[MS-CMOM]:
MSDTC Connection Manager:
OleTx Management Protocol Specification

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

Date	Revision History	Revision Class	Comments
07/20/2007	0.1	Major	MCPP Milestone 5 Initial Availability
09/28/2007	1.0	Major	Updated and revised the technical content.
10/23/2007	1.1	Minor	Revised a Windows Behavior note.
11/30/2007	1.1.1	Editorial	Revised a Windows Behavior note.

Date	Revision History	Revision Class	Comments
01/25/2008	1.1.2	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction

This document specifies the MSDTC Connection Manager: OleTx Management Protocol, which enables the remote management of an **OleTx Transaction Manager** and its extensions: MSDTC Connection Manager: OleTx Transaction Internet Extension ([MS-DTCM]), the Transaction Internet Protocol (TIP) Extensions ([MS-TIPP]), and MSDTC Connection Manager: OleTx XA Protocol Extension ([MC-DTCXA]).

Remote management allows a user to do the following:

- Subscribe to and receive **transaction** monitoring information by using the MSDTC Connection Manager: OleTx Multiplexing Protocol (as specified in [MS-CMP]).
- Get and set configuration information contained in registry keys by using the subscribe to and receive transaction monitoring information using the Windows Remote Registry Protocol (as specified in [MS-RRP]).
- Manage the **server** life cycle (such as starting and stopping it) by using the Service Control Manager Remote Protocol (as specified in [MS-SCMR]).

1.1 Glossary

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

Application
Authentication Level
Client
Connection
Connection Type
Contact Identifier (CID)
Endpoint
Globally Unique Identifier (GUID)
Incoming Authentication
Message
Message Tag (MTAG)
Mutual Authentication
Protocol Extension
Protocol Role
Recovery
Server
Session
Transaction
Transaction Identifier
Transaction Manager
Transaction Propagation

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

Enlistment
OleTx

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

Service
Service Control Manager (SCM)

The following terms are specific to this document:

Higher-Layer Business Logic: The **application** functionality that invokes the functionality that is specific to this protocol.

Name Object: An object that contains **endpoint** contact information, as specified in the Name Object section of [MS-CMPO].

Protocol Participant: An implementation of one of the **protocol roles**.

OleTx Transaction Manager: A **Transaction Manager** as specified in [MS-DTCO].

Transaction Monitoring: The functionality that allows the state and progress of transactions managed by a **Transaction Manager** to be monitored.

XA Transaction Manager: A **Transaction Manager** as specified in [MC-DTCXA].

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

1.2 References

1.2.1 Normative References

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

[ISO-8859-1] International Organization for Standardization, "Information Technology -- 8-Bit Single-Byte Coded Graphic Character Sets -- Part 1: Latin Alphabet No. 1", ISO/IEC 8859-1, 1998, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=28245&ICS1=35&ICS2=40&ICS3=

Note There is a charge to download the specification.

[MC-DTCXA] Microsoft Corporation, "MSDTC Connection Manager: OleTx XA Protocol Specification" October 2007.

[MS-CMP] Microsoft Corporation, "MSDTC Connection Manager: OleTx Multiplexing Protocol Specification", July 2007.

[MS-CMPO] Microsoft Corporation, "MSDTC Connection Manager: OleTx Transports Protocol Specification", July 2007.

[MS-DTCO] Microsoft Corporation, "MSDTC Connection Manager: OleTx Transaction Protocol Specification", July 2007.

[MS-DTCM] Microsoft Corporation, "MSDTC Connection Manager: OleTx Transaction Internet Protocol Specification", August 2007.

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

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

[MS-RRP] Microsoft Corporation, "Windows Remote Registry Protocol Specification", August 2007.

[MS-SECO] Microsoft Corporation, "Windows Security Overview", January 2007.

[MS-SCMR] Microsoft Corporation, "Service Control Manager Remote Protocol Specification", August 2007.

[MS-TIPP] Microsoft Corporation, "Transaction Internet Protocol (TIP) Extensions", September 2007.

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

1.2.2 Informative References

[MSDN-WINSVC] Microsoft Corporation, "Services", http://msdn2.microsoft.com/en-us/library/ms685141.aspx

1.3 Protocol Overview (Synopsis)

The OleTx Management Protocol facilitates remote management of the OleTx Transaction Manager and its extensions: OleTx Transaction Internet Extension ([MS-DTCM]), the Transaction Internet Protocol Extensions ([MS-TIPP]), and the OleTx XA Protocol Extension ([MC-DTCXA]).

Figure 1 illustrates the Management Client role and Management Server role, and lists the protocols managed by the OleTx Management Protocol.

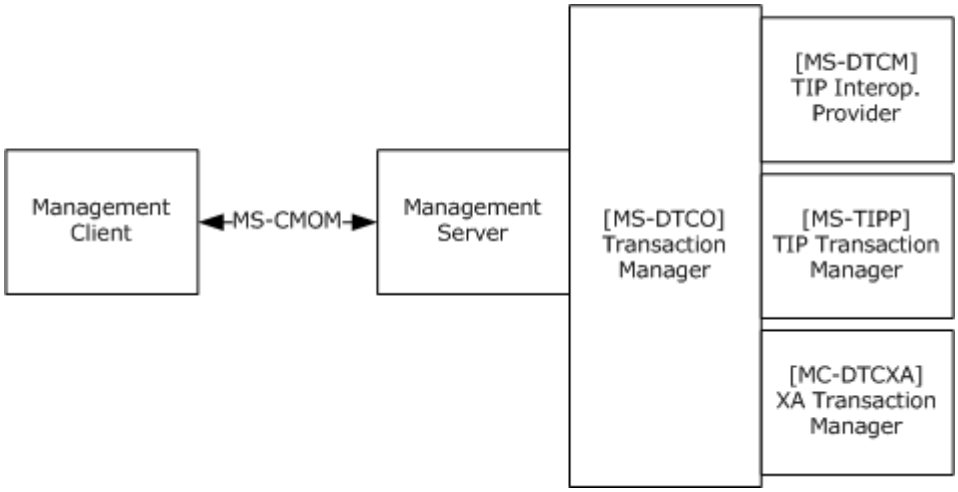


Figure 1: Overview

1.3.1 Management Client Role

The Management Client **protocol role** is remote to the **Transaction Manager**. It can take several forms, such as a console that monitors transactions, a configuration management tool, or a remote proxy to a Transaction Manager needing **connection** information.

This role performs the following tasks:

- Requests reads and writes of the configuration information for the managed protocols using registry keys exposed via the Windows Remote Registry Protocol ([MS-RRP]). The Management Client expects updates to the configuration keys to be durably stored. Some of the updates do

not take effect until the Management Server is restarted. The configuration keys include the following categories:

- Security keys: These contain configuration settings that enable different forms of distributed access.
- Remote proxy keys: These contain implementation-specific information needed by a proxy **application** to contact the Transaction Manager.
- Transaction Manager Contact Identifier (CID) keys: These contain information for the Transaction Manager **endpoints** corresponding to the four supported protocols.
  - The OleTx Transaction Manager ([MS-DTCO])
  - The OleTx XA Protocol Extension ([MC-DTCXA])
  - The OleTx Transaction Internet Extension ([MS-DTCM])
  - The OleTx Management Protocol (as specified in this document).
- Transaction monitoring keys: These keys contain settings concerning the frequency and filtering of Transaction monitoring information.
- Tracing configuration keys: These keys contain the settings for controlling the different degrees of tracing.

- Requests reads and writes of the configuration information about subscriptions to transaction monitoring information. This includes the rate of refresh and the levels of detail. This configuration information is common to all Management Clients. There are two ways to set this information.
  - If set by using the Windows Remote Registry Protocol ([MS-RRP]), configuration information will be durably stored but will not take effect until the Management Server is restarted.
  - If set by using the OleTx Multiplexing Protocol ([MS-CMP]), configuration information will not be durably stored, but will take effect immediately.
- Subscribes to receive transaction monitoring information by connecting to the Management Server via a CONNTYPE_TXUSER_DTCUIC connection using the OleTx Multiplexing Protocol (as specified in [MS-CMP]).
- Sends requests to the Management Server in order to update the settings controlling the frequency and detail of the transaction monitoring information.
- Receives transaction monitoring information at the configured rate and including the configured detail.
- Manages the life cycle of the Transaction Manager **service** (for example, starting and stopping) and queries for its status using the Service Control Manager Remote Protocol ([MS-SCMR]).

1.3.2 Management Server Role

The Management Server protocol role is an extension to the Transaction Manager.

This role performs the following tasks:

- Receives reads of the configuration information for the managed protocols by using the Windows Remote Registry Protocol ([MS-RRP]) and returns the requested information. The role receives writes of this information by using the Windows Remote Registry Protocol and durably stores it.
- Receives writes for the settings controlling the frequency and detail level of the transaction monitoring information that it pushes to all connected Management Clients. There are two ways that these settings can be updated:
  - If updated via the Windows Remote Registry Protocol ([MS-RRP]), the settings are durably stored but they do not take effect until the Management Server is restarted.
  - If updated via the OleTx Multiplexing Protocol ([MS-CMP]), the settings are not durably stored but they take effect immediately.
- Receives a subscription to transaction monitoring information. The Management Server does not durably store these requests.
- Captures transaction monitoring information.
- Publishes transaction monitoring information to all subscribers at the configured rate and including the configured detail.
- Receives actions that manage the server life cycle (e.g., start and stop) using the Service Control Manager Remote Protocol ([MS-SCMR]), and carries out these requests.

1.3.3 Common Scenarios

This section provides some common scenarios for the OleTx Management Protocol.

1.3.3.1 Setting and Enabling a Configuration Setting

Some configuration settings do not take effect until the Management Server is restarted. An example of this is the key "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDTC\Security\XaTransactions", which, when set to 1, enables the Transaction Manager to manage XA transactions.

Figure 2 illustrates a Transaction Manager configuration console using the Management Client protocol role to enable XA transactions, and then force the new setting to take effect by restarting the Management Server:

1. The first exchange reads XaTransactions and discovers that its value is 0, which means it is not enabled. The read operation is performed by using the Windows Remote Registry Protocol ([MS-RRP]).
2. The second exchange sets the XaTransactions key to 1. This operation is performed by using the Windows Remote Registry Protocol.
3. The third exchange stops the server. This operation is performed using the Service Control Manager Remote Protocol ([MS-SCMR]).
4. The fourth exchange starts the server. This operation is performed using the Service Control Manager Remote Protocol.

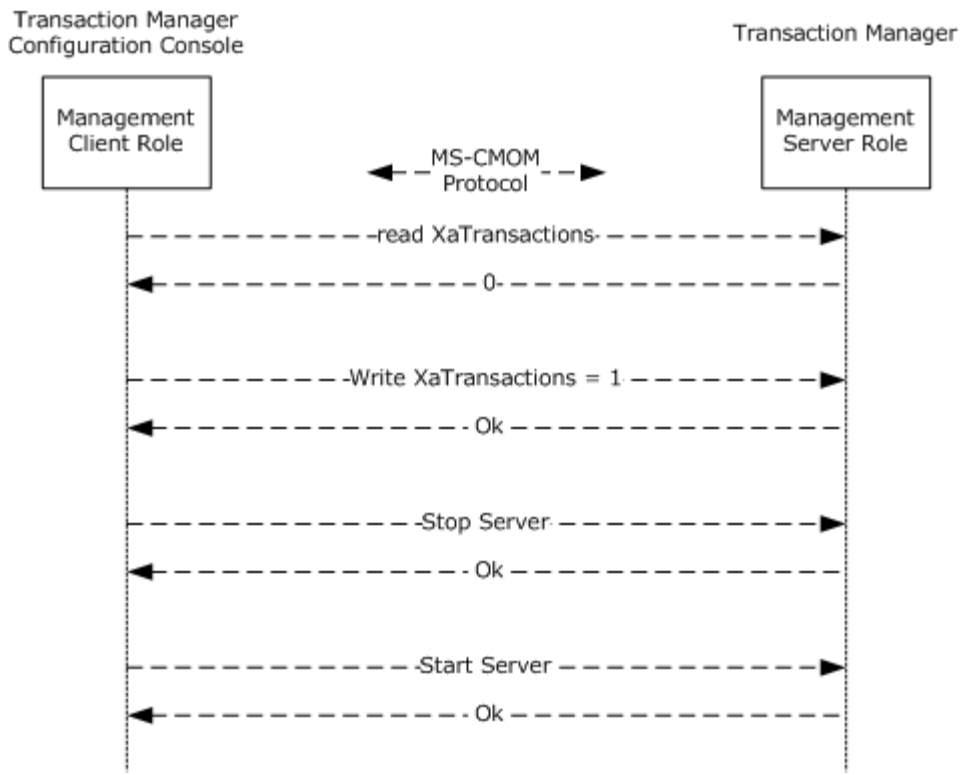


Figure 2: Setting and enabling a configuration setting

1.3.3.2 Subscribing to Transaction Monitoring Information

Figure 3 illustrates a Transaction monitor console using the Management Client protocol role to subscribe to transaction monitoring. After subscribing, transaction monitoring information is published to the Transaction monitor console until it unsubscribes.

1. The first exchange subscribes to Transaction monitoring.
2. The Management Server publishes Transaction monitoring information to all subscribers on a regular schedule. The schedule and the level of detail of Transaction monitoring information is determined by configuration settings.
3. The last exchange unsubscribes from Transaction monitoring.

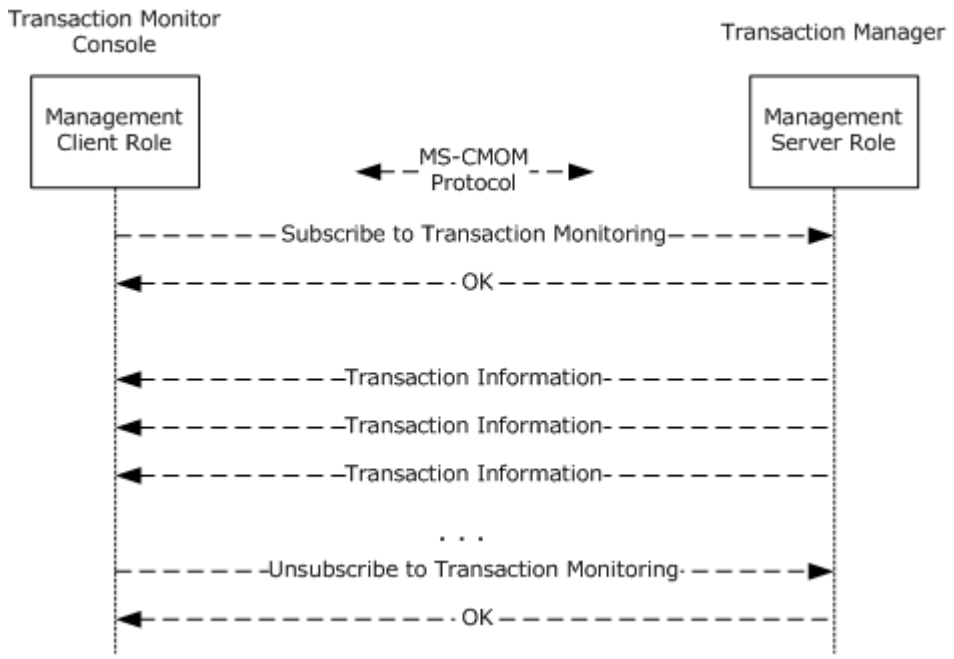


Figure 3: Subscribing to transaction monitoring information

1.3.3.3 Transaction Manager Remote Proxy

A Transaction Manager proxy allows an application that participates in transactions to communicate with a remote Transaction Manager. The proxy obtains the contact information for the protocol endpoints exposed by the Transaction Manager from a set of registry keys called the **contact id (CID)** keys.

In the case in which an application is configured to use a remote Transaction Manager, the proxy needs to obtain the contact information contained in the CID key on the machine hosting the Transaction Manager. Figure 4 illustrates a Transaction Manager proxy using the Management Client protocol role to get the remote Transaction Manager's CID and then forwarding Transaction requests to the Transaction Manager's OleTx Transaction Manager role:

1. The first exchange gets the information in the Transaction Manager's CID key. This communication uses the Windows Remote Registry Protocol ([MS-RRP]).

2. The proxy then forwards the OleTx Transaction Protocol ([MS-DTCO]) **messages** to the OleTx Transaction Manager role.

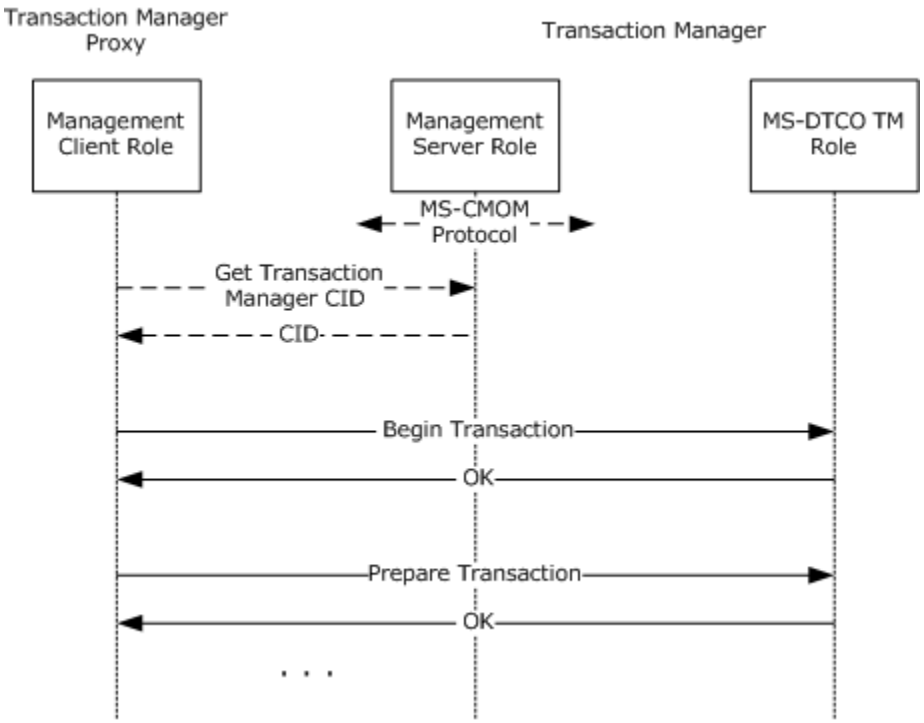


Figure 4: Transaction Manager proxy

1.4 Relationship to Other Protocols

Figure 5 illustrates which protocols the OleTx Management Protocol relies on.

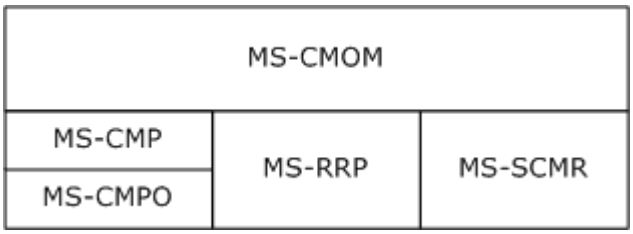


Figure 5: Protocols that the OleTx Management Protocol relies on

The OleTx Management Protocol affects the behavior of the following protocols and extensions:

- The OleTx Transaction Protocol ([MS-DTCO])
- The OleTx Transaction Internet Protocol ([MS-DTCM])
- The Transaction Internet Protocol (TIP) Extensions ([MS-TIPP])
- The OleTx XA Protocol ([MC-DTCXA])

1.5 Prerequisites/Preconditions

The operation of this protocol assumes the following:

- Implementations of both the Management Client protocol role and the Management Server protocol role possess implementations of the OleTx Multiplexing Protocol ([MS-CMP]) and the OleTx Transports Protocol ([MS-CMPO]).
- An implementation of the Management Server role operates as an extension to a OleTx Transaction Manager, and the following requirements are satisfied:
  - There is an unmediated, implementation-specific communication mechanism between the two modules.
  - The OleTx Transaction Manager implementation also possesses extension modules implementing the following roles:
    - The OleTx Transaction Internet Protocol ([MS-DTCM]) TIP Interoperability Provider.
    - The Transaction Internet Protocol Extensions ([MS-TIPP]) TIP Transaction Manager facets.
    - The OleTx XA Protocol ([MC-DTCXA]) **XA Transaction Manager** facets.
  - The OleTx Transaction Manager is implemented as a Service.
- An implementation of the Windows Remote Registry Protocol ([MS-RRP]) is present and operating, and the following conditions are satisfied:
  - The OleTx Transaction Manager uses specific keys, as specified in section 3.3.4, as either startup configuration settings or run-time settings.
  - The OleTx Management Protocol Management Server, the OleTx Transaction Protocol TIP Interoperability Provider, and the OleTx XA Protocol XA Transaction Manager use specific keys, as specified in section 3.3.4, to read their OleTx Transports Protocol contact information.

1.6 Applicability Statement

This protocol is applicable to scenarios that require the management of an OleTx Transaction Manager with its respective extensions for the OleTx Transaction Internet Protocol ([MS-DTCM]), Transaction Internet Protocol Extensions ([MS-TIPP]), and OleTx XA Protocol ([MC-DTCXA]). The prerequisites provided in Prerequisites/Preconditions, and all the prerequisites required for the operation of a OleTx Transaction Manager, a OleTx Transaction Protocol TIP Interoperability Provider, and a OleTx XA Protocol XA Transaction Manager need to be satisfied in order for this protocol to be employed successfully.<1>

1.7 Versioning and Capability Negotiation

This protocol covers versioning aspects in the following areas:

Protocol versions: This protocol provides four versions: 1, 2, 4, and 5 (version 3 is reserved and not used). More details about the protocol elements supported in each version are provided in Protocol Versioning.

Capability negotiation: Capability Negotiation Mechanisms describes the capability negotiation mechanisms for this protocol.

1.7.1 Version Negotiation Mechanisms

In order to negotiate a protocol version, this protocol uses the explicit version negotiation mechanism provided by the underlying OleTx Transports Protocol, as specified in [MS-CMPO], section 3.2.4.2.1. An implementation of this protocol uses this mechanism to specify which versions

of the protocol it supports and to negotiate a mutually agreeable version with its partners (see section 3.1.4).

1.7.2 Capability Negotiation Mechanisms

Within a protocol version, there are no optional messages or message fields, and therefore there are no capability negotiation mechanisms in that respect.

Certain protocol versions have optional support for certain registry keys (see section 2.1.3). If two **protocol participants** support an optional key, they will be able to use it as described in this protocol. Otherwise, the two protocol participants will not be able to use the respective key.

1.8 Vendor-Extensible Fields

The OleTx Management Protocol gives vendors the ability to define implementation-specific trace strings and trace sources to be used in connection with two of the protocol messages: MSG_DTCUIC_TRACE and MSG_DTCUIC_TRACESTRING.

1.9 Standards Assignments

The OleTx Management Protocol has no standards assignments.

2 Messages

This section defines how this protocol maps over lower-layer transport protocols and defines the syntax for the messages used by this protocol.

2.1 Transport

This protocol uses three forms of communication:

- OleTx Multiplexing Protocol ([MS-CMP]) messages over OleTx Transports Protocol ([MS-CMPO]) transport.
- Registry key values transmitted over the Windows Remote Registry Protocol ([MS-RRP]).
- Service control commands for the MSDTC service transmitted over the Service Control Manager Remote Protocol ([MS-SCMR]).

2.1.1 Parameters Passed to the Transport Layer

2.1.1.1 Parameters Passed to the OleTx Transports Protocol Transport

To establish a **session**, as specified in [MS-CMPO], the following values MUST be provided to the lower-layer protocol:

- A Security Level value that indicates the required RPC **authentication level**. The possible values for this element are as specified in [MS-CMPO].
- A **name object** that indicates the host name, the contact identifier, and the supported RPC network protocols of the remote endpoint against which the session is established. Name objects are specified in [MS-CMPO], section 3.1.1.4.
- The minimum and maximum values of the protocol version number, which specify the minimum and maximum protocol versions that are supported by the implementation.

2.1.1.1.1 Establishing a Security Level

Every protocol participant SHOULD use **mutual authentication** when establishing a new session. If the destination does not support mutual authentication, a protocol participant SHOULD use **incoming authentication**. If the destination does not support incoming authentication, a protocol participant MAY use no security.<2>

2.1.1.1.2 Obtaining a Name Object

The process of obtaining a Name object for a session partner is implementation-specific. <3>

2.1.1.1.3 Obtaining the Minimum and Maximum Protocol Version Numbers

The process for computing the minimum and maximum protocol version numbers used in initializing the underlying OleTx Transports Protocol transport is specified in Common Initialization Details.

2.1.1.2 Parameters Passed to the Windows Remote Registry Protocol Transport

This protocol does not pass any configuration parameters to the Windows Remote Registry Protocol ([MS-RRP]) transport.

2.1.1.3 Parameters Passed to the Service Control Manager Remote Protocol Transport

This protocol does not pass any configuration parameters to the Service Control Manager Remote Protocol ([MS-SCMR]) transport.

2.1.2 OleTx Multiplexing Protocol Messages and Connections

Each OleTx Multiplexing Protocol message MUST be sent by using an active OleTx Multiplexing Protocol connection, as specified in [MS-CMP]. The mechanisms used to initiate and accept connections (as specified in [MS-CMP]) are specified in section 3.1.2.

Each connection MUST be initiated inside an active session, as specified in [MS-CMPO]. The mechanism that is used to establish sessions is as specified in [MS-CMPO], section 1.3.3.1.

2.1.3 Protocol Versioning

This protocol has four versions: 1, 2, 4, and 5 (version 3 is reserved and not used). For each version, there is a set of protocol elements that MUST be supported (version-required elements), a set of optional elements that SHOULD be supported (version-optional elements), and a set of protocol elements that MUST NOT be supported (version-not supported elements).

2.1.3.1 Versioning of OleTx Transports Protocol Messages

The following table defines the versioning of the OleTx Transports Protocol ([MS-CMPO]) messages used by this protocol. The messages and data elements that are not shown in this table MUST be supported by all protocol versions.

Version-specific aspect	V1	V2	V4	V5
MSG_DTCUIC_STATS supports the **systemTimeTransactionsUp**, **dwTimeStamp**, and **cSinglePhaseInDoubt** fields.	Not supported	Required	Required	Required

2.1.3.2 Versioning of Windows Remote Registry Protocol Registry Keys

The following table defines the versioning of the Windows Remote Registry Protocol ([MS-RRP]) keys and values used by this protocol. The protocol keys and values that are not shown in this table MUST be supported by all protocol versions.

Version-specific aspect	V1	V2	V4	V5
Version supports "HKEY_LOCAL_MACHINE\Software\Microsoft\ MSDTC\ClientNetworkProtocol".	Not supported	Required	Required	Required
Version supports "HKEY_LOCAL_MACHINE\Software\Microsoft\ MSDTC\ServiceNetworkProtocol".	Not supported	Not supported	Required	Required
Version supports the "HKEY_LOCAL_MACHINE\Software\Microsoft\ Windows NT\CurrentVersion\Tracing*" keys and values defined in section 2.2.2.4.	Not supported	Not supported	Required	Required

Version-specific aspect	V1	V2	V4	V5
Version supports the following values of the "HKEY_LOCAL_MACHINE\Software\Microsoft\MSDTC\Security" key (see section 2.2.2.2): NetworkDtcAccess, NetworkDtcAccessAdmin, NetworkDtcAccessClients, NetworkDtcAccessTransactions, XaTransactions, AccountName.	Not supported	Not supported	Required	Required
Version supports the "HKEY_LOCAL_MACHINE\Software\Microsoft\MSDTC\Security\NetworkDtcAccessTip" value (see section 2.2.2.2).	Not supported	Required	Required	Required
Version supports the following values of the "HKEY_LOCAL_MACHINE\Software\Microsoft\MSDTC\Security" key: NetworkDtcAccessInbound, NetworkDtcAccessOutbound, AllowOnlySecureRpcCalls, FallbackToUnsecureRpcIfNecessary.	Not supported	Not supported	Not supported	Required
Version supports the "HKEY_LOCAL_MACHINE\Software\Microsoft\MSDTC\Security\TurnOffRpcSecurity" value.	Not supported	Not supported	Optional	Required
Version supports the "HKEY_LOCAL_MACHINE\Classes\OLETransactionManagers*" keys and values defined in section 2.2.2.5.	Not supported	Required	Required	Required

2.2 Message Syntax

2.2.1 OleTx Multiplexing Protocol Message Syntax

The layout of each OleTx Multiplexing Protocol message that is used by this protocol MUST extend the MESSAGE_PACKET structure, as specified in [MS-CMP], section 2.2.2.

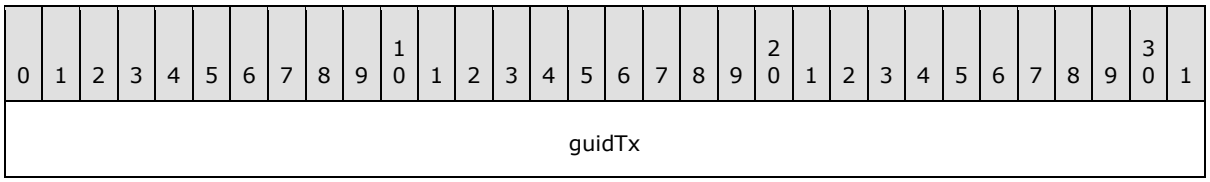
2.2.1.1 Connection Types

This protocol defines only one **connection type**: CONNTYPE_TXUSER_DTCUIC. The **Protocol Type** field for connections that implement this connection type, as specified in [MS-CMP], section 3.1.1.1, MUST be set to 0x00000000.

2.2.1.2 Data Structures

2.2.1.2.1 DtcUITransListElement

The DtcUITransListElement structure is used to represent the tracking information for a Transaction object.



...
...
...
ulIsol
szDesc
...
...
...
...
...
...
...
(szDesc cont'd for 2 rows)
dwStatus
szParent
...
...
...

guidTx (16 bytes): This field MUST contain a **GUID** that specifies the **transaction identifier** for the transaction.

ulIsol (4 bytes): This field MUST be set to an OLETX_ISOLATION_LEVEL (as specified in [MS-DTCO]) value that specifies the transaction's isolation level.

szDesc (40 bytes): The description of the transaction as a fixed-size array of 40 bytes containing a null-terminated Latin-1 ANSI string, as specified in [ISO-8859-1].

dwStatus (4 bytes): This field MUST be set to a **TRACKING_STATUS** value that specifies that status of the transaction.

szParent (16 bytes): A fixed-size array of 16 bytes containing a null-terminated Latin-1 ANSI character string, as specified in [ISO-8859-1], which MUST specify the host name of the transaction manager that is referenced in the Transaction object's **Superior Enlistment** field. If the transaction does not have a superior **enlistment**, the array MUST contain a null string.

2.2.1.3 Enumerations

2.2.1.3.1 TRACKING_STATUS

The **TRACKING_STATUS** enumeration defines a set of values indicating the status of a Transaction (managed by the OleTx Transaction Manager) as it is tracked by the Management Server through its progress. (See section 2.2.1.2.1.)

```
typedef  enum
{
  XACTSTAT_OPEN = 0x00000003,
  XACTSTAT_PREPARING = 0x00000004,
  XACTSTAT_PREPARED = 0x00000008,
  XACTSTAT_COMMITTING = 0x00000040,
  XACTSTAT_ABORTING = 0x00000100,
  XACTSTAT_ABORTED = 0x00000200,
  XACTSTAT_FORCED_ABORT = 0x00000201,
  XACTSTAT_COMMITTED = 0x00000400,
  XACTSTAT_FORCED_COMMIT = 0x00000401,
  XACTSTAT_NOTIFYING_COMMITTED = 0x00000801,
  XACTSTAT_ONLY_FAILED_COMMITTED_REMAIN = 0x00000C01,
  XACTSTAT_INDOUBT = 0x00020000,
  XACTSTAT_FORGET = 0x00080001
} TRACKING_STATUS;
```

XACTSTAT_OPEN: This status indicates the transaction is still in the active phase.

XACTSTAT_PREPARING: This status indicates the transaction is preparing.

XACTSTAT_PREPARED: This status indicates the transaction is prepared.

XACTSTAT_COMMITTING: This status indicates the transaction is committing.

XACTSTAT_ABORTING: This status indicates the transaction is aborting.

XACTSTAT_ABORTED: This status indicates the transaction is aborted.

XACTSTAT_FORCED_ABORT: This status indicates the transaction was forced to abort (by external intervention).

XACTSTAT_COMMITTED: This status indicates the transaction is committed.

XACTSTAT_FORCED_COMMIT: This status indicates the transaction was forced to commit.

XACTSTAT_NOTIFYING_COMMITTED: This status indicates the transaction is committed.

XACTSTAT_ONLY_FAILED_COMMITTED_REMAIN: This status indicates the transaction is in the aborting phase.

XACTSTAT_INDOUBT: This status indicates the transaction is in doubt.

XACTSTAT_FORGET: This status indicates the transaction has completed and the transaction manager does not manage it any more.

2.2.1.3.2 UPDATE_LIMIT

The **UPDATE_LIMIT** enumeration defines a set of values indicating how often the Management Server should transmit the transaction tracking data and statistics to the Management Client (see section 3.3.7.1).

```
typedef  enum
{
  UPDATE_20 = 0x00000000,
  UPDATE_10 = 0x00000001,
  UPDATE_5 = 0x00000002,
  UPDATE_3 = 0x00000003,
  UPDATE_1 = 0x00000004
} UPDATE_LIMIT;
```

UPDATE_20: A 20-second update period MUST be used.

UPDATE_10: A 10-second update period MUST be used.

UPDATE_5: A 5-second update period MUST be used.

UPDATE_3: A 3-second update period MUST be used.

UPDATE_1: A 1-second update period MUST be used.

2.2.1.3.3 SHOW_LIMIT

The **SHOW_LIMIT** enumeration defines a set of values indicating the minimum age required of a transaction in order to be tracked by the Management Server and reported to the Management Client (see section 2.2.1.2.1.)

```
typedef  enum
{
  SHOW_5_MIN = 0x00000000,
  SHOW_1_MIN = 0x00000001,
  SHOW_30_SEC = 0x00000002,
  SHOW_10_SEC = 0x00000003,
  SHOW_1_SEC = 0x00000004
} SHOW_LIMIT;
```

SHOW_5_MIN: A 5-minute minimum transaction age MUST be used.

SHOW_1_MIN: A 1-minute minimum transaction age MUST be used.

SHOW_30_SEC: A 30-second minimum transaction age MUST be used.

SHOW_10_SEC: A 10-second minimum transaction age MUST be used.

SHOW_1_SEC: A 1-second minimum transaction age MUST be used.

2.2.1.3.4 TRACE_LEVEL

The **TRACE_LEVEL** enumeration defines a set of values indicating the level of traces the Management Server should post to the Management Client (see section 2.2.1.4.1.2.)

```
typedef  enum
{
  TRACE_NONE = 0x00000000,
  TRACE_ERRORS = 0x00000001,
  TRACE_WARNINGS = 0x00000002,
  TRACE_INFORMATION = 0x00000003,
  TRACE_ALL = 0x00000004
} TRACE_LEVEL;
```

TRACE_NONE: This level indicates that no event MUST be traced.

TRACE_ERRORS: This level indicates that only significant problems, such as the failure of a critical task, MUST be traced.

TRACE_WARNINGS: This level indicates that all lower-level events plus warnings (events that not necessarily significant, but indicate the possible occurrence of a future problem) MUST be traced.

TRACE_INFORMATION: This level indicates that all lower-level event plus informational events (events that provide information about the normal evolution of the system) MUST be traced.

TRACE_ALL: All the error, warning, and information events MUST be traced.

2.2.1.3.5 TRACE_SEVERITY_LEVEL

The **TRACE_SEVERITY_LEVEL** enumeration defines the severity level values for a trace event (see sections 2.2.1.4.1.7 and 2.2.1.4.1.8.)

```
typedef  enum
{
  ERROR = 0x00000001,
  WARNING = 0x00000002,
  INFORMATION = 0x00000004,
  OBSCURE = 0x00000100,
  CRITICAL = 0x00001000
} TRACE_SEVERITY_LEVEL;
```

ERROR: This level MUST be used to trace errors in the operation of the system (such as the failure of some task).

WARNING: This level MUST be used to trace events that indicate the possible occurrence of a future problem.

INFORMATION: This level MUST be used to trace information about the normal progress of significant operations.

OBSCURE: This level MUST be used to trace information about the progress of less-important (obscure) operations.

CRITICAL: This level MUST be used to trace unrecoverable operation errors that preclude the operation of the whole system, or of an important subsystem.

2.2.1.4 Connection Type Details

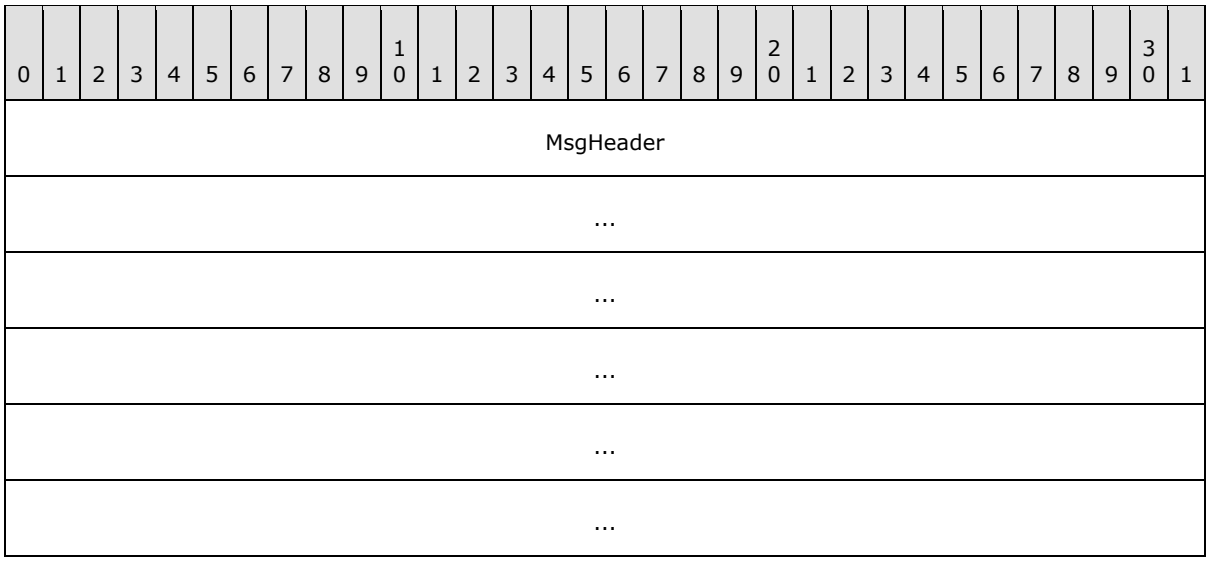
2.2.1.4.1 CONNTYPE_TXUSER_DTCUIC

This connection type is used for **transaction monitoring** and defines the following messages:

- MTAG_HELLO
- MSG_DTCUIC_TRACELIMIT
- MSG_DTCUIC_UPDATELIMIT
- MSG_DTCUIC_SHOWLIMIT
- MSG_DTCUIC_STATS
- MSG_DTCUIC_TRANSLIST
- MSG_DTCUIC_TRACE
- MSG_DTCUIC_TRACESTRING

2.2.1.4.1.1 MTAG_HELLO

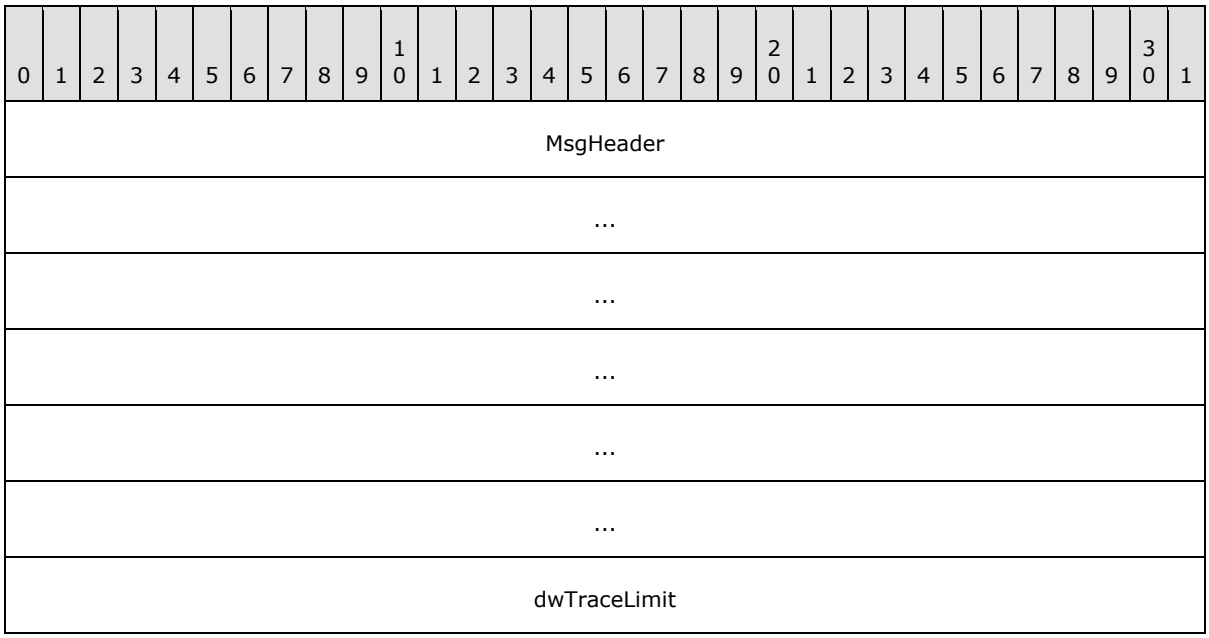
The MTAG_HELLO message is sent by the Management Client to the Management Server in order to verify that the connection with the Management Server is established (if not, the sending fails; see section 3.2.4.2).



MsgHeader (24 bytes): This field MUST contain a MESSAGE_PACKET structure (as specified in [MS-CMP]). The **dwUserMsgType** field MUST be 0x00003006. The **dwcbVarLenData** field MUST be 0.

2.2.1.4.1.2 MSG_DTCUIC_TRACELIMIT

The MSG_DTCUIC_TRACELIMIT message is used by the Management Client to transmit a new trace level to the Management Server (see section 3.2.4.5).

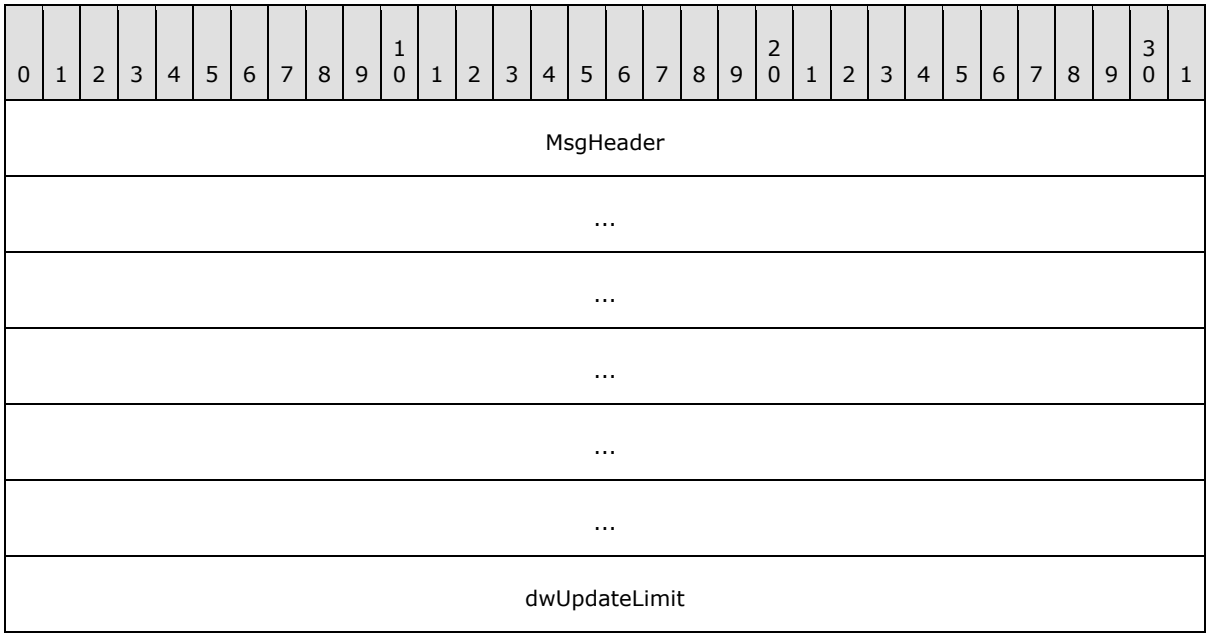


MsgHeader (24 bytes): This field MUST contain a MESSAGE_PACKET structure (as specified in [MS-CMP]). The **dwUserMsgType** field MUST be 0x00003003. The **dwcbVarLenData** field MUST be equal to 4.

dwTraceLimit (4 bytes): A **TRACE_LEVEL** value that indicates the requested trace level.

2.2.1.4.1.3 MSG_DTCUIC_UPDATELIMIT

The MSG_DTCUIC_UPDATELIMIT message is used by the Management Client to request the use of a new update interval from the Management Server (see section 3.2.4.3).



MsgHeader (24 bytes): This field MUST contain a MESSAGE_PACKET structure (as specified in [MS-CMP]). The **dwUserMsgType** field MUST be 0x00003004. The **dwcbVarLenData** field MUST be equal to 4.

dwUpdateLimit (4 bytes): An **UPDATE_LIMIT** value that indicates the requested update interval.

2.2.1.4.1.4 MSG_DTCUIC_SHOWLIMIT

The MSG_DTCUIC_SHOWLIMIT message is used by the Management Client to set a new minimum transaction age requested for tracking with the Management Server (see section 3.2.4.4).

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
MsgHeader																															
...																															
...																															
...																															
...																															
...																															
dwShowLimit																															

MsgHeader (24 bytes): This field MUST contain a MESSAGE_PACKET structure (as specified in [MS-CMP]). The **dwUserMsgType** field MUST be 0x00003005. The **dwcbVarLenData** field MUST be equal to 4.

dwShowLimit (4 bytes): A **SHOW_LIMIT** value that indicates the requested minimum transaction age.

2.2.1.4.1.5 MSG_DTCUIC_STATS

The MSG_DTCUIC_STATS message is sent by the Management Server to the Management Client to report current statistical information about transactions (see section 3.3.7.1).

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
MsgHeader																															
...																															
...																															
...																															
...																															
...																															

cOpen
cCommitted
cAborted
cInDoubt
cHeuristic
cOpenMax
cCommittedMax
cAbortedMax
cInDoubtMax
cHeuristicMax
cForcedCommit
cForcedAbort
cAvgResponseTime
cMinResponseTime
cMaxResponseTime
timeTransactionsUp
systemTimeTransactionsUp (optional)
...
...
...
dwTimeStamp (optional)

cSinglePhaseInDoubt (optional)

MsgHeader (24 bytes): This field MUST contain a MESSAGE_PACKET structure (as specified in [MS-CMP]). The **dwUserMsgType** field MUST be 0x00003001. The **dwcbVarLenData** field MUST be 60 or 84, depending on the protocol version (see section 2.1.3), plus the size in bytes of the **timeTransactionsUp** field.

cOpen (4 bytes): A 4-byte unsigned integer indicating the number of currently opened transactions.

cCommitted (4 bytes): A 4-byte unsigned integer indicating the current number of committed transactions.

cAborted (4 bytes): A 4-byte unsigned integer indicating the current number of aborted transactions.

cInDoubt (4 bytes): A 4-byte unsigned integer indicating the current number of in doubt transactions.

cHeuristic (4 bytes): A 4-byte reserved field that MUST be set to 0x00000000.

cOpenMax (4 bytes): A 4-byte unsigned integer indicating the current maximum number of open transactions.

cCommittedMax (4 bytes): A 4-byte unsigned integer indicating the current maximum number of committed transactions.

cAbortedMax (4 bytes): A 4-byte unsigned integer indicating the current maximum number of aborted transactions.

cInDoubtMax (4 bytes): A 4-byte unsigned integer indicating the current maximum number of in doubt transactions.

cHeuristicMax (4 bytes): A 4-byte reserved field that MUST be set to 0x00000000.

cForcedCommit (4 bytes): A 4-byte unsigned integer indicating the current number of transactions that were in doubt, and were forced to commit.

cForcedAbort (4 bytes): A 4-byte unsigned integer indicating the current number of transactions that were in doubt, and were forced to abort.

cAvgResponseTime (4 bytes): A 4-byte unsigned integer indicating the current average time in milliseconds for transaction committing.

cMinResponseTime (4 bytes): A 4-byte unsigned integer indicating the current minimum time in milliseconds for transaction committing.

cMaxResponseTime (4 bytes): A 4-byte unsigned integer indicating the current maximum time in milliseconds for transaction committing.

timeTransactionsUp (4 bytes): This field SHOULD contain an unsigned 4-byte integer that specifies the number of seconds that elapsed from an implementation-specific baseline time to when the service was started.<4>

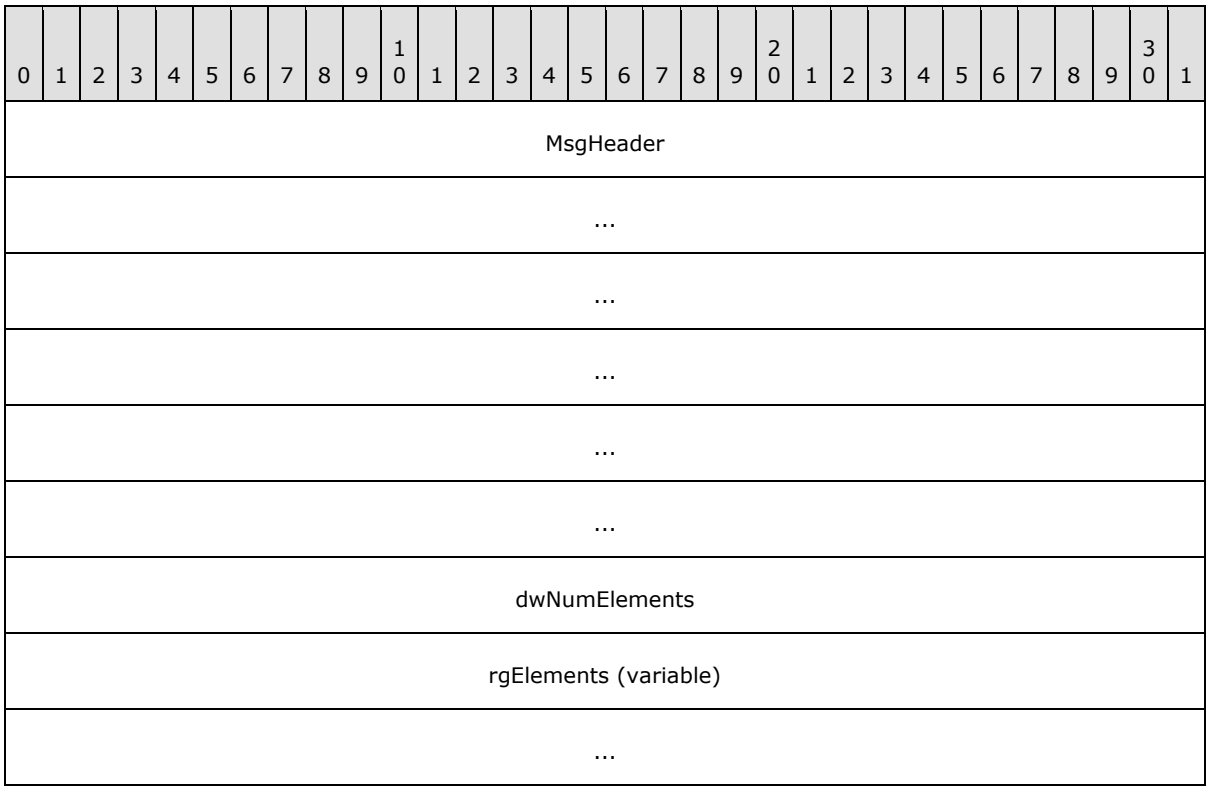
systemTimeTransactionsUp (16 bytes): A **SYSTEMTIME** structure (as specified in [MS-DTYP])indicating an implementation-specific time when the service was started.This field is either required or forbidden, depending on the protocol version (see section 2.1.3). <5>

dwTimeStamp (4 bytes): A 4-byte unsigned integer indicating an implementation-specific time when statistics were last collected. This field is either required or forbidden depending on the protocol version (see section 2.1.3). <6>

cSinglePhaseInDoubt (4 bytes): A 4-byte unsigned integer indicating the current number of transactions that have entered the single-phase in doubt state. This field is either required or forbidden, depending on the protocol version (see section 2.1.3).

2.2.1.4.1.6 MSG_DTCUIC_TRANSLIST

The MSG_DTCUIC_TRANSLIST message is sent by the Management Server to the Management Client to report current information about the transactions it is tracking (see section 3.3.7.1).



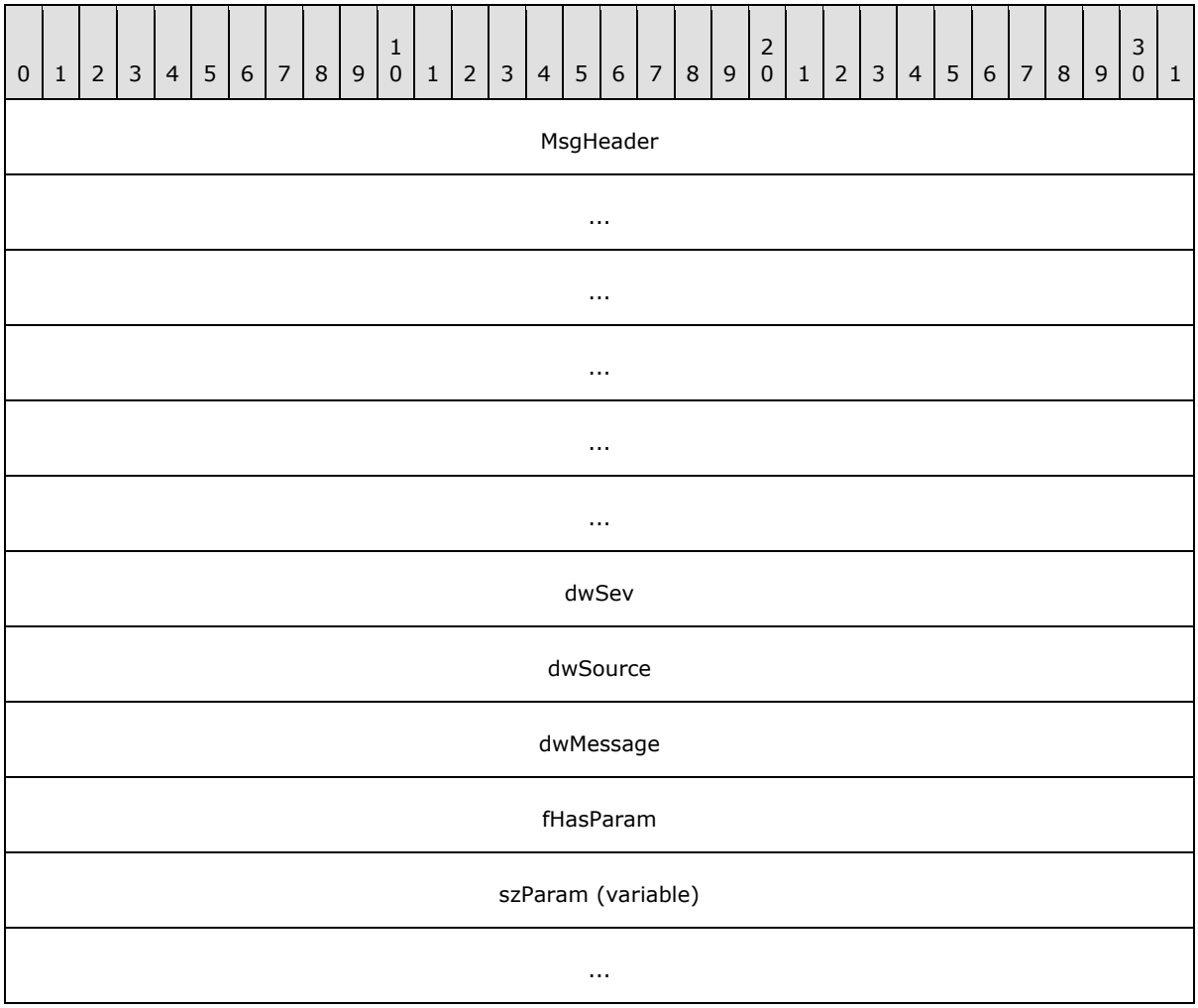
MsgHeader (24 bytes): This field MUST contain a MESSAGE_PACKET structure (as specified in [MS-CMP]). The **dwUserMsgType** field MUST be 0x00003002. The **dwcbVarLenData** field MUST be 4 plus the value of the dwNumElements field times 80.

dwNumElements (4 bytes): This field MUST contain an unsigned integer specifying the number of DtcUITransListElement structures present in the rgElements field.

rgElements (variable): This field MUST contain a list of DtcUITransListElement structures. The number of structures in this field is limited superiorly by the maximum size of an OleTx Multiplexing Protocol message, as defined in [MS-CMP].

2.2.1.4.1.7 MSG_DTCUIC_TRACE

The MSG_DTCUIC_TRACE message is sent by the Management Server to the Management Client to trace a formatted message (see section 3.3.8.2).



MsgHeader (24 bytes): This field MUST contain a MESSAGE_PACKET structure (as specified in [MS-CMP]). The **dwUserMsgType** field MUST be 0x00002FFF. The **dwcbVarLenData** field MUST be 16 plus the length of the string present in the **szParam** field (including its null terminator).

dwSev (4 bytes): This MUST be set to a **TRACE_SEVERITY_LEVEL** value.

dwSource (4 bytes): This field MUST be set to an implementation-specific unsigned integer that maps to one of the functional modules in the specific implementation (the mapping is implementation-specific and not part of this protocol). <7>

dwMessage (4 bytes): This field MUST contain an implementation-specific unsigned integer which maps to an implementation-specific trace string (the mapping is implementation-specific and not part of this protocol).<8>

fHasParam (4 bytes): This field MUST be set to 0x00000000 if the **szParam** field contains a non-null string, or 0x00000001 if the **szParam** field does not contain a non-null string.

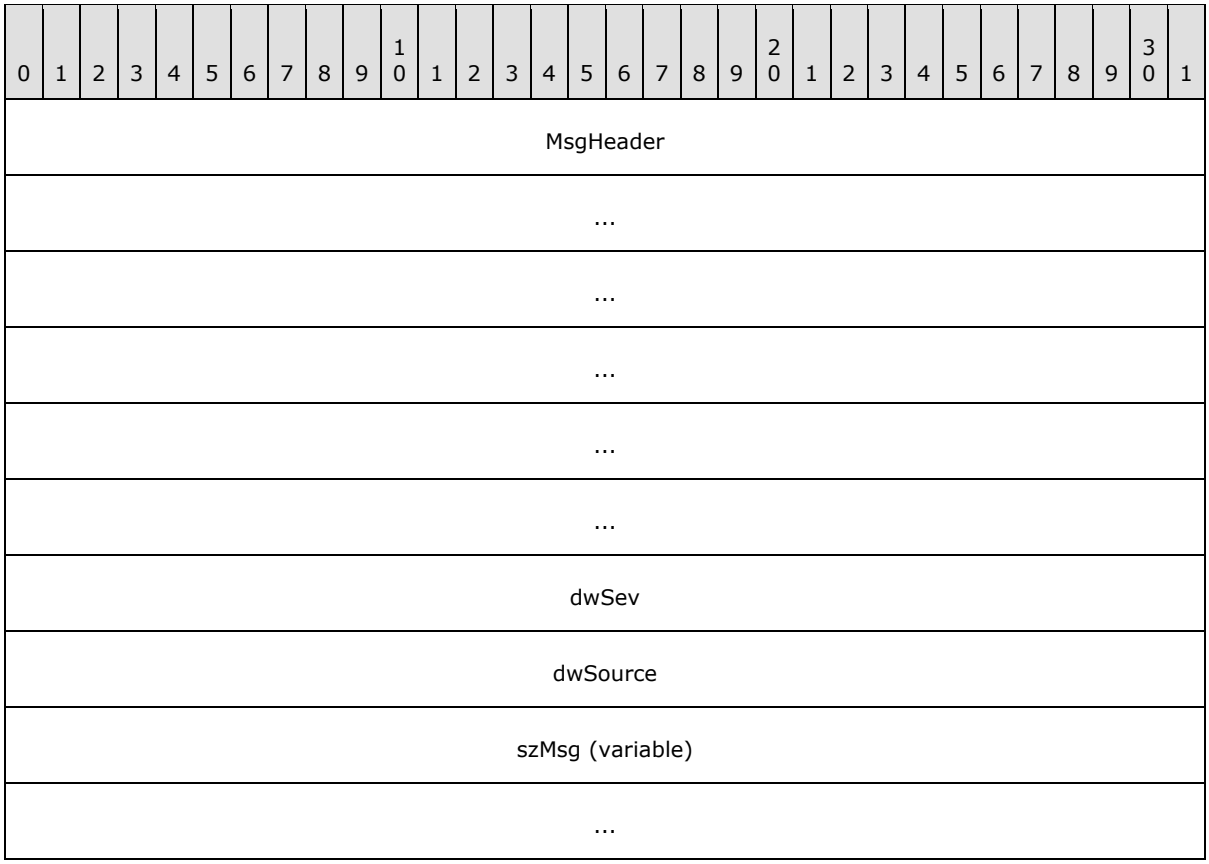
Possible Values:

Value	Meaning
0x00000000	Field contains a non-null string
0x00000001	Field does not contain a non-null string

szParam (variable): This field MUST contain a null-terminated Latin-1 ANSI string, as specified in [ISO-8859-1].

2.2.1.4.1.8 MSG_DTCUIC_TRACESTRING

The MSG_DTCUIC_TRACESTRING message is sent by the Management Server to the Management Client to trace an unformatted message (see section 3.3.8.2).



MsgHeader (24 bytes): This field MUST contain a MESSAGE_PACKET structure (as specified in [MS-CMP]). The **dwUserMsgType** field MUST be 0x00003000. The **dwcbVarLenData** field MUST be 8 plus the length of the string present in the **szMsg** field (including its null terminator).

dwSev (4 bytes): This field MUST be set to a **TRACE_SEVERITY_LEVEL** value.

dwSource (4 bytes): The field MUST be set to an implementation-specific unsigned integer that maps to one of the functional modules in the specific implementation. (See also the **dwSource** field in MSG_DTCUIC_TRACE).

szMsg (variable): This field MUST contain a null-terminated Latin-1 ANSI string, as specified in [ISO-8859-1]. The length of this string MUST be at least 1. An implementation MUST define its specific strings that are passed in this field, and represent implementation-specific traces.<9>

2.2.2 Registry Keys and Values Used with the Windows Remote Registry Protocol Transport

2.2.2.1 Enumerations

2.2.2.1.1 RPC_NETWORK_PROTOCOL

The **RPC_NETWORK_PROTOCOL** enumeration is used to specify the list of supported RPC network protocols.

```
typedef  enum
{
  TCP/IP = 0x00000001,
  SPX = 0x00000002,
  NetBEUI = 0x00000004
} RPC_NETWORK_PROTOCOL;
```

TCP/IP: The TCP/IP network protocol.

SPX: The SPX network protocol.<10>

NetBEUI: The NetBEUI network protocol.

2.2.2.2 Transaction Manager Security Registry Keys and Values

2.2.2.2.1 Values for the "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDTC\Security" key

Name	Description	Value type	Legal values
"NetworkDtcAccess"	This setting specifies if the network access to the Transaction Manager is enabled/disabled.	REG_DWORD	Any
"NetworkDtcAccessTransactions"	This setting specifies if the Transaction Manager is enabled for **transaction propagation** operations.	REG_DWORD	Any
"NetworkDtcAccessInbound"	This setting specifies if transactions can flow inbound to the Transaction Manager.	REG_DWORD	Any
"NetworkDtcAccessOutbound"	This setting determines if transactions can flow outbound from the Transaction Manager.	REG_DWORD	Any

Name	Description	Value type	Legal values
"NetworkDtcAccessAdmin"	This setting specifies if the Transaction Manager is enabled for remote administration from a network machine.	REG_DWORD	Any
"NetworkDtcAccessClients"	This setting specifies if the Transaction Manager accepts remote **clients**.	REG_DWORD	Any
"NetworkDtcAccessTip"	This setting specifies if the Transaction Manager is enabled for TIP transactions.	REG_DWORD	Any
"XaTransactions"	This setting specifies if the Transaction Manager is enabled for XA transactions.	REG_DWORD	Any
"AccountName"	This setting specifies the account name for the Transaction Manager's process.	REG_SZ	MUST be a string of 256 characters or less.

2.2.2.2.2 Values for the "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDTC" key

Name	Description	Value type	Legal values
"ClientNetworkProtocol"	This setting specifies the list of RPC network protocols used to initialize the client and server endpoints for clients, as specified in [MS-CMPO], section 1.3.2.	REG_DWORD	Any bitwise-OR combination of zero or more RPC_NETWORK_PROTOCOL values.
"ServiceNetworkProtocol"	This setting specifies the list of RPC network protocols used to initialize the server endpoints for the Transaction Manager, as specified in [MS-CMPO], section 1.3.2.	REG_DWORD	Any bitwise-OR combination of zero or more RPC_NETWORK_PROTOCOL values.
FallbackToUnsecureRPCIfNecessary	This setting specifies whether or not fallback to unsecured RPC is	REG_DWORD	Any

Name	Description	Value type	Legal values
	allowed.		
TurnOffRpcSecurity	This setting specifies if RPC security is turned off.	REG_DWORD	Any
AllowOnlySecureRpcCalls	This setting specifies if RPC security is strictly enforced.	REG_DWORD	Any

2.2.2.3 Transaction Manager Contact ID (CID) Registry Keys and Values

2.2.2.3.1 Values for the "HKEY_LOCAL_MACHINE\Software\Classes\CID\<GUID>" (Default) Subkeys

The variable <GUID> part in the key name is the string representation of a GUID value unique for the specific deployment instance (e.g. "HKEY_LOCAL_MACHINE\Software\Classes\CID\181ab538-d419-49c9-a225-f6bfadb85922"). These keys are called 'Contact Identifier' or CID keys because they specify contact information about endpoints exposed by a Transaction Manager. Specifically, the <GUID> part of a contact id key specifies the RPC unique identifier for the object that exposes the respective RPC endpoint. On a given system, there are several such keys, as defined in section 3.3.4.2. The following subkeys MUST be present under a "HKEY_LOCAL_MACHINE\Software\Classes\CID\<GUID>" key:

- Clsid
- CustomProperties
- Description
- Endpoint
- Host
- Protocol
- Svcid

The default values for these subkeys are specified in the following table:

Default value	Description	Type	Legal values
Clsid\(Default)	This setting specifies an implementation-specific GUID value.	REG_SZ	If Description\(Default) = "MSDTC" or "MSDTC Default", this value MUST be the string representation of the GUID (4364f170-81a9-11ce-9c32-00aa0051e517). If Description\(Default) = "MSDTCUIS", this value MUST be the string representation of the GUID (6e7120c0-ac8f-11ce-ad01-00aa0051e4a1).

Default value	Description	Type	Legal values
			For other values of Description\(Default), this value MUST be an empty string.
CustomProperties\(Default)	Reserved value.	REG_SZ	MUST be an empty string.
Description\(Default)	This setting specifies the description of the contact id.	REG_SZ	This value MUST be one of the following: "MSDTCUIS", "MSDTCXATM", "MSDTC", "MSDTCTIPGW", "MSDTCXATM Default", "MSDTC Default", "MSDTCTIPGW Default".
Endpoint\(Default)	Reserved value.	REG_SZ	MUST be an empty string.
Host\(Default)	This setting specifies the machine on which the transaction manager contact is present.	REG_SZ	MUST be a string of 16 characters or less. The empty string is a valid value.
Protocol\(Default)	This setting specifies the RPC network protocol to be used to connect to the respective endpoint.	REG_SZ	MUST be an empty string. <11>
Svcid\(Default)	This setting specifies the nature of the service provided through the respective endpoint.	REG_SZ	If Description\(Default) = "MSDTC" or "MSDTC Default", this value MUST be the string representation of the GUID (488091f0-bff6-11ce-9de8-00aa00a3f464). If Description\(Default) = "MSDTCUIS", this value MUST be the string representation of the GUID (ced2de40-bff6-11ce-9de8-00aa00a3f464). If Description\(Default) = " MSDTCXATM" or " MSDTCXATM Default", this value MUST be the string representation of the GUID (6407e780-7e5d-11d0-8ce6-00c04fdc877e). If Description\(Default) = " MSDTCTIPGW" or " MSDTCTIPGW Default", this value MUST be the string representation of the GUID (01366d42-c04e-11d1-b1c0-00c04fc2f3ef).

2.2.2.3.2 Custom Properties for the "HKEY_LOCAL_MACHINE\Software\Classes\CID\<MSDTC_GUID>" Key

The variable <MSDTC_GUID> part in the key name is the string representation of the GUID value for the contacts with the description "MSDTC" or "MSDTC Default", in a specific deployment instance. Either one of these contacts MUST contain the following subkeys:

- Log\Path
- Log\Size
- Service\Path

The default values for these subkeys are specified in the following table:

Name	Description	Value type	Legal values
Log\Path\(Default)	This setting specifies the path of the log file used by the Transaction Manager.	REG_SZ	MUST be a string of 256 characters or less.
Log\Size\(Default)	This setting specifies the size in megabytes (MB) of the log file used by the Transaction Manager.	REG_SZ	MUST be a string representation of an unsigned, 16-bit integer.
Service\Path\(Default)	This setting specifies the path of the executable file that contains the implementation of the Transaction Manager code.	REG_SZ	MUST be a string of 256 characters or less.

2.2.2.3.3 Custom Properties for the "HKEY_LOCAL_MACHINE\Software\Classes\CID\<MSDTCUIS_GUID>" Key

The variable <MSDTCUIS_GUID> part in the key name is the string representation of the GUID value for the contact with the description "MSDTCUIS", in a specific deployment instance. This contact MUST contain the following subkeys:

- DAC\ShowLimit
- DAC\UpdateLimit
- DAC\TraceLimit

The default values for these subkeys are specified in the following table:

Name	Description	Value type	Legal values
DAC\ShowLimit\(Default)	This setting specifies the minimum transaction age that is tracked by the Management Server (see also section 3.3.4.2.3).	REG_SZ	MUST be a string representing the **SHOW_LIMIT** value.
DAC\UpdateLimit\(Default)	This setting specifies the update limit that is used by the Management Server when reporting	REG_SZ	MUST be a string representing the **UPDATE_LIMIT** value.

Name	Description	Value type	Legal values
	information to clients (see also section 3.3.4.2.3).		
DAC\TraceLimit\(Default)	This setting specifies the trace limit used by the Management Server when tracing info to clients (see also section 3.3.4.2.3).	REG_SZ	MUST be a string representing the **TRACE_LEVEL** value.

2.2.2.4 Transaction Manager Internal Tracing Registry Keys and Values

2.2.2.4.1 Values for the "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Tracing\MSDTC\Modules" Keys

Name	Description	Value type	Legal values
"Active"	This setting specifies if tracing is enabled by default for all the Transaction Manager modules (which are implementation-specific).	REG_DWORD	Any

2.2.2.4.2 Values for the "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Tracing\MSDTC\Modules\Transaction_Transitions" Keys

Name	Description	Value type	Legal values
"Active"	This setting specifies if tracing is enabled for the Transaction_Transition module of the Transaction Manager. It overrides the general 'Active' setting under "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Tracing\MSDTC\Modules".	REG_DWORD	Any
"ControlFlags"	This setting specifies the implementation-specific categories of events that are traced out.	REG_DWORD	Any

2.2.2.4.3 Values for the "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Tracing\MSDTC\LoggingOptions" Keys

Name	Description	Value type	Legal values
"MaxBuffers"	This setting specifies an implementation-specific tracing parameter.	REG_DWORD	MUST be an unsigned integer between 1 and 999.
"RequestSessionUp"	This setting specifies whether or not the session tracing for the OleTx Transaction Manager is up.	REG_DWORD	Any

2.2.2.5 Proxy DLL Registry Keys and Values

2.2.2.5.1 Values for the "HKEY_LOCAL_MACHINE\Software\Classes\OLETransactionManagers" Key

Name	Description	Value type	Legal values
"DefaultTM"	This setting specifies the name of the default OleTx Transaction Manager installed on a machine.	REG_SZ	MUST be a string of 256 characters or less.

2.2.2.5.2 Values for the "HKEY_LOCAL_MACHINE\Software\Classes\OLETransactionManagers\<TM_NAME>" Keys

<TM_NAME> is a generic string that MUST be the name of one of the OleTx Transaction Managers installed on the machine. This string is dependent on the specific implementation and deployment scenario.

Name	Description	Value type	Legal values
"DLL"	This setting specifies the name of the dynamic link library (DLL) that an OleTx Transaction Protocol ([MS-DTCO]) application MUST load in order to work with the <TM_NAME> OleTx Transaction Manager.	REG_SZ	MUST be a string of 256 characters or less.

2.2.3 Service Names Used with the Service Control Manager Remote Protocol Transport

The Service Name (as specified in [MS-SCMR]) used for service control operations is "MSDTC".

3 Protocol Details

This section defines the behavior of the Protocol Roles introduced in the Protocol Overview (Synopsis) section: the Management Client role and the Management Server role.

3.1 Common Details

This section contains protocol details that are common to all protocol roles.

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 that described in this document.

Note It is possible to implement the conceptual data defined in this section using a variety of techniques. An implementation is at liberty to implement such data in any way it pleases.

3.1.1.1 Protocol Connection Objects

The Connection objects used in this specification extend the definition of a Connection object as defined in [MS-CMP], section 3.1.1, to include the following data elements:

State: A State enumeration value that represents the current state of the Connection while participating in the interactions associated with a certain Connection Type.

The following rules apply with respect to Connection states:

- When a Protocol Participant initiates or accepts a Connection, the Connection's State field MUST be set to the Idle state. When the Connection is disconnected, the Connection's state MUST be set to the Ended state.
- If a Connection enters the Ended state and the Connection is not disconnected, it MUST be disconnected. See [MS-CMP], section 3.1.4, for more information about Connection disconnection.

3.1.2 OleTx Multiplexing Protocol Communication Details

A Protocol Role that uses the OleTx Multiplexing Protocol transport layer (see section 2.1) to send or receive Protocol Messages MUST satisfy the following requirements:

- It MUST use OleTx Multiplexing Protocol Connections as a transport protocol for sending Messages. Sections 2.1 and 3.1.4 define the mechanisms by which this protocol initializes and makes use of an OleTx Multiplexing Protocol implementation.
- It MUST maintain all of the data elements required in the Abstract Data Model section of the[MS-CMP] specification.
- It MUST support both initiating and accepting multiple concurrent Connections both inside the same OleTx Transports Protocol Session and inside different OleTx Transports Protocol Sessions (as specified in [MS-CMPO]). Consequently, a role MUST support the existence of multiple Connection instances implementing the same Connection Type. A role MUST also support initiating multiple concurrent Sessions to a number of different Endpoints.

3.1.3 Protocol Versioning Negotiation

Protocol Versioning Negotiation is as specified in [MS-DTCO], section 3.1.6.

3.1.4 Common Initialization Details

Related to protocol versioning, when a protocol role is initialized, it MUST do the following:

- Set the **Minimum Level 3 Version Number** data field of the underlying OleTx Transports Protocol transport to 0x00000001 (as specified in [MS-CMPO], section 3.2.1.1). Note that the OleTx Management Protocol is layered on top of the OleTx Multiplexing Protocol, which is layered on top of the OleTx Transports Protocol; hence, the OleTx Management Protocol is a level-three protocol for OleTx Transports Protocol, as defined in [MS-CMPO]).
- Set the Maximum Level 3 Version Number data field of the underlying OleTx Transports Protocol ([MS-CMPO]) transport to the value of the maximum supported the OleTx Management Protocol version (as specified in section 3.1.3).

When an OleTx Transports Protocol session is successfully established between two protocol participants (as specified in [MS-CMPO]), the value of the **dwLevelThreeAccepted** field of the Session object's **Version** field (see [MS-CMPO], section 3.2.1.2) indicates the negotiated protocol version (for example, if the value of the **dwLevelThreeAccepted** field is 5, the negotiated protocol version is 5).

3.1.5 Common Message Processing Events and Sequencing Rules

The Common Message Processing Events and Sequencing Rules are as specified in [MS-DTCO], section 3.1.6.

3.1.6 Common Local Events

A Protocol Participant that uses a Connection object MUST be prepared to handle the following event at any time during the lifetime of that Connection.

3.1.6.1 Connection Disconnected

When a Connection is disconnected, a Protocol Participant MUST:

- Perform all of the actions required for a valid disconnection, as specified in [MS-CMP], section 3.1.4.3. If the Connection state is not already Ended, the state MUST be set to Ended.

3.2 Management Client Role 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.

Note that it is possible to implement the conceptual data defined in this section using a variety of techniques. An implementation is at liberty to implement such data in any way it pleases.

The Management Client role MUST extend the common Abstract Data Model specified in section 3.1.1 to include the following data elements:

- **Management Client Name**: A Name Object used to identify the Management Client with the underlying OleTx Transports Protocol ([MS-CMPO]) transport infrastructure.
- **Management Server Name**: A Name Object identifying the Management Server that the Management Client MUST use.
- **Management Connection**: A Connection that is established with the Management Server.

3.2.1.1 CONNTYPE_TXUSER_DTCUIC Initiator States

The Management Client MUST act as an initiator for the CONNTYPE_TXUSER_DTCUIC Connection Type. In this role, the Management Client MUST provide support for the following states:

- Idle
- Active
- Ended

The following figure depicts the relationship between the CONNTYPE_TXUSER_DTCUIC Initiator states:

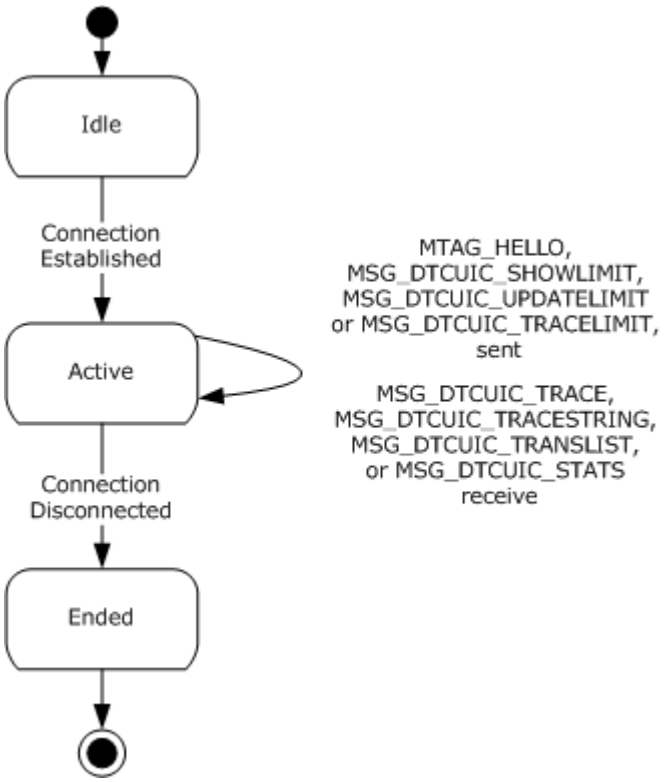


Figure 6: Relationship between the CONNTYPE_TXUSER_ DTCUI Initiator states

3.2.1.1.1 Idle

This is the initial state. The following events are processed in this state:

- Connecting to the Management Server.

3.2.1.1.2 Active

The following events are processed in this state:

- Initiating the connection with the Management Server.
- Testing the connection by sending an MTAG_HELLO message.
- Setting the Update Limit on the Management Server by sending the MSG_DTCUIC_UPDATELIMIT message.
- Setting the Show Limit on the Management Server by sending the MSG_DTCUIC_SHOWLIMIT message.
- Setting the Trace Limit on the Management Server by sending the MSG_DTCUIC_TRACELIMIT message.
- Receiving Transaction statistics in MSG_DTCUIC_TRACESTRING, MSG_DTCUIC_TRACE, MSG_DTCUIC_STATS or MSG_DTCUIC_TRANSLIST messages.
- Disconnecting from the Management Server.

3.2.1.1.3 Ended

This is the final state.

3.2.2 Timers

The Management Client role does not use any timers particular to this specification.

3.2.3 Initialization

When a Management Client is initialized:

- The Management Client Name field MUST be set to a value obtained from an implementation-specific source. This field MUST be used when initializing the underlying OleTx Transports Protocol implementation as the Local Name Object (as specified in [MS-CMPO], section 3.2.3).
- The Management Server Name field MUST be obtained from the "MSDTCUIS" CID key on the machine on which the Management Server is located (see also section 3.3.4.2).
- The Management Connection field MUST be assigned a null value.

3.2.4 Higher-Layer Triggered Events

The Management Client MUST be prepared to process a set of events triggered by the **Higher-Layer Business Logic**, as specified in the following sections. The details of the Higher-Layer Business Logic are implementation-specific, and therefore they are outside of the scope of this specification.

3.2.4.1 Connecting to the Management Server

If the higher-layer business logic decides to connect to the Management Server, the Management Client MUST perform the following actions:

- Establish a new CONNTYPE_TXUSER_DTCUIC Connection using the application's Management Server Name field as the partner's Name Object.
- Assign the new Connection object to the Management Connection field.
- Set the state of the Connection to Active.

3.2.4.2 Testing the Connection

If the higher-layer business logic decides to send an MTAG_HELLO message to test the connection with the Management Server, the Management Client MUST perform the following actions:

- If the Management Connection field is not set to a CONNTYPE_TXUSER_DTCUIC Connection in the Active state:
  - Signal the higher layer that no active Connection with the Management Server exists.
- Otherwise:
  - Send a MTAG_HELLO message using the Connection.
  - If the send fails (as specified in [MS-CMP]):
    - Reset the Management Connection field.
    - Signal the higher layer that no active Connection with the Management Server exists.
    - Set the state of the Connection to Ended.

3.2.4.3 Setting the Update Limit on the Management Server

If the higher-layer business logic decides to set the update limit on the Management Server, the Management Client MUST perform the following actions:

- Send a MSG_DTCUIC_UPDATELIMIT message using the Connection specified by the Management Connection field:
  - The **dwUpdateLimit** field MUST be set to the provided value.

3.2.4.4 Setting the Show Limit on the Management Server

If the higher-layer business logic decides to set the show limit on the Management Server, the Management Client MUST perform the following actions:

- Send a MSG_DTCUIC_SHOWLIMIT message using the Connection specified by the Management Connection field:
  - The **dwShowLimit** field MUST be set to the provided value.

3.2.4.5 Setting the Trace Limit on the Management Server

If the higher-layer business logic decides to set the trace limit on the Management Server, the Management Client MUST perform the following actions:

- Send a MSG_DTCUIC_TRACELIMIT message using the Connection specified by the Management Connection field:
  - The **dwTraceLimit** field MUST be set to the provided value.

3.2.4.6 Disconnecting from the Management Server

If the higher-layer business logic decides to disconnect from the Management Server, the Management Client MUST perform the following actions:

- Disconnect the CONNTYPE_TXUSER_DTCUIC Connection as specified in [MS-CMP], section 3.1.4.3.
- Set the Management Connection field to null.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 CONNTYPE_TXUSER_DTCUIC as Initiator

For all Messages received in this Connection type, the Management Client MUST process the Message as specified in section 3.1.5. The application MUST additionally follow the processing rules specified in the following sections.

3.2.5.1.1 Receiving a MSG_DTCUIC_TRACESTRING, MSG_DTCUIC_TRACE, MSG_DTCUIC_STATS, or MSG_DTCUIC_TRANLIST Message

When the Management Client receives a MSG_DTCUIC_TRACESTRING, MSG_DTCUIC_TRACE, MSG_DTCUIC_STATS or MSG_DTCUIC_TRANLIST message, it MUST perform the following actions:

- Return the data contents of the message received to the higher-layer business logic using an implementation-specific format.

3.2.5.1.2 Connection Disconnected

When a CONNTYPE_TXUSER_DTCUIC Connection is disconnected, the Management Client MUST set the Management Connection field to a null value.

3.2.6 Timer Events

This role has no protocol-specific timer events.

3.2.7 Other Local Events

None.

3.3 Management Server Role Details

3.3.1 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.

Note It possible to implement the conceptual data defined in this section using a variety of techniques. An implementation is at liberty to implement such data in any way it pleases.

The Management Server MUST maintain all of the data elements specified in section 3.1.1. The Management Server MUST also maintain the following data elements:

- **Management Server Name**: A Name object identifying the Management Server with the underlying OleTx Transports Protocol ([MS-CMPO]) transport infrastructure.
- **Show Limit**: An unsigned integer value specifying the minimum age a Transaction MUST have to be reported to the Management Client. This is common to all connections in the Management Connection List.
- **Update Limit**: An unsigned integer value specifying the minimum age a Transaction MUST have to be reported to the Management Client. This is common to all connections in the Management Connection List.
- **Trace Limit**: A field that MUST have one of the values in TRACE_LEVEL, and specifies what level a trace event MUST have to be reported to the Management Client. This is common to all connections in the Management Connection List.
- **Management Connection List**: A list of CONNTYPE_TXUSER_DTCUIC connections.
- **Tracked Transaction List**: A list of Transaction Objects.

3.3.1.1 Connection States

The Management Server MUST provide the following states for its supported Connection Types:

3.3.1.1.1 CONNTYPE_TXUSER_DTCUIC Acceptor States

The Management Server MUST act as an acceptor for the CONNTYPE_TXUSER_DTCUIC Connection Type. In this role, the Management Server MUST provide support for the following states:

- Idle
- Active
- Ended

The following figure depicts the relationship between the CONNTYPE_TXUSER_DTCUIC acceptor states:

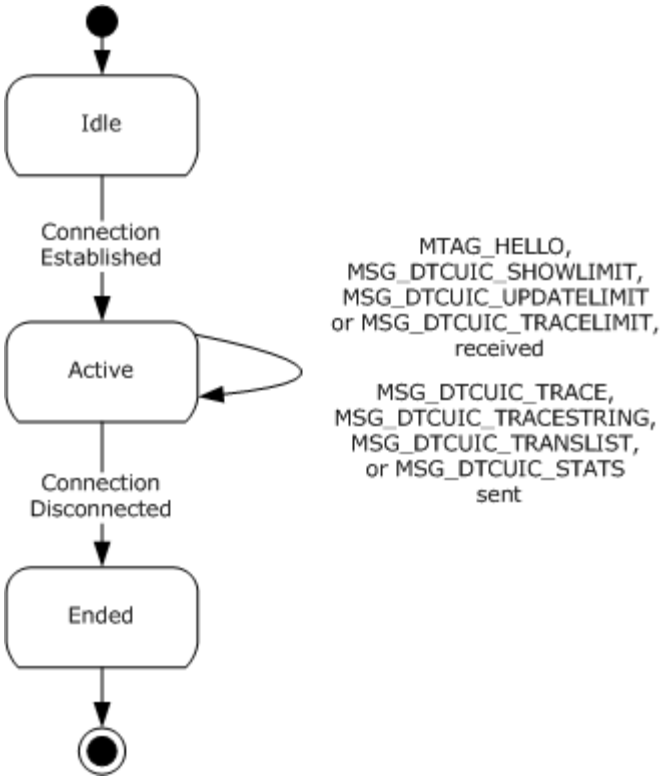


Figure 7: Relationship between the CONNTYPE_TXUSER_ DTCUI acceptor states

3.3.1.1.1.1 Idle

This is the initial state. The following events are processed in this state:

- Incoming Connection Request event.

3.3.1.1.1.2 Active

The following events are processes in this state:

- Receiving a MSG_DTCUIC_TRACELIMIT Message
- Receiving a MSG_DTCUIC_UPDATELIMIT Message
- Receiving a MSG_DTCUIC_SHOWLIMIT Message
- Update Timer elapse
- Trace Event
- Trace String Event

3.3.1.1.1.3 Ended

This is the final state.

3.3.2 Timers

The Management Server MUST provide the following timers:

3.3.2.1 Update Timer

This timer MUST be set when a CONNTYPE_TXUSER_DTCUIC connection request is accepted, and the Management Connection List maintained by the Management Server is empty. The timer MUST be cancelled when the last connection in the Management Connection List has been closed and removed from the list.

The value of the timer MUST be expressed in milliseconds. The default value of the timer MUST be specific to the implementation. <12>

The minimum and maximum values of the timer are implementation-specific and MUST be greater than zero.

3.3.3 Initialization

When the Management Server role is initialized, it MUST perform the following actions:

- The value of Management Server Name field MUST be obtained from the "MSDTCUIS" CID key (see section 3.3.4.2). This field MUST be used when initializing the underlying OleTx Transports Protocol implementation (as specified in [MS-CMPO], section 3.2.3).
- The Show Limit field MUST be set to the value of the "DAC\ShowLimit" custom property of the "MSDTCUIS" CID key (see section 3.3.4.2.3).
- The Update Limit field MUST be set to the value of the "DAC\UpdateLimit" custom property of the "MSDTCUIS" CID key.
- The Trace Limit field MUST be set to the value of the "DAC\TraceLimit" custom property of the "MSDTCUIS" CID key.
- The Management Connection List MUST be initialized with an empty list of Connection objects.
- The Tracked Transaction List MUST be initialized with an empty list of Transaction Objects.

3.3.4 Registry Keys

The set of registry keys and values specified in section 2.2.2 is used for communicating configuration values between the Management Client and the Management Server. These keys MUST be exposed by the Management Server via the Windows Remote Registry Protocol. The store used by the Windows Remote Registry Protocol implementation MUST be durable so that the keys survive the restart of the Management Server. The Management Server MUST map (through an implementation-specific data transfer mechanisms) between the set of registry keys and Abstract Data Model elements pertaining to the OleTx Transaction Manager, the **OleTx** XA Transaction Manager, and the OleTx Transaction Protocol TIP Interoperability Provider with which it interacts.

Section 3.3.4.1 specifies the mapping between registry keys and Abstract Data Model elements, as well as time constraints on when registry key changes take effect. These mappings and time constraints MUST be supported by an implementation-specific mechanism.

3.3.4.1 Security Key Mappings

The following security key mappings MUST be implemented by the Management Server:

Key value	Abstract data model element	Takes effect
"NetworkDtcAccess"	Allow Network Access flag defined in [MS-DTCO], section 3.2.1. If the key value is 0x00000000, is missing, or is invalid, then the flag MUST be set to FALSE, otherwise the flag MUST be set to TRUE.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
"NetworkDtcAccessTransactions"	Allow Network Transactions flag defined in [MS-DTCO], section 3.2.1. If the key value is 0x00000000, is missing, or is invalid, then the flag MUST be set to FALSE, otherwise the flag MUST be set to TRUE.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
"NetworkDtcAccessInbound"	Allow Inbound Transactions flag defined in [MS-DTCO], section 3.2.1. If the key value is 0x00000000, is missing, or is invalid, then the flag MUST be set to FALSE, otherwise the flag MUST be set to TRUE.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
"NetworkDtcAccessOutbound"	Allow Outbound Transactions flag defined in [MS-DTCO], section 3.2.1. If the key value is 0x00000000, is missing, or is invalid, then the flag MUST be set to FALSE, otherwise the flag MUST be set to TRUE.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
"NetworkDtcAccessAdmin"	Allow Remote Administration flag defined in [MS-DTCO], section 3.2.1. If the key value is 0x00000000, is missing, or is invalid, then the flag MUST be set to FALSE, otherwise the flag MUST be set to TRUE.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
"NetworkDtcAccessClients"	Allow Remote Clients flag defined in [MS-DTCO], section 3.2.1. If the key value is 0x00000000, is missing, or is invalid, then the flag MUST be set to FALSE, otherwise the flag MUST be set to TRUE.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
"NetworkDtcAccessTip"	Allow Tip flag defined in [MS-DTCO], section 3.2.1. If the key value is 0x00000000, is missing, or is invalid, then the flag MUST be set to FALSE, otherwise the flag MUST be set to TRUE.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
"XaTransactions"	Allow XA flag defined in [MS-DTCO],	The mapping MUST take

Key value	Abstract data model element	Takes effect
	section 3.2.1. If the key value is 0x00000000, is missing, or is invalid, then the flag MUST be set to FALSE, otherwise the flag MUST be set to TRUE.	effect when the OleTx Transaction Manager starts up.
"AccountName"	The account name of the "MSDTC" service which is used to register the OleTx Transaction Protocol ([MS-DTCO]) implementation with the Service Control Manager Remote Protocol ([MS-SCMR]).	The value is for informational purposes only.
"AllowOnlySecureRpcCalls", "FallbackToUnsecureRPCIfNecessary", and "TurnOffRpcSecurity"	The AllowOnlySecureRpcCalls, FallbackToUnsecureRPCIfNecessary, and TurnOffRpcSecurity flags are used to set the [MS-DTCO] Security Level. If a key value is 0x00000000, missing, or invalid, then the flag MUST be treated as if it were set to FALSE; otherwise the flag MUST be treated as if it were set to TRUE. If the AllowOnlySecureRpcCalls flag is TRUE, then the Security Level MUST be set to Mutual Authentication. Else, if the FallbackToUnsecureRPCIfNecessary flag is TRUE, then the Security Level MUST be set to Incoming Authentication. Else if the TurnOffRpcSecurity flag is TRUE, then the Security Level MUST be set to No Security. Else the Security Level MUST be set to the default value. <13>	The mapping MUST take effect when the OleTx Transaction Manager starts up.
"ClientNetworkProtocol"	The list of protocols specified by ClientNetworkProtocol MUST be mapped to the Protocol list value that is passed to the underlying OleTx Transports Protocol layer ([MS-CMPO], section 1.3.2). If the key value is 0x00000000 or is missing, then the list of protocols MUST contain a single entry specifying TCP/IP.	The mapping MUST take effect during partner initialization ([MS-CMPO], section 1.3.2).
"ServiceNetworkProtocol"	The list of protocols specified by ServiceNetworkProtocol MUST be mapped to the list of supported RPC protocols passed to the underlying OleTx Transports Protocol layer ([MS-CMPO], section 1.3.2. If the key value is 0x00000000 or is missing, then the list of protocols MUST contain a single entry specifying TCP/IP.	The mapping MUST take effect during partner initialization ([MS-CMPO], section 1.3.2).

3.3.4.2 Contact ID Key Mappings

A Management Server implementation MUST expose via the Windows Remote Registry Protocol a minimum of four CID keys (see section 2.2.2.3) as follows:

- A CID key with the description "MSDTCUIS".
- A CID key with the description "MSDTC".
- A CID key with the description "MSDTCXATM".
- A CID key with the description "MSDTCTIPGW".

For each CID key, the following values:

- The name of the key (a deployment instance-specific GUID).
- The Protocol\(Default) value.
- The Host\(Default) value.

are used to construct a Name object as follows:

- The **CID** field of the Name object is set to the name of the key.
- The **Protocols** field of the Name object is set to the Protocol\(Default) value.
- The **Hostname** field of the Name object is set to the Host\(Default) value.

The Name objects for each CID MUST be mapped as follows:

Name object	Abstract Data Model element	Takes effect
Name object for the "MSDTCUIS" CID	Management Server Name field of the OleTx Management Protocol Management Server.	The mapping MUST take effect when the OleTx Management Protocol Management Server starts up.
Name object for the "MSDTC" CID	The Transaction Manager Name field of the the OleTx Transaction Manager.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
Name object for the "MSDTCXATM" CID	The XA Transaction Manager Name of the OleTx XA Transaction Manager.	The mapping MUST take effect when the OleTx XA Transaction Manager starts up.
Name object for the "MSDTCTIPGW" CID	The TIP Interoperability Provider Name field of the OleTx Transaction Protocol ([MS-DTCM]) TIP Interoperability Provider.	The mapping MUST take effect when the OleTx Transaction Protocol ([MS-DTCM]) TIP Interoperability Provider starts up.

The rest of the values of a CID key MUST be either not mapped (Description\(Default), Endpoint\(Default)) or mapped to implementation-specific data elements (Clsid\(Default), Svcid\(Default)).

In the case in which the machine on which the Management Server resides is set up to work with a remote OleTx Transaction Manager, a set of three more CID keys MUST be exposed by the Management Server:

- A CID key with the description "MSDTC Default".

- A CID key with the description "MSDTCXATM Default".
- A CID key with the description "MSDTCTIPGW Default".

These CID keys MUST be copied from the "MSDTC", "MSDTCXATM", and "MSDTCTIPGW" CID keys exposed on the machine of the remote OleTx Transaction Manager, at the time when the remote relationship is established.

3.3.4.2.1 Custom Property Key Mappings

3.3.4.2.2 Custom Property Key Mappings for the "MSDTC" contact

The custom property values for the "MSDTC" contact (see section 2.2.2.3.2) MUST have the following mappings:

Key value	Abstract Data Model element	Takes effect
Log\Path\(Default)	It MUST map to the path of the implementation-specific log file used by the OleTx Transaction Manager for transaction logging.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
Log\Size\(Default)	It MUST map to the size of the implementation-specific log file used by the OleTx Transaction Manager for transaction logging.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
Service\Path\(Default)	It MUST map to the path of the executable file that contains the implementation of the Transaction Manager code.	The mapping MUST take effect when the OleTx Transaction Manager starts up.

3.3.4.2.3 Custom Property Key Mappings for the "MSDTCUIS" contact

The custom property values for the "MSDTCUIS" contact (see section 2.2.2.3.3) MUST have the following mappings:

Key value	Abstract Data Model element	Takes effect
DAC\ShowLimit\(Default)	The Show Limit field of the Management Server.	The mapping MUST take effect when the Management Server starts up.
DAC\UpdateLimit\(Default)	The Update Limit field of the Management Server.	The mapping MUST take effect when the Management Server starts up.
DAC\TraceLimit\(Default)	The Trace Limit field of the Management Server.	The mapping MUST take effect when the Management Server starts up.

3.3.4.3 Transaction Manager Tracing Key Mappings

3.3.4.3.1 Mappings for "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Tracing\MSDTC\Modules" Key

Key value	Abstract Data Model element	Takes effect
"Active"	It MUST be mapped to an implementation-specific data element that enables/disables tracing for all the OleTx Transaction Manager modules.	The mapping MUST take effect when the OleTx Transaction Manager starts up, and then when the key value is changed.

3.3.4.3.2 Mappings for the "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Tracing\MSDTC\Modules\Transaction_Transitions" Key

Key value	Abstract Data Model element	Takes Effect
"Active"	It MUST be mapped to an implementation-specific data element that enables/disables tracing for transaction transitions with the OleTx Transaction Manager.	The mapping MUST take effect when the OleTx Transaction Manager starts up and then when the key value is changed.
"ControlFlags"	It MUST be mapped to an implementation-specific data element that enables/disables transaction-transition tracing for certain classes of transactions with the OleTx Transaction Manager.	The mapping MUST take effect when the OleTx Transaction Manager starts up and then when the key value is changed.

3.3.4.3.3 Mappings for the "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Tracing\MSDTC\LoggingOptions" Key

Key value	Abstract Data Model element	Takes effect
"MaxBuffers"	It MUST be mapped to an implementation-specific data element specifying the maximum number of buffers used by the OleTx Transaction Manager's tracing functionality.	The mapping MUST take effect when the OleTx Transaction Manager starts up.
"RequestSessionUp"	It MUST be mapped to an implementation-specific data element specifying whether the tracing internal to the OleTx Transaction Manager is started.	The mapping MUST take effect when the OleTx Transaction Manager starts up and then when the key value is changed.

3.3.4.4 Proxy DLL Key Mappings

These keys are defined in Proxy DLL Registry Keys and Values section.

3.3.4.4.1 Mappings for the "HKEY_LOCAL_MACHINE\Software\Classes\OLETransactionManagers" Key

Key value	Abstract Data Model element	Takes effect
"DefaultTM"	It MUST be mapped to an implementation-specific data element	This setting MUST take

Key value	Abstract Data Model element	Takes effect
	specifying the name of the default OleTx Transaction Manager installed on a system.	effect when it is changed.

3.3.4.4.2 Mappings for the "HKEY_LOCAL_MACHINE\Software\Classes\OLETransactionManagers\<TM_NAME>" Keys

Name	Description	Legal values
"DLL"	This setting MUST be mapped to the name of the implementation-specific dynamic link library (DLL) that a OleTx Transaction Protocol ([MS-DTCO]) application MUST load in order to work with the <TM_NAME> OleTx Transaction Protocol) Transaction Manager.	This setting MUST take effect when the <TM_NAME> Transaction Manager is installed.

3.3.5 Higher-Layer Triggered Events

There are no higher-layer triggered events specific to this role.

3.3.6 Message Processing Events and Sequencing Rules

Note that the Management Server Abstract Data Model data elements that these messages modify are reset during Management Server initialization (see section 3.2.3).

3.3.6.1 CONNTYPE_TXUSER_DTCUIC as Acceptor

For all Messages received in this Connection Type, the Management Server MUST process the Message in accordance with section 3.1.5. The TIP Management Server MUST also follow the processing rules specified in the following sections.

3.3.6.1.1 Receiving a MTAG_HELLO Message

When the Management Server receives a MTAG_HELLO Message, it MUST ignore it. (Note that this message is sent by the client just to see if sending it fails or succeeds, which means the connection is down or up, respectively.)

3.3.6.1.2 Receiving a MSG_DTCUIC_UPDATELIMIT Message

When the Management Server receives a MSG_DTCUIC_UPDATELIMIT Message, it MUST perform the following actions:

- If the Connection state is Active:
  - Set the Update Limit field to the value of the **dwUpdateLimit** field from the message.
- Otherwise, the Message MUST be processed as an invalid Message according to the rules defined in section 3.1.5.

3.3.6.1.3 Receiving a MSG_DTCUIC_SHOWLIMIT Message

When the Management Server receives a MSG_DTCUIC_SHOWLIMIT Message, it MUST perform the following actions:

- If the Connection state is Active:
  - Set the Show Limit field to the value of the **dwShowLimit** field from the message.
- Otherwise, the Message MUST be processed as an invalid Message according to the rules specified in section 3.1.5.

3.3.6.1.4 Receiving a MSG_DTCUIC_TRACELIMIT Message

When the Management Server receives a MSG_DTCUIC_TRACELIMIT Message, it MUST perform the following actions:

- If the Connection state is Active:
  - Set the Trace Limit field to the value of the **dwTraceLimit** field from the message.
- Otherwise, the Message MUST be processed as an invalid Message according to the rules specified in section 3.1.5.

3.3.6.1.5 Connection Disconnected

When a CONNTYPE_TXUSER_DTCUIC Connection is disconnected, the Management Server MUST do the following:

- Remove the Connection object from the Management Connection List.
- If the Management Connection List is now empty, cancel and destroy the Update Timer.

3.3.7 Timer Events

3.3.7.1 Update Timer

When this timer expires, the Management Server MUST perform the following actions:

- For each connection in the Management Connection List
  - Send a MSG_DTCUIC_STATS using the connection:
    - The fields of this message MUST be set to implementation-specific values as follows:
      - **cOpen**: The current number of open transactions.
      - **cCommitted**: The current number of committed transactions.
      - **cAborted**: The current number of aborted transactions.
      - **cInDoubt**: The current number of in-doubt transactions.
      - **cHeuristic**: The current number of heuristically completed transactions.
      - **cOpenMax**: The current maximum number of open transactions.
      - **cCommittedMax**: The current maximum number of committed transactions.
      - **cAbortedMax**: The current maximum number of aborted transactions.
      - **cInDoubtMax**: The current maximum number of in-doubt transactions.

- **cForcedCommit**: The current number of transactions that were in-doubt, and were forced to commit.
- **cForcedAbort**: The current number of transactions that were in-doubt, and were forced to abort.
- **cAvgResponseTime**: The current average time for transaction committing.
- **cMinResponseTime**: The current minimum time for transaction committing.
- **cMaxResponseTime**: The current maximum time for transaction committing.
- **timeTransactionsUp**: Implementation-specific time value.
- If the implementation supports the **systemTimeTransactionsUp**, **dwTimestamp**, and **cSinglePhaseInDoubt** fields for the MSG_DTCUIC_STATS message (see section 2.1.3), these fields MUST be set as follows:
  - **systemTimeTransactionsUp**: Implementation-specific time value.
  - **dwTimestamp**: Time value representing the current time when the last statistics were collected.
  - **cSinglePhaseInDoubt**: The current number of transactions that have entered the single-phase in doubt state.

- Create an empty list of DtcUICTransListElement structures with an implementation-specific maximum capacity. <14>
- For each Transaction object in the Transaction Table maintained by the OleTx Transaction Manager that is either in the In Doubt state or older than the Show Limit value (as determined through an implementation-specific mechanism):
  - If the Transaction object is not present in the Tracked Transaction List:
    - Add the Transaction object to the Tracked Transaction List.
- For each Transaction object in the Tracked Transaction List, do the following:
  - If the list of DtcUICTransListElement structures is not full,
    - Create a DtcUICTransListElement and set its fields as follows:
      - Set the **guidTx** field to the Transaction object's identifier.
      - Set the **ulIsol** field to the Transaction object's isolation level.
      - Set the **szDesc** field to the Transaction object's description.
      - Set the **dwStatus** field to the **TRACKING_STATUS** value that corresponds to the state of the transaction, as specified in section 2.2.1.3.1.
      - Set the **szParent** field to the host name specified by the Name field of the Transaction object's Superior Enlistment field.
      - Add the DtcUICTransListElement to the list of DtcUICTransListElement structures.
    - If the value of the **dwStatus** field is XACTSTAT_FORGET, remove the Transaction object from the Tracked Transaction List.

- If the list of DtcUICTransListElement structures is not empty, then send a MSG_DTCUIC_TRANSLIST message using the connection:
  - The **dwNumElements** field MUST be set to the number of elements in the list of DtcUICTransListElement structures.
  - The **elements** field MUST be set to the list of DtcUICTransListElement structures.

- Set the Update Timer's period to the Update Limit field and reset the timer.

3.3.8 Other Local Events

A Management Server MUST be prepared to process the local events defined in the following sections:

3.3.8.1 Incoming Connection Request

This event MUST be signaled by the underlying OleTx Multiplexing Protocol transport layer when an MTAG_CONNECTION_REQ message is received (as specified in [MS-CMP]). The event MUST be signaled with the following arguments:

- Connection: An OleTx Multiplexing Protocol ([MS-CMP]) Connection object.

When the Incoming Connection Request event is signaled, the Management Server MUST do the following:

- If the connection request is for an CONNTYPE_TXUSER_DTCUIC connection (as specified by the Protocol Type field of the connection object) and the connection is accepted:
  - Create an extended Connection object, as specified in Protocol Connection Objects using the Connection object argument.
  - If the Management Connection List is empty:
    - Create an Update Timer and set its period to the Update Limit field; start the timer.
    - Add the extended Connection object to the Management Connection List.
    - Set the Connection state to Active.

3.3.8.2 Trace

This event MUST be signaled with the following arguments:

- Severity: A **TRACE_SEVERITY_LEVEL** enumeration value indicating the severity of the event.
- Source: A 4-byte unsigned integer value specifying the implementation-specific source of the trace event (see the **dwSource** field in section 2.2.1.4.1.7 for more details).
- MessageId: An implementation-specific unsigned integer that maps to a string describing the event to be traced (see the **dwMessage** field in section 2.2.1.4.1.7 for more details).
- Parameter: An implementation-specific null-terminated ANSI string specifying the variable part (if any) of the string identified by MessageId.

If the Trace event is signaled, the Management Server MUST perform the following actions:

- For each connection in the Management Connections list:

- If the value of the Severity argument is greater than the value of the Trace Limit field, send a MSG_DTCUIC_TRACE message using the connection:
  - The **dwSev** field MUST be set to the value of the Severity argument.
  - The **dwSource** field MUST be set to the value of the Source argument.
  - The **dwMessage** field MUST be set to the value of MessageId argument.
  - If the **FormatString** argument is non-null
    - The **fHasParam** field of the message MUST be set to 0x00000001.
    - The **szParam** field of the message MUST be set to the Parameter argument.
  - Otherwise
    - The **fHasParam** field of the message MUST be set to 0x00000000.
    - The **szParam** field of the message MUST be null.

3.3.8.3 Trace String

This event MUST be signaled with the following arguments:

- Severity: A **TRACE_SEVERITY_LEVEL** enumeration value specifying the severity of the event.
- Source: A 4-byte unsigned integer value specifying the implementation-specific source of the trace event (see the **dwSource** field in section 2.2.1.4.1.7 for more details).
- MessageString: An implementation-specific, null-terminated ANSI string describing the trace event (see the **szMsg** field in section 2.2.1.4.1.7 for more details).

If the Trace String event is signaled, the Management Server MUST perform the following actions:

- For each connection in the Management Connections list:
  - If the value of the Severity argument is greater than the value of the Trace Limit field, send a MSG_DTCUIC_TRACESTRING message using the connection:
    - The **dwSev** field MUST be set to the value of the Severity argument.
    - The **dwSource** field MUST be set to the value of the Source argument.
    - The **szMsg** field of the message MUST be set to the FormatString string.

3.3.8.4 Service Control Events

The Management Server MUST respond to **Service Control Manager (SCM)** requests (as specified in [MS-SCMR]) to perform the following actions:

- Start the OleTx Transaction Manager Service.
- Stop the OleTx Transaction Manager Service.
- Interrogate the service status of the OleTx Transaction Manager Service.

4 Protocol Examples

The following section describes common scenarios to illustrate the function of the OleTx Management Protocol. These protocol examples assume that an OleTx transports session, as specified in [MS-CMPO], has already been established between the two participants.

Participants communicate with each other by using OleTx multiplexing connections (as specified in [MS-CMP]), that are in turn layered on top of the OleTx transports infrastructure (as specified in [MS-CMPO]). In these examples, messages are sent from one participant to another by submitting a MESSAGE_PACKET (as specified in [MS-CMP], section 2.2.2) to the underlying OleTx multiplexing layer (as specified in [MS-CMP]).

4.1 Simple Management Client Scenario

This scenario exhibits how an OleTx Management Client creates a connection to an OleTx Management Server. The scenario begins by the application establishing a transport session with an OleTx Management Server and negotiating its connection resources.

4.1.1 Beginning a Management Client

This packet sequence is initiated by starting a connection on a transport session between an application and an OleTx Management Server.

CONNTYPE_TXUSER_DTCUIC: The packet sequence starts when an OleTx Management Client initiates a connection using CONNTYPE_TXUSER_DTCUIC.

Field	Value	Value description
MsgTag	0x00000005	MTAG_CONNECTION_REQ
fIsMaster	0x00000001	1
dwConnectionId	0x00000001	1
dwUserMsgType	0x00000000	CONNTYPE_TXUSER_DTCUIC
dwcbVarLenData	0x00000000	0
dwReserved1	0xCD64CD64	Reserved

The OleTx Management Client then sends a MTAG_HELLO user message to the OleTx Management Server.

Field	Value	Value Description
MsgTag	0x00000FFF	MTAG_USER_MESSAGE
fIsMaster	0x00000001	1
dwConnectionId	0x00000001	1
dwUserMsgType	0x00003006	MTAG_HELLO
dwcbVarLenData	0x00000000	0
dwReserved1	0xCD64CD64	Reserved

When the OleTx Management Server receives the connection request from the underlying OleTx Multiplexing Protocol ([MS-CMP]) layer, the server initiates a timer thread that notifies the OleTx Management Server to send transaction statistics and transaction list information to the OleTx Management Client. The timer is set to notify the OleTx Management Server in accordance to its UI Update Limit value.

When the timing thread notifies the OleTx Management Server, the server sends a MSG_DTCUIC_STATS user message to the OleTx Management Client. In this message, the OleTx Management Server reports that two (2) transactions are open, seventeen (17) transactions have been committed, and as many as eight (8) transactions were open at the one time since the transaction manager was started up on 6/14/2007.

Field	Value	Value description
MsgTag	0x00000FFF	MTAG_USER_MESSAGE
fIsMaster	0x00000001	1
dwConnectionId	0x00000001	1
dwUserMsgType	0x00003001	MSG_DTCUIC_STATS
dwcbVarLenData	0x00000058	88
dwReserved1	0xCD64CD64	Reserved
cOpen	0x00000002	2
cCommitted	0x00000011	11
cAborted	0x00000000	0
cInDoubt	0x00000000	0
cHeuristic	0x00000000	0
cOpenMax	0x00000008	8
cCommittedMax	0x00000011	17
cAbortedMax	0x00000000	0
cInDoubtMax	0x00000000	0
cHeuristicMax	0x00000000	0
cForcedCommit	0x00000000	0
cForcedAbort	0x00000000	0
cAvgResponseTime	0x00002364	9060
cMinResponseTime	0x00001F4F	8015
cMaxResponseTime	0x0000B508	46344
timeTransactionsUp	0x46709338	1181782840
systemTimeTransactionsUp	0x07D70006	6/14/2007 1:00:40.640 AM (UTC)

Field	Value	Value description
	0x0004000E	
	0x00010000	
	0x00280280	
dwTimestamp	0x00000000	0
cSinglePhaseInDoubt	0x00000001	1

The OleTx Management Server then sends a MSG_DTCUIC_STATS user message to the OleTx Management Client if there are any open transactions older than specified by the UI Show Limit value. In this message, the OleTx Management Server reports the Transaction Id, the Isolation Level, the description, the status and Parent name for two (2) open transactions. If there are no open transactions older than specified by the UI Show Limit value, then this message is not sent.

Field	Value	Value description
MsgTag	0x00000FFF	MTAG_USER_MESSAGE
fIsMaster	0x00000001	1
dwConnectionId	0x00000001	1
dwUserMsgType	0x00003002	MSG_DTCUIC_TRANLIST
dwcbVarLenData	0x000000A4	164
dwReserved1	0xCD64CD64	Reserved
dwNumElements	0x00000002	2
guidTx	0xB30F0859	b30f0859-f3cf-4866-8db1-287e81cc69f2
	0x4866F3CF	
	0x7E28B18D	
	0xF269CC81	
ulIsol	0x00100000	ISOLATIONLEVEL_SERIALIZABLE
szDesc	0x6E617254	"Transaction #1"
	0x74636173	

Field	Value	Value description
	0x206E6F69	
	0x00003123	
	0x00000000	
	0x00000000	
	0x00000000	
	0x00000000	
	0x00000000	
	0x00000000	
dwStatus	0x00000C01	XACTSTAT_ONLY_FAILED_COMMITED_REMAIN
szParent	0x6863614D	"Machine2"
	0x32656E69	
	0x00000000	
	0x00000000	
guidTx	0x2489B646	2489b646-94f0-41c6-a470-2b618d9f1ef2
	0x41C694F0	
	0x612B70A4	
	0xF21E9F8D	
ulIsol	0x00100000	ISOLATIONLEVEL_SERIALIZABLE
szDesc	0x6E617254	"Transaction #2"

Field	Value	Value description
	0x74636173	
	0x206E6F69	
	0x00003223	
	0x00000000	
	0x00000000	
	0x00000000	
	0x00000000	
	0x00000000	
	0x00000000	
dwStatus	0x00020000	XACTSTAT_INDOUBT
szParent[16]	0x6863614D	"Machine2"
	0x32656E69	
	0x00000000	
	0x00000000	

The OleTx Management Server continues to send these messages until the OleTx Management Client closes the connection by initiating the disconnect sequence.

4.1.2 Adjusting the UI Update Limit

While the management connection is running, the OleTx Management Client can adjust the UI Update Limit value by sending a MSG_DTCUIC_UPDATELIMIT user message to the OleTx Management Server, specifying the UI Update Limit value. In this message, the OleTx Management Client specifies a UI Update Limit of UPDATE_5 (0x00000002), indicating that the OleTx Management Client wants updates sent every five (5) seconds.

Field	Value	Value description
MsgTag	0x00000FFF	MTAG_USER_MESSAGE
fIsMaster	0x00000001	1
dwConnectionId	0x00000001	1
dwUserMsgType	0x00003004	MSG_DTCUIC_UPDATELIMIT
dwcbVarLenData	0x00000004	4
dwReserved1	0xCD64CD64	dwReserved1: 0xCD64CD64
dwUpdateLimit	0x00000002	UPDATE_5

When the OleTx Management Server receives the message, the server updates its UI Update Limit value with the new value. After the timer notifies the OleTx Management Server to send transaction statistics and transaction list information, the timer will wait five seconds before notifying the server to send management data to the client.

4.1.3 Adjusting the UI Show Limit

To adjust the UI Show Limit value, the OleTx Management Client sends a MSG_DTCUIC_SHOWLIMIT user message to the OleTx Management Server, specifying the UI Show Limit value. In this message, the OleTx Management Client specifies a UI Show Limit of SHOW_10_SEC (0x00000003), indicating that the OleTx Management Client wants the OleTx Management Server to only send transaction list data on open transactions that have been open for ten (10) seconds or more.

Field	Value	Value description
MsgTag	0x00000FFF	MTAG_USER_MESSAGE
fIsMaster	0x00000001	1
dwConnectionId	0x00000001	1
dwUserMsgType	0x00003005	MSG_DTCUIC_SHOWLIMIT
dwcbVarLenData	0x00000004	4
dwReserved1	0xCD64CD64	dwReserved1: 0xCD64CD64
dwShowLimit	0x00000003	SHOW_10_SECS

When the OleTx Management Server receives the message, the server updates its UI Show Limit value with the new value. When the timer notifies the OleTx Management Server to send transaction statistics and transaction list information, the timer will only send transaction list information of transactions that have been open for ten (10) seconds or more.

4.2 Enabling XA Transactions Scenario

This scenario exhibits how an OleTx Management Client enables XA transactions on a remote machine; generally this scenario would be initiated by higher-layer logic.

4.2.1 Setting the XaTransactions Registry Key on the Remote Machine

The OleTx Management Client enables XA transactions on a remote machine by setting the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDTC\Security\XaTransactions registry value on the remote machine to one (DWORD: 0x00000001).

To do this, the OleTx Management Client establishes a Windows Remote Registry Protocol ([MS-RRP]) connection to the remote machine. The client then sends an OpenLocalMachine method (as specified in [MS-RRP], section **3.1.5.3**) with the following values for the parameters:

```
ServerName = 0
samDesired = 0x00020006// KEY_WRITE
phKey      = NULL
```

When the Windows Remote Registry Protocol server receives this request from the client, the server opens the handle to the root key HKEY_LOCAL_MACHINE with write access and returns 0 (ERROR_SUCCESS) and the pointer to the opened handle in the phKey parameter of the response.

The Windows Remote Registry Protocol client then uses the handle that is returned in phKey to open the subkey "SOFTWARE\Microsoft\MSDTC\Security" for write access by sending a BaseRegOpenKey (as specified in [MS-RRP], section **3.1.5.15**) method with the following parameter values:

```
hKey       = Handle (phKey) returned in
             previous server response.
lpSubKey   = L"SOFTWARE\\Microsoft\\MSDTC\\Security"
dwOptions  = 0
samDesired = 0x00020006// KEY_WRITE
phkResult  = NULL
```

When the Windows Remote Registry Protocol server receives this request from the client, the server opens the handle to the registry key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDTC\Security with write access and returns 0 (ERROR_SUCCESS) and the pointer to the opened handle in the phkResult parameter of the response.

The Windows Remote Registry Protocol client then uses the handle that is returned in phkResult to set "XaTransactions" value to one (0x00000001) in the registry subkey "SOFTWARE\Microsoft\MSDTC\Security" for write access by sending a BaseRegSetValue (as specified in [MS-RRP], section **3.1.5.22**) method with the following parameter values:

```
hSubKey     = Handle (phkResult) returned in
              previous server response.
lpValueName = L"XaTransactions"
dwType      = 4// REG_DWORD
lpData      = pointer(0x00000001)
cbData      = 4
```

When the Windows Remote Registry Protocol server receives this request from the client, the server uses the handle to the registry subkey HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSDTC\Security to set the REG_DWORD value "XaTransactions" to one (0x00000001) and returns 0 (ERROR_SUCCESS).

The Windows Remote Registry Protocol client then closes the registry subkey handles by sending a BaseRegCloseKey (as specified in [MS-RRP], section **3.1.5.6**) method with the following parameter value:

```
hSubKey = Handle (phkResult) returned in
          previous server response.
```

When the Windows Remote Registry Protocol server receives this request from the client, the server closes the registry subkey handle and returns 0 (ERROR_SUCCESS).

The Windows Remote Registry Protocol client then closes the registry key handle by sending a BaseRegCloseKey (as specified in [MS-RRP], section **3.1.5.6**) method with the following parameter value:

```
hKey = Handle (phKey) returned in previous server response.
```

When the Windows Remote Registry Protocol server receives this request from the client, the server closes the registry key handle and returns 0 (ERROR_SUCCESS).

4.2.2 Restarting the Transaction Manager on the Remote Machine

Since the XaTransactions registry value is only read during transaction manager startup, the remote transaction manager has to be restarted by using the Service Control Manager Remote Protocol ([MS-SCMR]).

4.2.2.1 Stopping the Transaction Manager Service

To restart the transaction manager on the remote machine, the OleTx Management Client established a Service Control Manager Remote Protocol ([MS-SCMR]) connection with the remote machine (e.g. Machine2) and sends an ROpenSCManagerW call with the following values for the parameters.

```
lpMachineName   = L"Machine2"
lpDatabaseName  = SERVICES_ACTIVE_DATABASE
dwDesiredAccess = 0x00000001 // SC_MANAGER_CONNECT
lpScMgrHandle   = NULL
```

On receiving this request from the client, the server opens the handle to the SCM database with read access and returns from the method with an error code of 0 and the pointer to the opened handle in the lpScMgrHandle parameter of the response.

The client then uses the handle returned in lpScMgrHandle to open the transaction manager service with stop access by sending an RGetServiceDisplayNameW call with the following values for the parameters.

```
hSCManager      = Handle returned in the lpScMgrHandle
                  parameter of the previous server response.
lpServiceName   = L"MSDTC"
dwDesiredAccess = 0x00000020 // SERVICE_STOP
lpScHandle      = NULL
```

Upon receiving this request from the client, the server queries the SCM database for the service named "MSDTC" and returns from the method with an error code of 0, and the pointer to the opened handle in the lpScHandle parameter of the response.

The client then uses the handle returned in lpScHandle to stop the transaction manager service by sending an RControlServiceW call with the following values for the parameters.

```
hScHandle   = Handle returned in the lpScHandle
              parameter of the previous server response.
dwControl   = 0x00000001 // SERVICE_CONTROL_STOP
lpSvcStatus = NULL
```

On receiving this request from the client, the server uses the handle to stop the service and returns from the method with an error code of 0 and the pointer to the last reported service status in the lpSvcStatus parameter of the response.

The client closes the service handle by sending an RCloseServiceHandle with the following values for the parameters.

```
hScHandle = Handle returned in the lpScHandle
            parameter of the previous server response.
```

On receiving this request from the client, the server closes the handle to the service and returns from the method with an error code of 0.

The client closes the SCM database by sending an RCloseServiceHandle with the following values for the parameters.

```
hSCManager = Handle returned in the lpScMgrHandle
             parameter of the previous server response.
```

On receiving this request from the client, the server closes the handle to the open SCM database and returns from the method with an error code of 0.

4.2.2.2 Starting the Transaction Manager Service

To restart the transaction manager on the remote machine, the OleTx Management Client established a Service Control Manager Remote Protocol ([MS-SCMR]) connection with the remote machine (e.g. Machine2) and sends an ROpenSCManagerW call with the following values for the parameters.

```
lpMachineName   = L"Machine2"
lpDatabaseName  = SERVICES_ACTIVE_DATABASE
dwDesiredAccess = 0x00000001 // SC_MANAGER_CONNECT
lpScMgrHandle   = NULL
```

On receiving this request from the client, the server opens the handle to the SCM database with read access and returns from the method with an error code of 0 and the pointer to the opened handle in the lpScMgrHandle parameter of the response.

The client then uses the handle returned in lpScMgrHandle to open the transaction manager service with stop access by sending an RGetServiceDisplayNameW call with the following values for the parameters.

```
hSCManager      = Handle returned in the lpScMgrHandle
                  parameter of the previous server response.
lpServiceName   = L"MSDTC"
dwDesiredAccess = 0x00000010 // SERVICE_START
lpScHandle      = NULL
```

Upon receiving this request from the client, the server queries the SCM database for the service named "MSDTC" and returns from the method with an error code of 0, and the pointer to the opened handle in the lpScHandle parameter of the response.

The client then uses the handle returned in lpScHandle to start the transaction manager service by sending an RStartServiceW call with the following values for the parameters.

```
hScHandle = Handle returned in the lpScHandle
            parameter of the previous server response.
argc      = 0
argv      = NULL
```

On receiving this request from the client, the server uses the handle to start the service and returns from the method with an error code of 0.

The client closes the service handle by sending an RCloseServiceHandle with the following values for the parameters.

```
hScHandle = Handle returned in the lpScHandle
            parameter of the previous server response.
```

On receiving this request from the client, the server closes the handle to the service and returns from the method with an error code of 0.

The client closes the SCM database by sending an RCloseServiceHandle with the following values for the parameters.

```
hSCManager = Handle returned in the lpScMgrHandle
             parameter of the previous server response.
```

On receiving this request from the client, the server closes the handle to the open SCM database and returns from the method with an error code of 0.

5 Security

This protocol employs the security mechanism of the underlying transport infrastructure specified in [MS-CMP] and [MS-CMPO]. Since the information exchanged in Messages by this protocol can contain sensitive data, like the transaction identifiers and Transaction Manager addresses, implementers SHOULD use Mutual Authentication, as specified in [MS-CMPO], section 2.1.3. <15>

Regarding the [MS-RRP] registry keys exposed by the Management Server:

- The HKLM\Software\Microsoft\MSDTC key
- The HKEY_LOCAL_MACHINE\Software\Microsoft\MSDTC\Security key
- The HKEY_LOCAL_MACHINE \Software\Classes\CID key and its subkeys
- The HKEY_LOCAL_MACHINE\Classes\OLETransactionManagers key and its subkeys
- The HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Tracing\MSDTC key and its subkeys

SHOULD be protected for access as follows:

- Read access (KEY_READ) SHOULD be granted to all authenticated users (AUTHENTICATED_USERS, as defined in [MS-SECO]).
- Write access (KEY_WRITE) SHOULD to be granted to a restricted group of users.
- Full access (KEY_ALL_ACCESS) SHOULD be granted to a restricted group of users. <16>

6 Appendix A: Windows Behavior

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

- Windows Server 2008
- Windows Vista
- Windows Server 2003
- Windows XP
- Windows 2000
- Windows NT 4.0 Option Pack

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

<1> Section 1.6: This protocol is supported in Windows Server 2008, Windows Vista, Windows Server 2003 and its service packs, Windows XP and its service packs, and Windows 2000.

<2> Section 2.1.1.1.1: Mutual authentication is used by default on Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP SP2 and later. No security is used on Windows XP SP1 or Windows 2000.

<3> Section 2.1.1.1.2: Data necessary to construct the Name object for the Management Server implementation is found in the Windows registry.

<4> Section 2.2.1.4.1.5: In 64-bit Windows implementations, this field contains 4 bytes of padding (containing non-deterministic values), followed by an 8-byte unsigned integer. Therefore, the total size of the field is 12 bytes. The baseline time used by Windows implementations is midnight January 1, 1970, Coordinated Universal Time (UTC). This data is for informational purposes only.

<5> Section 2.2.1.4.1.5: Windows uses Coordinated Universal Time (UTC). This data is for informational purposes only.

<6> Section 2.2.1.4.1.5: In Windows, this specifies the number of milliseconds that elapsed from the time the machine was last started until the time when statistics were last recorded. This data is for informational purposes only.

<7> Section 2.2.1.4.1.7: The following table displays the trace sources defined on Windows implementations and their corresponding identifiers:

Identifier	Trace source
0x00000001	The MSDTC service functionality.
0x00000002	The Transaction Manage core.
0x00000003	The connection manager.
0x00000004	The log related functionality.
0x00000005	The XA TM module.

Identifier	Trace source
0x00000006	The MTXOCI module.
0x00000007	The Asynchronous Events functionality.
0x00000008	Functionality in none of the defined categories.
0x00000009	The proxy bootstrap (xolehlp) functionality.
0x0000000A	The MSDTC proxy functionality.
0x0000000B	The TIP gateway module.
0x0000000C	The TCP connection manager.
0x0000000D	The trace module.
0x0000000E	The cluster related functionality.
0x0000000F	The MSDTC user interface.
0x00000029	The kernel transaction manager resource manager (KTMRM).

<8> Section 2.2.1.4.1.7: On Windows platforms, the implementation provides a resource DLL which contains the mapping between the integers passed in the dwMessage field of the MSG_DTCUIC_TRACE messages and the trace string specific to the implementation. The following table provides shows the mappings contained in that resource DLL:

4-byte unsigned integer value	Trace string
0x4000101C	MS DTC logging has started.
0x4000101D	The MS DTC log file has been decompressed.
0x4000101E	The MS DTC Log Manager is stopping.
0x4000102E	A log client has reported taking a checkpoint.
0x8000D008	MS DTC detected dirty log pages on **recovery**, and is reconstructing state from other pages.
0xC000101C	Unable to decompress the MS DTC log file. Please ensure that there is sufficient available disk space on the MS DTC log device.
0xC000101D	The MS DTC log file is full and cannot accept new log records.
0xC000101E	The MS DTC log file attributes are invalid. The log file must not be read only.
0xC000101F	The call to MapViewOfFile failed for the MS DTC log file.
0xC000103A	The MS DTC log file is unreadable. After ensuring that all Resource Managers coordinated by MS DTC have no indoubt transactions, please run "msdtc -resetlog" to reset the log file.
0xC0001042	The MS DTC log file is an incompatible version.
0xC0001043	MS DTC log file not found. After ensuring that all Resource Managers coordinated by

4-byte unsigned integer value	Trace string
	MS DTC have no indoubt transactions, please run msdtc -resetlog to create the log file.
0xC0001046	The maximum number of active transactions that the MS DTC log file can accommodate has been exceeded. An implementer must increase the size of the MS DTC log file to initiate more concurrent transactions.
0xC000104B	MS DTC Transaction Manager log write failed with error %1.
0xC000104C	MS DTC Transaction Manager start failed. GetClassObject on LogMgr returned error %1.
0xC000104D	MS DTC Transaction Manager start failed. GetCurrentLogRecord returned error %1.
0xC000104E	MS DTC Transaction Manager start failed. GetCheckpoint returned error %1.
0xC000104F	MS DTC Transaction Manager start failed. Init on LogRead returned error %1.
0xC0001050	MS DTC Transaction Manager start failed. LogRead returned error %1.
0xC0001051	MS DTC Transaction Manager start failed. LogFlush returned error %1.
0xC0001052	MS DTC Transaction Manager start failed. LogWrite returned error %1.
0xC0001053	MS DTC Transaction Manager start failed. OpenLogStream for Read returned error %1.
0xC0001054	MS DTC Transaction Manager start failed. OpenLogStream for Write returned error %1.
0xC0001055	MS DTC Transaction Manager start failed. QueryInterface on LogStorage returned error %1.
0xC0001056	MS DTC Transaction Manager start failed. QueryInterface on LogRecordPointer returned error %1.
0xC0001057	MS DTC Transaction Manager start failed. QueryInterface on TransactionTrackerFactory returned error %1.
0xC0001058	MS DTC Transaction Manager start failed. SetCheckpoint returned error %1.
0xC0001059	MS DTC Transaction Manager start failed. LogInit returned error %1.
0xC000105A	MS DTC Transaction Manager start failed. QueryInterface on LogWriteAsynch returned error %1.
0xC000105B	MS DTC Transaction Manager start failed. Init on LogWriteAsynch returned error %1.
0xC000105C	MS DTC Transaction Manager start failed. Out of memory.
0xC000105D	MS DTC Transaction Manager start failed. Log record too small.
0xC000105E	MS DTC Transaction Manager start failed. Log record type unknown.
0xC000105F	MS DTC Transaction Manager start failed. Unable to unpack name object, error %1.
0xC0001061	A resource manager performed recovery and called ReenlistmentComplete indicating that recovery was complete. However, there is at least one transaction that was

4-byte unsigned integer value	Trace string
	enlisted with the resource manager whose state is still "In Doubt".

Note that some trace strings in the above table take a variable parameter which is designated by "%1". This parameter, when needed, is provided in the **szParam** field of the MSG_DTCUIC_TRACE message.

<9> Section 2.2.1.4.1.8: Windows implementations define the following trace strings that are passed in the szMsg field:

- "DebugTrace should be a string type."
- "Could not create <filename>", where <filename> is the name of a file.
- "Unable to open/create key: <key>. Return value from RegCreateKeyEx is: <retval>", where
  - <key> is the name of a registry key
  - <retval> is a return value from calling RegCreateKeyEx
- "The value <valname> in key: <key> is of type: <valtype>. It should be a string type instead.", where
  - <valname> is a the name of a registry key value
  - <key> is the name of a registry key
  - <valtype> is the type of a registry key value
- "Could not read value <valname> in key: <key>. RegQueryValueEx returned: <retval>", where
  - <valname> is the name of a registry key value
  - <key> is the name of a registry key
  - <retval> is the return value from calling RegQueryValueEx.
- "Could not copy <valname> value to key: <key>. Return value from RegSetValueEx is: <retval>", where
  - <valname> is the name of a registry key value
  - <key> is the name of a registry key
  - <retval> is the return value from calling RegSetValueEx.
- "MTxOCI--Enlist API : Call failed due to time out. CurrState = <transactionstate>\n", where <transactionstate> is the state of a transaction.
- "OCI8 API is being requested on an OCI7.X machine"
- "MTxOCI--Enlist API : Call failed due to time out. CurrState = <transactionstate>\n", where <transactionstate> is the state of a transaction.
- "xa_open call failed. Error=<retval>\n", where <retval> is the value returned from the call.
- "olog call failed Error=<retval>", where <retval> is the value returned from the call.

- "xa_start call failed with error <error> \n", where <error> is the error code from the call.
- "sqlld2 call failed with error <error> \n", where <error> is the error code from the call.
- "ENLIST xa_end call failed with error <error> \n", where <error> is the error code from the call.
- "xa_end call failed inside of PrepareRequest."
- "CommitRequest failed with error <error>, RMGuid=<guid> \n", where <error> is an error code specific to the request and <guid> is a GUID.
- "xa_end failed with error <error> \n", where <error> is the error code of the failure.
- "xa_rollback failed with error <error>, RMGuid=<guid>\n", where <error> is the error code for the failure and <guid> is a GUID.
- "xa_rollback caused an exception. \n"
- "TM failed to connect to the RM. Following was the connect string"
- "opinit call failed with error <error>\n", where <error> is the error code for the failure.
- "Disconnected from XATM. RECOVERY = <status>, RMGuid = <guid>", where <status> is the recovery status and <guid> is a GUID.

<10> Section 2.2.2.1.1: SPX is not valid on Windows Vista or Windows Server 2008.

<11> Section 2.2.2.3.1: An empty string is interpreted as TCP/IP (section 2.2.2.1.1) on Windows NT 4.0. On all other platforms this value is ignored.

<12> Section 3.3.2.1: For Windows implementations, the default value of this timer is 1 second.

<13> Section 3.3.4.1: The default security level is Mutual Authentication on Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP SP2 and later. The default security level is No Security on Windows XP SP1 and Windows 2000.

<14> Section 3.3.7.1: In Windows, the number of elements in this list MUST NOT exceed 30.

<15> Section 5: Mutual authentication is used by default on Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP SP2 and later. No security is used on Windows XP SP1 or Windows 2000.

<16> Section 5: Windows implementations grant KEY_WRITE and KEY_ALL_ACCESS rights only to the following group of users: BUILTIN_ADMINISTRATORS, LOCAL_SYSTEM, and the MSDTC service account.

7 Index

T

U

V

W