      - PartitionID := *iPartition.PartitionID*
      - MachineName := *pathName*
      - LastAckedTime := 0
    - Add *newBSCNeighbor* to **BSCNeighborCollection**.
    - Start a Change Propagation Timer with the following arguments:<3>
      - iNeighbor := *newBSCNeighbor*
      - iScope := **Intrasite**
- Set *rStatus* to MQ_OK.

3.1.7.2.6 Handle Directory Update

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) that represents the directory partition with which the directory change is associated.
- *iObjectType*: A value of **Directory Object Types** that identifies the directory object type.

- *iPathName*: A null-terminated **WCHAR** buffer that contains the path name of the directory object.
- *iGuidIdentifier*: A GUID that identifies the directory object.
- *iNumberOfProperties*: A 32-bit unsigned integer that indicates the size (in elements) of the arrays *iPropertyID* and *iPropertyValue*. The arrays *iPropertyID* and *iPropertyValue* MUST have an identical number of elements.
- *iPropertyID*: An array of **PROPID** that identifies the attributes for updating the object.
- *iPropertyValue*: An array of **PROPVARIANT** that contains the property values that are associated with all property identifiers in *iPropertyID*.
- *iSeqNumber*: A sequence number that is associated with this directory change.
- *iPostUpdate*: A Boolean value that indicates whether this event is triggered after the update operation is already performed. If it is true, this event only handles the protocol state changes.

Return Values:

- *rStatus*: An **HRESULT** status code.
- *rNeedFlush*: A Boolean value that indicates whether the change should be flushed to the BSCs of the site to which the queue manager belongs.

The queue manager MUST perform the following actions to process this event:

- Set *rNeedFlush* to false.
- If (queue manager is PEC and *iPartition* is **QueueManager.MyEnterprisePartition**) or *iPartition* is **QueueManager.MySitePartition**:
  - Set *myPartition* to true.
- Else:
  - Set *myPartition* to false.
- If *iPostUpdate* is false:
  - Raise an [MS-MQDS] **Update Directory Object** event with the following arguments:
    - iObjectType := *iObjectType*
    - iPathName := *iPathName*
    - iGuidIdentifier := *iGuidIdentifier*
    - iNumberOfProperties := *iNumberOfProperties*
    - iPropertyID := *iPropertyID*
    - iPropertyValue := *iPropertyValue*
    - iPartitionID := *iPartition.PartitionID*
    - iSeqNumber := *iSeqNumber*

        - iReplicate := *myPartition*
    - Set *rStatus* to the return value *rStatus* of the **Update Directory Object** event.
    - If *rStatus* is not MQ_OK:
        - Take no further action.
- If *iObjectType* is MQDS_SITE:
    - Raise a **Handle Site Update** event (section 3.1.7.2.11) with the following arguments:
        - iPathName := *iPathName*
        - iGuidIdentifier := *iGuidIdentifier*
        - iNumberOfProperties := *iNumberOfProperties*
        - iPropertyID := *iPropertyID*
        - iPropertyValue := *iPropertyValue*
        - iSync := false
    - Set *rStatus* to the return value *rStatus* of the Handle Site Update event.
    - Set *rNeedFlush* to the return value *rNeedFlush* of the Handle Site Update event.
- If *iObjectType* is MQDS_ENTERPRISE:
    - Raise a **Handle Enterprise Update** event (section 3.1.7.2.10) with the following arguments:
        - iPartition := *iPartition*
        - iNumberOfProperties := *iNumberOfProperties*
        - iPropertyID := *iPropertyID*
        - iPropertyValue := *iPropertyValue*
    - Set *rStatus* to the return value *rStatus* of the Handle Enterprise Update event.
    - Set *rNeedFlush* to the return value *rNeedFlush* of the Handle Enterprise Update event.
- If queue manager is a PSC and *iObjectType* is MQDS_MACHINE and *iPartition* is the same as **QueueManager.MySitePartition**:
    - Raise a **Handle Machine Update** event (section 3.1.7.2.12) with the following arguments:
        - iPartition := *iPartition*
        - iPathName := *iPathName*
        - iGuidIdentifier := *iGuidIdentifier*
        - iNumberOfProperties := *iNumberOfProperties*
        - iPropertyID := *iPropertyID*
        - iPropertyValue := *iPropertyValue*

        - iSync := false
    - Set *rStatus* to the return value *rStatus* of the Handle Machine Update event.
    - Set *rNeedFlush* to the return value *rNeedFlush* of the Handle Machine Update event.

3.1.7.2.7 Handle Directory Delete

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) that represents the directory partition with which the directory change is associated.
- *iObjectType*: A value of **Directory Object Types** that identifies the directory object type.
- *iScope*: A **UCHAR** that indicates the scope of the deleted object.
- *iPathName*: A null-terminated **WCHAR** buffer that contains the path name of the directory object.
- *iGuidIdentifier*: A GUID that identifies the directory object.
- *iSeqNumber*: A sequence number that is associated with this directory change.
- *iPostDelete*: A Boolean value that indicates whether this event is triggered after the deletion operation is already performed. If it is true, this event only handles the protocol state changes.

Return Values:

- *rStatus*: An **HRESULT** status code.
- *rNeedFlush*: A Boolean value that indicates whether the change should be flushed to the BSCs of the site to which the queue manager belongs.

The queue manager MUST perform the following actions to process this event:

- Set *rNeedFlush* to false.
- If (queue manager is a PEC and *iPartition* is **QueueManager.MyEnterprisePartition**) or *iPartition* is **QueueManager.MySitePartition**:
    - Set *myPartition* to true.
- Else:
    - Set *myPartition* to false.
- If *iPostDelete* is false:
    - Raise an [MS-MQDS] **Delete Directory Object** event (section 3.2.6.2) with the following arguments:
        - iObjectType := *iObjectType*
        - iPathName := *iPathName*
        - iGuidIdentifier := *iGuidIdentifier*
        - iScope := *iScope*

        - iReplicate := *myPartition*
    - Set *rStatus* to the return value of the Delete Directory Object event.
    - If *rStatus* is not MQ_OK:
        - Take no further action.
- If queue manager is a PEC and *iObjectType* is MQDS_SITE:
    - Find a DirectoryPartition object, referred to as *dirPartition*, in **DirectoryPartitionCollection**, where:
        - *dirPartition.PartitionID* is the same as *iGuidIdentifier*.
    - If *dirPartition* is found:
        - Remove *dirPartition* from **DirectoryPartitionCollection**.
        - Remove any **DirectoryNeighbor** object (section 3.1.1.2.4) whose <MachineName> is the same as *dirPartition.AuthorityName* from **PSCNeighborCollection**.
    - Set *rNeedFlush* to true.
- If queue manager is a PSC and *iObjectType* is MQDS_MACHINE:
    - Remove any DirectoryNeighbor object, referred to as *dirBSCNeighbor*, from **BSCNeighborCollection** where:
        - *dirBSCNeighbor.MachineName* is the same as *iPathName*.
- Set *rStatus* to MQ_OK.

3.1.7.2.8 Handle Directory Sync

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) that represents the directory partition with which the directory change is associated.
- *iObjectType*: A value of **Directory Object Types** that identifies the directory object type.
- *iPathName*: A null-terminated **WCHAR** buffer that contains the path name of the directory object.
- *iGuidIdentifier*: A GUID that identifies the directory object.
- *iNumberOfProperties*: A 32-bit unsigned integer that indicates the element size of the arrays *iPropertyID* and *iPropertyValue*. The arrays *iPropertyID* and *iPropertyValue* MUST have an identical number of elements.
- *iPropertyID*: An array of **PROPID** that identifies the attributes for synchronizing the object.
- *iPropertyValue*: An array of **PROPVARIANT** that contains the property values that are associated with all property identifiers in *iPropertyID*.
- *iSeqNumber*: A sequence number that is associated with this directory change.

Return Values:

- *rStatus*: An **HRESULT** status code.
- *rNeedFlush*: A Boolean value that indicates whether the change should be flushed to the BSCs of the site to which the queue manager belongs.

The queue manager MUST perform the following actions to process this event:

- Set *rNeedFlush* to false.
- Raise a [MS-MQDS] **Read Directory Object** event with the following arguments:
  - iObjectType := *iObjectType*
  - iPathName := *iPathName*
  - iGuidIdentifier := *iGuidIdentifier*
  - iNumberOfProperties := *iNumberOfProperties*
  - iPropertyID: Contains one **PROPID** element that is set according to *iObjectType*, as follows.

iObjectType	iPropertyID
MQDS_ENTERPRISE	PROPID_E_ID
MQDS_SITE	PROPID_S_ID
MQDS_ROUTINGLINK	PROPID_L_ID
MQDS_CN	PROPID_CN_ID
MQDS_USER	PROPID_U_ID
MQDS_MACHINE	PROPID_QM_ID
MQDS_QUEUE	PROPID_Q_ID

  - iPropertyValue: Contains one **PROPVARIANT** element with vt set to VT_NULL.
- Set *rStatus* to the return value *rStatus* of the **Read Directory Object** event.
- If *rStatus* is MQDS_OBJECT_NOT_FOUND:
  - Raise a [MS-MQDS] **Create Directory Object** event (section 3.2.6.1) with the following arguments:
    - iObjectType := *iObjectType*
    - iPathName := *iPathName*
    - iNumberOfProperties := *iNumberOfProperties*
    - iPropertyID := *iPropertyID*
    - iPropertyValue := *iPropertyValue*
    - iPartitionID := *iPartition.PartitionID*
    - iSeqNumber := *iSeqNumber*

        - iReplicate := false
    - Set *rStatus* to the return value *rStatus* of the Create Directory Object event.
- Else if *rStatus* is MQ_OK:
    - Raise an [MS-MQDS] **Update Directory Object** event with the following arguments:
        - iObjectType := *iObjectType*
        - iPathName := *iPathName*
        - iNumberOfProperties := *iNumberOfProperties*
        - iPropertyID := *iPropertyID*
        - iPropertyValue := *iPropertyValue*
        - iPartitionID := *iPartition.PartitionID*
        - iSeqNumber := *iSeqNumber*
        - iReplicate := false
    - Set *rStatus* to the return value *rstatus* of the **Update Directory Object** event.
- If *rStatus* is not MQ_OK:
    - Take no further action.
- If *objectType* is MQDS_ENTERPRISE:
    - Raise a **Handle Enterprise Update** event (section 3.1.7.2.10) with the following arguments:
        - iPartition := *iPartition*
        - iNumberOfProperties := *iNumberOfProperties*
        - iPropertyID := *iPropertyID*
        - iPropertyValue := *iPropertyValue*
    - Set *rStatus* to the return value *rStatus* of the Handle Enterprise Update event.
    - Set *rNeedFlush* to the return value *rNeedFlush* of the Handle Enterprise Update event.
- If *objectType* is MQDS_SITE:
    - Raise a **Handle Site Update** event (section 3.1.7.2.11) with the following arguments:
        - iPathName := *iPathName*
        - iGuidIdentifier := *iGuidIdentifier*
        - iNumberOfProperties := *iNumberOfProperties*
        - iPropertyID := *iPropertyID*
        - iPropertyValue := *iPropertyValue*

        - iSync := true
    - Set *rStatus* to the return value *rStatus* of the Handle Site Update event.
    - Set *rNeedFlush* to the return value *rNeedFlush* of the Handle Site Update event.
- If the queue manager is a PSC and *objectType* is MQDS_MACHINE:
    - Raise a **Handle Machine Update** event (section 3.1.7.2.12) with the following arguments:
        - iPartition := *iPartition*
        - iPathName := *iPathName*
        - iGuidIdentifier := *iGuidIdentifier*
        - iNumberOfProperties := *iNumberOfProperties*
        - iPropertyID := *iPropertyID*
        - iPropertyValue := *iPropertyValue*
    - Set *rStatus* to the return value *rStatus* of the Handle Machine Update event.
    - Set *rNeedFlush* to the return value *rNeedFlush* of the Handle Machine Update event.

3.1.7.2.9 Purge Deleted Objects

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) that represents the directory partition from which the deleted objects are purged.

Return Values:

- None

The queue manager MUST perform the following actions to process this event:

- Set *purgeSeqNumber* to (*iPartition.LastSeqNumber* - 1024). This is to have a buffer for deleted objects. Deleted objects within this buffer are not purged. By having a buffer, the protocol reduces the chance of a full-partition synchronization.
- If *iPartition.PurgeState* is not 0 (PURGE_STATE_NORMAL) or *iPartition.PurgedSeqNumber* >= *purgeSeqNumber*:
    - Take no further action.
- If (queue manager is PEC and *iPartition* is the same as **QueueManager.MyEnterprisePartition**) or (queue manager is PSC and *iPartition* is the same as **QueueManager.MySitePartition**):
    - If *iPartition.PartitionID* is PEC_PARTITION_ID as defined in section 1.9:
        - Set *minAckedSeqNumber* to the minimum **AckedPECSeqNumber** field of objects in **PSCNeighborCollection**.
    - Else:

        - Set *minAckedSeqNumber* to the minimum **AckedSeqNumber** field of objects in **PSCNeighborCollection**.
    - If *minAckedSeqNumber* <= *iPartition.PurgedSeqNumber*:
        - Take no further action because not all PSCs have acknowledged the deletion operations.
    - Set *purgeSeqNumber* to *minAckedSeqNumber*.
- Else (queue manager is not the object authority of *iPartition*):
    - If *iPartition.AllowedPurgeSeqNumber* < *purgeSeqNumber*:
        - If *iPartition.AllowedPurgeSeqNumber* < *iPartition.PurgedSeqNumber*:
            - Take no further action.
        - Set *purgeSeqNumber* to *iPartition.AllowedPurgeSeqNumber*.
- Remove each object, referred to as *objDeleted*, from **DeletedObjectCollection**, where:
    - *objDeleted.PartitionID* is equal to *iPartition.PartitionID* and *objDeleted.SeqNumber* <= *purgeSeqNumber*.

3.1.7.2.10 Handle Enterprise Update

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) that represents the enterprise-wide object partition.
- *iNumberOfProperties*: A 32-bit unsigned integer that indicates the size (in elements) of the arrays *iPropertyID* and *iPropertyValue*. The arrays *iPropertyID* and *iPropertyValue* MUST have an identical number of elements. This argument is implicitly used during the search for a property identifier in *iPropertyID* or for a property value in *iPropertyValue*.
- *iPropertyID*: An array of **PROPID** that identifies the attributes for updating the enterprise object.
- *iPropertyValue*: An array of **PROPVARIANT** that contains the property values that are associated with all property identifiers in *iPropertyID*.

Return Values:

- *rStatus*: An **HRESULT** status code.
- *rNeedFlush*: A Boolean value that indicates whether the change should be flushed to the BSCs of the site to which the queue manager belongs.

The queue manager MUST perform the following actions to process this event:

- Set *rNeedFlush* to false.
- If PROPID_E_PECNAME is found in *iPropertyID*:
    - Set *newPECName* to the property value in *iPropertyValue* that is associated with PROPID_E_PECNAME.
    - If *newPECName* is different from *iPartition.AuthorityName*:

        - Set *iPartition.AuthorityName* to *newPECName*.
        - Set *rNeedFlush* to true.
- Set *rStatus* to MQ_OK.

3.1.7.2.11 Handle Site Update

This event MUST be generated with the following arguments:

- *iPathName*: A null-terminated **WCHAR** buffer that contains the path name of the site object.
- *iGuidIdentifier*: A GUID that identifies the site object.
- *iNumberOfProperties*: A 32-bit unsigned integer that indicates the size (in elements) of the arrays *iPropertyID* and *iPropertyValue*. The arrays *iPropertyID* and *iPropertyValue* MUST have an identical number of elements. This argument is implicitly used during the search for a property identifier in *iPropertyID* or a property value in *iPropertyValue*.
- *iPropertyID*: An array of **PROPID** that identifies the attributes for updating the site object.
- *iPropertyValue*: An array of **PROPVARIANT** that contains the property values that are associated with all property identifiers in *iPropertyID*.
- *iSync*: A Boolean value that indicates whether the change is from a **SyncReplyMessage** (section 2.2.6.5).

Return Values:

- *rStatus*: An **HRESULT** status code.
- *rNeedFlush*: A Boolean value that indicates whether the change should be flushed to the BSCs of the queue manager.

The queue manager MUST perform the following actions to process this event:

- Set *rStatus* to MQ_OK.
- Set *rNeedFlush* to false.
- If PROPID_S_PSC is not found in *iPropertyID*:
    - Take no further action.
- Set *newPSCName* to the property value in *iPropertyValue* that is associated with PROPID_S_PSC.
- Find a **DirectoryPartition** object (section 3.1.1.2.3), referred to as *dirPartition*, in **DirectoryPartitionCollection**, where *dirPartition.PartitionID* is equal to *iGuidIdentifier*.
- If *dirPartition* is found:
    - Set *dirPartition.AuthorityName* to *newPSCName*.
    - Set *rNeedFlush* to true.
    - Find the **DirectoryNeighbor** object (section 3.1.1.2.4), referred to as *dirPSCNeighbor*, in **PSCNeighborCollection**, where:
        - *dirPSCNeighbor.PartitionID* is equal to *dirPartition.PartitionID*.

        - If *dirPSCNeighbor* is found:
            - Set *dirPSCNeighbor.MachineName* to *newPSCName*.
    - Else if *iSync* is true:
        - Create a new DirectoryPartition object, referred to as *newPartition*, with the following attributes:
            - PartitionID := the property value in *iPropertyValue* that is associated with the property identifier PROPID_S_SITEID in *iPropertyID*.
            - AuthorityName := *newPSCName*
            - LastSeqNumber := MIN_SEQ_NUMBER as defined in section 2.2.2.
            - PurgedSeqNumber := MIN_SEQ_NUMBER as defined in section 2.2.2.
            - AllowedPurgeSeqNumber := MIN_SEQ_NUMBER as defined in section 2.2.2.
            - PurgeState := 0 (PURGE_STATE_NORMAL)
        - Add *newPartition* to **DirectoryPartitionCollection**.
        - If *newPSCName* is different from **QueueManager.MachineName**:
            - Raise a **Send Sync Request** event (section 3.3.7.3) with the following arguments:
                - iPartition := *newPartition*
            - Create a new DirectoryNeighbor object, referred to as *newPSCNeighbor*, with the following attributes:
                - PartitionID := *newPartition.PartitionID*
                - MachineName := *newPSCName*
                - AckedSeqNumber := MIN_SEQ_NUMBER as defined in section 2.2.2.
                - AckedPECSeqNumber := MIN_SEQ_NUMBER as defined in section 2.2.2.
            - Add *newPSCNeighbor* to **PSCNeighborCollection**.
            - Start a **Change Propagation Timer** (section 3.3.2.1) with the following arguments:<4>
                - iNeighbor := *newPSCNeighbor*
                - iScope := **Intersite**
        - Else If *newPartition.PartitionID* is equal to **QueueManager.SiteIdentifier**:
            - Set **QueueManager.MySitePartition** to *newPartition*.
    - Set *rNeedFlush* to true.
    - Set *rStatus* to MQ_OK.

3.1.7.2.12 Handle Machine Update

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a **DirectoryPartition** object (section 3.1.1.2.3) that represents the directory partition with which the directory change is associated.
- *iPathName*: A null-terminated **WCHAR** buffer that contains the path name of the machine object.
- *iNumberOfProperties*: A 32-bit unsigned integer that indicates the size (in elements) of the arrays *iPropertyID* and *iPropertyValue*. The arrays *iPropertyID* and *iPropertyValue* MUST have an identical number of elements. This argument is implicitly used during the search for a property identifier in *iPropertyID* or for a property value in *iPropertyValue*.
- *iPropertyID*: An array of **PROPID** that identifies the attributes for updating the machine object.
- *iPropertyValue*: An array of **PROPVARIANT** that contains the property values that are associated with all property identifiers in *iPropertyID*.

Return Values:

- *rStatus*: An **HRESULT** status code.

The queue manager MUST perform the following actions to process this event:

- Set *rStatus* to MQ_OK.
- If PROPID_QM_SERVICE is not found in *iPropertyID*:
  - Take no further action.
- Set *qmService* to the property value that is associated with PROPID_QM_SERVICE.
- If *iPartition* is the same as *QueueManager.MySitePartition*:
  - If *qmService* is SERVICE_BSC (0x2):
    - If there is no element in **BSCNeighborCollection** whose <MachineName> matches *iPathName*:
      - Create a new **DirectoryNeighbor** object (section 3.1.1.2.4), referred to as *newBSCNeighbor*, with the following attributes:
        - PartitionID := *iPartition.PartitionID*
        - MachineName := *iPathName*
        - LastAckedTime := 0
      - Append *newBSCNeighbor* to **BSCNeighborCollection**.
      - Start a **Change Propagation Timer** (section 3.3.2.1) with the following arguments:<5>
        - iNeighbor := *newBSCNeighbor*
        - iScope := **Intrasite**
  - Else:

    - Remove the element, if any, from **BSCNeighborCollection** where <MachineName> matches *iPathName*.

- Else:
  - Take no further action.

3.2 MQDSRP Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The abstract data model for this protocol comprises elements that are private to this protocol and others that are shared between multiple Microsoft Message Queuing (MSMQ) protocols that are collocated at a common queue manager. The shared abstract data model is defined in [MS-MQDMPR] section 3.1.1, and the relationship between this protocol, a queue manager, and other protocols that share a common queue manager is described in [MS-MQOD].

3.2.2 Timers

No timers are required by the protocol server.

3.2.3 Initialization

The server MUST perform the initializations specified in section 3.1.3.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

Section 3.1.5 specifies the local events that are raised by the protocol when processing replication messages. The following events are specifically raised by the protocol server when processing incoming replication messages:

- **Receive Change Propagation**, section 3.2.7.1
- **Receive Change Request**, section 3.2.7.2
- **Receive Sync Request**, section 3.2.7.3
- **Receive PSC Ack**, section 3.2.7.4
- **Receive BSC Ack**, section 3.2.7.5

Section 3.2.7 specifies how each event MUST be handled by the queue manager.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

The events that are listed in this section are subscribed and processed by the protocol server.

The queue manager MUST process the events described in this section.

3.2.7.1 Receive Change Propagation

This event MUST be generated with the following arguments:

- *iChangePropagationMessage*: A reference to a ChangePropagationMessage (section 2.2.6.9) data element.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- For each DirectoryChange (section 2.2.4) object, referred to as *dirChange* in *iChangePropagationMessage.DirectoryChanges*:
  - Process *dirChange* as specified in section 3.1.5.2.
- If *iChangePropagationMessage.SeqNumberHeader.Count* > 0:
  - Set *machineName* to *iChangePropagationMessage.SeqNumberHeader.MachineName*.
  - For each PARTITION_SEQ_NUMBERS (section 2.2.3) element, referred to as *seqNumber* in *iChangePropagationMessage.SeqNumberHeader.PartitionSeqNumbers*:
    - Find a DirectoryPartition (section 3.1.1.2.3) object, referred to as *dirPartition* in **DirectoryPartitionCollection**, where:
      - *dirPartition.PartitionID* is the same as *seqNumber.PartitionID*.
    - If *dirPartition* is not found:
      - Continue with the next element.
    - If *seqNumber.PurgedSeqNumber* < *dirPartition.PurgedSeqNumber*:
      - Continue with the next element.
    - If *seqNumber.PurgedSeqNumber* > *dirPartition.AllowedPurgeNumber*:
      - Set *dirPartition.AllowedPurgeNumber* to *seqNumber.PurgedSeqNumber*.
    - If *seqNumber.LastSeqNumber* <= *dirPartition.LastSeqNumber* or *seqNumber.LastSeqNumber* < *dirPartition.ChangeMissingWindow*:
      - Continue with the next element.
    - If the queue manager is a PSC:
      - Set *dirPartition.AuthorityName* to *machineName*.
    - Else (queue manager is a BSC):

    - Set *QueueManager.MyPSCName* to *machineName*.
  - Set *dirPartition.ChangeMissingWindow* to (*seqNumber.LastSeqNumber* + 1).
- Raise a [Send Sync Request (section 3.3.7.3)]() event with the following arguments:
  - iPartition := *dirPartition*

3.2.7.2 Receive Change Request

This event MUST be generated with the following arguments:

- iChangeRequestMessage: A reference to a [ChangeRequestMessage (section 2.2.6.2)]() data element.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- Set *changeResult* to MQ_OK.
- Set *partitionID* to *iChangeRequestMessage.ChangeRequestHeader.PartitionID*.
- Set *myEnterprise* to false.
- Set *mySite* to false.
- If the queue manager is a PEC and *partitionID* is equal to **QueueManager.MyEnterprisePartition.PartitionID**:
  - Set *myEnterprise* to true.
- If the queue manager is a PSC and *partitionID* is equal to **QueueManager.SiteIdentifier**:
  - Set *mySite* to true.
- If *myEnterprise* or *mySite* is true:
  - Set *myPartition* to the element in **DirectoryPartitionCollection** whose PartitionID is *partitionID*.
  - Set *dirChange* to *iChangeRequestMessage.DirectoryChange*.
  - Use the first **PROPID** in *dirChange.PropertyId* to determine the object type, referred to as *objectType*, that is associated with this change, as specified in section 3.1.1.4.
  - Set *newSeqNumber* to (*myPartition.LastSeqNumber* + 1).
  - Case *dirChange.Command* Of:
    - 0x00 (create):
      - Raise a [Handle Directory Create (section 3.1.7.2.5)]() event with the following arguments:
        - iPartition := *myPartition*
        - iObjectType := *objectType*

    - iPathName := *dirChange.PathName*
    - iNumberOfProperties := *dirChange.NumberOfProperties*
    - iPropertyID := *dirChange.PropertyID*
    - iPropertyValue := *dirChange.PropertyValue*
    - iSeqNumber := *newSeqNumber*
    - iPostCreate := false
  - Set *changeResult* to the return value *rStatus* of the **Handle Directory Create** event.
- 0x01 (update):
  - Raise a Handle Directory Update (section 3.1.7.2.6) event with the following arguments:
    - iPartition := *myPartition*.
    - iObjectType := *objectType*.
    - iPathName: If *dirChange.UseGuid* is 0x00, set it to *dirChange.PathName*; otherwise, set it to NULL.
    - iGuidIdentifier := If *dirChange.UseGuid* is 0x00, set it to NULL; otherwise, set it to *dirChange.GuidIdentifier*.
    - iNumberOfProperties := *dirChange.NumberOfProperties*.
    - iPropertyID := *dirChange.PropertyID*.
    - iPropertyValue := *dirChange.PropertyValue*.
    - iSeqNumber := *newSeqNumber*.
    - iPostUpdate := false.
  - Set *changeResult* to the return value *rStatus* of the **Handle Directory Update** event.
- 0x02 (delete):
  - Raise a Handle Directory Delete (section 3.1.7.2.7) event with the following arguments:
    - iPartition := *myPartition*.
    - iObjectType := the property value of the second element in *dirChange.PropertyValue*.
    - iScope := the property value of the first element in *dirChange.PropertyValue*.
    - iPathName: If *dirChange.UseGuid* is 0x00, set it to *dirChange.PathName*; otherwise, set it to NULL.
    - iGuidIdentifier := If *dirChange.UseGuid* is 0x00, set it to NULL; otherwise, set it to *dirChange.GuidIdentifier*.
    - iSeqNumber := *newSeqNumber*.
    - iPostDelete := false.

            - Set *changeResult* to the return value *rStatus* of the **Handle Directory Delete** event.
            - If *changeResult* is MQ_OK:
                - Create a DeletedObject (section 3.1.1.2.2) object, referred to as *objDeleted*, with the following attributes:
                    - Identifier := *dirChange.GuidIdentifier*.
                    - PartitionID := *myPartition.PartitionID*.
                    - SeqNumber := *newSeqNumber*.
                    - ObjectType: the property value of the second element in *dirChange.PropertyValue*.
                    - Scope: the property value of the first element in *dirChange.PropertyValue*.
                - Append *objDeleted* to **DeletedObjectCollection**.
        - 0x03 (synchronize):
            - Raise a Handle Directory Sync (section 3.1.7.2.8) event with the following arguments:
                - iPartition := *myPartition*.
                - iObjectType := *objectType*.
                - iPathName := *dirChange.PathName*.
                - iNumberOfProperties := *dirChange.NumberOfProperties*.
                - iPropertyID := *dirChange.PropertyID*.
                - iPropertyValue := *dirChange.PropertyValue*.
                - iSeqNumber := *newSeqNumber*.
                - iPostCreate := false.
            - Set *changeResult* to the return value *rStatus* of the Handle Directory Sync event.
        - Other:
            - Set *changeResult* to MQ_ERROR.
    - If *changeResult* is MQ_OK:
        - Set *myPartition.LastSeqNumber* to *newSeqNumber*.
- Else (queue manager is not the object authority, so it forwards the request):
    - Raise a Process Change Request (section 3.3.7.1) event with the following arguments:
        - iChangeRequestMessage := *iChangeRequestMessage*.
    - Set *changeResult* to the return result *rStatus* of the **Process Change Request** event.
- Send a reply to the requester by generating a Send Change Reply (section 3.2.7.6) event with the following arguments:

    - iStatus := *changeResult*
    - iChangeRequestMessage := *iChangeRequestMessage*

3.2.7.3 Receive Sync Request

This event MUST be generated with the following arguments:

- *iSyncRequestMessage*: A reference to a SyncRequestMessage (section 2.2.6.4) data element.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- Find a DirectoryPartition (section 3.1.1.2.3) object, referred to as *dirPartition*, in **DirectoryPartitionCollection**, where:
    - *dirPartition.PartitionID* is the same as *iSyncRequestMessage.PartitionID*.
- If *dirPartition* is not found:
    - Take no further action.
- If *dirPartition.PurgeState* is not 0 (PURGE_STATE_NORMAL):
    - Take no further action.
- Set *fromSeqNum* to *iSyncRequestMessage.FromSeqNumber*.
- Set *toSeqNum* to *iSyncRequestMessage.ToSeqNumber*.
- If *fromSeqNum* < *dirPartition.PurgedSeqNumber* and *iSyncRequestMessage.KnownPurgedSeqNumber* < *dirPartition.PurgedSeqNumber*:
    - Raise a Send Already Purged (section 3.2.7.7) event with the following arguments:
        - iPartition := *dirPartition*
        - iRequesterName := *iSyncRequestMessage.RequesterName*
    - Take no further action.
- If *toSeqNum* is MAX_SEQ_NUMBER, as defined in section 2.2.2:
    - Set *toSeqNum* to *dirPartition.LastSeqNumber*.
- If the queue manager is a PEC and *dirPartition* is the same as **QueueManager.MyEnterprisePartition**:
    - Set *syncObjectTypes* to [MQDS_SITE, MQDS_CN, MQDS_ENTERPRISE, MQDS_USER, MQDS_ROUTINGLINK].
- If the queue manager is a PSC and *dirPartition* is the same as **QueueManager.MySitePartition**:
    - Set *syncObjectTypes* to [MQDS_MACHINE, MQDS_QUEUE].

- Create an empty array of DirectoryChange (section 2.2.4) objects, referred to as *dirSyncReplyChanges*.
- For each type, referred to as *objectType*, in *syncObjectTypes*:
  - Raise a Lookup Directory Objects (section 3.2.7.8) event with the following arguments:
    - iPartition := *dirPartition*
    - iObjectType := *objectType*
    - iFromSeqNumber := *fromSeqNum*
    - iToSeqNumber := *toSeqNum*
    - iScope := *iSyncRequestMessage.Scope*
  - Append the return result *rDirectoryChanges* of the **Lookup Directory Objects** event to *dirSyncReplyChanges*.
- For each element, referred to as *objDeleted*, in **DeletedObjectCollection**:
  - Create a **DirectoryChange** object, referred to as *dirChange*, with the following attributes:
    - Command := 0x02 (delete)
    - UseGuid := 0x01
    - GuidIdentifier := *objDeleted.Identifier*
    - PartitionID := *dirPartition.PartitionID*
    - PreviousSeqNumber: This attribute is not initialized at this moment.
    - SeqNumber := *objDeleted.SeqNumber*
    - PurgedSeqNumber := *dirPartition.PurgedSeqNumber*
    - NumberOfProperties := 2
    - PropertyID := [PROPID_D_SCOPE, PROPID_D_OBJTYPE]
    - PropertyValue := [value copied from *objDeleted.Scope*, value copied from *objDeleted.ObjectType*]
  - Append *dirChange* to *dirSyncReplyChanges*.
- Sort *dirSyncReplyChanges* by the **SeqNumber** field in descending order.
- Set the **PreviousSeqNumber** field of the first element in *dirSyncReplyChanges* to *iSyncRequestMessage.FromSeqNumber*. For all but the first element in *dirSyncReplyChanges*, set the **PreviousSeqNumber** field to the **SeqNumber** field of the previous element.
- Create a SyncReplyMessage (section 2.2.6.5) object, referred to as *syncReplyMessage*, with the following attributes:
  - BaseReplicationHeader.Operation := 0x03
  - Initialize the fields in *syncReplyMessage.SyncReplyHeader* as follows:

        - PartitionID := *dirPartition.PartitionID*
        - FromSeqNumber := *fromSeqNumber*
        - ToSeqNumber := *toSeqNumber*
        - PurgedSeqNumber := *dirPartition.PurgedSeqNumber*
        - Count := size of *dirSyncReplyChanges*
        - CompleteSync0: MUST be set as follows:

CompleteSync0	Condition
0	*iSyncRequestMessage.IsSync0* is 0.
1	*iSyncRequestMessage.IsSync0* is 1, and *toSeqNum* is different from *dirPartition.LastSeqNumber* (indicating that new changes happened while building this reply message).
2	*iSyncRequestMessage.IsSync0* is 1, and *toSeqNum* is the same as *dirPartition.LastSeqNumber* (all available changes have been synchronized).

    - DirectoryChanges := *dirSyncReplyChanges*
- Raise a Send Replication Message (section 3.1.7.2.1) event with the following arguments:
    - iDestination := *iSyncRequestMessage.RequesterName*
    - iReplicationMessage := *syncReplyMessage*
    - iTimeout := 20 * 60
    - iPriority := 3
    - iAckRequested := None
    - iAdminQueueFormatName := NULL
    - iResponseQueueFormatName := NULL

3.2.7.4 Receive PSC Ack

This event MUST be generated with the following arguments:

- iPSCAck: A reference to a PSCAckMessage (section 2.2.6.7) data element.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- If queue manager is not PSC:
    - Take no further action.

- Find a DirectoryNeighbor (section 3.1.1.2.4) object, referred to as *dirPSCNeighbor*, in **PSCNeighborCollection**, where:
  - *dirPSCNeighbor.MachineName* is the same as *iPSCAck.PSCName*.
- If *dirPSCNeighbor* is not found:
  - Take no further action.
- If the queue manager is a PEC and *iPSCAck.AckedPartitionID* is the same as **QueueManager.MyEnterprisePartition.PartitionID**:
  - Set *dirPSCNeighbor.AckedPECSeqNumber* to *iPSCAck.AckedSeqNumber*.
- If *iPSCAck.AckedPartitionID* is the same as **QueueManager.MySitePartition.PartitionID**:
  - Set *dirPSCNeighbor.AckedSeqNumber* to *iPSCAck.AckedSeqNumber*.

3.2.7.5 Receive BSC Ack

This event MUST be generated with the following arguments:

- *iBSCAck*: A reference to a BSCAckMessage (section 2.2.6.6) data element.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- If the queue manager is not a PSC:
  - Take no further action.
- Find a DirectoryNeighbor (section 3.1.1.2.4) object, referred to as *dirBSCNeighbor*, in **BSCNeighborCollection**, where:
  - *dirBSCNeighbor.MachineName* is the same as *iBSCAck.BSCName*.
- If *dirBSCNeighbor* is not found:
  - Take no further action.
- Set *dirBSCNeighbor.LastAckedTime* to the current time.

3.2.7.6 Send Change Reply

This event MUST be generated with the following arguments:

- *iStatus*: An **HRESULT** status code that indicates the result of the change request operation.
- *iChangeRequestMessage*: The change request that the reply is for.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- Set *serverName* to *iChangeRequestMessage.ChangeRequestHeader.RequesterName*.
- Create a ChangeReplyMessage (section 2.2.6.3), referred to as *changeReply*, with the following attributes:
  - BaseReplicationHeader.Operation := 0x04
  - RequestIdentifier := *iChangeRequestMessage.ChangeRequestHeader.RequestIdentifier*
  - Result := *iStatus*
  - RequesterName := *serverName*
- Find a DirectoryNeighbor (section 3.1.1.2.4) object, referred to as *dirNeighbor*, in the **PSCNeighborCollection** and **BSCNeighborCollection**, whose <MachineName> matches *serverName*.
- If *dirNeighbor* is not found:
  - Set *serverName* to *iChangeRequestMessage.ChangeRequestHeader.PSCName*.
- Generate a Send Replication Message (section 3.1.7.2.1) event with the following arguments:
  - iDestination := *serverName*
  - iReplicationMessage := *changeReply*
  - iTimeout := 10
  - iPriority := 8
  - iAckRequested := **AckNackReachQueue**
  - iAdminQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager **GUID** [MS-DTYP] section 2.3.4
  - iResponseQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager **GUID**

3.2.7.7 Send Already Purged

This event is raised under certain conditions when a SyncRequestMessage (section 2.2.6.4) is processed by the protocol server in the **Receive Sync Request** event, as specified in section 3.2.7.3. It MUST be generated with the following arguments:

- *iPartition*: A reference to a DirectoryPartition (section 3.1.1.2.3) data element that represents the directory partition to which the AlreadyPurgedMessage (section 2.2.6.8) applies.
- *iRequesterName*: The requester name of the original synchronization request.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- Create an **AlreadyPurgedMessage** object, referred to as *alreadyPurged*, with the following attributes:

- BaseReplicationHeader.Operation := 0x05
  - PartitionID := *iPartition.PartitionID*
  - PurgedSeqNumber := *iPartition.PurgedSeqNumber*
- Raise a Send Replication Message (section 3.1.7.2.1) event with the following arguments:
  - iDestination := *iRequesterName*
  - iReplicationMessage := *alreadyPurged*
  - iTimeout := 20 * 60
  - iPriority := 3
  - iAckRequested := None
  - iAdminQueueFormatName := NULL
  - iResponseQueueFormatName := NULL

3.2.7.8 Lookup Directory Objects

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a DirectoryPartition (section 3.1.1.2.3) data element that represents the directory partition to which the directory lookup applies.
- *iObjectType*: A value of **Directory Object Types** that identifies the directory object type.
- *iFromSeqNumber*: A SEQUENCE_NUMBER (section 2.2.2) element that contains the sequence number where the lookup starts.
- *iToSeqNumber*: A SEQUENCE_NUMBER element that contains the sequence number where the lookup ends.
- *iScope*: A UCHAR specifying the object scope for the lookup. This value is used only when *iObjectType* is MQDS_QUEUE, and it MUST NOT be used for other object types.

Return Values:

- *rDirectoryChanges*: An array of **DirectoryChange** objects, as specified in section 2.2.4. Each directory object that satisfies the given lookup constraints is converted to a **DirectoryChange** object and saved in this array.

The queue manager MUST perform the following actions to process this event:

- Set *rDirectoryChanges* to empty.
- Maintain the following table, referred to as *refTable*, for reference as specified in the actions.

Object type	LookupPropID	DirectoryChangePropID
MQDS_SITE	PROPID_S_SEQNUM PROPID_S_SITEID PROPID_S_GATES	PROPID_S_GATES PROPID_S_INTERVAL1 PROPID_S_INTERVAL2

Object type	LookupPropID	DirectoryChangePropID
	PROPID_S_INTERVAL1 PROPID_S_INTERVAL2 PROPID_S_PATHNAME PROPID_S_PSC PROPID_S_PSC_SIGNPK PROPID_S_SECURITY	PROPID_S_PATHNAME PROPID_S_PSC PROPID_S_PSC_SIGNPK PROPID_S_SECURITY
MQDS_CN	PROPID_CN_SEQNUM PROPID_CN_GUID PROPID_CN_NAME PROPID_CN_PROTOCOLID PROPID_CN_SECURITY	PROPID_CN_NAME PROPID_CN_PROTOCOLID PROPID_CN_SECURITY
MQDS_ENTERPRISE	PROPID_E_SEQNUM PROPID_E_ID PROPID_E_NAME PROPID_E_NAMESTYLE PROPID_E_CSP_NAME PROPID_E_PECNAME PROPID_E_LONG_LIVE PROPID_E_VERSION PROPID_E_SECURITY	PROPID_E_NAME PROPID_E_NAMESTYLE PROPID_E_CSP_NAME PROPID_E_PECNAME PROPID_E_LONG_LIVE PROPID_E_VERSION PROPID_E_SECURITY PROPID_E_S_INTERVAL1 PROPID_E_S_INTERVAL2 PROPID_E_CRL PROPID_E_CSP_TYPE PROPID_E_ENCRYPT_ALG PROPID_E_SIGN_ALG PROPID_E_HASH_ALG PROPID_E_CIPHER_MODE
MQDS_USER	PROPID_U_SEQNUM PROPID_U_ID PROPID_U_SID PROPID_U_DIGEST PROPID_U_SIGN_CERT	PROPID_U_SID PROPID_U_DIGEST PROPID_U_SIGN_CERT
MQDS_ROUTINGLINK	PROPID_L_SEQNUM PROPID_L_ID PROPID_L_COST PROPID_L_NEIGHBOR1 PROPID_L_NEIGHBOR2	PROPID_L_COST PROPID_L_NEIGHBOR1 PROPID_L_NEIGHBOR2
MQDS_MACHINE	PROPID_QM_SEQNUM PROPID_QM_MACHINE_ID PROPID_QM_FOREIGN PROPID_QM_QUOTA RROPID_QM_JOURNAL_QUOTA PROPID_QM_CNS	PROPID_QM_FOREIGN PROPID_QM_QUOTA PROPID_QM_JOURNAL_QUOTA PROPID_QM_CNS PROPID_QM_MACHINE_TYPE PROPID_QM_OS

Object type	LookupPropID	DirectoryChangePropID
	PROPID_QM_MACHINE_TYPE PROPID_QM_OS PROPID_QM_ADDRESS PROPID_QM_CREATE_TIME PROPID_QM_ENCRYPT_PK PROPID_QM_INFRS PROPID_QM_MODIFY_TIME PROPID_QM_OUTFRS PROPID_QM_PATHNAME PROPID_QM_SERVICE PROPID_QM_SIGN_PK PROPID_QM_SITE_ID PROPID_QM_SECURITY	PROPID_QM_ADDRESS PROPID_QM_CREATE_TIME PROPID_QM_ENCRYPT_PK PROPID_QM_INFRS PROPID_QM_MODIFY_TIME PROPID_QM_OUTFRS PROPID_QM_PATHNAME PROPID_QM_SERVICE PROPID_QM_SIGN_PK PROPID_QM_SITE_ID PROPID_QM_SECURITY PROPID_QM_HASHKEY
MQDS_QUEUE	PROPID_Q_SEQNUM PROPID_Q_INSTANCE PROPID_Q_SCOPE PROPID_Q_QMID PROPID_Q_TYPE PROPID_Q_BASEPRIORITY PROPID_Q_JOURNAL PROPID_Q_QUOTA PROPID_Q_JOURNAL_QUOTA PROPID_Q_CREATE_TIME PROPID_Q_MODIFY_TIME PROPID_Q_PATHNAME PROPID_Q_LABEL PROPID_Q_AUTHENTICATE PROPID_Q_PRIV_LEVEL PROPID_Q_TRANSACTION PROPID_Q_SECURITY	PROPID_Q_SCOPE PROPID_Q_QMID PROPID_Q_TYPE PROPID_Q_BASEPRIORITY PROPID_Q_JOURNAL PROPID_Q_QUOTA PROPID_Q_JOURNAL_QUOTA PROPID_Q_CREATE_TIME PROPID_Q_MODIFY_TIME PROPID_Q_PATHNAME PROPID_Q_LABEL PROPID_Q_AUTHENTICATE PROPID_Q_PRIV_LEVEL PROPID_Q_TRANSACTION PROPID_Q_SECURITY PROPID_Q_HASHKEY PROPID_Q_LABEL_HASHKEY

- Create an array of **PROPID** elements, referred to as *aLookupPropID*, as specified in *refTable* column LookupPropID for object type *iObjectType*.
- Raise an [MS-MQDS] **Begin Directory Lookup** event with the following arguments:
  - iRestriction:
    - cRes: set to the number of **MQPROPERTYRESTRICTION** elements in the value column of paPropRes.
    - paPropRes: set for each object type as specified in the following table.

Object type	Value
MQDS_SITE	[PREQ, PROPID_S_PARTITIONID, PV(dirPartition.PartitionID)] [PRGE, PROPID_S_SEQNUM, PV(fromSeqNum)]

Object type	Value
	[PRLE, PROPID_S_SEQNUM, PV(toSeqNum)]
MQDS_CN	[PREQ, PROPID_CN_PARTITIONID, PV(dirPartition.PartitionID)] [PRGE, PROPID_CN_SEQNUM, PV(fromSeqNum)] [PRLE, PROPID_CN_SEQNUM, PV(toSeqNum)]
MQDS_ENTERPRISE	[PREQ, PROPID_E_PARTITIONID, PV(dirPartition.PartitionID)] [PRGE, PROPID_E_SEQNUM, PV(fromSeqNum)] [PRLE, PROPID_E_SEQNUM, PV(toSeqNum)]
MQDS_USER	[PREQ, PROPID_U_PARTITIONID, PV(dirPartition.PartitionID)] [PRGE, PROPID_U_SEQNUM, PV(fromSeqNum)] [PRLE, PROPID_U_SEQNUM, PV(toSeqNum)]
MQDS_ROUTINGLINK	[PREQ, PROPID_L_PARTITIONID, PV(dirPartition.PartitionID)] [PRGE, PROPID_L_SEQNUM, PV(fromSeqNum)] [PRLE, PROPID_L_SEQNUM, PV(toSeqNum)]
MQDS_MACHINE	[PREQ, PROPID_QM_PARTITIONID, PV(dirPartition.PartitionID)] [PRGE, PROPID_QM_SEQNUM, PV(fromSeqNum)] [PRLE, PROPID_QM_SEQNUM, PV(toSeqNum)]
MQDS_QUEUE	[PREQ, PROPID_Q_PARTITIONID, PV(dirPartition.PartitionID)] [PRGE, PROPID_Q_SEQNUM, PV(fromSeqNum)] [PRLE, PROPID_Q_SEQNUM, PV(toSeqNum)] If iScope is 0x01: [PREQ, PROPID_Q_SCOPE, PV(0x01)]

PV(GUID): a PROPVARIANT with vt=VT_CLSID and puuid=GUID.

PV(UCHAR[8]): a PROPVARIANT with vt=VT_BLOB, blob.cbSize=8, and blob.pBlobData = the array of UCHAR.

PV(UCHAR): a PROPVARIANT with vt=VT_UI1 and bVal=UCHAR.

- iColumnSet:
  - cCol: set to the number of elements in *aLookupPropID*
  - aCol: set to *aLookupPropID*:
- Set *dirLookupDescriptor* to the return value *rLookupDescriptor* of the **Begin Directory Lookup** event.
- Repeat the following actions until *rNumberOfProperties* returned from the **Lookup Directory Next** event is zero:
  - Raise an [MS-MQDS] **Lookup Directory Next** event with the following arguments:
    - iLookupDescriptor := *dirLookupDescriptor*
  - Divide *rPropertyValue* returned from the **Lookup Directory Next** event into subsets. The size of each subset MUST be equal to the size of *aLookupPropID*. For each subset, referred to as *aObjectPropertyValue*:

        - Create a **DirectoryChange** object, referred to as *dirChange*, with the following attributes:
            - Command := 0x03
            - UseGuid := 0x01
            - GuidIdentifier := the property value of the second element in *aObjectPropertyValue*.
            - PartitionID := *dirPartition.PartitionID*.
            - PreviousSeqNumber: This attribute is not initialized at this moment.
            - SeqNumber := the property value of the first element in *aObjectPropertyValue*.
            - PurgedSeqNumber := *dirPartition.PurgedSeqNumber*.
            - NumberOfProperties := the number of property identifiers in *refTable* column **DirectoryChangePropID** associated with *iObjectType*.
            - PropertyID := a new array of **PROPID** containing the property identifiers in *refTable* column **DirectoryChangePropID** associated with *iObjectType*.
            - PropertyValue: A new array of **PROPVARIANT**. The size of the array MUST be the same as the size of PropertyID. The value of each element MUST be set as follows:
                - If the property identifier, referred to as *propID*, at the same index in **PropertyID**, is found in *aLookupPropID*:
                    - Set the element to the property value in *aObjectPropertyValue* associated with *propID*.
                - Else:
                    - The variant type MUST be set according to the definition of *propID* in [MS-MQMQ]. The variant value is undefined and MUST NOT be interpreted by the protocol implementation.
        - Append *dirChange* to *dirSyncReplyChanges*.
- Raise an [MS-MQDS] **End Directory Lookup** event with the following arguments:
    - iLookupDescriptor := *lookupDescriptor*

3.3 MQDSRP Client Details

3.3.1 Abstract Data Model

The abstract data model for this protocol comprises elements that are private to this protocol and others that are shared between multiple Microsoft Message Queuing (MSMQ) protocols that are co-located at a common queue manager. The shared abstract data model is defined in [MS-MQDMPR] section 3.1.1, and the relationship between this protocol, a queue manager, and other protocols that share a common queue manager, is described in [MS-MQOD].

3.3.1.1 Shared Data Elements

The MQDSRP protocol client manipulates the shared data elements that are listed in section 3.1.1.1.

3.3.2 Timers

The MQDSRP protocol client uses the timers that are described in this section to control directory change replication.

3.3.2.1 Change Propagation Timer

The change propagation timer regulates the amount of time, in milliseconds, that a PSC waits before propagating directory changes to one neighbor. The default, minimum, and maximum values of this timer are implementation-specific.<6>

The queue manager initializes one change propagation timer for each of its directory neighbors. When this timer expires, another instance of this timer is initialized with the same expiration time.

3.3.2.2 BSC Ack Timer

This timer regulates the amount of time, in milliseconds, that the queue manager waits before sending a BSC acknowledgment message to its PSC. The default, minimum, and maximum values of this timer are implementation-specific. <7>

When this timer expires, another instance of this timer is initialized with the same expiration time.

3.3.3 Initialization

The MQDSRP protocol client MUST perform the common initialization that is specified in section 3.1.3.

The MQDSRP protocol client MUST perform the following initializations:

- If the queue manager is a BSC:
  - Start the **BSC Ack Timer** (section 3.3.2.2). <8>
- If the queue manager is a PSC:
  - For each **DirectoryNeighbor** object (section 3.1.1.2.4), referred to as *dirBSCNeighbor*, in the **BSCNeighborCollection**, start a **Change Propagation Timer** with the following arguments: <9>
    - iNeighbor := *dirBSCNeighbor*
    - iScope := **Intrasite**
  - For each DirectoryNeighbor object, referred to as *dirPSCNeighbor* in the **PSCNeighborCollection**, start a **Change Propagate Timer** with the following arguments: <10>
    - iNeighbor:= *dirPSCNeighbor*
    - iScope := **Intersite**
- For each **DirectoryPartition** object (section 3.1.1.2.3), referred to as *dirPartition*, in the **DirectoryPartitionCollection**:
  - If *dirPartition.AuthorityName* is different from **QueueManager.MachineName** or the queue manager is a BSC:

    - Raise a **Send Sync Request** event (section 3.3.7.3) with the following argument:
        - iPartition := *dirPartition*

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

Section 3.1.5 specifies the local events that are raised by the protocol when processing replication messages. The following events are specifically raised by the MQDSRP protocol client when processing incoming replication messages:

- Receive Change Reply
- Receive Already Purged
- Receive Sync Reply
- Receive Change Request Nack
- Receive Request Nack

Section 3.3.7 specifies how each event MUST be handled by the queue manager.

3.3.6 Timer Events

This section specifies the timer events that are handled by the MQDSRP protocol client.

3.3.6.1 Change Propagation Timer Event

The **Change Propagation Timer** (section 3.3.2.1) has the following arguments:

- *iNeighbor*: A reference to a **DirectoryNeighbor** data element (section 3.1.1.2.4) representing the directory neighbor to which a directory update message is sent.
- *iScope*: MUST be set to either **Intrasite** or **Intersite**.

When the timer fires, the queue manager MUST perform the following actions to process the timer event:

- Raise a **Send Change Propagation** event (section 3.3.7.2) with the following arguments:
    - iNeighbor := *iNeighbor*
    - iScope := *iScope*
- Initialize another instance of the **Change Propagation Timer** with the same arguments.

3.3.6.2 BSC Ack Timer Event

When the timer fires, the queue manager MUST perform the following actions to process the timer event:

- Create a **BSCAckMessage** object (section 2.2.6.6), referred to as *bscAck*, with the following attributes:

    - BaseReplicationHeader.Operation := 0x07
    - BSCMachineID := **QueueManager.Identifier**
    - BSCName := **QueueManager.MachineName**
- Raise a **Send Replication Message** event (section 3.1.7.2.1) with the following arguments:
    - iDestination := **QueueManager.MyPSCName**
    - iReplicationMessage := *bscAck*
    - iTimeout := 20 * 60
    - iPriority := 3
    - iAckRequested := **AckNackReachQueue**
    - iAdminQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to QueueManager.Identifier.
    - iResponseQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to QueueManager.Identifier.

3.3.7 Other Local Events

The events listed in this section are subscribed and processed by the MQDSRP protocol client.

3.3.7.1 Process Change Request

This event MUST be generated with the following arguments:

- *iChangeRequestMessage*: A reference to the ChangeRequestMessage data element (section 2.2.6.2).

Return Values:

- *rStatus*: An **HRESULT** that indicates the result of the change request operation.

The queue manager MUST perform the following actions to process this event:

- If the queue manager is BSC:
    - Set *nextHop* to **QueueManager.MyPSCName**.
- Else:
    - Find a DirectoryPartition object (section 3.1.1.2.3), referred to as *dirPartition*, in the **DirectoryPartitionCollection**, where:
        - *dirPartition.PartitionID* is the same as *iChangeRequestMessage.ChangeRequestHeader.PartitionID*
    - If *dirPartition* is not found:
        - Set *rStatus* to MQDS_UNKNOWN_SOURCE.
        - Take no further action.

        - Else:
            - Set *nextHop* to *dirPartition.AuthorityName*.
- Generate a Send Replication Message event (section 3.1.7.2.1) with the following arguments:
    - iDestination := *nextHop*
    - iReplicationMessage := *iChangeRequestMessage*
    - iTimeout := 10
    - iPriority := 8
    - iAckRequested := **AckNackReachQueue**
    - iAdminQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager GUID.
    - iResponseQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager GUID.
- Wait until one of the following conditions is true and perform the corresponding actions:
    - An implementation-specific waiting time<11> has elapsed:
        - Set *rStatus* to MQDS_NO_RSP_FROM_OWNER.
    - A Receive Change Reply event (section 3.3.7.6) occurs:
        - Set *rStatus* to the status code passed from the Receive Change Reply event.
    - A Receive Change Request Nack event (section 3.3.7.8) occurs:
        - Set *rStatus* to the status code passed from the Receive Change Request Nack event.

3.3.7.2 Send Change Propagation

The event MUST be triggered with the following arguments:

- *iNeighbor*: A reference to a DirectoryNeighbor object (section 3.1.1.2.4) representing the directory server to which a DirectoryChange message (section 2.2.4) is sent.
- *iScope*: MUST be either **Intrasite** or **Intersite**.

Return Values:

- *None*.

The queue manager MUST perform the following actions to process this event:

- Create a SeqNumberHeader object (section 2.2.5.4), referred to as *seqNumberHeader*, with the following attributes:
    - Count := 0.
    - Other fields: Not set.

- If an implementation-specific time period<12> has elapsed since the last time a SeqNumberHeader (section 2.2.5.4) was sent, initialize *seqNumberHeader* as follows:
  - *seqNumberHeader.MachineName* := *QueueManager.MachineName*.
  - If *iScope* is **Intersite**:
    - If the queue manager is PEC:
      - *Set seqNumberHeader.Count* to 2.
      - Initialize the first element of *seqNumberHeader.PartitionSeqNumbers* as follows:
        - PartitionID := PEC_PARTITION_ID as defined in section 1.9.
        - LastSeqNumber := **QueueManager.MyEnterprisePartition.LastSeqNumber**.
        - PurgedSeqNumber := **QueueManager.MyEnterprisePartition.PurgedSeqNumber**.
      - Move to the second element.
    - Else:
      - Set *seqNumberHeader.Count* to 1.
      - Move to the first element.
    - Initialize the current element as follows:
      - PartitionID := **QueueManager.MySitePartition.PartitionID**.
      - LastSeqNumber := **QueueManager.MySitePartition.LastSeqNumber**.
      - PurgedSeqNumber := **QueueManager.MySitePartition.PurgedSeqNumber**.
  - Else:
    - Set *iSeqNumberHeader.Count* to the size of the **DirectoryPartitionCollection**.
    - For each DirectoryPartition object (section 3.1.1.2.3), referred to as *dirPartition* in the **DirectoryPartitionCollection**, initialize the corresponding element in *seqNumberHeader.PartitionSeqNumbers* as follows:
      - PartitionID := *dirPartition.PartitionID*.
      - LastSeqNumber := *dirPartition.LastSeqNumber*.
      - If *dirPartition* is **QueueManager.MyEnterprisePartition** or *dirPartition* is **QueueManager.MySitePartition**:
        - PurgedSeqNumber := *dirPartition.PurgedSeqNumber*.
      - Else:
        - PurgedSeqNumber := *dirPartition.AllowedPurgeSeqNumber*.
- Create a ChangePropagationMessage (section 2.2.6.9), referred to as *dirChangeMessage*, with the following attributes:

    - BaseReplicationHeader.Operation := 0x00.
    - Flush := 0x00.
    - Count := the size of *iNeighbor.AvailableChangeCollection*.
    - DirectoryChanges := *iNeighbor.AvailableChangeCollection*.
    - SeqNumberHeader := *seqNumberHeader*.
- Raise a Send Replication Message event (section 3.1.7.2.1) with the following arguments:
    - iDestination := *iNeighbor.MachineName*.
    - iReplicationMessage := *dirChangeMessage*.
    - iTimeout := 20 * 60.
    - iPriority := 3.
    - iAckRequested := **None**.
    - iAdminQueueFormatName := NULL.
    - iResponseQueueFormatName := NULL.
- Remove all elements in *iNeighbor.AvailableChangeCollection*.

3.3.7.3 Send Sync Request

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a DirectoryPartition object (section 3.1.1.2.3) representing the directory partition with which the SyncRequestMessage (section 2.2.6.4) is associated.

Return Values:

- *None*.

The queue manager MUST perform the following actions to process this event:

- If *iPartition.PurgeState* is not 0 (PURGE_STATE_NORMAL) and *iPartition.PurgeState* is not 2 (PURGE_STATE_SYNC0):
    - Take no further action.
- Create a SyncRequestMessage, referred to as *syncRequest,* with the following attributes:
    - BaseReplicationHeader.Operation := 0x02.
    - PartitionID := *iPartition.PartitionID*.
    - FromSeqNumber := *iPartition.LastSeqNumber*.
    - ToSeqNumber := *iPartition.ChangeMissingWindow*.
    - KnownPurgedSeqNumber := *iPartition.PurgedSeqNumber*.
    - **IsSync0** MUST be set as follows.

IsSync0	Condition
0x00	*iPartition.PurgeState* is not equal to 2 (PURGE_STATE_SYNC0).
0x01	*iPartition.PurgeState* is equal to 2 (PURGE_STATE_SYNC0).

- Scope MUST be set as follows.

Scope	Condition
0x00	The queue manager is a BSC.
0x01	The queue manager is a PSC.

- RequesterName := **QueueManager.MachineName**.

- Raise a Send Replication Message event (section 3.1.7.2.1) with the following arguments:
  - iDestination: If the queue manager is a BSC, set it to **QueueManager.MyPSCName**; otherwise, set it to *iPartition.AuthorityName*.
  - iReplicationMessage := *syncRequest*.
  - iTimeout := 20 * 60.
  - iPriority := 3.
  - iAckRequested := **AckNackReachQueue**.
  - iAdminQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager GUID.
  - iResponseQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager GUID.

3.3.7.4 Send PSC Ack

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a DirectoryPartition object (section 3.1.1.2.3) that represents the directory partition that is targeted by the PSCAckMessage (section 2.2.6.7).
- *iLastSeqNumber*: The last sequence number that is associated with the partition.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- Create a PSCAckMessage, which is referred to as *pscAck*, with the following attributes:
  - BaseReplicationHeader.Operation := 0x06.
  - PSCPartitionID := **QueueManager.PartitionID**.
  - AckedPartitionID := *iPartition.PartitionID*.

    - AckedSeqNumber **:=***iLastSeqNumber*.
    - PSCName **:= QueueManager.MachineName**.
- Generate a Send Replication Message event (section 3.1.7.2.1) with the following arguments:
    - iDestination := *iPartition.AuthorityName*.
    - iReplicationMessage := *pscAck*.
    - iTimeout := 20 * 60.
    - iPriority := 3.
    - iAckRequested := AckNackReachQueue.
    - iAdminQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager GUID.
    - iResponseQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager GUID.

3.3.7.5 Receive Sync Reply

This event MUST be generated with the following arguments:

- *iSyncReplyMessage*: A reference to a SyncReplyMessage object (section 2.2.6.5).

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- Find a DirectoryPartition object (section 3.1.1.2.3), referred to as *dirPartition*, in the **DirectoryPartitionCollection**, where:
    - *dirPartition.PartitionID* is the same as *iSyncReplyMessage.SyncReplyHeader.PartitionID*.
- If *dirPartition* is not found:
    - Take no further action.
- If *iSyncReplyMessage.SyncReplyHeader.PurgedSeqNumber* < *dirPartition.PurgedSeqNumber*:
    - Take no further action.
- For each DirectoryChange object (section 2.2.4), referred to as *dirChange*, in *iSyncReplyMessage.DirectoryChanges*:
    - Process *dirChange* as specified in section 3.1.5.2.
- Set *sync0State* to *iSyncReplyMessage.SyncReplyHeader.CompleteSync0*.
- Update the purge state according to *sync0State* as follows:
    - Set *toSeqNumber* to *iSyncReplyMessage.SyncReplyHeader.ToSeqNumber*.
    - Set *purgedSeqNumber* to *iSyncReplyMessage.SyncReplyHeader.PurgedSeqNumber*.

- If *dirPartition.PurgeState* is 2 (PURGE_STATE_SYNC0):
  - If *purgedSeqNumber* < *dirPartition.PurgedSeqNumber*:
    - Take no further action.
  - If *sync0State* is 2 (PURGE_STATE_SYNC0):
    - Raise a Complete Sync0 State event (section 3.3.7.11) with the following arguments:
      - iPartition := *dirPartition.*
  - Else if *sync0State* is 1:
    - There are more updates for this partition. Raise a Send Sync Request event (section 3.3.7.3) with the following arguments:
      - iPartition := *dirPartition.*
- Else If *dirPartition.PurgeState* is 0 (PURGE_STATE_NORMAL):
  - If *dirPartition.ChangeMissingWindow* is MAX_SEQ_NUMBER:
    - Set *dirPartition.ChangeMissingWindow* to MIN_SEQ_NUMBER.
  - Else if *iSyncReplyMessage.SyncReplyHeader.Count* is 0:
    - If *dirPartition.ChangeMissingWindow* is equal to *toSeqNumber*:
      - Set *dirPartition.ChangeMissingWindow* to MIN_SEQ_NUMBER.
    - Set *dirPartition.LastSeqNumber* to *toSeqNumber*.
  - Raise a Check Pending Changes event (section 3.1.7.2.4) with the following arguments:
    - iPartition := *dirPartition.*

3.3.7.6 Receive Change Reply

This event MUST be generated with the following arguments:

- *iChangeReplyMessage*: A reference to a ChangeReplyMessage data element (section 2.2.6.3).

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- Find the Process Change Request event (section 3.3.7.1) that is identified by *iChangeReplyMessage.RequestIdentifier*.
- If event is not found:
  - Take no further action.
- Continue processing the found Process Change Request event with the following value, as specified in section 3.3.7.1:

    - *iChangeReplyMessage.Status*.

3.3.7.7 Receive Already Purged

This event MUST be generated with the following arguments:

- *iAlreadyPurgedMessage*: A reference to an AlreadyPurgedMessage object (section 2.2.6.8).

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- Find a DirectoryPartition object (section 3.1.1.2.3), referred to as *dirPartition* in the **DirectoryPartitionCollection**, where:
    - *dirPartition.PartitionID* is the same as *iAlreadyPurgedMessage.PartitionID*.
- If *dirPartition* is not found:
    - Take no further action.
- If *dirPartition.LastSeqNumber* >= *iAlreadyPurgedMessage.PurgedSeqNumber*:
    - Take no further action.
- If *dirPartition.PurgedSeqNumber* >= *iAlreadyPurgedMessage.PurgedSeqNumber*:
    - Take no further action.
- Raise a Start Sync0 State event (section 3.3.7.10) with the following arguments:
    - iPartition := *dirPartition.*

3.3.7.8 Receive Change Request Nack

This event MUST be generated with the following arguments:

- *iChangeRequestMessage*: A reference to a ChangeRequestMessage object (section 2.2.6.2).

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- Locate the Process Change Request event (section 3.3.7.1) that is identified by *iChangeRequestMessage.RequestIdentifier*.
- If the event is not found:
    - Take no further action.
- Continue processing the Process Change Request event with the following value, as specified in section 3.3.7.1:
    - MQDS_OWNER_NOT_REACHED.

3.3.7.9 Receive Request Nack

This event MUST be generated with the following arguments:

- *iNackMessage*: A reference to a message object.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- If *iNackMessage.Class* is not **NackBadSignature**:
  - Take no further action.
- Raise a Send Replication Message event (section 3.1.7.2.1) with the following arguments:
  - iDestination := Machine in *iNackMessage.DestinationQueueFormatName*.
  - iReplicationMessage := *iNackMessage.Body*.
  - iTimeout := 10.
  - iPriority := 3.
  - iAckRequested := **AckNackReachQueue**.
  - iAdminQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager GUID.
  - iResponseQueueFormatName := REPLICATION_QUEUE_PRIVATE, where <Queue Manager GUID> is set to the sending queue manager GUID.

3.3.7.10 Start Sync0 State

This event MUST be generated with the following arguments:

- *iPartition*: A reference to a DirectoryPartition object (section 3.1.1.2.3) representing the directory partition that is starting the **sync0** state.
- *iPurgedSeqNumber*: A sequence number returned in the AlreadyPurgedMessage (section 2.2.6.8) that triggered this event.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- If *dirPartition.LastSeqNumber* >= *iPurgedSeqNumber* or *dirPartition.PurgedSeqNumber* >= *iPurgedSeqNumber*:
  - Take no further action.
- Set *iPartition.PurgeState* to 1 (PURGE_STATE_STARTSYNC0).
- Set *iPartition.LastSeqNumber* to MIN_SEQ_NUMBER.

- Set *iPartition.ChangeMissingWindow* to MAX_SEQ_NUMBER.
- Set *iPartition.PurgedSeqNumber* to *iPurgedSeqNumber*.
- Remove all elements from *iPartition.PendingChangeCollection*.
- Raise an MQDS **Reset Directory Partition** event with the following arguments:
  - iPartitionID := *iPartition.PartitionID*.
  - iOperation := 1 (mark all objects as deleted by setting their sequence numbers to MIN_SEQ_NUMBER).
- If the **Reset Directory Partition** event does not return MQ_OK:
  - Retry the **Reset Directory Partition** event with the same arguments until it succeeds.
- For each element, referred to as *objDeleted* in **DeletedObjectCollection**:
  - If *objDeleted.PartitionID* is equal to *iPartition.PartitionID*:
    - Set *objDeleted.SeqNumber* to MIN_SEQ_NUMBER.
- Set *iPartition.PurgeState* to 2 (PURGE_STATE_SYNC0).
- Raise a Send Sync Request event (section 3.3.7.3) with the following arguments:
  - iPartition := *iPartition*.

3.3.7.11 Complete Sync0 State

This event MUST be generated with the following argument:

- *iPartition*: A reference to a DirectoryPartition object (section 3.1.1.2.3) representing the directory partition that is completing the **sync0** state.

Return Values:

- None.

The queue manager MUST perform the following actions to process this event:

- If *iPartition.PurgeState* is not 2 (PURGE_STATE_STARTSYNC0) or 3 (PURGE_STATE_SYNC0):
  - Take no further condition.
- Raise an MQDS **Reset Directory Partition** event with the following arguments:
  - iPartitionID := *iPartition.PartitionID*.
  - iOperation := 2 (delete all objects whose sequence number is MIN_SEQ_NUMBER).
- If the **Reset Directory Partition** event does not return MQ_OK:
  - Retry the **Reset Directory Partition** event with the same arguments until it succeeds.
- For each element, referred to as *objDeleted* in the **DeletedObjectCollection**:

    - If *objDeleted.PartitionID* is equal to *iPartition.PartitionID* and *objDeleted.SeqNumber* is equal to MIN_SEQ_NUMBER:
        - Remove *objDeleted* from **DeletedObjectCollection**.
- Set *iPartition.PurgeState* to 0 (PURGE_STATE_NORMAL).
- Set *iPartition.ChangeMissingWindow* to MIN_SEQ_NUMBER.
- Raise a Check Pending Changes event (section 3.1.7.2.4) with the following argument:
    - iPartition := *iPartition*.

4 Protocol Examples

This section provides common scenarios that exemplify MQDSRP protocol functionality. The examples in this section begin with an MSMQ enterprise, as described in the following figure.

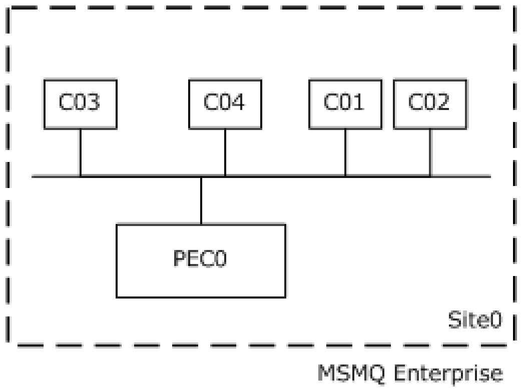

Figure 7: Default site in the MSMQ enterprise

In this MSMQ enterprise, there is one site (site0). The enterprise controller (also the site controller for site0) is PEC0. There are four directory clients, C01 through C04. Because there is only one directory server in the enterprise, replication is not required.

Changes to the enterprise may require different types of directory objects to be created or updated in the directory. The protocol examples in this section show how the MQDSRP protocol interacts with other protocols and how directory changes are replicated throughout the enterprise.

In the examples in this section, the term "MQDS server" refers to the MQDS protocol server that is co-located with the MQDSRP protocol on a common queue manager.

4.1 Adding a Site to the Enterprise

When a site is added, a site object is created in the enterprise partition and a directory partition is created for the new site. Assuming that the new site is named site1 and the queue manager acting as the PSC resides on machine PSC1, the following figure shows the sequences of adding a site to the enterprise from the data replication perspective.

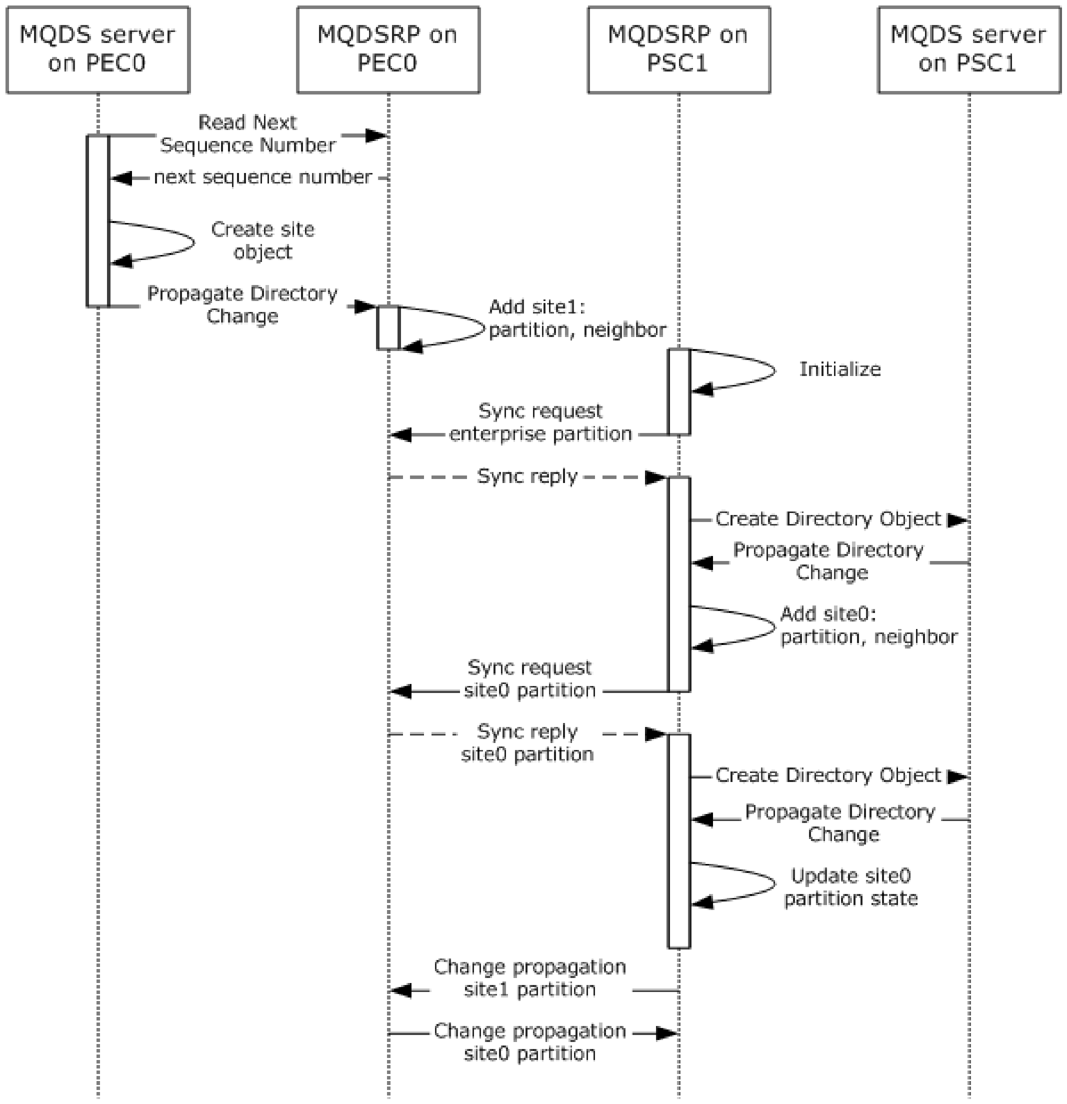
MQDS server on PEC0
MQDSRP on PEC0
MQDSRP on PSC1
MQDS server on PSC1
Read Next Sequence Number
next sequence number
Create site object
Propagate Directory Change
Add site1: partition, neighbor
Initialize
Sync request enterprise partition
Sync reply
Create Directory Object
Propagate Directory Change
Add site0: partition, neighbor
Sync request site0 partition
Sync reply site0 partition
Create Directory Object
Propagate Directory Change
Update site0 partition state
Change propagation site1 partition
Change propagation site0 partition

Figure 8: Adding a site to the enterprise

A site object is created to represent the new site by the object authority, which is PEC0 in this example. The MQDS server on PEC0 creates the site object and triggers the Propagate Directory Change (section 3.1.7.1.2) event. The MQDSRP event handler creates a new site partition and adds PSC1 as a PSC neighbor of PEC0.

When the PSC1 queue manager performs the MQDSRP protocol client initialization, as specified in section 3.3.3, the **DirectoryPartitionCollection** is empty. Therefore, it creates a directory partition for enterprise-wide objects and raises a Send Sync Request (section 3.3.7.3) event for that partition, which causes a SyncRequestMessage (section 2.2.6.4) to be sent to PEC0.

PEC0 receives the synchronization request and prepares and sends back a SyncReplyMessage (section 2.2.6.5) that contains all objects in the enterprise partition.

When the PSC1 queue manager receives the synchronization reply from PEC0, it triggers the MQDS Create Directory Object event to create all objects, including an enterprise object and two site objects (for site0 and site1). For each site object, the protocol creates a directory partition, as specified in sections 3.1.7.2.8 and 3.1.7.2.11. For site0, the protocol also creates a PSC neighbor object that represents PEC0. The PSC1 queue manager raises a **Send Sync Request** event, which causes a **SyncRequestMessage** to be sent to PEC0 for the default site partition.

PEC0 receives the synchronization request and prepares and sends back a **SyncReplyMessage** that contains all objects in the site0 partition.

When the PSC1 queue manager receives the synchronization reply from PEC0, it triggers the MQDS **Create Directory Object** event to create all objects in the site0 partition.

At this point, PSC1 has obtained all directory data from PEC0 for PEC0's partitions.

When new directory changes occur in site 1, PSC1 propagates these changes to PEC0 because PEC0 is now a PSC neighbor of PSC1.

Another site (site2) can be added with the same process. The difference is that the site controller PSC2 creates two site partitions when it receives the first synchronization reply from PEC0, and it requests site partition synchronization with both PEC0 and PSC1 to populate objects in these two partitions.

After the two sites are added, the MSMQ enterprise configuration is as follows:

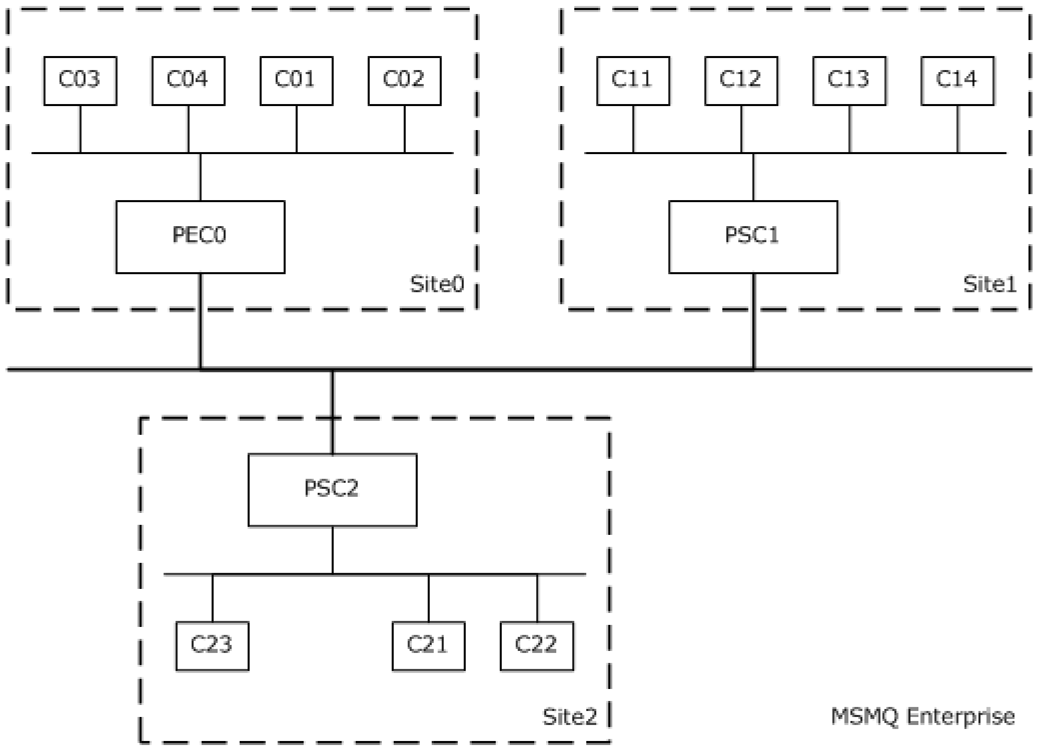

Figure 9: The MSMQ enterprise with three sites

Each of the three sites has a site controller, and directory service replication can now occur among the three controllers.

4.2 Adding a BSC to a Site

A site can have zero or more BSCs. This example describes how the directory data is replicated after C03 in site0 is assigned with a BSC role. C03 is renamed to BSC01 after it becomes a BSC. The sequence of adding a BSC to site0 is described in the following figure.

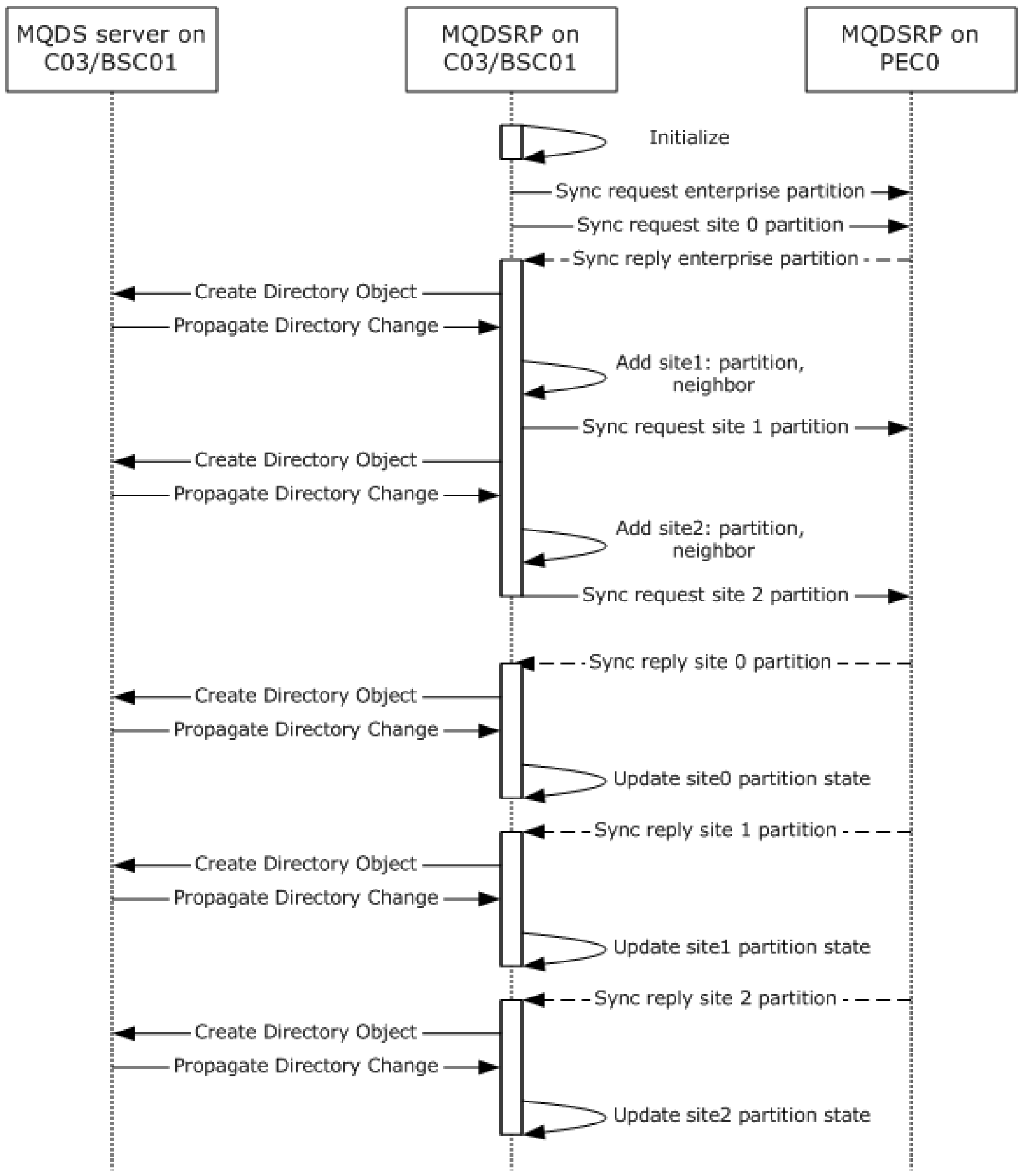
MQDS server on C03/BSC01
MQDSRP on C03/BSC01
MQDSRP on PEC0
Initialize
Sync request enterprise partition
Sync request site 0 partition
Sync reply enterprise partition
Create Directory Object
Propagate Directory Change
Add site1: partition, neighbor
Sync request site 1 partition
Create Directory Object
Propagate Directory Change
Add site2: partition, neighbor
Sync request site 2 partition
Sync reply site 0 partition
Create Directory Object
Propagate Directory Change
Update site0 partition state
Sync reply site 1 partition
Create Directory Object
Propagate Directory Change
Update site1 partition state
Sync reply site 2 partition
Create Directory Object
Propagate Directory Change
Update site2 partition state

Figure 10: Adding a BSC to a site

When the BSC01 queue manager performs the MQDSRP protocol client initialization, as specified in section 3.3.3, the **DirectoryPartitionCollection** is empty. Therefore, it creates a directory partition for enterprise-wide objects and then raises a Send Sync Request (section 3.3.7.3) event for that partition, which causes a SyncRequestMessage (section 2.2.6.4) to be sent to PEC0.

The synchronization request asks for all directory changes in the enterprise partition, so PEC0 reads the directory objects in that partition and sends them as directory changes back to BSC01 in a SyncReplyMessage (section 2.2.6.5).

When BSC01 receives the reply message, it creates the directory objects and three site partitions for the three sites in the enterprise, as specified in sections 3.1.7.2.8 and 3.1.7.2.11. For each new partition, a **Send Sync Request** event is raised and the data for that site is similarly synchronized. Because BSC01 is a BSC, the **SyncRequestMessage** is sent to its PSC, which is PEC0 in the example, as specified in section 3.3.7.3.

The same process can be used to add more BSCs to a site. This results in the example enterprise referenced in section 1.3.1.1. The following figure shows the same enterprise with the machine names labeled.

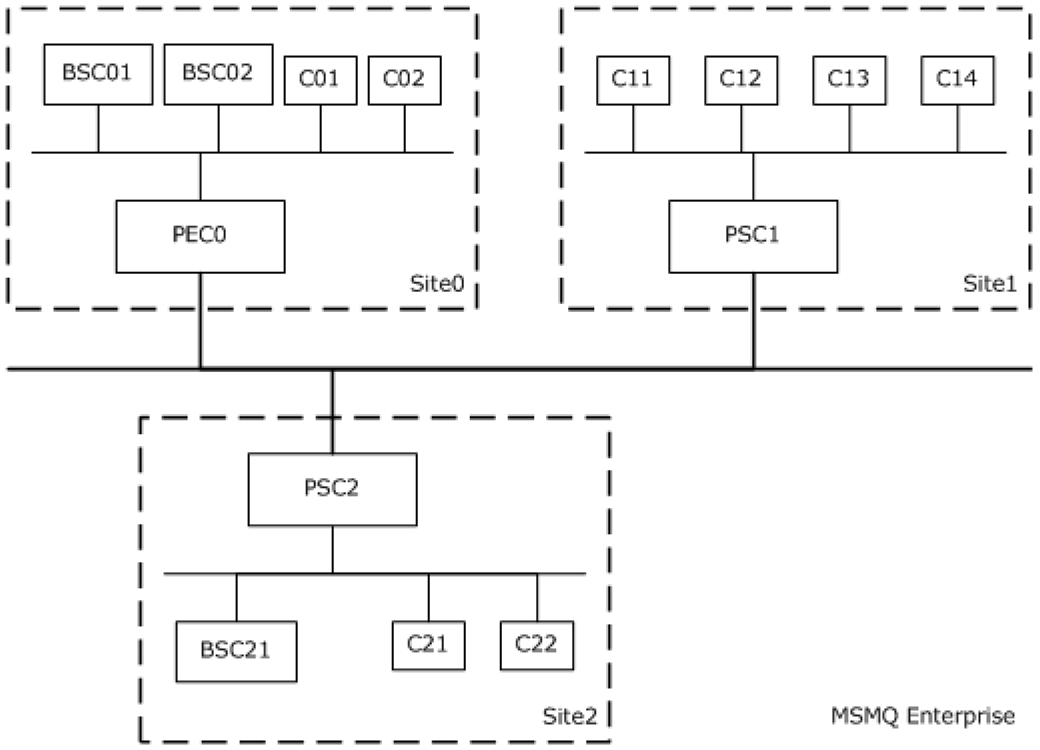

Figure 11: Example MSMQ enterprise

4.3 Creating a Remote Queue

This example shows how directory client C22 (in the MSMQ enterprise illustration) creates a queue object "C14\testq". Assuming that C22 uses BSC21 as the MQDS supporting server for this operation, the following figure summarizes the sequence of actions.

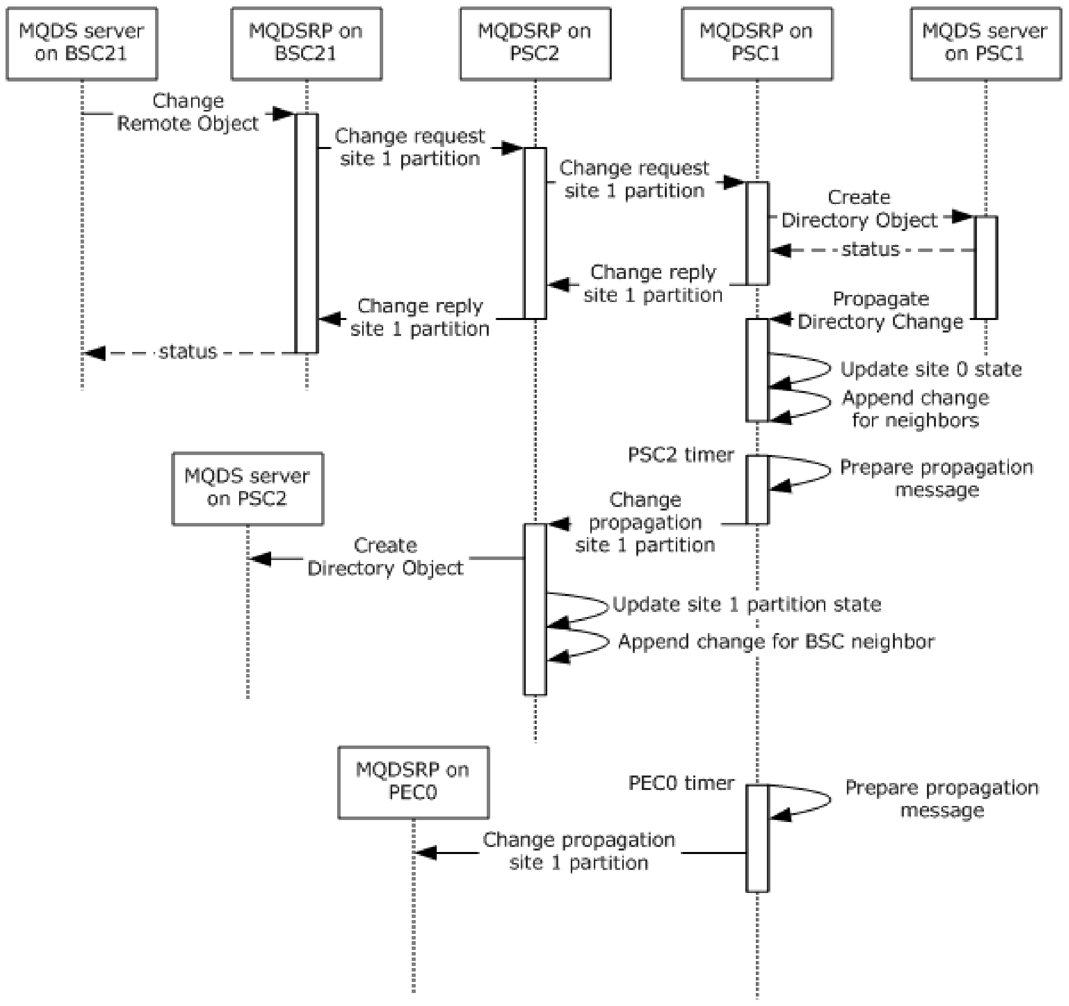
MQDS server on BSC21
MQDSRP on BSC21
MQDSRP on PSC2
MQDSRP on PSC1
MQDS server on PSC1
Change Remote Object
Change request site 1 partition
Change request site 1 partition
Create Directory Object
status
Change reply site 1 partition
Change reply site 1 partition
Propagate Directory Change
status
Update site 0 state
Append change for neighbors
PSC2 timer
Prepare propagation message
MQDS server on PSC2
Change propagation site 1 partition
Create Directory Object
Update site 1 partition state
Append change for BSC neighbor
MQDSRP on PEC0
PEC0 timer
Prepare propagation message
Change propagation site 1 partition

Figure 12: Sequence of creating a remote queue and change propagation

In this scenario, directory client C22 calls the MQDS protocol client when trying to create the queue. The MQDS server on BSC21 receives this request. It checks the ownership of the object by extracting the machine name from the queue name and locating the partition that the machine belongs to. Because the machine C14 belongs to site 1, BSC21 is not the authority of the queue object, so the MQDS server on BSC21 raises a MQDSRP Change Remote Object (section 3.1.7.1.1) event for creating this remote queue.

The MQDSRP on BSC21 handles this event by constructing a ChangeRequestMessage (section 2.2.6.2) and sending it to its PSC, machine PSC2.

PSC2 receives the **ChangeRequestMessage** and forwards the request to PSC1, the object authority.

PSC1 receives the **ChangeRequestMessage** and raises an MQDS **Create Directory Object** event to create the queue object as specified in the directory change contained in the message. The status returned from the MQDS server is sent back to PSC2 inside a ChangeReplyMessage (section 2.2.6.3).

PSC2 receives the **ChangeReplyMessage** and forwards the status to BSC21 inside another **ChangeReplyMessage**.

BSC21 receives the **ChangeReplyMessage** and returns the contained status to the MQDS server.

After creating the queue object, the MQDS server on PSC1 also raises a Propagate Directory Change (section 3.1.7.1.2) event. The MQDSRP event handler on PSC1 updates the replication state of the site 1 partition and appends the directory change to the available change collection of all its neighbors (PSC2 and PEC0).

When the PSC2 propagation timer fires, MQDSRP on PSC1 creates a ChangePropagationMessage (section 2.2.6.9) that contains this queue creation operation, and it sends the message to PSC2.

PSC2 receives the **ChangePropagationMessage**, and it raises an MQDS **Create Directory Object** event to create the queue in its directory neighbor, updates the replication state of the site 1 partition, and appends the change to the available change collection of all its BSC neighbors (BSC21).

When the BSC21 propagation timer fires on PSC2, this change is propagated to BSC21 in a similar process.

Finally, when the PEC0 propagation timer fires on PSC1, this change is propagated to PEC0 and then to its BSC neighbors BSC01 and BSC02.

4.4 Deleting a Queue

Handling a change request for deleting a queue is the same as creating it, as indicated in the example in section 4.3. The following figure illustrates how a deletion is propagated and acknowledged by neighbors.

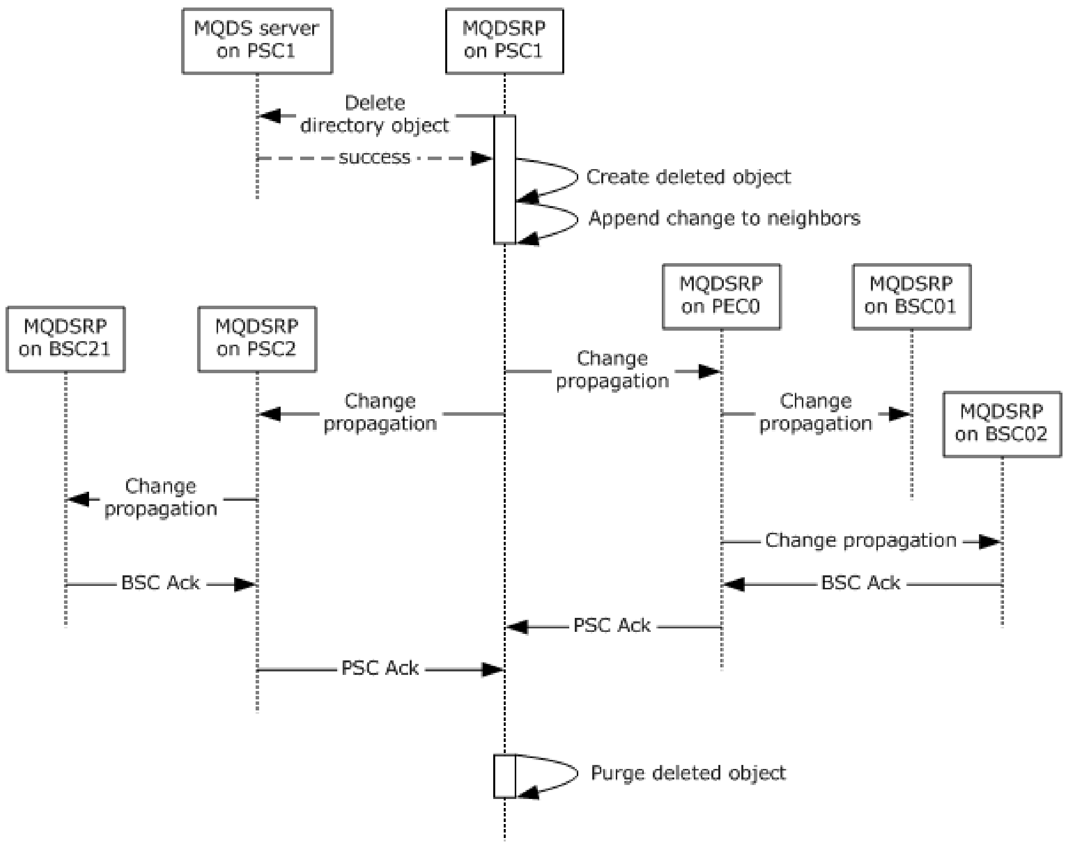

Figure 13: Sequence of deleting a queue and change propagation

After the object authority PSC1 deletes the queue object, it creates a deleted object that represents this deletion operation. A directory change is also created for this operation and appended to the available change collection of neighbors PEC0 and PSC2.

The change is propagated to PEC0 and PSC2 and then to their BSC neighbors BSC01, BSC02, and BSC21.

When the BSC Ack Timer (section 3.3.2.2) event fires, BSC02 and BSC21 send a BSCAckMessage (section 2.2.6.6) to their PSCs, PEC0 and PSC2, respectively. In this example, there is no **BSC Ack** from BSC01 to PEC0. This may happen if the BSC01 queue manager failed to send the message or the machine crashed, or for some other reasons. Later PEC0 and PSC2 send PSCAckMessages (section 2.2.6.7) to PSC1.

Note that the **BSC Ack** is controlled by a timer (sections 3.3.2.2 and 3.3.6.2), and the **PSC Ack** is controlled by the incoming directory changes count (section 3.1.7.2.3). They are not a direct response to the ChangePropagationMessage (section 2.2.6.9) that contains the queue deletion operation.

Assuming that the sequence number associated with the queue deletion is 8 (zero bytes omitted), the acknowledged sequence number from PSC2 is 10, and the acknowledged sequence number from PEC0 is 9, PSC1 purges the deleted object when the Purge Deleted Objects (section 3.1.7.2.9) event is triggered.

4.5 Full Partition Synchronization

In the example of section 4.4, assume that BSC01 crashed after it received the change propagation from PEC0 and remained offline for some time. The last sequence number it maintains for the site 1 partition is 8 (the one associated with the queue deletion operation).

With the replication occurring, the last sequence number and purged sequence number that PEC0 maintains for the site 1 partition are now both 20.

When BSC01 comes online again, it sends a SyncRequestMessage (section 2.2.6.4) asking for directory changes with sequence number from 9 to MAX_SEQ_NUMBER (section 2.2.2).

PEC0 provides a Send Already Purged (section 3.2.7.7) reply to BSC01 because the requested beginning sequence number is less than the purged sequence number for the site 1 partition.

BSC01 receives an AlreadyPurgedMessage (section 2.2.6.8). It then begins a full-partition synchronization for the site 1 partition by setting the sequence number of all objects in the site 1 partition to 0, which resets the replication state of the site 1 partition and resends a **SyncRequestMessage** to PEC0.

PEC0 receives the new **SyncRequestMessage**, generates directory changes for all objects in the site 1 partition, and sends them to BSC01 in a SyncReplyMessage (section 2.2.6.5).

BSC01 rebuilds the site 1 partition according to the received directory changes.

The sequence of the full-partition synchronization is described in section 3.1.1.5.3.

5 Security

5.1 Security Considerations for Implementers

This protocol sends directory replication data in MSMQ messages. It may allow attacks that tamper with the replication data and cause data corruption in the directory. An implementation of this protocol should develop mechanisms to prevent such attacks, such as using the MSMQ message integrity security feature to ensure that the messages are not altered during transfer.<13>

5.2 Index of Security Parameters

There are no security parameters for this protocol.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows NT Server operating system
- Windows 2000 Server operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 1.6: Windows NT Server contains an MSMQ Directory Service that implements the MSMQ: Directory Service Replication Protocol (MQDSRP). Windows 2000 Server implements the MQDSRP protocol to support interoperability between the MSMQ Directory Service in the Windows NT Server environment and the directory service that is provided by Active Directory in the Windows 2000 Server environment. Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows 8, and Windows Server 2012 do not support this protocol.

<2> Section 3.1.7.2.5: The default value of an intersite (from a PSC to its PSC neighbor) Change Propagation Timer is 10 seconds. The default value of an intrasite (from a PSC to its BSC neighbor) Change Propagation Timer is 2 seconds.

<3> Section 3.1.7.2.5: The default value of an intersite (from a PSC to its PSC neighbor) Change Propagation Timer is 10 seconds. The default value of an intrasite (from a PSC to its BSC neighbor) Change Propagation Timer is 2 seconds.

<4> Section 3.1.7.2.11: The default value of an intersite (from a PSC to its PSC neighbor) Change Propagation Timer is 10 seconds. The default value of an intrasite (from a PSC to its BSC neighbor) Change Propagation Timer is 2 seconds.

<5> Section 3.1.7.2.12: The default value of an intersite (from a PSC to its PSC neighbor) Change Propagation Timer is 10 seconds. The default value of an intrasite (from a PSC to its BSC neighbor) Change Propagation Timer is 2 seconds.

<6> Section 3.3.2.1: The default value of an inter-site (from a PSC to its PSC neighbor) **Change Propagation Timer** is 10 seconds. The default value of an intra-site (from a PSC to its BSC neighbor) **Change Propagation Timer** is 2 seconds.

<7> Section 3.3.2.2: The default value of a **BSC Ack Timer** is 5 seconds for initialization and 12 hours for the next instances.

<8> Section 3.3.3: The default value of a BSC Ack Timer is 5 seconds for initialization and 12 hours for the next instances.

<9> Section 3.3.3: The default value of an inter-site (from a PSC to its PSC neighbor) **Change Propagation Timer** is 10 seconds. The default value of an intra-site (from a PSC to its BSC neighbor) **Change Propagation Timer** is 2 seconds.

<10> Section 3.3.3: The default value of an inter-site (from a PSC to its PSC neighbor) **Change Propagation Timer** is 10 seconds. The default value of an intra-site (from a PSC to its BSC neighbor) **Change Propagation Timer** is 2 seconds.

<11> Section 3.3.7.1: The default value of the waiting time is set as follows:

- 10 seconds if the next hop is the object authority.
- 20 seconds if the next hop is not the object authority. This happens when a BSC accepts a directory change request and the object authority is not its PSC. In such cases, the BSC sends the request to its PSC and the PSC forwards the request to the object authority.

<12> Section 3.3.7.2: The default value of the interval for sending the SeqNumberHeader in a ChangePropagationMessage (section 2.2.6.9) is 20 minutes.

<13> Section 5.1: In the Windows implementation of this protocol, the MSMQ messages that contain replication data are signed by the sending queue manager.

7 Change Tracking

This section identifies changes that were made to the [MC-MQDSRP] protocol document between the October 2012 and January 2013 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.2.6.2 ChangeRequestMessage	67392 Updated details of the ChangeRequestHeader Operation field value.	N	Content updated.
2.2.6.5 SyncReplyMessage	67392 Updated details of the SyncReplyHeader element Operation field value.	N	Content updated.
3.1.7.1.2 Propagate Directory Change	67393 Added details for the synchronize (0x03) iOperation case.	N	Content updated.
3.2.7.2 Receive Change Request	67394 Added details for the synchronize (0x03) dirChange.Command case.	N	Content updated.

8 Index