The MyDelivery Server

Frank Walker and Girish Lingappa

June 2010

The MyDelivery Server

National Library of Medicine
8600 Rockville Pike, Bethesda, MD - 20894

The MyDelivery Server

Summary:

This document explains all aspects of the MyDelivery system architecture and server design. It should be sufficient to allow an organization to deploy its own MyDelivery server system. The minimum requirement of one Windows-based server is capable of handling several hundred Windows-based clients. The server system is easily expandable to allow thousands more users or to enhance server reliability.

Table of Contents

1.0 System Architecture ***5***

1.2 Synchronization of Client Communication 6
1.3 Expanded Server Architecture 7
1.4 Alternative Technologies 8
1.5 System Performance 10

2.0 Client-Server SOAP Communications ***11***

2.1 GetVersion 11
2.2 Initialize 12
2.3 CSP (ClientStatusPage) 14
2.4 UploadUserInformation 16
2.5 ChangePassword 19
2.6 UpdateStatus 20
2.7 CheckRecipient 24
2.8 StartDeliveryUpload 26
2.9 UploadDeliveryHeader1 28
2.10 RequestToUpload 35
2.11 UploadAttachment 37
2.12 IsDeliveryValid 38
2.13 DownloadDeliveryHeader1 40
2.14 DownloadAttachment 41
2.15 DeliveryStatus 43
2.16 TerminateDelivery 44

3.0 MyDelivery Server Architecture ***46***

3.1 MyDelivery Windows Service: mdClientListService 48
3.2 MyDelivery Windows Service: mdFileXFERService 50
3.3 MyDelivery Windows Service: mdMonitor 51
3.4 MyDelivery Windows Service: mdFlusher 52
3.5 MyDelivery Windows Service: mdPinger 53
3.6 MyDelivery Windows Service: mdClientManager 54

4.0. Server Setup ***56***

5.0 MyDelivery SQL Server Database Tables ***84***

5.1 ClientsOnline 85
5.2 Deliveries 86
5.3 IDs 88
5.4 InvalidLogin 89
5.5 PendingSenders 90
5.6 Processors 91
5.7 RegCodes 92
5.8 SystemInfo 93
5.9 Users 94

6.0 SQL Server Database Configuration using SQL Server Express______________________ 95

7.0 Implementation of Database Mirroring via SQL Server 2005 _____________________ 105

7.1 Setting up Security____ 105

7.2 Backup MyDelivery ____ 113

7.3 Detach, Attach and Restore MyDelivery____________________________________ 116

7.4 Configure Mirroring____ 120

1.0 System Architecture

MyDelivery is Windows-based client-server architecture. As illustrated in Figure 1.1, the minimal system configuration consists of a single server and two clients. The server, running Microsoft's IIS, keeps track of which clients are online, and it facilitates communication between the two clients. Each Windows-based client is capable of communicating messages with any other client. Throughout this document an exchange of a message with optional attachments will be termed a 'delivery'. Each user has a personal address book that lists other users with which he or she wishes to communicate using the client software. Clients can communicate only with other users listed in a user's address book. All communication occurs through the server. At no time do the two clients communicate with each other directly, as in peer-to-peer systems.

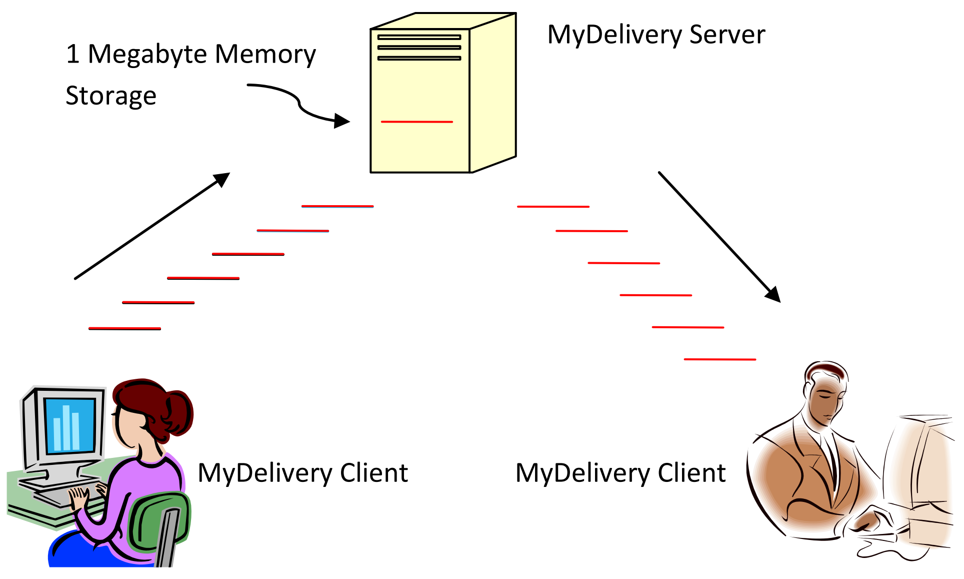


Figure 1.1 Segmented Data Communication

Furthermore, no user delivery is stored on the server hard disk; instead, the data resides temporarily in the server memory in 1 megabyte segments or chunks. When the server establishes connectivity between each two users, it temporarily allocates a 1 megabyte segment of memory for that communication. Ten pairs of users will be allocated 1 megabyte of memory per pair, for a total of 10 megabytes. One thousand pairs of users will be allocated 1,000 segments of memory, for a total of 1 gigabyte. Because memory is inexpensive (less than $50 per gigabyte in 2010), with 3 or more gigabytes typically available on a web server, this architecture makes it possible to have 1,000 pairs of users on a single server using up only 1 gigabyte of memory. If more than 1 megabyte of data needs to be sent in a delivery, then the delivery is segmented into 1 megabyte chunks. A delivery that has a 1 gigabyte file attached will be divided into 1,000 chunks, each of which is sent separately and stored temporarily in the server memory allocated to that data exchange. Figure 1.2 shows how multiple files are packed into the data segments. The sending client first compresses all file attachments, and then packs them into 1 megabyte chunks, which are then sent to the server. The sending and receiving clients negotiate with the server to exchange one chunk at a time. Once all chunks in a delivery are downloaded to the receiving client, that client separates them into compressed files and decompresses them to get the original files.

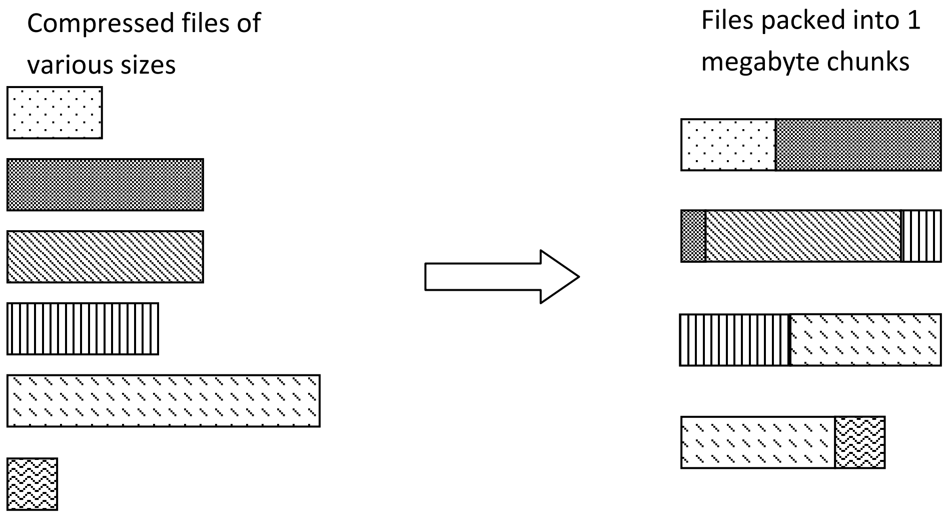

Compressed files of various sizes
Files packed into 1 megabyte chunks

Figure 1.2 Segmented File Stream

1.2 Synchronization of Client Communication

Communications between two clients takes place only when both clients are online simultaneously. This is similar to the case of two fax machines that communicate with each other, and is in contrast to other, more conventional methods of Internet communication, such as email or FTP, which allow one client to be offline. It is the job of the MyDelivery server to detect the online status of all clients. When one client requests to send a delivery to a second client, the server establishes whether the second client is online, and if so, facilitates the data exchange. If the recipient is not online, the sending client retains the delivery for future exchange with the recipient.

The communication protocol between the two clients and server allow the server to synchronize the sending client's data stream with the recipient's data stream. The two clients exchange a 1 megabyte chunk of data stored temporarily in server memory, and the protocol governs when the sending client may upload its data chunk to the server, and when the receiving client may download it. As a router, the server negotiates the data chunk exchange between the two clients. The two client data streams may be of differing speed; the sending stream may, for instance, be much faster or slower than the receiving stream, or vice versa. The communications protocol ensures synchronization between the two streams by slowing down the effective rate of the faster stream to that of the slower stream. If the sending client is faster than the receiving client, then the server will tell the sending client to slow down until its speed matches that of the receiving client. Similarly, if the receiving client is faster than the sending client, the server will force it to slow down to match the sending client's speed. Due to the nature of the Internet, communication speeds will vary throughout the day depending on usage. Thus, at any given moment, the original client that had been faster may turn into a slower client. The server will recognize this, and attempt to increase a client's speed to match that of the faster client. The process of matching speeds is dynamic: the server will increase or decrease a client's speed so that it matches that of the other client, and vice versa. The server's task is then to synchronize the speeds of the two clients so that they communicate with each other at the highest possible rate.

1.3 Expanded Server Architecture

While a single server can handle several hundred online users, an organization implementing MyDelivery may at some point need to expand its capacity, whether to add more users, or to achieve a higher degree of system reliability. Figure 1.3 shows a multi-CPU server architecture that allows thousands of online users, with sufficient redundancy to handle cases where one or more machines may fail.

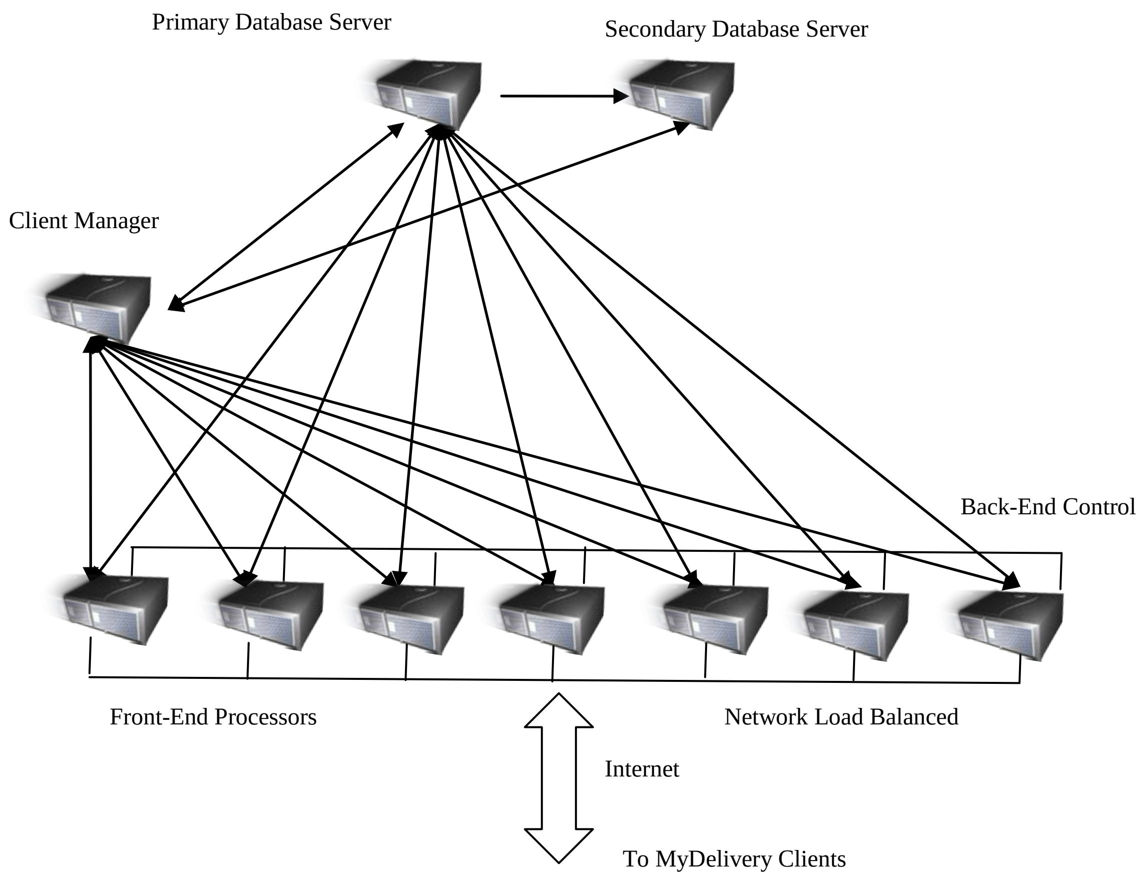


Figure 1.3. Multi-CPU MyDelivery Server

This document explains how to configure servers for small or large configurations. Each machine runs Windows server 2003 or 2008. In this example, seven machines are "Front-End Processors," for handling communication with clients. They run in a network load balanced configuration, which permits traffic to be evenly distributed across all Front-end Processors. Furthermore, network load balancing allows one or more Front-End Processors to fail, and the remaining machines will continue operating to ensure error-free client communication. The Client Manager processor keeps track of all online clients, and performs periodic system maintenance. The Primary Database Server and Secondary Database Server each run SQL Server 2005 or 2008, which allows database mirroring and automatic failover protection. With database mirroring, the Secondary Database Server maintains a current up-to-date copy of the relevant tables on the Primary Database Server. If the Primary Database Server goes down, the system detects this, and transfers operation to the Secondary Database Server, which then becomes the Primary. Then, when the downed server comes back online, it becomes the Secondary Database Server, and maintains an up-to-date copy of the tables on the new Primary Database Server.

1.4 Alternative Technologies

If there is any technology that might closely resemble MyDelivery, it might be a secure fax machine that would be capable of sending files. Fax machines use the telephone system for communication and synchronization. No fax data is ever stored on servers where it could be lost. The problem is that fax machines send copies of paper, not files, and the communication is usually not secure. MyDelivery uses the Internet for data communication, and there are a number of existing technologies that permit two clients to exchange information. These include email, secure email, FTP, instant messaging, and web delivery services. The following table compares MyDelivery with alternative Internet technologies.

Tool	Large Datasets	Personal Communications	Easy to set up and use	Secure	Reliable over poor networks	User Data Safe if Server is Lost
MyDelivery	✓	✓	✓	✓	✓	✓
Email		✓	✓			
Secure Email		✓		✓		
FTP	✓					
Peer-to-Peer Instant Messaging	✓	✓	✓			✓
Web-based Instant Messaging		✓	✓	✓		
Web Delivery Services	✓		✓	✓		

Email, a 30-year old technology, is easy to use, safe and usually free. However, it often limits message attachment size, quantity, file types and inbox size. Secure email often presents users a hassle with obtaining (for a fee) and managing security certificates. However, it provides end-to-end HIPAA-compliant encryption and verification for limited attachment sizes and quantity.

Like email, FTP has been around for 30 years, and it allows users to exchange large files or large numbers of files. An FTP server is tricky to set up, as it requires network administrators to poke holes through firewalls. Neither FTP nor email protocols compensate for intermittent network connections, requiring retransmission of data in the event of network failure.

Instant Messaging is easy to use, and usually allows large file exchange. There are two basic types of Instant Messaging systems: one that uses peer-to-peer client communication and another that is web-based. Peer-to-peer Instant Messaging systems offer file exchange that usually requires a server to run on a user's desktop computer (potentially allowing hackers to enter). Web-based instant messaging systems do not use peer-to-peer communications, but employ a web server for temporary buffer for user data. These systems may or may not permit file exchange. If they do, then some of these systems may potentially archive user data for long periods of time, during which the data could be compromised.

A wide variety of web delivery services have sprung up in recent years, including BigUpload.com, DropSend.com, SendSpace.com, YouSendit.com, MegaUpload.com, SendThisFile.com, GigaSize.com, MailBigFile.com, WebCargo.com, and DropSend.com. These usually permit a user to upload a file via a browser to a server. After the file arrives at the server, an email message is sent to the recipient, who then clicks on a link in the email message to the server for file retrieval. Some services limit file size to between 1 and 50 MB, while others permit up to 2 GB files. None offer HIPAA-compliant verification, and none offer protection for communicating over poor networks.

The big advantage of MyDelivery over alternative technologies is its ability to offer HIPAA-compliant exchange of user information while avoiding storage of that information on a server's hard disk. The avoidance of using a server's hard disk for temporary storage of user data makes MyDelivery a very low-cost solution to implement, while at the same time offering users an increased level of security. MyDelivery also detects intermittent network communication and automatically recommences communication at the point where communication was lost without having to restart from the beginning.

1.5 System Performance

MyDelivery has been tested in an office environment over an Ethernet local area network, with a measured speed of about 13 minutes to transfer a 1 gigabyte file from one client to another. This time will vary depending on the compressibility of the data sent. It could be considerably longer if using a wireless network or if transferring data over the Internet, where speeds can vary considerably.

2.0 Client-Server SOAP Communications

A unique communications protocol has been designed for MyDelivery client-server communication and control. All communication takes place using HTTP or HTTPS, which nearly all firewalls permit. Sixteen functions were created using Simple Object Access Protocol (SOAP), which is the transmission of XML-based messages over HTTP. The functions are listed below, followed by a brief description given for each.

GetVersion
Initialize
CSP
UploadUserInformation
ChangePassword
UpdateStatus
CheckRecipient
StartDeliveryUpload
UploadDeliveryHeader1
RequestToUpload
UploadAttachment
IsDeliveryValid
DownloadDeliveryHeader1
DownloadAttachment
DeliveryStatus
TerminateDelivery

2.1 GetVersion

A MyDelivery client uses the GetVersion function to find out the latest version of the client available on the server. It checks the version number with its own version number, and determines whether the client software needs to be updated to the new version. This function is typically executed once per session and it is usually the first SOAP function executed.

Parameters sent to the Server: None

Parameters received from the Server:

Parameter	Data Type	Description
Status	Integer	Integer indication of whether the function was successful.

		0 = Function completed successfully. 4 = Server problems; try again later. Client should delay 15 minutes before trying to contact the server again.
Version	String	Most recent version of client software that is available for download from the server.
NewSoftwareURL	String	URL where the latest client software may be downloaded. The client is designed to ask the user whether the new software should be downloaded and installed. It goes to this URL to download the software, which is in a self-extracting executable file.

2.2 Initialize

The MyDelivery client uses this function to:

- Log into the system
- Tell the server how much free space is on the client computer, and the size of the largest file
- Get the user's Address Book, Signature and SpamControl
- Get the user's First and Last name as supplied during registration
- Get a Key uniquely identifying this user for executing other SOAP functions

Parameters sent to the Server:

Parameter	Data Type	Description
UserID	String	The MyDelivery UserID/password
DeviceCaps		An XML string containing two parameters: "LargestFile" - the size of the largest file permitted on the hard disk of the client's computer, and "FreeDiskSpace" - the total free disk space available for deliveries on

		the client computer.

Parameters received from the Server

Parameter	Data Type	Description
Status	Integer	Integer indication of whether the function was successful. 0 = Function completed successfully. 1 = Not used 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes
ErrorStatus	String	String containing status 3-specific information. To be used for debugging purposes only, since status 3 should never occur in a production client.
UserName	String	A string containing the user's first and last name, separated by a space (as supplied during registration).
AddressBook	Zip-compressed byte array containing an XML string	XML String containing the signature, spam control policy, and address book belonging to this user.
Key	String	String containing a 10-digit key that uniquely identifies this client. This key shall be used in all other SOAP functions to identify this client.

2.3 CSP (ClientStatusPage)

Each client shall execute CSP (ClientStatusPage) from the MyDelivery Server periodically, whose period depends on the overall system load. A typical period will be as fast once every 5 seconds, and could be as slow as a minute or more. It will be specified by a variable, HeartBeat, returned by CSP. One purpose of the CSP is to let the server know that the client is online. It also lets the client know whether the system is up or down, and whether deliveries are pending, or clients are ready for a delivery. CSP plays an important role in error recovery after a communications problem. The client uses it to probe the network to see if it is up and running before executing all other SOAP functions.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	String containing the unique key assigned to this user via Initialize.

Parameters received from the Server

Parameter		Description
Status	String	String containing status for this specific client in the form “N1,N2”

When the client executes CSP, this lets the MyDelivery Server know that the client is online, and for those clients configured to receive deliveries, it lets those clients know something about the status of deliveries. The Server sends back to the client a brief message that may be as short as three ASCII characters. Here is the scheme for the message:
“N1,N2” where:

N1 is a decimal encoding of ServerDown, InvalidKey, DoUpdateStatus, and PendingSender status where the status of each of these is indicated by a Boolean flag (true or false).

N2 is a decimal number representing the HeartBeat, which is the number of seconds the client must wait prior to repeating the execution of CSP.

Here is the relationship between the bit positions of N1 and the five status flags:

```
                     |-----------------------------ServerDown
                          |------------------------InvalidKey
                               |-------------------DoUpdateStatus
                                    |--------------Not Used
                                         |--------PendingSender
N1 bit value        0/1  0/1  0/1  0/1  0/1
N1 bit position      4    3    2    1    0
```

Here are the status indications, their possible values, and meanings:

Key Word	Value	Meaning
ServerDown	1 (True) or 0 (False)	A value of True indicates that the server is Down. The client shall immediately stop doing whatever it is doing, wait 15 minutes, and attempt to recover. If recovery fails, the client shall start from the beginning by executing the Initialize function. A value of False indicates the server is up.
InvalidKey	1 (True) or 0 (False)	A value of True indicates that either another client has logged into the system with the same UserID, or this client has provided an invalid key. This client must notify the user and shut down automatically, stopping all other processes it may be conducting. A value of False indicates no key problem.
DoUpdateStatus	1 (True) or 0 (False)	A value of True indicates that the client should execute UpdateStatus to determine the status of deliveries. This bit is set to False after the sending client executes UpdateStatus.
Not Used		This bit is not currently used.
PendingSender	1 (True) or 0 (False)	A value of True indicates that either another client wants to make a delivery to this client, but is unable to do so because the other client is not in this client's address book. A value of False indicates there are no pending senders.
HeartBeat	Integer >= 5	HeartBeat is the amount of time in seconds that the client should wait before executing the next CSP. The Server will vary this, depending on its total CPU utilization. The Front-end processors are able to throttle their CPU utilization by varying the frequency of CSPs received by the machine.

The following are sample responses sent by the Server back to the client through CSP:
0,5

This is normal status with a heartbeat of 5 seconds:

ServerDown = 0

InvalidKey = 0

DoUpdateStatus = 0

(Not Used) = 0

PendingSender = 0

HeartBeat = 5

4,5

This indicates that client should execute the UpdateStatus function:

ServerDown = 0

InvalidKey = 0

DoUpdateStatus = 1

(Not Used) = 0

PendingSender = 0

HeartBeat = 5

16,5

This indicates the server is down, and the client should go away for 15 minutes:

ServerDown = 1

InvalidKey = 0

DoUpdateStatus = 0

(Not Used) = 0

PendingSender = 0

HeartBeat = 5

2.4 UploadUserInformation

The MyDelivery client uses UploadUserInformation to upload to the MyDelivery Server its address book, which also contains the user signature, and spam control policy.

Parameters sent to the Server

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize.
AddressBook	Zip-compressed byte array containing an XML string	XML String containing the signature, spam control policy, and address book belonging to this user.

Parameters received from the Server

Parameter	Data Type	Description
Status	Integer	Status – Integer indication of whether the function was successful. 0 = Function completed successfully. 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes 5 = Invalid MyDelivery ID in Address Book. Invalid ID's will be listed in ErrorStatus
ErrorStatus	String	Comma-separated list of invalid MyDelivery ID's (if the Status received is "5")

AddressBook – XML String containing the signature, spam control policy, and address book belonging to this user.

It will contain a SIG element, which is a string containing the user's signature.

It will contain a POLICY element, which is a string containing one of two values (BLOCKALL or ADDRESSBOOK) for controlling unwanted receptions:

BLOCKALL – Prevent all receptions

ADDRESSBOOK – Allow receptions only from users listed in the user's address book.

It will contain a WHITELIST section for users permitted to communicate with the user. The WHITELIST contains:

- NUM_ENTRIES – number of entries in the WHITELIST
  - ADDRx – entry for user x
    - FIRST – user's first name
    - LAST – user's last name
    - HANDLE – MyDelivery ID of this user
    - NOTES – up to 256 characters of free text

It will contain a BLOCKED_SENDERS section for users NOT permitted to communicate with the user. This section contains:

- NUM_BLK_ENTRIES – number of entries in the section
  - ADDRx – entry for user x
    - FIRST – user's first name
    - LAST – user's last name
    - HANDLE – MyDelivery ID of this user
    - NOTES – up to 256 characters of free text

Here is a sample address book:

```
<?xml Version="1.0"?>
<MyDelivery>
     <SIG>John Dow</SIG>
     <POLICY>BLOCKALL</POLICY>
      <WHITELIST>
            <NUM_ENTRIES>1</NUM_ENTRIES>
            <ADDR1>
                  <FIRST>Tom</FIRST>
                  <LAST>Jones</LAST>
                  <HANDLE>tomj</HANDLE>
                  <NOTES>tomjones@yahoo.com</NOTES>
            </ADDR1>
      </WHITELIST>
      <BLOCKED_SENDERS>
            <NUM_BLK_ENTRIES>1</NUM_BLK_ENTRIES>
            <ADDR1>
                  <FIRST>John</FIRST>
                  <LAST>Doe</LAST>
                  <HANDLE>jdoe</HANDLE>
                  <NOTES>jdoe@yahoo.com</NOTES>
            </ADDR1>
      </BLOCKED_SENDERS>
```

</MyDelivery>

If a potential sender to a recipient is not in the recipient's address book, and attempts to send a delivery to the recipient, the following will occur:

If POLICY contains BLOCKALL, the MyDelivery server will reject the delivery.

If POLICY contains ADDRESSBOOK, information about the sender will be presented to the recipient, who will be asked whether the sender should be added to the address book and the delivery transaction completed.

2.5 ChangePassword

Users are permitted to change their MyDelivery password through the client when the ChangePassword function is executed.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize.
OldPassword	String	String containing the user's current password. This is the user's current password. If the user sends an incorrect OldPassword, the server will return a Status of 2, and will not store the NewPassword.
NewPassword	String	String containing the user's new password. This is the user's new password. A password must be between 6 and 15 alphanumeric characters (a-z, A-Z, or 0-9).

Parameters received from the Server

Parameter	Data Type	Description

Status	Integer	Integer indication of whether the function was successful. 0 = Function completed successfully. 1 = User logged out. Must execute ChangePassword again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes
ErrorStatus	String	String containing status 3-specific information

2.6 UpdateStatus

The MyDelivery client uses UpdateStatus in several ways:

1. To receive from the MyDelivery server a list of MyDelivery IDs that have recently attempted to send to this user's client, but which cannot send because they are not listed in the user's address book.
2. To receive from the MyDelivery server the status of previous deliveries or attempted deliveries – whether they were successfully delivered, whether they are pending delivery, whether the receiving client is blocking deliveries from this user (anti-spam), or whether they could not be delivered.
3. To tell the server how much free disk space is available on the client for receiving a delivery.

There are three instances when the client executes UpdateStatus:

1. Right after running Initialize. This enables a client to immediately find out whether it has pending deliveries, or other clients are not included in its address book.
2. When CSP indicates a TRUE state in DoUpdateStatus.
3. When there is a major change in FreeDiskSpace on the client computer. The client should keep track of FreeDiskSpace, and when it changes by 10 percent or more since the last Initialize or UpdateStatus, it should run UpdateStatus again, giving the new value to the server.

Parameters sent to the Server

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize.
DeliveryStatusRequest	String	XML string containing a list of DeliveryID's for deliveries previously sent to the MyDelivery Server.
FreeDiskSpace	String	The amount (in bytes) of space available on the client computer for receiving deliveries. The MyDelivery Server can use this number to reject deliveries that exceed the disk space available on a potential recipient computer.

Parameters received from the Server

Parameter	Data Type	Description
Status	Integer	Indicates whether the function was successful. 0 = Function completed successfully. 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes

ErrorStatus	String	String containing status 3-specific information.
DeliveriesReadyForDownload	String	XML string containing a list of pending deliveries. Each list entry contains two items for each pending delivery: DeliveryID assigned to delivery DeliverySize -the size of the delivery.
DeliveryStatus	String	XML string containing the delivery status for those deliveries whose DeliveryID was given in DeliveryStatusRequest

An example of the DeliveryStatusRequest parameter:

```
 <?xml version="1.0" ?>
- <MyDelivery>
 - <DeliveryStatusRequest>
   <NUM_ENTRIES>3</NUM_ENTRIES>
   <ID1>234899009837779</ID1>
   <ID2>982349872277999</ID2>
   <ID3>198098347798739</ID3>
 </DeliveryStatusRequest>
  </MyDelivery>
```

An example of the DeliveryStatus parameter:

```
 <?xml version="1.0" ?>
- <MyDelivery>
 - <DeliveriesReadyForDownload>
 <NUM_ENTRIES>2</NUM_ENTRIES>
 <Delivery1>
   <ID>123989820937</ID>
   <TotalSize> 23399</TotalSize>
 </Delivery1>
 <Delivery2>
   <ID>892839876390</ID>
   <TotalSize>2388399</TotalSize>
</Delivery2>
```

```
 </DeliveriesReadyForDownload>
 </MyDelivery>
```

DeliveryStatus will be one of the following:

Pending – The Delivery is still waiting for the client to initiate its download. The sending client should retain the entire delivery until the recipient has received all of it.
Restart – The delivery should either be restarted or continued. The Pointer parameter indicates where the delivery should recommence.
Success – The Delivery was successfully completed. The sending client may delete the delivery from its hard disk since it is no longer needed.
Failure – The Delivery failed. The sending client should retain the delivery for future action.
Pointer will indicate where the sending client should restart or continue. If 0, the client shall send UploadDeliveryHeader1. If not zero, this indicates the attachment number for resumption of communications.

Example:

```
  <?xml version="1.0" ?>
- <MyDelivery>
  <DeliveryStatus>
  <NUM_ENTRIES>3</NUM_ENTRIES>
<Delivery1>
  <ID>234899009837779</ID>
  <Status>Failure</Status>
 <Pointer></Pointer>
</Delivery1>
<Delivery2>
 <ID>982349872277999<ID>
 <Status>Pending</Status>
<Pointer></Pointer>
</Delivery2>
<Delivery3>
 <ID>198098347798739<ID>
  <Status> Success </Status>
<Pointer></Pointer>
</Delivery3>
<Delivery4>
 <ID>198098347798739<ID>
 <Status>Restart </Status>
<Pointer>25</Pointer>
</Delivery4>
 </DeliveryStatus>
</MyDelivery>
```

2.7 CheckRecipient

The client uses CheckRecipient to find out if a delivery can be made to a specific recipient. This function is executed immediately after the user clicks the “Start” button in the New Delivery dialog box. The results returned by the function allow the delivery to either be processed or rejected immediately. For rejections, the client software shall present the user with a dialog box informing the user of the problem.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize
ReceiverHandle	String	The MyDeliveryID of the intended recipient client.
TotalSize	String	String representation of an int64 number that is an estimate of the total size (in bytes) of all attachments to be sent in this delivery to the recipient.
LargestFile	String	String representation of an int64 number that is the size (in bytes) of the largest file (prior to encryption) to be sent as an attachment in this delivery to the recipient. The MyDelivery Server shall use this to determine whether the recipient can accept files of certain sizes. For FAT16 and FAT32, the maximum file size is $2^{32} – 1$ bytes. For NTFS, the maximum file size is $2^{44} – 64$kb.

Parameters received from the Server

Parameter	Data Type	Description
Status	Integer	Indicates whether the function was successful.

		0 = Function completed successfully. Delivery to the specified recipient may proceed. 1 = User logged out. Must execute Initialize again. 2 = Not used 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes 5 = Recipient has never run MyDelivery. Notify the user through a dialog box: "Recipient needs to run MyDelivery, and has never run it before." 6 = Blocked. The recipient is not allowing reception from this user. Do not attempt a delivery to this recipient. 7 = DiskFull - the recipient does not have adequate hard disk space to accept the delivery from the sending client. 8 = FileSystemOverload – the recipient computer's file system cannot accept the largest file to be sent. 9 = Pending - the recipient does not have the sender in its address book, and it is not blocking this sender. The next time the recipient executes ClientStatusPage, it will receive "PendingSender" status, indicating there is a pending sender. Upon receiving this status, the recipient should execute the UpdateStatus

		function to find out who the pending sender is. 10 = Problem with recipient's MyDelivery ID; either the recipient is not in the sender's address book, or the recipient does not exist.
ErrorStatus	String	String containing status 3-specific information

2.8 StartDeliveryUpload

The client uses StartDeliveryUpload to initiate a single delivery to a recipient. The client must specify the MyDeliveryID of the recipient, and in return receives the DeliveryID of the recipient that it should use for other functions associated with sending a delivery: UploadDeliveryHeader, UploadAttachment, IsDeliveryValid and TerminateDelivery.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize.
ReceiverHandle	String	The MyDelivery ID of the intended recipient client.
TotalSize	String	String representation of an int64 number that is an estimate of the total size (in bytes) of all attachments to be sent in this delivery to the recipient.
LargestFile	String	String representation of an int64 number that is the size (in bytes) of the largest file (prior to encryption) to be sent as an attachment in this delivery to the recipient. The MyDelivery Server shall use this to determine whether the recipient can accept files of certain sizes. For FAT16

		and FAT32, the maximum file size is $2^{32} - 1$ bytes. For NTFS, the maximum file size is $2^{44} - 64$kb.

Parameters received from the Server:

Parameter	DataType	Description
Status	Integer	Indicates whether the function was successful. 0 = Function completed successfully. The client may proceed next with UploadDeliveryHeader. 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes 5 = Not used. 6 = Blocked. The recipient is not allowing reception from this user. Do not attempt a delivery to this recipient. 7 = DiskFull - the recipient does not have adequate hard disk space to accept the delivery from the sending client. 8 = FileSystemOverload - the recipient computer's file system cannot accept the largest file to be sent by the sending MyDelivery. 9 = Pending - the recipient does not

		have the sender in its address book, and it is not blocking this sender. The next time the recipient executes UpdateStatus, it will receive status indicating the pending sender. 10 = Problem with recipient's MyDelivery ID; either the recipient is not in the sender's address book, or the recipient does not exist. 11 = Recipient is offline.
ErrorStatus	String	String containing status 3-specific information.
DeliveryID	String	If the Server determines the delivery to the specified recipient can proceed (Status = 0), it creates the DeliveryID, which is the string representation of an int64 number identifying the delivery. The sending and receiving clients will use this identifier for all functions regarding this specific delivery.

2.9 UploadDeliveryHeader1

The client uses UploadDeliveryHeader1 to upload the header information for a single delivery, where a delivery may consist of zero, one or more file attachments. The UploadDeliveryHeader initiates a delivery, and it is sent once per delivery. It contains general information describing the delivery. For the case where there are no attachments, the delivery contains only a text message. Attachments will be numbered 1, 2, 3, and so on. If there are no attachments, then execution of UploadDeliveryHeader completes the delivery. Otherwise if there are attachments, UploadDeliveryHeader1 must be followed with one or more UploadAttachment executions. The number of attachments is calculated from the total size of all files listed in the Files parameter. The attachment size is fixed at one megabyte (1,048,576 bytes). Files smaller than this size are packed together to create a one megabyte attachment. Files larger than one megabyte are split into one megabyte attachments.

The number of files in the delivery is obtained from the NUM_ENTRIES element in Files (see below).

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize.
DeliveryID	String	A string representing an int64 number that identifies the delivery; obtained from StartDeliveryUpload.
Subject	Byte Array containing a string	Byte array (base-64-encoded by SOAP) containing the text subject.
TextMessage	Byte Array containing a string	Byte array (base-64-encoded by SOAP) containing a text message that accompanies the attached delivery. This may be empty if there is no text message.
Files	Byte Array containing a string	XML string containing the node ID of the file (1, 2, 3, etc.), the names of the encrypted files, the size (in bytes) of each file, and the number of parts required to send each file. It also contains a compression map that describes which attached files have been compressed. Each file may be subdivided into one or more "parts". If the file has only one part, this means the file is completely sent in one attachment. If the file needs to be divided into more than one part, then the parts may be spread out over two or more attachments. For multiple parts, part 1 is the first portion of the file, part 2 follows part 1, part 3 follows part 2, and so on. A single file is the concatenation of its parts. The number of files in the entire delivery is denoted by NUM_ENTRIES in this XML string.

		There are no files if this is an empty string.
Tree	Zip-compressed Byte Array containing a string	Byte array (base-64-encoded by SOAP) containing an always compressed representation of the tree structure. The Tree structure that will be used by MyDelivery will contain a delimiter (‘*’) separated list of values. These values are used to reconstruct the hierarchical information of the attached folders, files and their timestamps. Each line in this file represents a single node, and the lines are separated by a new line character ‘\n’.
CheckCode	Byte Array	Contains a checkcode for verifying the Subject, TextMessage, Files and Tree. The receiving client shall hash these fields to determine whether they match the CheckCode. If there is a match, then the fields were not altered during transmission. If there is not a match, then this function must be repeated until a match is received.

Parameters received from the Server

Parameter	Data Type	Description
Status	Integer	Indicates whether the function was successful. 0 = Function completed successfully. The client should next execute the UploadAttachment function if there are any to be uploaded. 1 = User logged out. Must execute

		Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes
ErrorStatus	String	String containing status 3-specific information.

Files – XML string containing the node ID of the file (1, 2, 3, etc.), the names of the files, the size (in bytes) of each file, and the number of parts required to send each file. It also contains a compression map that describes which attached files have been compressed. Each file may be subdivided into one or more “parts”. If the file has only one part, this means the file is completely sent in one attachment. If the file needs to be divided into more than one part, then the parts may be spread out over two or more attachments. For multiple parts, part 1 is the first portion of the file, part 2 follows part 1, part 3 follows part 2, and so on. A single file is the concatenation of its parts. The number of files in the entire delivery is denoted by NUM_ENTRIES in this XML string. There are no files if this is an empty string.

The Compression map is a base-64-encoded byte array that describes which attached files are compressed. This is the format for the byte array and the correspondence between the file attachment and the bit within the byte:

Bit Position	Attached File
7	1
6	2
5	3
4	4
3	5
2	6
1	7
0	8

If the file is compressed, then the bit is set to 1. If the file is uncompressed, then the bit is set to 0. Three factors are used to determine whether a file gets compressed. First, if the file size is less than 10,240 bytes, the file will not be compressed. Second, if the first 1,048,576 bytes of the file can be compressed, then the entire file will be compressed. Third, if the resulting size of the compressed file is greater than the uncompressed size, the file will not be compressed.

If there is only one attached file, bit 0 corresponds to that file. If there are two attached files, the first attached file corresponds to bit 1, and the second file corresponds to bit 0. The above table is the case for 8 attached files. A second byte will be added if there are 9 through 16 attached files. In this case, the first byte corresponds to files 1 through 8, and the second file corresponds to files 9 through 16. Additional bytes are added for additional attached files.

Once the byte array is created, it is converted to base-64, which results in a text representation of the bytes. In base-64 encoding, four bytes are produced for every three input. The Compression element in the xml-based Files parameter is this base-64 conversion of the byte array.

Example:

```
 <?xml version="1.0" ?>
 <MyDelivery>
 <Files>
  <NUM_ENTRIES>3</NUM_ENTRIES>
  <COMPRESSION>BQ==</COMPRESSION>
  </Files>
  <FILEINFO>1,2489034433,3:3,20909,1:4, 3489034433,4</FILEINFO>
</MyDelivery>
```

In the example above, the first file, has a node id of 1, a size of 2489034433 bytes, and is divided into 3 parts. The second file, has a node id of 3, has a size of 20909, and consists of only one part. The FILEINFO field will be empty if there are no attachments.

The client and server shall each assume that the total number of attachments is the total size of all files divided by 1,048,576. This means that each “part” or transmitted attachment has a maximum size of 1,048,576 bytes.

Tree – Byte array (base-64-encoded by SOAP) containing an always compressed representation of the tree structure. The Tree structure that will be used by MyDelivery will contain a delimiter (‘*’) separated list of values. These values are used to reconstruct the hierarchical information of the attached folders, files and their timestamps. Each line in this file represents a single node, and the lines are separated by a new line character ‘\n’. The following table lists the fields and their data types:

Field	Type	Max Size
NodeID	Integer	32 bits
ParentID	Integer	32 bits
LastWriteTime	FILETIME	64 bits
Attribute	Integer	32 bits
FileName	String	256 bytes

Ordering:

Each line has the fields in the same order as in the table above.

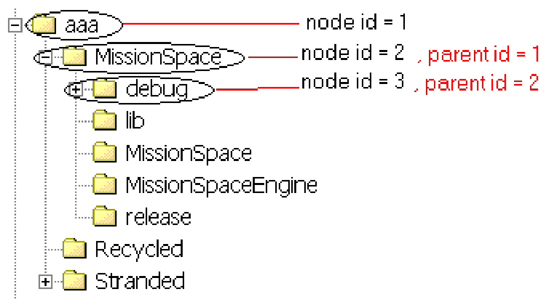
*NodeID*ParentID*LastWriteTime*Attribute*FileName’\n’*

NodeID

The tree structure used by MyDelivery consists of a hierarchy of nodes. Each of these nodes represents either a file or a folder. Each Node has a unique node id (integer) that identifies it. These node ids start at 1 (excluding the reserved root MyDelivery node id of '0') and continuously increase by 1 with no gaps in between the node ids. That is every node id that starts on a new line should be the next higher integer of the previous lines node id.

ParentID:

Each node has a parent id that is the node id of its parent. The root node does not have a parent and its parent id field has the value -1.



LastWriteTime:

The LastWriteTime field has the datatype **FILETIME** that is really an (unsigned __int64). This structure specifies when the file or directory was modified. The **FILETIME** structure is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

```
typedef struct _FILETIME {
  DWORD dwLowDateTime;
  DWORD dwHighDateTime;} FILETIME,
```

Attribute:
The Attribute Field specifies the attribute of the file or folder and has one of these values (defined in winnt.h)

#define FILE_ATTRIBUTE_READONLY	0x00000001
#define FILE_ATTRIBUTE_HIDDEN	0x00000002
#define FILE_ATTRIBUTE_SYSTEM	0x00000004
#define FILE_ATTRIBUTE_DIRECTORY	0x00000010
#define FILE_ATTRIBUTE_ARCHIVE	0x00000020
#define FILE_ATTRIBUTE_DEVICE	0x00000040
#define FILE_ATTRIBUTE_NORMAL	0x00000080
#define FILE_ATTRIBUTE_TEMPORARY	0x00000100
#define FILE_ATTRIBUTE_SPARSE_FILE	0x00000200
#define FILE_ATTRIBUTE_REPARSE_POINT	0x00000400

#define FILE_ATTRIBUTE_COMPRESSED	0x00000800
#define FILE_ATTRIBUTE_OFFLINE	0x00001000
#define FILE_ATTRIBUTE_NOT_CONTENT_INDEXED	0x00002000
#define FILE_ATTRIBUTE_ENCRYPTED	0x00004000

FileName:

The FileName is a string that holds the name of the file on disk and can be MAX_PATH (256) bytes long.

FileSize:
The FileSize is a string containing the size in bytes of the uncompressed original file.

Samples:

Case 1: A folder c:\aaa containing subdirectories and files is attached

The first line always describes the root folder MyDelivery. This pseudo folder is a parent for all attachments included in this delivery. The following is a listing of the tree when a folder c:\aaa is attached. This listing specifies all files and folders that appear below the directory c:\aaa.

```
0*-1*0*16*MyDelivery
1*0*128894147203484931*16*aaa
2*1*128226147780354931*16*MissionSpace
3*2*128226147217876531*16*debug
4*3*128226105885869991*16*images
5*4*128223879320000000*128*cassini.jpg*1001
6*4*128225236300000000*128*ecell.jpg*10011
7*4*128222923260000000*128*exhaust.jpg*10012
8*4*128225240680000000*128*galaxy.jpg*100111
9*4*128222599020000000*128*planet1.bmp*100131
10*4*127892533520000000*128*quitDown.png*10044
11*4*127892533540000000*128*quitOver.png*1005
12*4*127892533540000000*128*quitUp.png*10054
13*4*128223328800000000*128*ssbody.jpg*100433
14*4*128223316660000000*128*ssglass.jpg*1002
15*4*127892533540000000*128*startDown.png*10021
16*4*127892533540000000*128*startOver.png*100123
17*4*127892533540000000*128*startUp.png*10032
18*3*128226133174665811*16*objects
19*18*128226104176404387*128*artifact.x*10033
20*18*128226133174197079*128*planet.x*10043
21*18*128226124963418635*128*skybox.x*1002
22*18*128226106165702995*128*spaceship.x*1005
23*3*128226090650361307*16*sound
24*23*128224689780000000*128*laser.wav*1005
25*23*128224687920000000*128*launch.wav*1003
26*23*128224694020000000*128*meteor.wav*10033
```

```
27*23*128224686500000000*128*travel.wav*100532
28*3*128226127624566443*128*mission.txt*10052
```

Case 2: A folder c:\aaa that contains only files is attached

The first line always describes the root folder MyDelivery. This pseudo folder is a parent for all attachments included in this delivery. The following is a listing of the tree when a folder c:\aaa is attached. This listing specifies all files that appear below the directory c:\aaa.

```
0*-1*0*16*MyDelivery
1*0*128894147203484931*16*aaa
2*1*128223879320000000*128*cassini.jpg*1012
3*1*128225236300000000*128*ecell.jpg*10033
4*1*128222923260000000*128*exhaust.jpg*10044
5*1*128225240680000000*128*galaxy.jpg*10032
6*1*128222599020000000*128*planet1.bmp*10012
7*1*127892533540000000*128*quitOver.png*10043
8*1*127892533540000000*128*quitUp.png*10021
9*1*128223328800000000*128*ssbody.jpg*10044
10*1*128223316660000000*128*ssglass.jpg*1006
```

Case 3: Only files are attached

The first line always describes the root folder MyDelivery. This pseudo folder is a parent for all attachments included in this delivery. The following is a listing of the tree when only files are attached to a delivery like in WebDelivery.

```
0*-1*0*16*MyDelivery*0
1*0*128223879320000000*128*cassini.jpg*103
2*0*128225236300000000*128*ecell.jpg*10034
3*0*128222923260000000*128*exhaust.jpg*10234
4*0*128225240680000000*128*galaxy.jpg*1054
5*0*128222599020000000*128*planet1.bmp*10053
6*0*127892533540000000*128*quitOver.png*10022
7*0*127892533540000000*128*quitUp.png*100223
8*0*128223328800000000*128*ssbody.jpg*1002
9*0*128223316660000000*128*ssglass.jpg*10023
```

2.10 RequestToUpload

The client uses RequestToUpload to get permission from the server to upload an attachment via UploadAttachment. RequestToUpload is executed at least once for each UploadAttachment. If the 1 megabyte of server memory is not available (i.e., the receiving client has not downloaded the previous attachment), this function returns a nonzero value in the Wait parameter. This value will be the number of milliseconds the sending client must wait prior to re-executing the RequestToUpload function. The sending client must keep repeating RequestToUpload until the value returned in the Wait parameter is 0, after which it is permitted to execute UploadAttachment.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize.
DeliveryID	String	String identifying the delivery. Represents an int64 data type.
Attachment	Integer	The attachment number (1, 2, 3, …)

Parameters received from the Server:

Parameter	Data Type	Description
Status	Integer	Integer indication of whether the function was successful. 0 = Function completed successfully. 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes 5= Receiver is offline 6= Resync: The receiving client has indicated through DeliveryStatus that the download failed. the sending client must execute

		UpdateStatus to determine the new starting point for sending the delivery.
Wait	Integer	Number of milliseconds the client must wait before repeating this function. A value of 0 indicates the client may proceed immediately with UploadAttachment.

2.11 UploadAttachment

The client uses UploadAttachment to upload an attachment of a delivery. This function may be executed more than once, if multiple attachments must be sent. All attachments stored at the MyDelivery Server will be one megabyte (1,048,576 bytes) in size, unless it is the final attachment in a delivery, in which case it can be from one byte to one megabyte in size. This function permits the client to upload variable-sized DIME attachments.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize.
DeliveryID	String	String identifying the delivery. Represents an int64 data type.
Attachment	Integer	The attachment number (1, 2, 3, …)
CheckCode	String	Hash checkcode for the attachment.
DIMEAttachment	Byte stream	File part attached to the message using DIME.

Parameters received from the Server:

Parameter	Data Type	Description
Status	Integer	Integer indication of whether the

		function was successful. 0 = Function completed successfully. 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes 5= Receiver is offline 6= Resync: The receiving client has indicated through DeliveryStatus that the download failed. the sending client must execute UpdateStatus to determine the new starting point for sending the delivery.
ErrorStatus	String	String containing status 3-specific information.
Wait	Integer	Number of milliseconds the client must wait before repeating this function. A value of 0 indicates the Server has stored the attachment, and the client may proceed with the next RequestToUpload.

2.12 IsDeliveryValid

The client uses IsDeliveryValid to determine whether a delivery is still available for reception or delivery. The function also returns status indicating whether a delivery has been terminated.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize.
DeliveryID	String	String identifying the delivery (represents an int64 data type).

Parameters received from the Server

Parameter	Data Type	Description
Status	Integer	Indicates whether the function was successful. 0 = Function completed successfully; Delivery is valid 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again later
ErrorStatus	String	String containing status 3-specific information.
DeliveryValid	Boolean	TRUE if the delivery is still available on the server, or FALSE if not available.
DeliveryTerminated	Boolean	TRUE if the delivery was terminated, or FALSE if not terminated.

2.13 DownloadDeliveryHeader1

The client uses DownloadDeliveryHeader1 to download the header information for a single delivery, where a delivery may consist of zero, one or more attachments.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize.
DeliveryID	String	String identifying the delivery (represents an int64 data type)

Parameters received from the Server

Parameter	Data Type	Description
Status	Integer	Indicates whether the function was successful. 0 = Function completed successfully. 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes 5= Sending client is offline
ErrorStatus	String	String containing status 3-specific information.
DeliveryTerminated	Boolean	Indicates whether the sending client has terminated the delivery (TRUE) or not (FALSE).

SenderFirstName	String	The sender's first name
SenderLastName	String	The sender's last name
SenderHandle	String	The sender's MyDelivery ID
Subject	Byte Array containing a string	The text subject.
TextMessage	Byte Array containing a string	The text message
Files	String	XML string describing the attached files and/or folders. If the string is empty, this indicates there are no attachments.
Tree	Byte Array containing a string	Byte array (base-64-encoded by SOAP) containing an always compressed representation of the tree structure. This tree structure contains information such as the folders; hierarchy, display names, extensions, timestamps and other file attribute information.
CheckCode	Byte Array	A hash checkcode for the Subject, TextMessage, Files and Tree.

2.14 DownloadAttachment

The client uses DownloadAttachment to download a specific attachment for a single delivery, where a delivery may consist of zero, one or more one megabyte (1,048,576 bytes) attachments. This function may be executed more than once, if multiple attachments must be received.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize
DeliveryID	String	Identifies the delivery. Represents

		an int64 data type.
Attachment	Integer	The attachment number (1, 2, 3, …). Attachments shall be downloaded in order, starting with 1, then 2, etc.

Parameters received from the Server:

Parameter	Data Type	Description
Status	Integer	Indicates whether the function was successful. 0 = Function completed successfully. 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again in 15 minutes 5=Sender is offline; cannot download the attachment
ErrorStatus	String	Contains status 3-specific information.
MaxAttachmentReady	Integer	The number of the highest numbered attachment on the server available for download. The client should use this number, if possible, instead of executing IsAttachmentReady.
Wait	Integer	Number of milliseconds the client must wait before repeating this function. If 0, this indicates the specified attachment is available as a DIME attachment in this function. If non zero, the specified attachment is not available, and the

		client must repeat the function after waiting.
CheckCode	String	Hash checkcode for the attachment.
DeliveryTerminated	Boolean	Indicates whether the sending client has terminated the delivery (TRUE) or not (FALSE). If the delivery has been terminated, the receiving client shall delete whatever it has received from the delivery.
DIMEAttachment	Byte stream	File part attached to the message using DIME.

2.15 DeliveryStatus

The client uses DeliveryStatus to tell the server all of the attachments for a delivery were correctly received. Once the server receives a successful delivery indication, it can notify the sending client that the delivery was successfully completed.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	The unique key assigned to this user via Initialize
DeliveryID	String	Identifies the delivery. Represents an int64 data type.
DeliveryResult	String	“Success” if the delivery was successfully received. “Failure” if the delivery was not properly received.
RestartPointer	Integer	If DeliveryResult is "Failure", RestartPointer indicates where the sending client should restart the delivery 0 = Restart with UploadDeliveryHeader > 0 = The attachment number where UploadAttachment should

		be restarted

Parameters received from the Server

Parameter	Data Type	Description
Status	Integer	Indicates whether the function was successful. 0 = Function completed successfully. 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again later
ErrorStatus	String	Contains status 3-specific information.

2.16 TerminateDelivery

The client uses TerminateDelivery to inform the server that it is killing one or more deliveries.

Parameters sent to the Server:

Parameter	Data Type	Description
Key	String	Contains the unique key assigned to this user via Initialize
TerminationList	String	XML string of DeliveryID's to be terminated.

Parameters received from the Server

Parameter	Data Type	Description

Status	Integer	Indicates whether the function was successful. 0 = Function completed successfully. 1 = User logged out. Must execute Initialize again. 2 = Invalid MyDelivery ID or password 3 = Serious problem; details given in ErrorStatus 4 = Server problems; try again later
ErrorStatus	String	Contains status 3-specific information.

Example of the TerminationList parameter:

```
<?xml version="1.0"?>
- <MyDelivery>
<Terminate>
<NUM_ENTRIES>2</NUM_ENTRIES>
<ID1>123345677</ID1>
<ID2>345345247</ID2>
 </Terminate>
</MyDelivery>
```

3.0 MyDelivery Server Architecture

The MyDelivery server can be configured as one or more machines, depending on the amount of redundancy the administrator desires, and the number of users. The cheapest solution is to run the MyDelivery server as a single machine, which should be able to handle several hundred online users. This is a list of Windows services running on a single-machine server:

1. mdClientListService
2. mdFileXFERService
3. mdFlusher
4. mdMonitor
5. mdPinger
6. mdClientManager
7. SQL Server Express

For a large production system capable of handling thousands of users, rack-mounted blade servers are recommended. Each server runs Windows Server 2003 or 2008, and the Internet Information Server (IIS) is the HTTP server that interfaces with the Internet. Figure 1.3 illustrates a multi CPU system that uses Microsoft's Network Load Balancing for the Front-End Processors. These are the services running on each Front-End Processor:

1. mdClientListService
2. mdFileXFERService
3. mdFlusher
4. mdMonitor
5. mdPinger
6. Secondary ClientManager (running on one Front-End Processor)

These are the services running on the ClientManager processor:

1. Primary ClientManager
2. SQL Server Express acting as a Witness for SQL Server database mirroring

Primary SQL Server Database Machine

1. SQL Server 2005 or SQL Server 2008 (for database mirroring)

   or SQL Server Express (without database mirroring)

Secondary SQL Server Database Machine (for database mirroring)

1. SQL Server 2005 or 2008

3.1 MyDelivery Windows Service: mdClientListService

The mdClientListService is a Window service that runs on each of the Front-End Processors. It maintains ClientsOnlineArray, which is a memory cache version of a table in the SQL Server database: ClientsOnline. Its main function is to reduce load on the database machine by having the equivalent table available in the cache memory of each Front-End Processor. In addition to containing information about online client status, the table contains information intended to signal a client of a change in delivery status when it executes the CSP (ClientStatusPage) function.

The ClientsOnlineArray (and the SQL Server's ClientsOnline table) maintains a list of numbers identifying each online MyDelivery user, where each user is assigned a unique random ID. A MyDelivery client is online if it executes the CSP function regularly. Each time CSP is executed, an entry is made for that user in the ClientsOnlineArray. This service is called remotely via .NET Remoting from any computer in the system using TCP Channel via port 9080. It is set up to run a "Singleton" object, which means there is only one copy of the object created for all clients that use this service. The name of the object it runs is called ClientList. Periodically another Windows Service, mdClientManager, queries the mdClientListService running on each Front-End Processor to get its current list of all online clients. Once the mdClientManager gets the list from a machine, that machine's online client list is emptied. If a user's client is shut down, it no longer periodically executes CSP, and the entry for that user will no longer be found in the mdClientListService. Then, when the mdClientManager finds no entry for a particular user, it will assume that this user is offline.

The ClientList object exposes several important functions via .NET Remoting:

1. CheckClient - Allows ClientStatusPage (CSP) to check the status of a client, and report the client as being online to mdClientManager. Periodically triggers refreshing the ClientsOnlineArray.

2. AddClient - Allows a remote program to tell the mdClientListService to add a client to its ClientsOnlineArray. This is executed when a user logs into MyDelivery (via the Initialize SOAP function)

3. DeleteClient - Allows a remote program (mdClientManager) to tell the mdClientListService to remove a client from its ClientsOnlineArray.

4. ChangeClient - Allows a remote program to tell the mdClientListService to change the status of a client in its ClientsOnlineArray.

5. GetClients - Called by the mdClientManager to get a list of identifying numbers for all clients that have reported to be online.

ClientList uses a SyncLock to ensure that its functions run exclusively of each other. It is not possible for them to conflict. Furthermore, since mdClientListService is running ClientList as a Singleton object, only one ClientList object is actually created, which means there is only one ClientListArray created in each Front-End Processor.

3.2 MyDelivery Windows Service: mdFileXFERService

The mdFileXFERService is a Windows Service that is installed and run automatically upon boot up on all Front-End Processors. It provides a variety of functions for an external client to transfer portions of deliveries to and from the server. The mdFileXFERService can be called remotely via .NET Remoting from any computer in the system using TCP Channel via port 9082. It is set up to run a "SingleCall" object, which means there a copy of the object it controls is created for each calling client. The name of the object it runs is called FileXFER.

The FileXFER object offers memory-based transfer functions:

1. AddDeliveryHeader – Allows a remote process to add a delivery header to a memory-based collection of delivery headers. Each header consists of a MyDeliveryID, Subject, TextMessage, Files, Tree and CheckCode.

2. GetDeliveryHeader – Allows a remote process to get the delivery header for a specified MyDeliveryID.

3. AddAttachment – Allows a remote process to add a 1 megabyte attachment and corresponding CheckCode to a memory-based collection of delivery attachments.

4. GetAttachment – Allows a remote process to get a 1 megabyte attachment and corresponding CheckCode from the memory-based collection of delivery attachments.

5. DeleteDelivery – Allows a remote process to delete the header and attachment for a specific MyDeliveryID from the memory-based collections of delivery headers and attachments.

6. CheckExist – Allows a remote process to determine whether a delivery header or attachment for a specific MyDelivery is in the memory-based collection of headers and attachments.

In addition to memory-based transfer functions, this service provides a disk-based transfer function:

FileIn –Allows a remote process (the mdClientManager) to send an HTML file named "mdStatus.htm" to the server's hard disk. This file lists the status of the overall MyDelivery server system, and is accessible over the Internet. It allows a remote administrator to monitor the health of the MyDelivery system.

mdFileXFERService is useful because it makes it possible to store or access a delivery on a computer other than the one on which it is received. This overcomes a problem that arises due to Network Load Balancing, which creates the scenario where files may not be stored on a computer with which a MyDelivery client is communicating. It easily allows software running on one computer to access files on another computer via .Net Remoting.

3.3 MyDelivery Windows Service: mdMonitor

The mdMonitor service is a Windows Service that is installed and run automatically upon boot up on all Front-End Processors. It has three functions for monitoring the machine on which it is running:

Every two minutes it monitors the CPU utilization of the machine, and enters this number in the SQL Server database. One of the factors contributing to high CPU utilization is the CSP function, if there are hundreds of remote user clients executing it periodically. The utilization can be decreased by slowing down the heartbeat (period of the CSP), or increased by speeding up the CSP period. mdMonitor decides whether to increase or decrease the heartbeat, and this is reflected in the heartbeat parameter sent to the client via the CSP. If there is more than one Front-End Processor used in the MyDelivery server system, each Front-End Processor will individually control its own heartbeat.

Once every 24 hours mdMonitor removes web server log files from the server's hard disk that are older than 7 days.

It checks the mdStatus.htm file on the current server to see if mdClientManager has updated it recently. If this file has not been updated, this indicates that mdClientManager is down, and the remote administrator must be notified. The notification is made by mdMonitor, which creates an mdStatus.htm file containing the appropriate error status.

3.4 MyDelivery Windows Service: mdFlusher

The mdFlusher service is a Windows Service that is installed and run automatically upon boot up on all Front-End Processors. Its purpose is to allow a remote administrator to control two parameters used by the system, and each parameter is stored temporarily in system Cache. This service can be called remotely via .NET Remoting from any computer in the system using TCP Channel via port 9081. The only process in the MyDelivery system that calls mdFlusher is mdClientManager. It is set up to run a “SingleCall” object, which means there a copy of the object it controls is created for each calling client. The name of the object it runs is called CacheData.

The CacheData object has one function: UpdateNow. Depending on the parameter sent via UpdateNow, CacheData either modifies a file called flush.clientversion, or a file called flush.serverstatus. When either of these files is modified, a parameter stored in system cache is subsequently updated.

Flush.serverstatus is used by the system to maintain a current copy of ServerStatus in Cache. This has a value of 1 (if the system is up) or 2 (if the system is down). The only way to bring the system up or down is through a web page named Syscontrol.aspx (a password-controlled function). This function uses mdFlusher when the ServerStatus variable stored in Cache on all Front-End Processors needs to be changed (system status goes from up to down or vice versa).

Flush.Clientversion is used by the system to maintain the current version number of the latest client. When a remote client executes GetVersion, this function goes to cache memory to get the version value, rather than to the SQL Database server. This saves execution time on the SQL server.

3.5 MyDelivery Windows Service: mdPinger

The mdPinger service is a Windows Service that is installed and run automatically upon boot up on all Front-End Processors. It acts in a way similar to a ping server: it just responds to a remote request (or fails to respond). It serves to let a remote processor know if the server on which mdPinger is running is actually up. If mdPinger responds, this indicates the server is alive; otherwise, if mdPinger does not respond, this indicates the server is down.

The mdPinger service can be called remotely via .NET using TCP Channel via port 9084. It is set up to run a “SingleCall” object, which means there a copy of the object it controls is created for each calling client. The name of the object it runs is called Pinger. This function allows the mdClientManager service to determine whether a Front-End Processor is up. It can take the Front-End Processor out of service if mdPinger fails to respond.

3.6 MyDelivery Windows Service: mdClientManager

The mdClientManager runs 24/7, and is active on one or perhaps two computers in the MyDelivery Server system. If the MyDelivery Server consists of only one processor, then the mdClientManager runs on it. If the MyDelivery Server consists of at least one Front-End Processor and a Client Manager, then a primary mdClientManager runs on the Client Manager machine, and a secondary mdClientManager runs on any Front-End processor as a backup in case the Client Manager machine goes down. The mdClientManager does three main things:

Once every two hours it cleans up deliveries. Users are required to complete a delivery. If mdClientManager finds an incomplete delivery (from the SQL Server Deliveries table), and that either sending or receiving client has not serviced a delivery within a time period called the "Window", then mdClientManager uses the mdFileXFERService function DeleteFolder to delete the folder and files from the appropriate Document Exchange Server, and updates the Deliveries table in the SQL server database. The "Window" is permitted to vary from 1 hour to 336 hours. It is set by the system administrator through the syscontrol.aspx web page.

Once every three minutes it updates lists of clients that are online. The mdClientManager uses the Processors table in the SQL server database to get a list of all active Front-End Processors. Once every three minutes it gathers a list of online clients from each Front-End Processor, which it uses to create and maintain a master list of all online clients. It checks the master list to determine which clients are online (executing CSP every heartbeat, typically 5 seconds) and which are offline (not executing a CSP for three minutes). The mdClientManager determines when a client goes physically offline, and makes a note of it. This is done when the mdClientManager contacts the mdClientListService at each Status Server computer. When any client status changes, mdClientManager updates the ClientsOnline table in the SQL server database. It notifies all Front-End Processors of changes in the ClientsOnline table only when a client is physically offline. Each Status Server maintains a cache version of ClientsOnline, called ClientsOnlineArray, which it uses for servicing ClientStatusPage. When mdClientManager notifies each Front-End Processor of changes in ClientsOnline, the Front-End Processor's mdClientListService updates its cache version from the SQL Server database. The notifications of changes in the ClientsOnline table through the mdFlusher service, which runs on each Front-End Processor. The mdFlusher service makes a change to the flush.clientsonline file, residing on the machine's hard disk. The ClientsOnlineArray, stored in Cache on the server, is a representation of the ClientsOnline table in the SQL Server database. The ClientsOnlineArray is removed automatically from Cache upon the change to the flush.clientsonline file, and must be refreshed from the ClientsOnline table.

Every ten minutes it monitors system status and produces a web page named mdstatus.htm that it transfers to each Front-End Processor, for remote system monitoring over the Internet. It checks a few basic operations in the MyDelivery Server system to determine whether it is running properly. First, it queries the database server to get a list of all Front-End Processors. If the query to the database server fails, the SystemMonitor will not send a status page, mdstatus.htm, to the web servers running on the

Front-End Processors. Then it attempts to get the default web page from each Front-End Processor. Any problems are noted in the mdstatus.htm file it creates and distributes to each Front-End Processor.

Primary-Secondary mdClientManager Interaction

For redundancy, there may be two ClientManagers used in the system: a Primary mdClientManager and a Secondary mdClientManager. The Primary ClientManager will run on the mdprod-manager processor in the production system. The way a mdClientManager knows whether it is classified as a Primary or Secondary ClientManager will be through the

c:\bin\mydelivery.ini file. The computer where the Primary mdClientManager runs will

contain the entry: ClientManager:=Primary . The computer where the Secondary ClientManager runs contains the entry: ClientManager:=Secondary . When the mdClientManager service starts up, it shall read the mydelivery.ini file to find out whether it is a Primary or Secondary mdClientManager.   When the Primary mdClientManager finishes executing its ten-minute system check, it updates the ClientManagerAccess variable in the SystemInfo table, with the current DayTime. If for some reason the Primary mdClientManager is not functioning, this variable will not get updated.

The Secondary ClientManager in each of its timer functions shall check the ClientManagerAccess variable in the SystemInfo table. If the value is off by more than 30 minutes, it shall execute its system checks. It will never update the value of the ClientManagerAccess variable, because only the Primary mdClientManager does that. When the Primary ClientManager comes back online, it will recommence updating the ClientManagerAccess variable with the current DayTime. At that time the Secondary The mdClientManager service shall stop further work, and remain as a potential backup.

4.0. Server Setup

This section details the steps necessary for setting up a MyDelivery server for using HTTP (non-SSL) communication. Slight changes to the procedure are required if HTTPS (TLS or SSL) is to be used. It is possible to construct a low-cost MyDelivery server using only one machine. This machine houses the MyDelivery website, SQL Server Express, and all MyDelivery web services. In simulated testing, we believe that a single server can handle several hundred simultaneous users, which is good enough for small organizations. We have a created a test server housing all MyDelivery server functions and found that it works well.

It is also possible to create a highly redundant, very reliable, but expensive MyDelivery Server system. We have also created one of these systems, consisting of ten rack-mounted processors. For large organizations requiring thousands of MyDelivery users, this server system consists of three categories of machine:

Front-End Processor – 1 to 32 servers in a Network Load Balanced (NLB) cluster that handle status requests from clients. If NLB is used, then each Front-End Processor shall have two Ethernet ports, one of which is used for front-end communication with the outside world, and the second which shall be used for internal communication with other MyDelivery processors. Rack-mounted blade servers are recommended for these machines

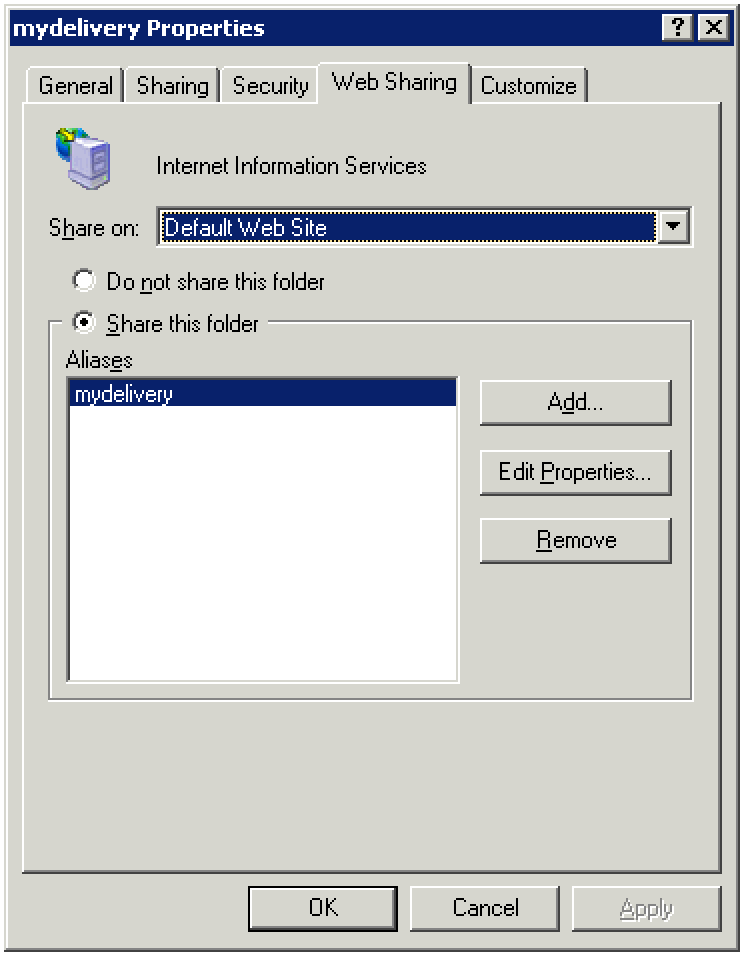
Primary/Secondary Database Server – Two rack-mounted blade servers running SQL Server 2005 or SQL Server 2008. One is the primary server and the second machine is a standby backup.

ClientManager Server – one rack-mounted blade server that runs the Client Manager. It also runs a SQL Server Express Witness for database mirroring. The Witness helps to coordinate automatic failover for the primary and secondary database SQL servers.

- Install Windows Server 2003 Standard Edition on all machines. We have not tested a configuration using Windows Server 2008, but this should work equally well as Windows Server 2003.
- Download and install Microsoft security updates on all machines.
- There are three website folders required for each Front-End Processor:
    1. MyDeliveryWeb – This folder contains a website explaining MyDelivery. It also allows a user to register to use MyDelivery, and download the client software. Our test system uses this folder as the default home folder for the website. Most pages for the distribution MyDeliveryWeb have been deleted, as they pertained only to our original beta test. The only ones that remain are the ones for user registration and system monitoring.
    2. MyDelivery – This folder contains the MyDelivery SOAP web services for communication with MyDelivery clients.

3. NonSSL – This is used only for servers that use HTTPS (secure communication via TLS or SSL). Any user attempting to access the server via HTTP using a browser will be directed via this folder to an HTTPS connection to MyDelivery.

- For each Front-End Processor, use Windows Explorer and right-click on the mydelivery folder name. Go to Sharing and Security. Go to Web Sharing. Share the MyDelivery folder as "mydelivery." Share the MyDeliveryWeb folder as mydeliveryweb. Share the NonSSL folder as nonssl.


mydelivery Properties

General | Sharing | Security | Web Sharing | Customize

Internet Information Services

Share on: Default Web Site

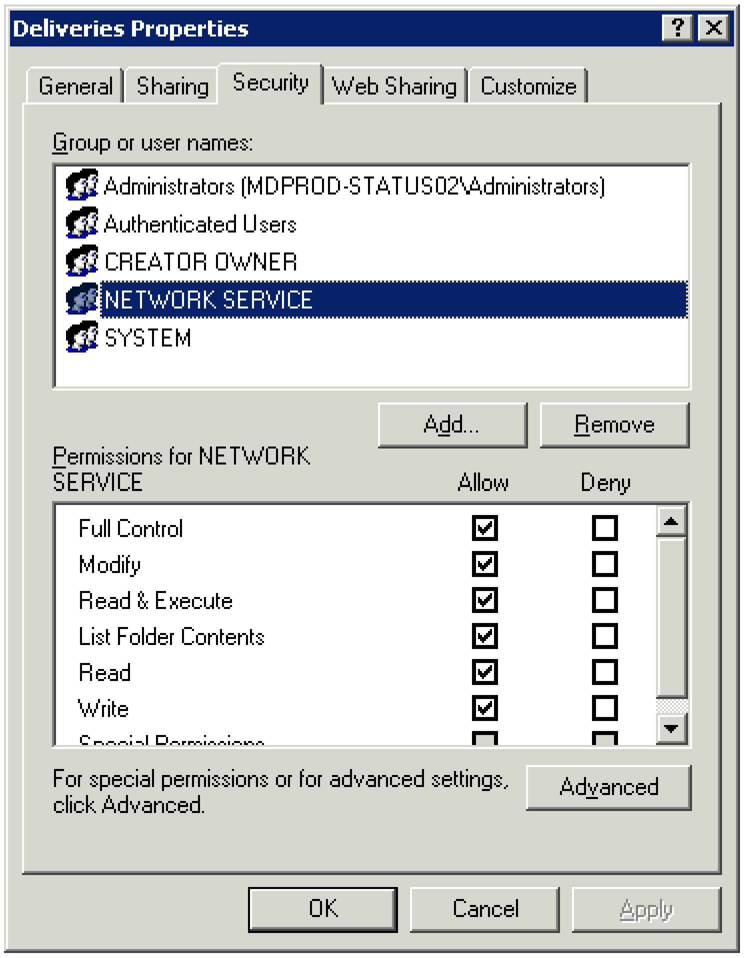
( ) Do not share this folder

(●) Share this folder

Aliases

mydelivery

- Create c:\deliveries folder on all Front-End Processors. Give Network Service full control over the c:\deliveries folder created in the last step. Use the Windows Explorer for this step. Right click on the folder name, and select Sharing and Security. Click the Security tab in the dialog box.


Deliveries Properties

General | Sharing | Security | Web Sharing | Customize

Group or user names:

- Administrators (MDPROD-STATUS02\Administrators)
- Authenticated Users
- CREATOR OWNER
- NETWORK SERVICE
- SYSTEM

Add... Remove

Permissions for NETWORK SERVICE

	Allow	Deny
Full Control	✓	
Modify	✓	
Read & Execute	✓	
List Folder Contents	✓	
Read	✓	
Write	✓	
Special Permissions		

For special permissions or for advanced settings, click Advanced.

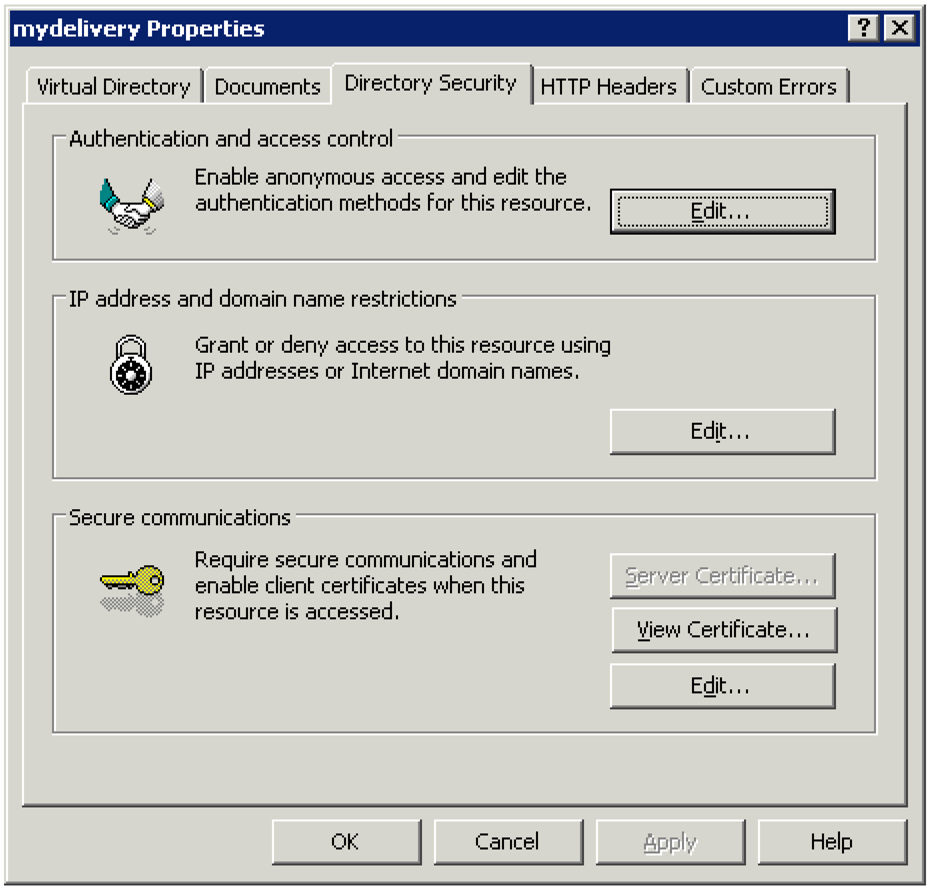
Advanced

OK Cancel Apply

- Install IIS and ASP.NET on the server. To do this, go to Manage Your Server. Then select "Add or Remove a role." In the "Configure Your Server Wizard", click Next. Select a role to add: Application Server (IIS, ASP.NET). Enable ASP.NET. Insert the Windows 2003 (or 2008) Server Standard Edition CD where requested. Both ASP.NET versions 1.0 and 2.0 should be allowed.

- Use IIS Manager to enable ASP.NET version 3.5 on the default web server. You may use version 4.0 or higher if you have it installed. This will permit it to run on the MyDelivery web site. Make sure that the Network Service has the security privilege to use ASP.NET version 3.5 (look at aspnet_isapi.dll in the ASP.NET 2 folder). If not, reinstall ASP.NET version 2 and make sure NetWork Service has the right level of access.

- Use IIS Manager to allow Anonymous access to the MyDelivery, MyDeliveryWeb and NonSSL web sites on all Front-End Processors. In IIS Manager, right-click on mydelivery, then select Properties. Click on Directory Security.


mydelivery Properties

Virtual Directory | Documents | Directory Security | HTTP Headers | Custom Errors

Authentication and access control

Enable anonymous access and edit the authentication methods for this resource.

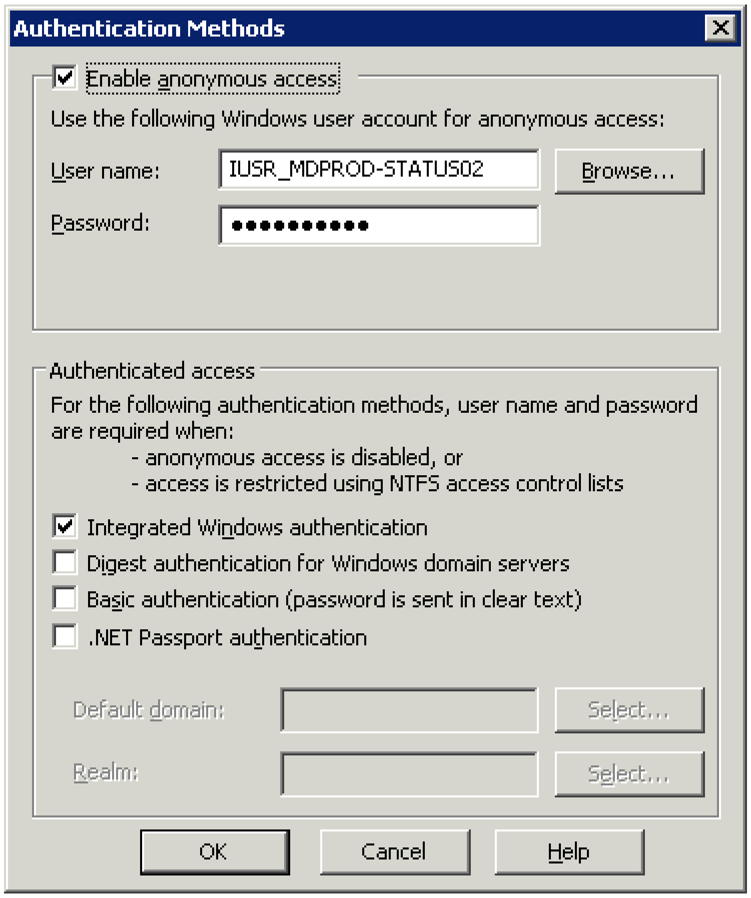
IP address and domain name restrictions

Grant or deny access to this resource using IP addresses or Internet domain names.

Secure communications

Require secure communications and enable client certificates when this resource is accessed.

- Under Authentication and access control, click Edit …


Authentication Methods

[✓] Enable anonymous access

Use the following Windows user account for anonymous access:

User name: IUSR_MDPROD-STATUS02

Password: ●●●●●●●●●●●

Authenticated access

For the following authentication methods, user name and password are required when:
- anonymous access is disabled, or
- access is restricted using NTFS access control lists

[✓] Integrated Windows authentication

[ ] Digest authentication for Windows domain servers

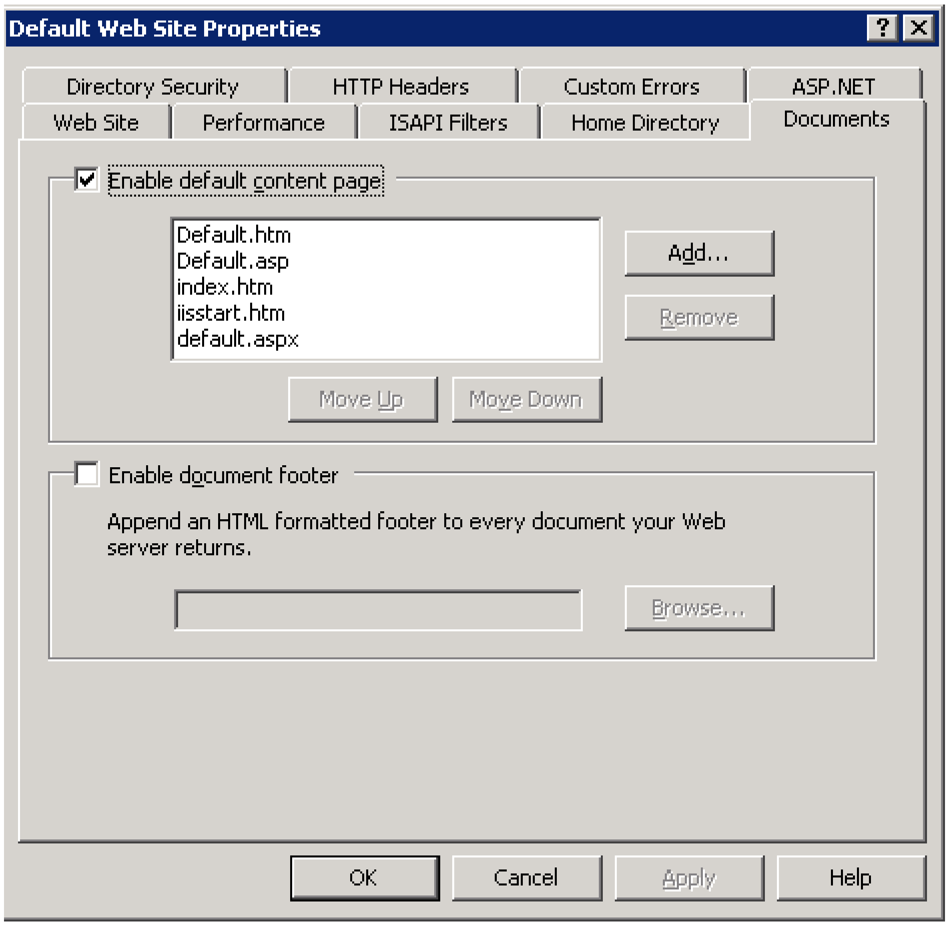
[ ] Basic authentication (password is sent in clear text)

[ ] .NET Passport authentication

Default domain:

Realm:

- Check the box that says "Enable anonymous access". Do not enter the user name in the picture above; use the one automatically suggested by IIS.
- In IIS Manager, disable logging. This maximizes system speed.
- In IIS Manager on the MyDeliveryWeb web site, go to Documents. Add default.aspx to the list of default page contents for MyDeliveryWeb. Eliminate all the other default pages. Ignore this step for the MyDelivery and NonSSL web sites.


Default Web Site Properties

Documents

☑ Enable default content page

Default.htm
Default.asp
index.htm
iisstart.htm
default.aspx

☐ Enable document footer

Append an HTML formatted footer to every document your Web server returns.

- Create a c:\bin folder on each server. Make this folder a part of the Path for the user, so that programs located in this folder can be executed from any folder. This folder shall contain several files:

    a. Mydelivery.ini. The mydelivery.ini file is used by MyDelivery Windows services to contact the proper SQL Server. It contains three parameters: myServer, domain, and CLientmanagaer, which are used by the system software. Here is an example:

        i. myServer:=Server=db01.mycompany.com\SQL2005; Failover Partner=db02.mycompany.com\SQL2005;Database=MyDelivery;Trusted_Connection=yes;

        ii. DOMAIN:=mycompany.com

        iii. Clientmanager:=Primary

    b. Installutil.exe This is a Microsoft utility used for installing MyDelivery web services. Copy it to the c:\bin folder.

c. Aspnet_setreg.exe This is a Microsoft utility used to create a registry-based UserID and password for accessing the MyDelivery SQL database server. Copy it to the c:\bin folder.

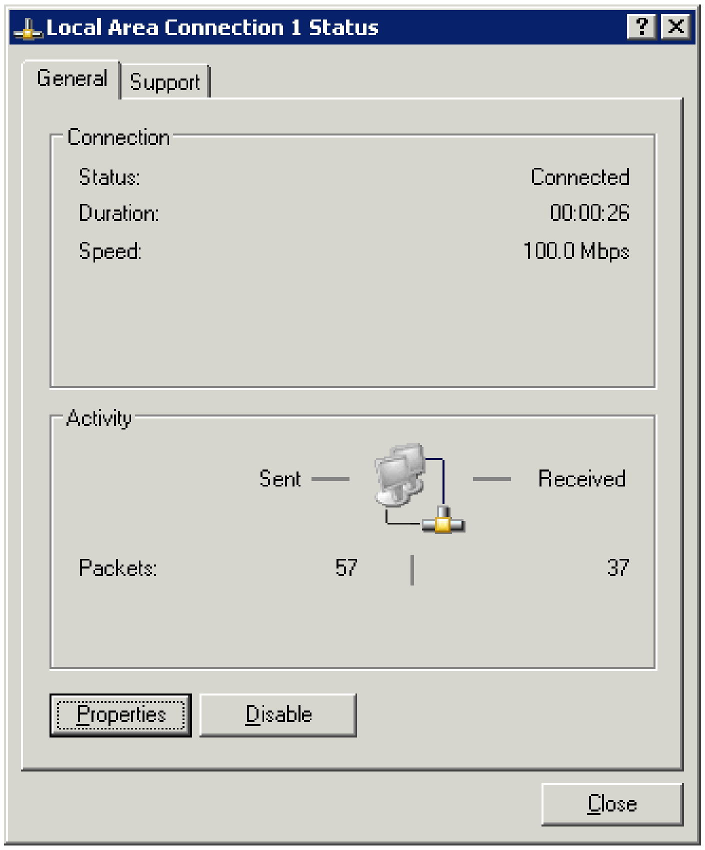
d. Mdsetup.bat – This is a batch file for installing MyDelivery web services.

- Install Web Service Enhancements 1.0 runtime (WSE 1.0) on the computer. This is available as a download from Microsoft. It provides SOAP DIME attachment communication compatible with the SOAP3 toolkit used by the MyDelivery client.

- Add the Anonymous user to allow access to c:\windows\Microsoft.NET\Framework (and the folders/files beneath it).

- If more than one Front-End Processor is used, then the multiple configuration shall use Network Load Balancing (NLB). Set up NLB from the ClientManager Server. For NLB to work properly, all Front-End Processors, which have two Ethernet ports each, will be accessible through a common URL, which for this example is mydelivery.nlm.nih.gov. Each machine has two Ethernet ports: one public and one private. The private port should be the only one initially configured and running after the operating system is installed. The public ports will be configured in the nest steps.

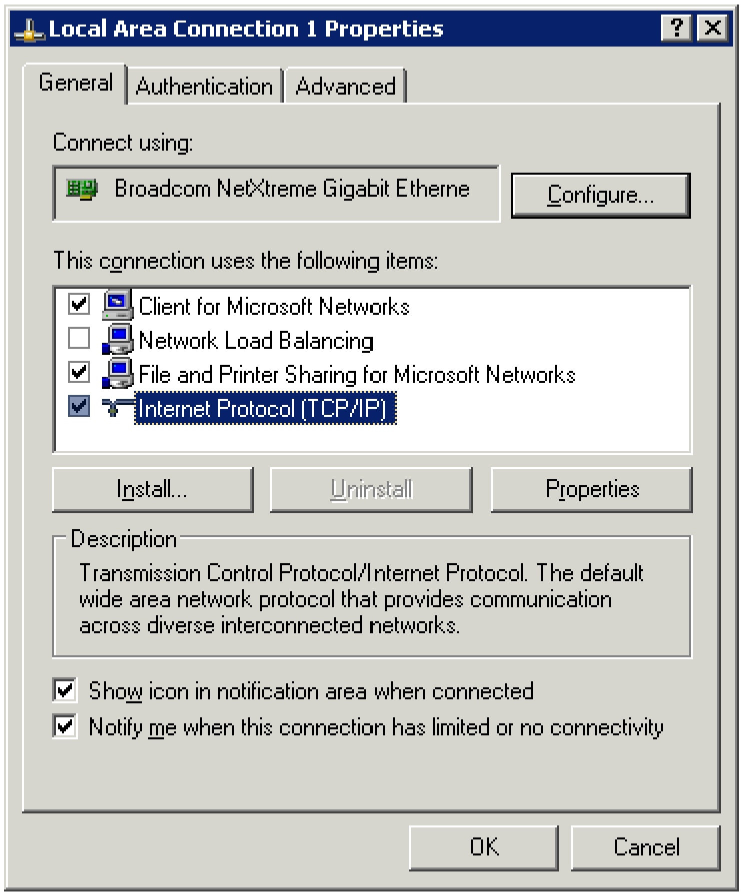
- For the public port, examine the network status:


Local Area Connection 1 Status

Connection
Status: Connected
Duration: 00:00:26
Speed: 100.0 Mbps

Activity
Sent — Received
Packets: 57 | 37

- Click the Properties button, and get the next dialog box:



Connect using:

Broadcom NetXtreme Gigabit Etherne

This connection uses the following items:

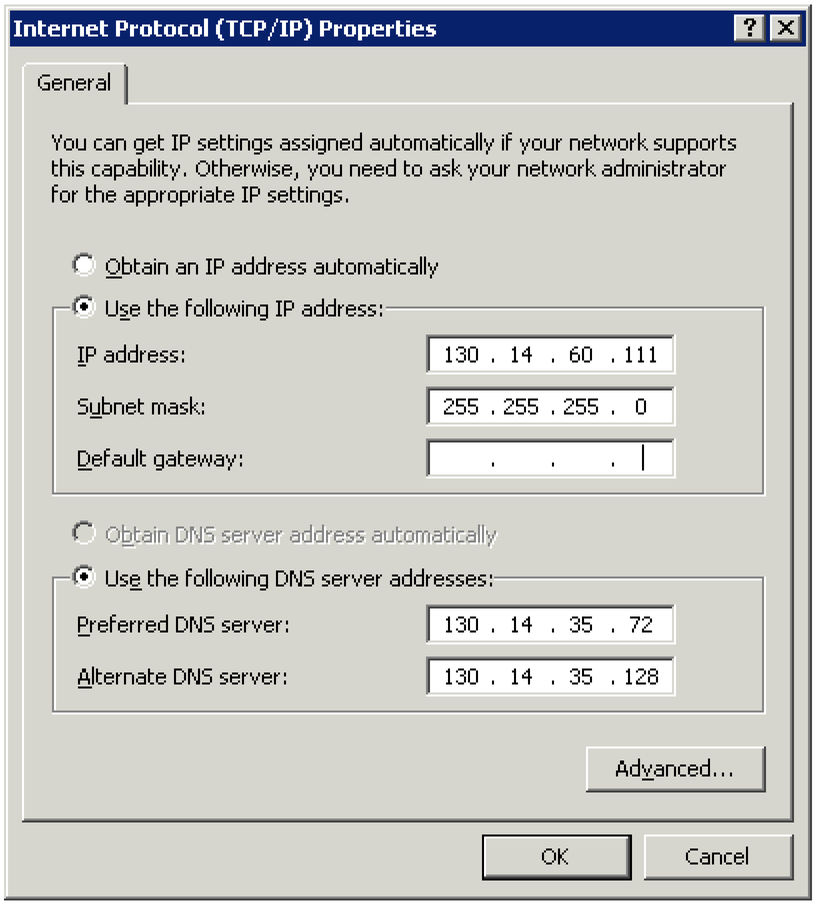
- [✓] Client for Microsoft Networks
- [ ] Network Load Balancing
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] Internet Protocol (TCP/IP)

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

- [✓] Show icon in notification area when connected
- [✓] Notify me when this connection has limited or no connectivity

- Select Internet Protocol (TCP/IP), and click the Properties button.


Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically

(•) Use the following IP address:

IP address:	130 . 14 . 60 . 111
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

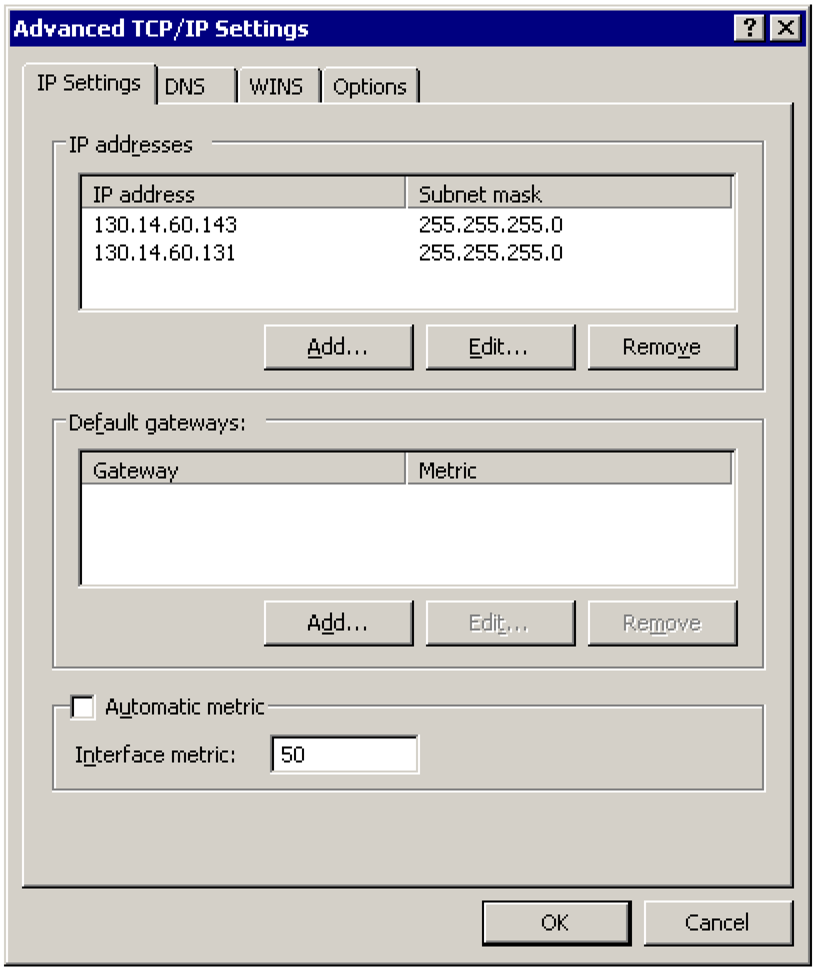
( ) Obtain DNS server address automatically

(•) Use the following DNS server addresses:

Preferred DNS server:	130 . 14 . 35 . 72
Alternate DNS server:	130 . 14 . 35 . 128

- Select “Use the following IP address:” radio button, and under IP address enter the dedicated IP address assigned to the public Ethernet port. Enter the values above for Subnet mask, Default gateway (leave blank), Preferred DNS server, and Alternate DNS server. Enter addresses appropriate for your organization; do not use the ones in the picture. They are examples only, and will not work.

- Next, we need to set the public Ethernet interface at a lower priority than the private interface for outgoing control from internal-running software. This will ensure that all outgoing communication goes out the private Ethernet interface, rather than through the public Ethernet interface. To do this, click the Advanced button again. On the IP Settings tab, un-check the Automatic metric checkbox. Enter 50 into the Interface metric text box. Click OK.


Advanced TCP/IP Settings

IP Settings | DNS | WINS | Options

IP addresses

IP address	Subnet mask
130.14.60.143	255.255.255.0
130.14.60.131	255.255.255.0

Default gateways:

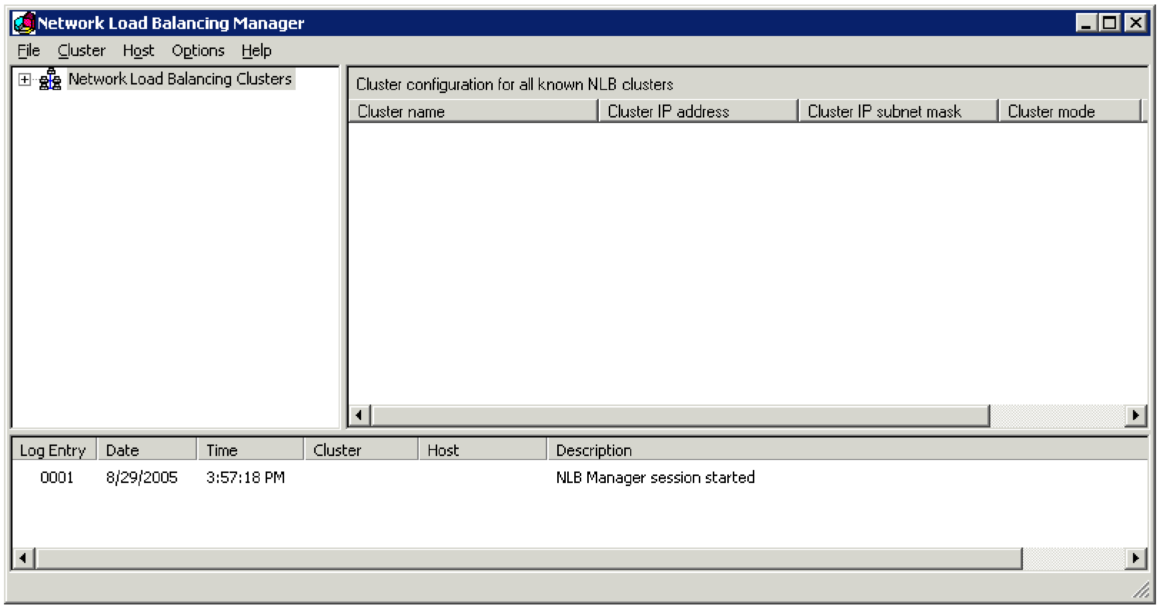
Gateway	Metric

☐ Automatic metric

Interface metric: 50

- Close the dialog boxes that called the one above, to start up the Ethernet port.
- You can verify that the 50 metric has been entered into the public interface by going to a command prompt window, and running “route print”. The private interface will have a metric of 20, while the public interface will have a metric of 50. The operating system always selects the interface with the lower-numbered metric for outgoing communication.

- Start up the Network Load Balancing Manager:

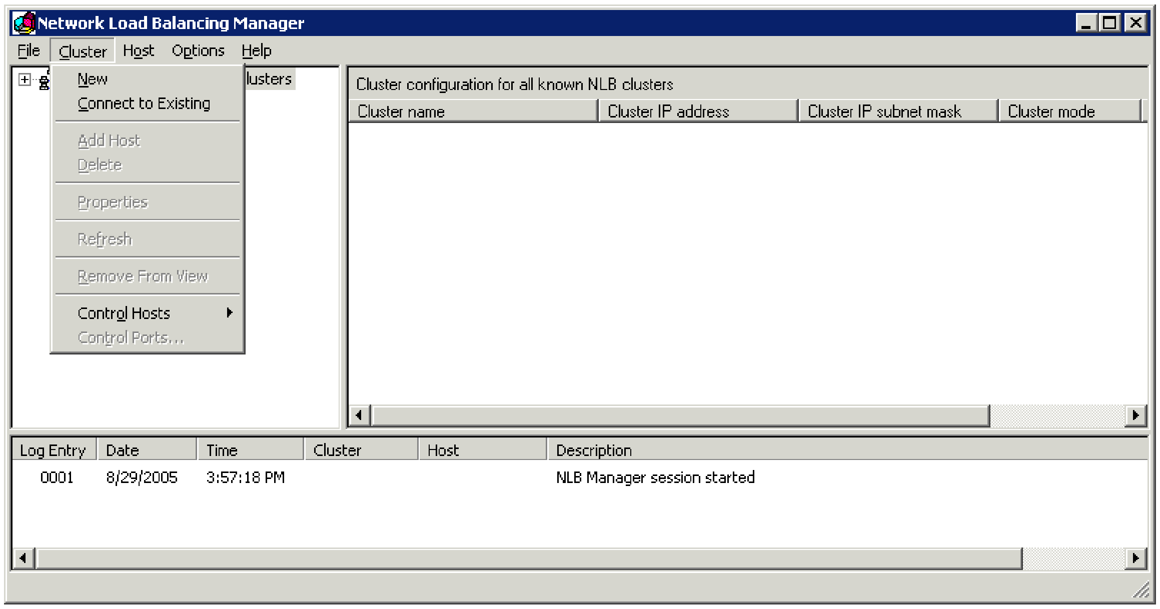

Network Load Balancing Manager
File Cluster Host Options Help
Network Load Balancing Clusters

Cluster configuration for all known NLB clusters

Cluster name	Cluster IP address	Cluster IP subnet mask	Cluster mode

Log Entry	Date	Time	Cluster	Host	Description
0001	8/29/2005	3:57:18 PM			NLB Manager session started

- Select Cluster, then select New:



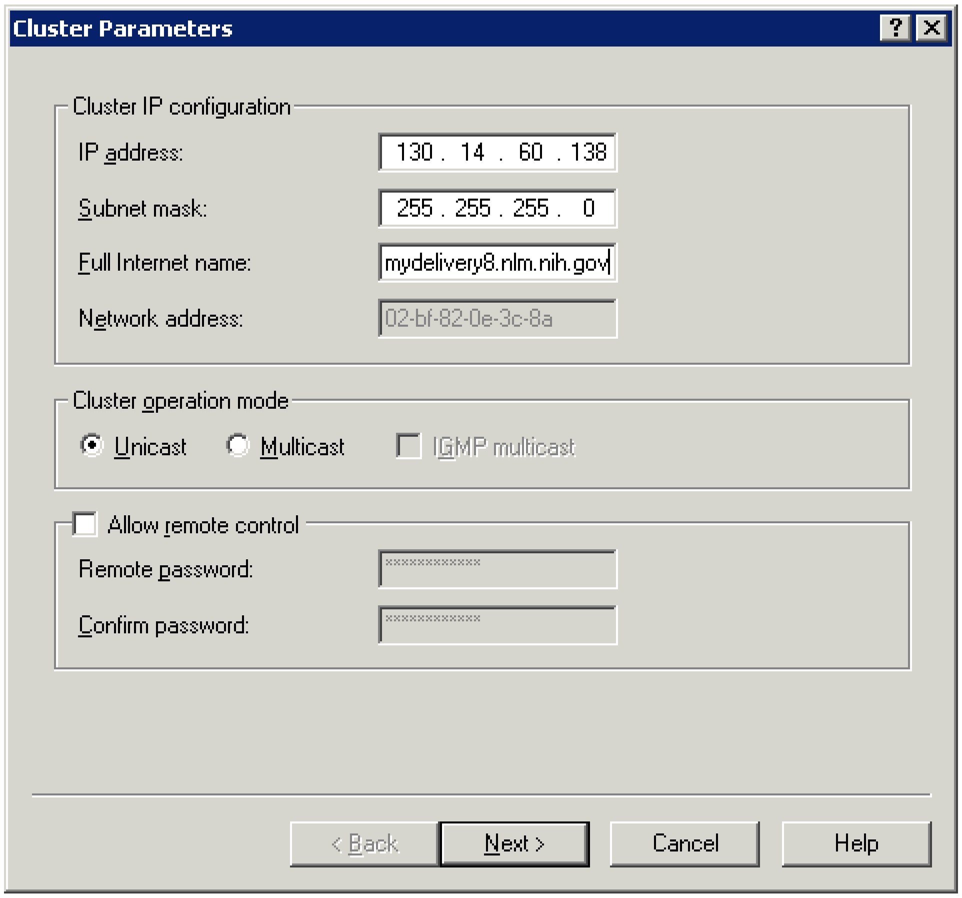
Cluster configuration for all known NLB clusters

Cluster name	Cluster IP address	Cluster IP subnet mask	Cluster mode

Log Entry	Date	Time	Cluster	Host	Description
0001	8/29/2005	3:57:18 PM			NLB Manager session started

- Set up the Cluster parameters:
- IP address – 130.14.60.130 (use the IP address assigned to your NLB)
- Subnet Mask – 255.255.255.0

- Full Internet Name – Enter the appropriate URL for the server, such as mydelivery.mycompany.com.



Cluster IP configuration

IP address:	130 . 14 . 60 . 138
Subnet mask:	255 . 255 . 255 . 0
Full Internet name:	mydelivery8.nlm.nih.gov
Network address:	02-bf-82-0e-3c-8a

Cluster operation mode

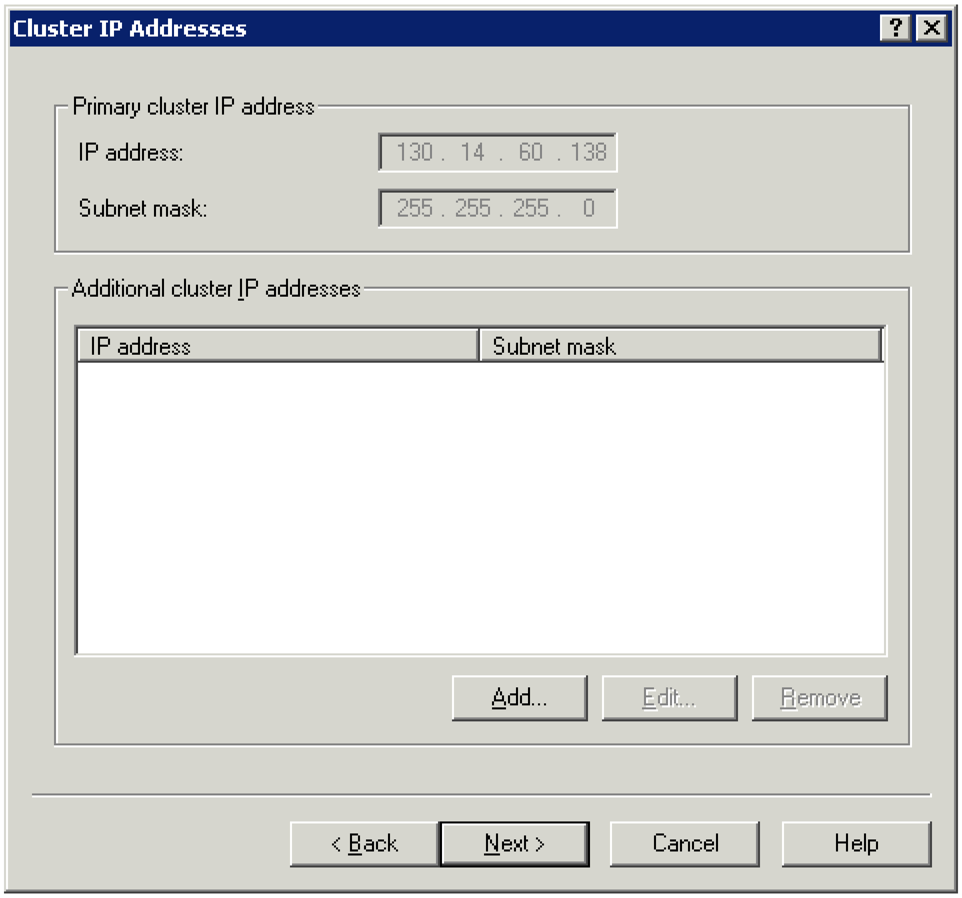
[✓ Unicast] Multicast IGMP multicast

☐ Allow remote control

Remote password:	************
Confirm password:	************

- Select Unicast, and leave "Allow remote control" unchecked. Then click Next.

- On the next dialog box, click Next:



Cluster IP Addresses

Primary cluster IP address

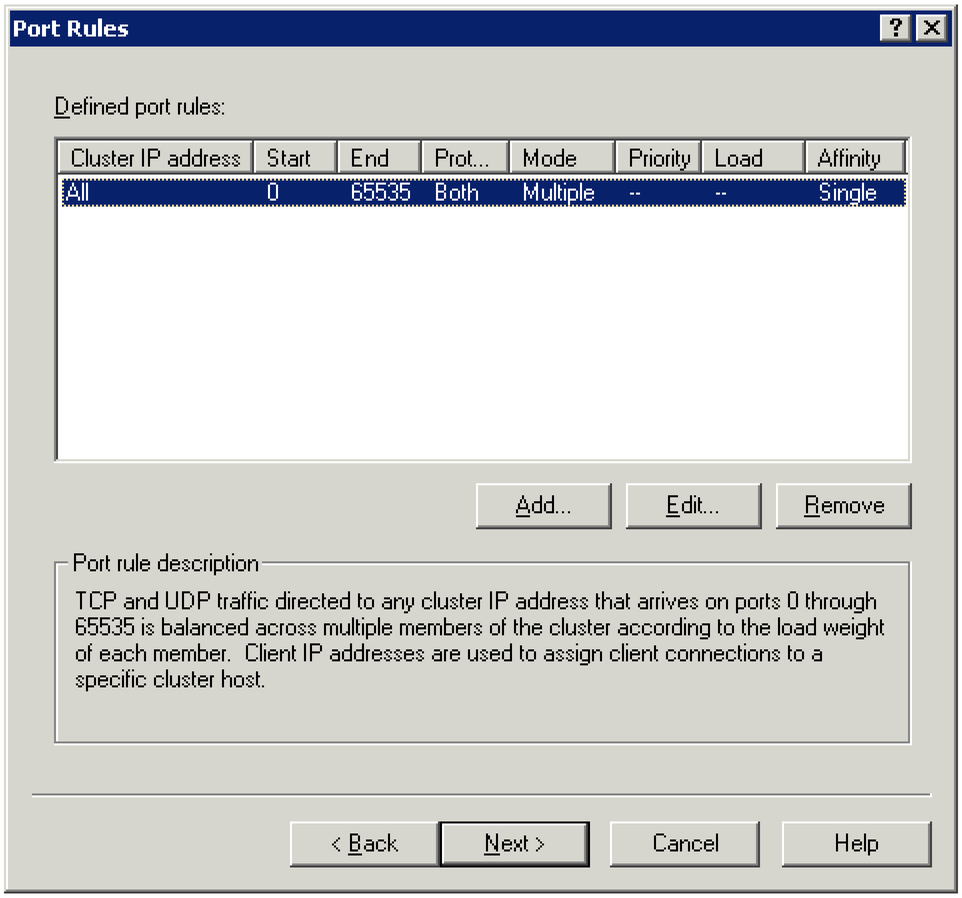
IP address: 130 . 14 . 60 . 138

Subnet mask: 255 . 255 . 255 . 0

Additional cluster IP addresses

IP address	Subnet mask

- On the next dialog box, click Next:



Port Rules

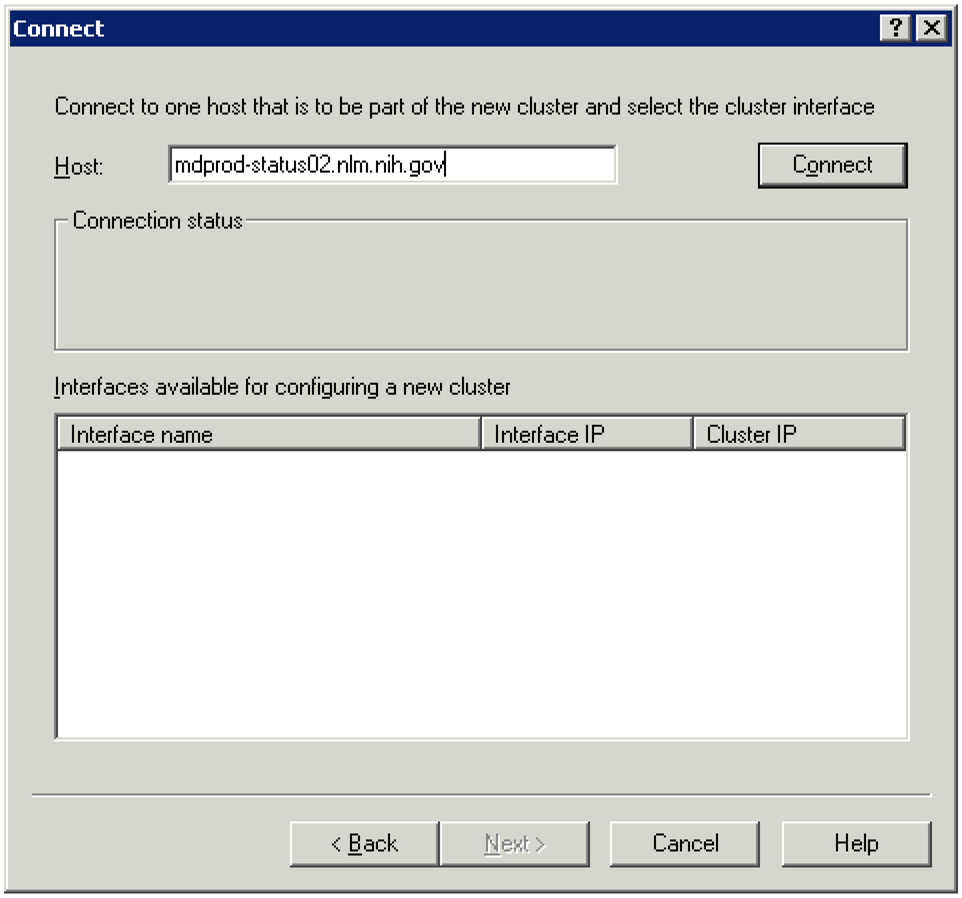
Defined port rules:

Cluster IP address	Start	End	Prot...	Mode	Priority	Load	Affinity
All	0	65535	Both	Multiple	--	--	Single

Port rule description

TCP and UDP traffic directed to any cluster IP address that arrives on ports 0 through 65535 is balanced across multiple members of the cluster according to the load weight of each member. Client IP addresses are used to assign client connections to a specific cluster host.

- In the next dialog box, enter the name of the server being added to the cluster, then click Connect:



Connect

Connect to one host that is to be part of the new cluster and select the cluster interface

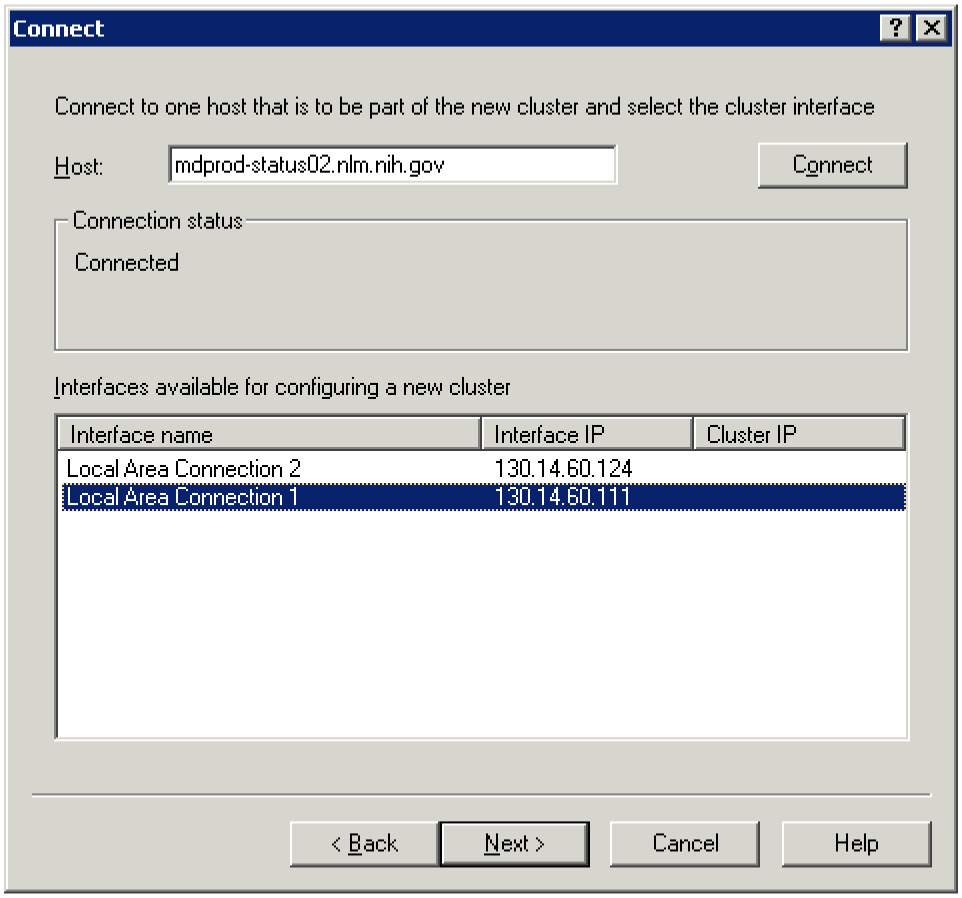
Host: mdprod-status02.nlm.nih.gov

Connection status

Interfaces available for configuring a new cluster

Interface name	Interface IP	Cluster IP

- Select the port that was added in the Internet Protocol Properties dialog box, then click Next:



Connect to one host that is to be part of the new cluster and select the cluster interface

Host: mdprod-status02.nlm.nih.gov

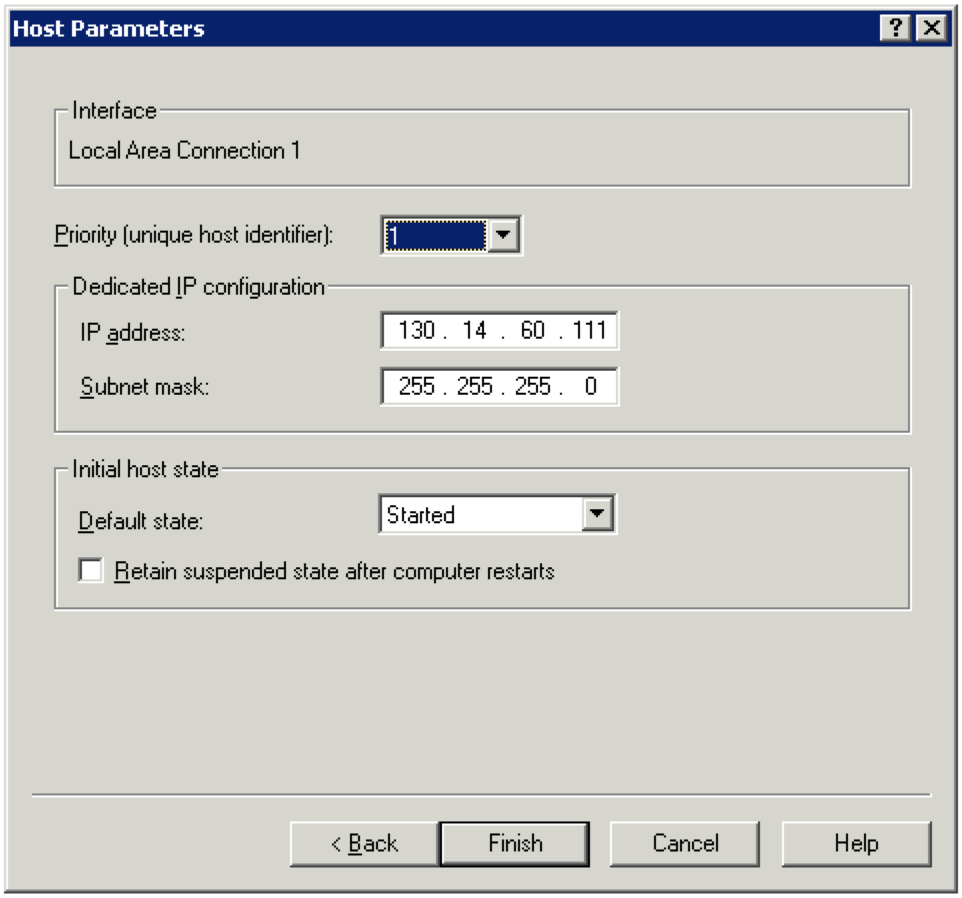
Connection status

Connected

Interfaces available for configuring a new cluster

Interface name	Interface IP	Cluster IP
Local Area Connection 2	130.14.60.124	
Local Area Connection 1	130.14.60.111	

- Click Finish:



Host Parameters

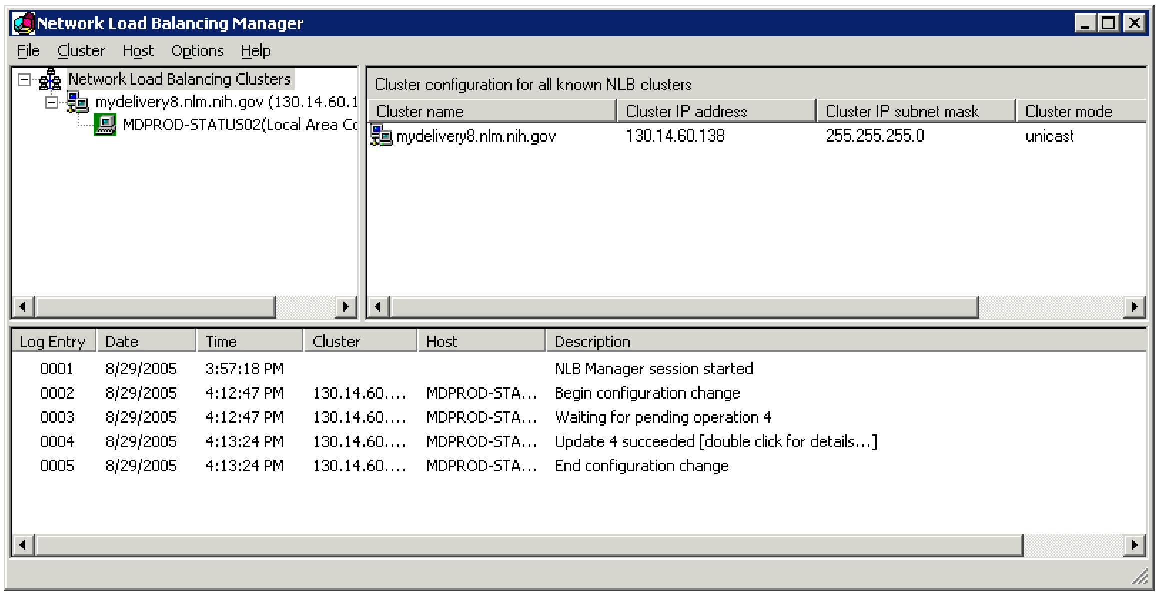
Interface
Local Area Connection 1

Priority (unique host identifier): 1

Dedicated IP configuration
IP address: 130 . 14 . 60 . 111
Subnet mask: 255 . 255 . 255 . 0

Initial host state
Default state: Started
☐ Retain suspended state after computer restarts

- The next dialog box will show the successful addition of the host server to the cluster:



Network Load Balancing Manager
File Cluster Host Options Help

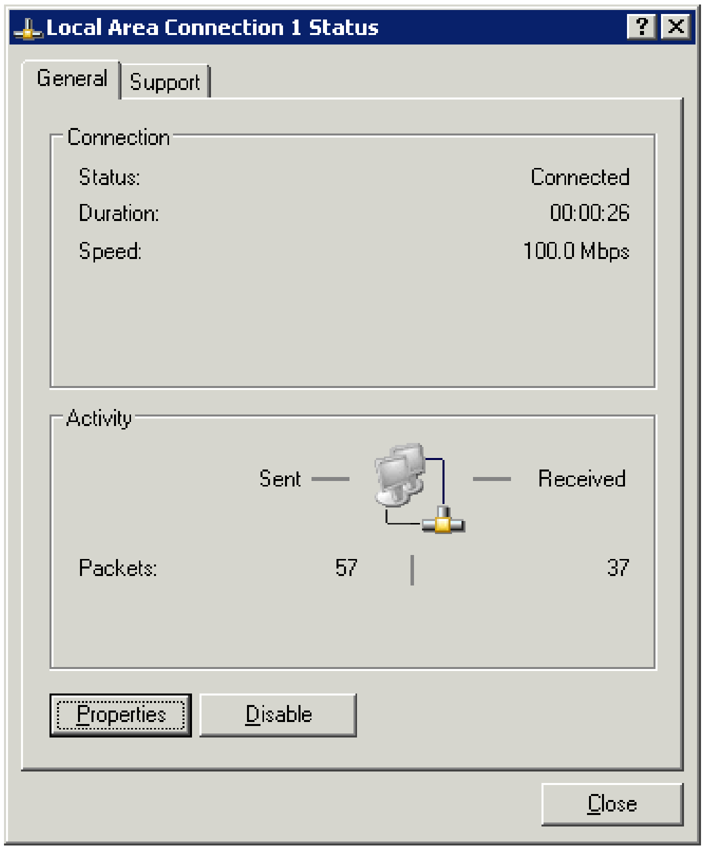
Network Load Balancing Clusters
mydelivery8.nlm.nih.gov (130.14.60.1
MDPROD-STATUS02(Local Area Co

Cluster configuration for all known NLB clusters

Cluster name	Cluster IP address	Cluster IP subnet mask	Cluster mode
mydelivery8.nlm.nih.gov	130.14.60.138	255.255.255.0	unicast

Log Entry	Date	Time	Cluster	Host	Description
0001	8/29/2005	3:57:18 PM			NLB Manager session started
0002	8/29/2005	4:12:47 PM	130.14.60....	MDPROD-STA...	Begin configuration change
0003	8/29/2005	4:12:47 PM	130.14.60....	MDPROD-STA...	Waiting for pending operation 4
0004	8/29/2005	4:13:24 PM	130.14.60....	MDPROD-STA...	Update 4 succeeded [double click for details...]
0005	8/29/2005	4:13:24 PM	130.14.60....	MDPROD-STA...	End configuration change

- Next we need to add more hosts to this cluster. To start, go to the new host's public port and examine the network status:



Local Area Connection 1 Status

General | Support

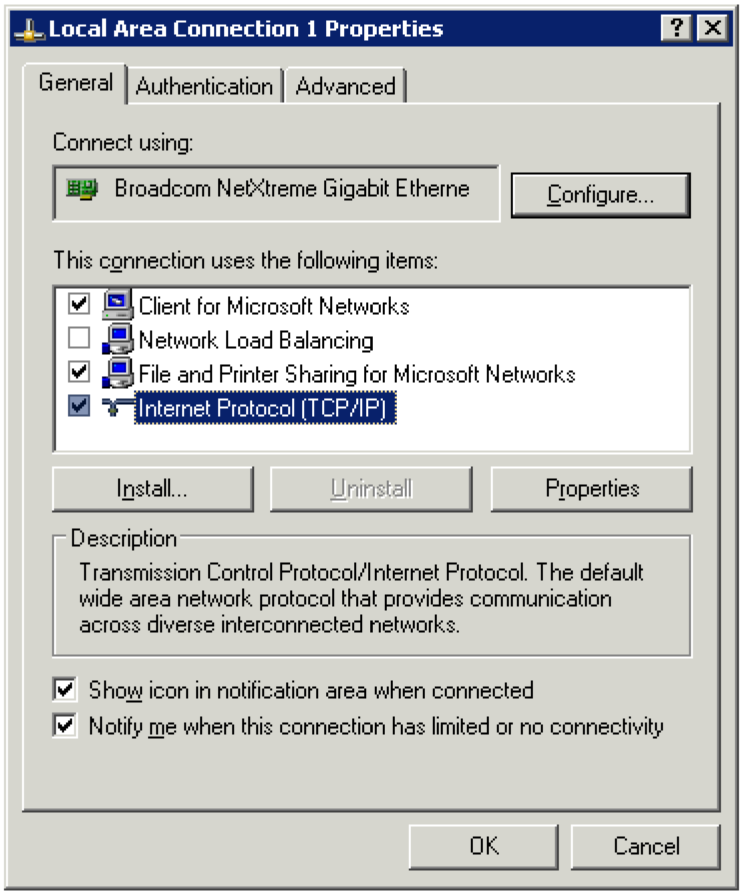
Connection

Status:	Connected
Duration:	00:00:26
Speed:	100.0 Mbps

Activity

	Sent	Received
Packets:	57	37

- Click the Properties button, and get the next dialog box:



Connect using:

Broadcom NetXtreme Gigabit Etherne

This connection uses the following items:

- [✓] Client for Microsoft Networks
- [ ] Network Load Balancing
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] Internet Protocol (TCP/IP)

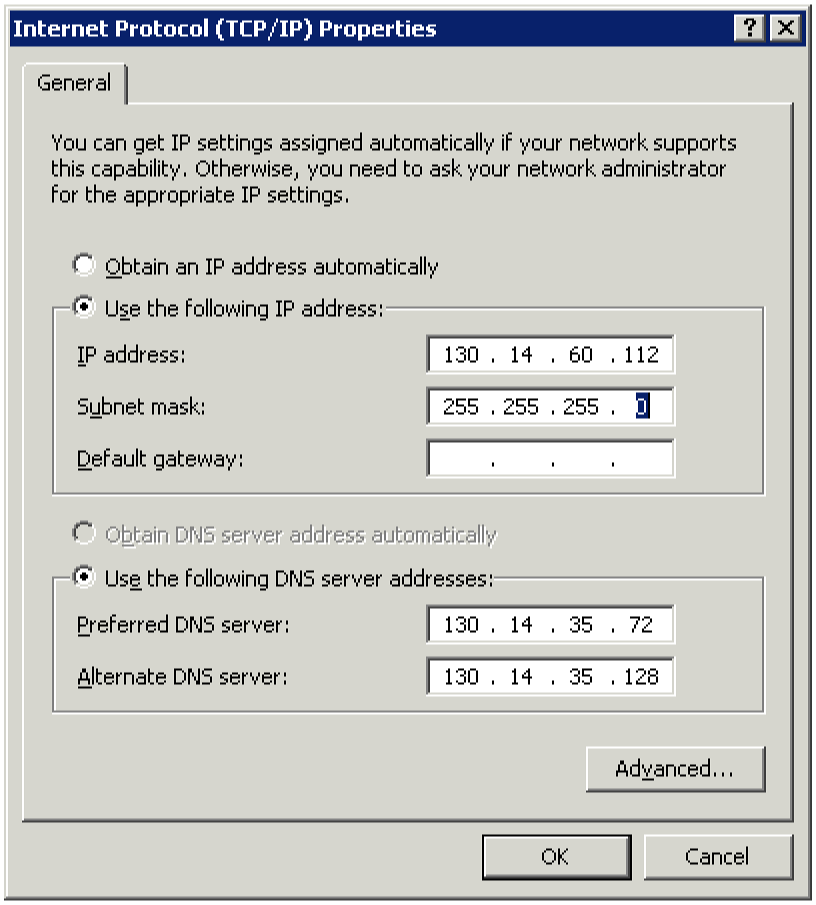
Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

[✓] Show icon in notification area when connected

[✓] Notify me when this connection has limited or no connectivity

- Select Internet Protocol (TCP/IP), and click the Properties button.


Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically

(●) Use the following IP address:

IP address:	130 . 14 . 60 . 112
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

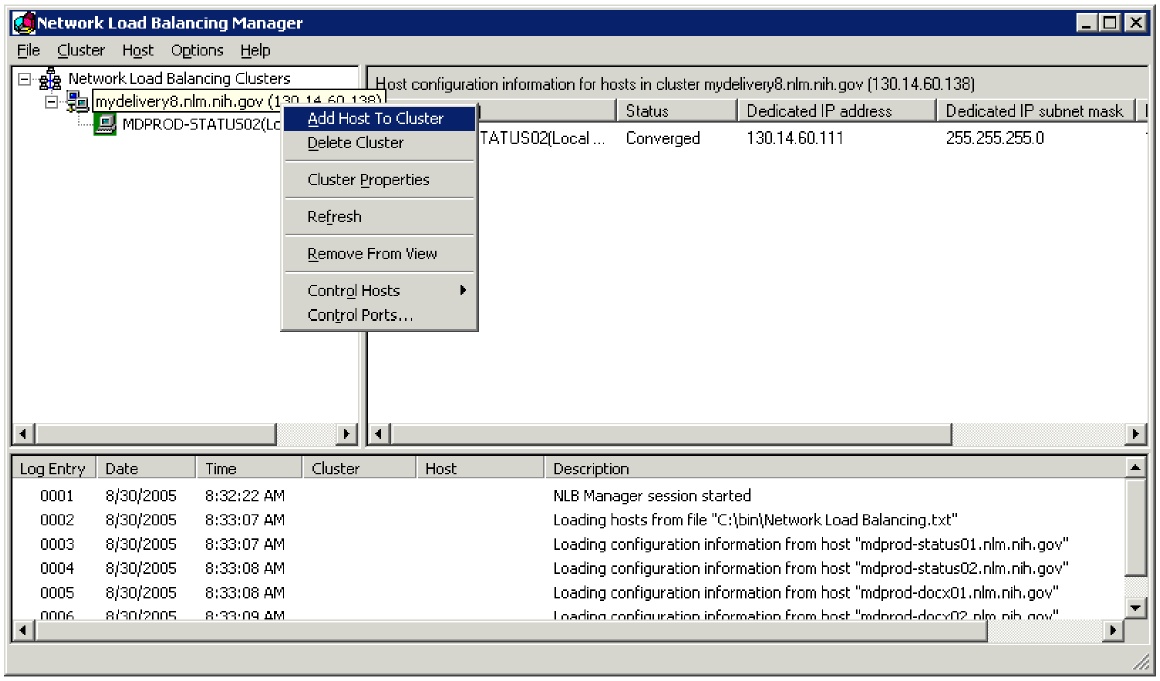
( ) Obtain DNS server address automatically

(●) Use the following DNS server addresses:

Preferred DNS server:	130 . 14 . 35 . 72
Alternate DNS server:	130 . 14 . 35 . 128

- Enter the unique local IP address to be used by this port, subnet mask, and leave the default gateway blank. Click OK.

- Go to the Network Load Balancing Manager to add the new host to the NLB cluster. Right click on the cluster name, and select "Add Host to Cluster".

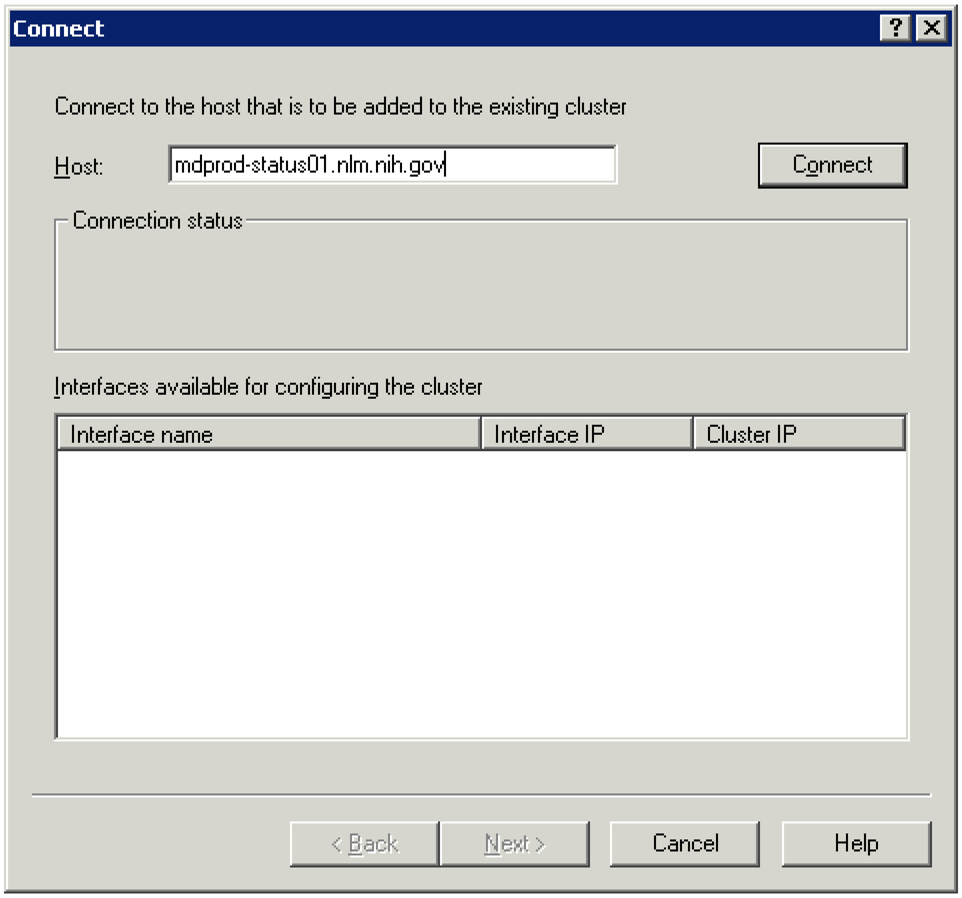

Network Load Balancing Manager
File Cluster Host Options Help
Network Load Balancing Clusters
mydelivery8.nlm.nih.gov (130.14.60.138)
MDPROD-STATUS02(Lc
Add Host To Cluster
Delete Cluster
Cluster Properties
Refresh
Remove From View
Control Hosts
Control Ports...

Host configuration information for hosts in cluster mydelivery8.nlm.nih.gov (130.14.60.138)

	Status	Dedicated IP address	Dedicated IP subnet mask
TATUS02(Local ...	Converged	130.14.60.111	255.255.255.0

```
Log Entry  Date       Time        Cluster  Host  Description
0001       8/30/2005  8:32:22 AM                 NLB Manager session started
0002       8/30/2005  8:33:07 AM                 Loading hosts from file "C:\bin\Network Load Balancing.txt"
0003       8/30/2005  8:33:07 AM                 Loading configuration information from host "mdprod-status01.nlm.nih.gov"
0004       8/30/2005  8:33:08 AM                 Loading configuration information from host "mdprod-status02.nlm.nih.gov"
0005       8/30/2005  8:33:08 AM                 Loading configuration information from host "mdprod-docx01.nlm.nih.gov"
0006       8/30/2005  8:33:09 AM                 Loading configuration information from host "mdprod-docx02.nlm.nih.gov"
```

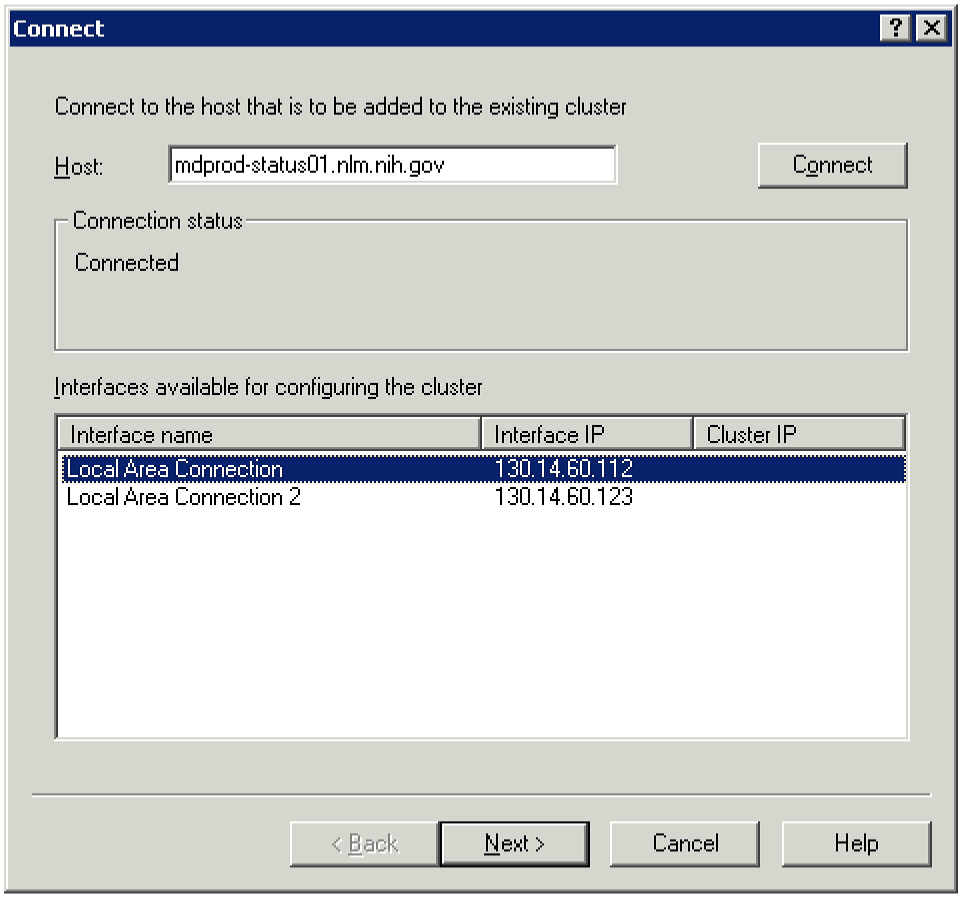
- Enter the private name of the host to be added, and click Connect.


Connect
Connect to the host that is to be added to the existing cluster
Connect
Connection status
Interfaces available for configuring the cluster
< Back
Next >
Cancel
Help

Host: mdprod-status01.nlm.nih.gov

Interface name	Interface IP	Cluster IP

- Select the IP address just created for the public port of this server, and click Next:



Connect

Connect to the host that is to be added to the existing cluster

Host: mdprod-status01.nlm.nih.gov

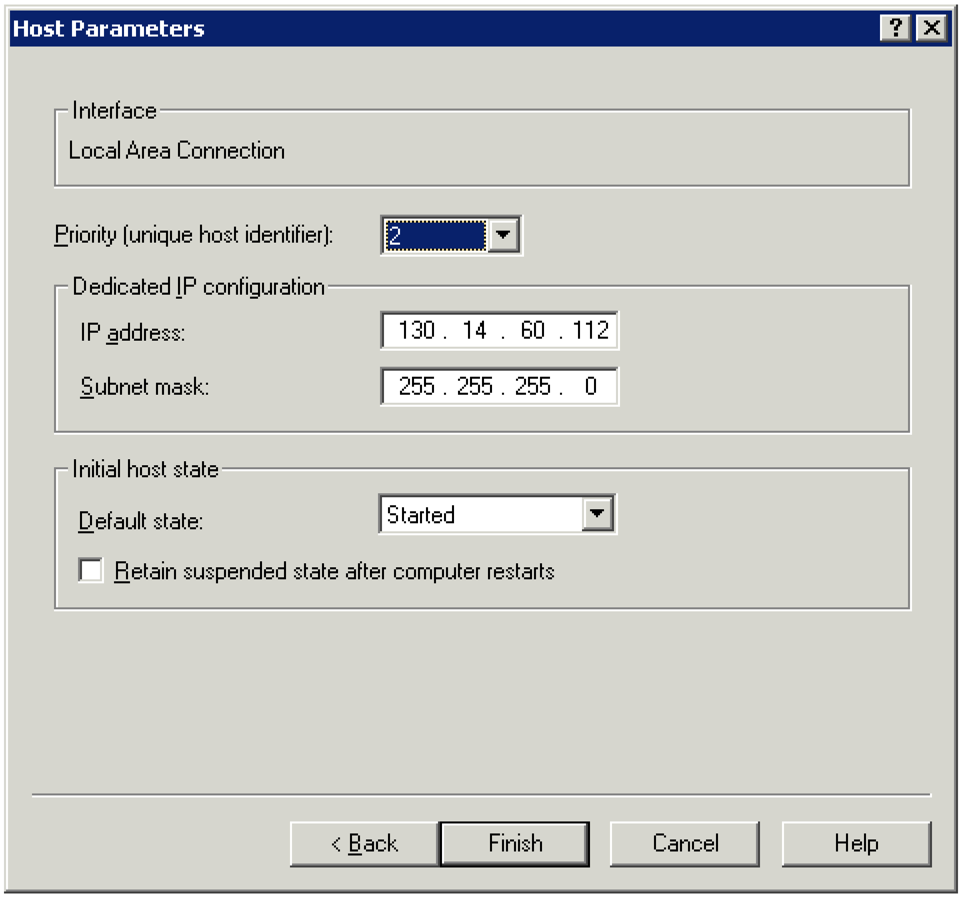
Connection status

Connected

Interfaces available for configuring the cluster

Interface name	Interface IP	Cluster IP
Local Area Connection	130.14.60.112	
Local Area Connection 2	130.14.60.123	

- Click Next:



Host Parameters

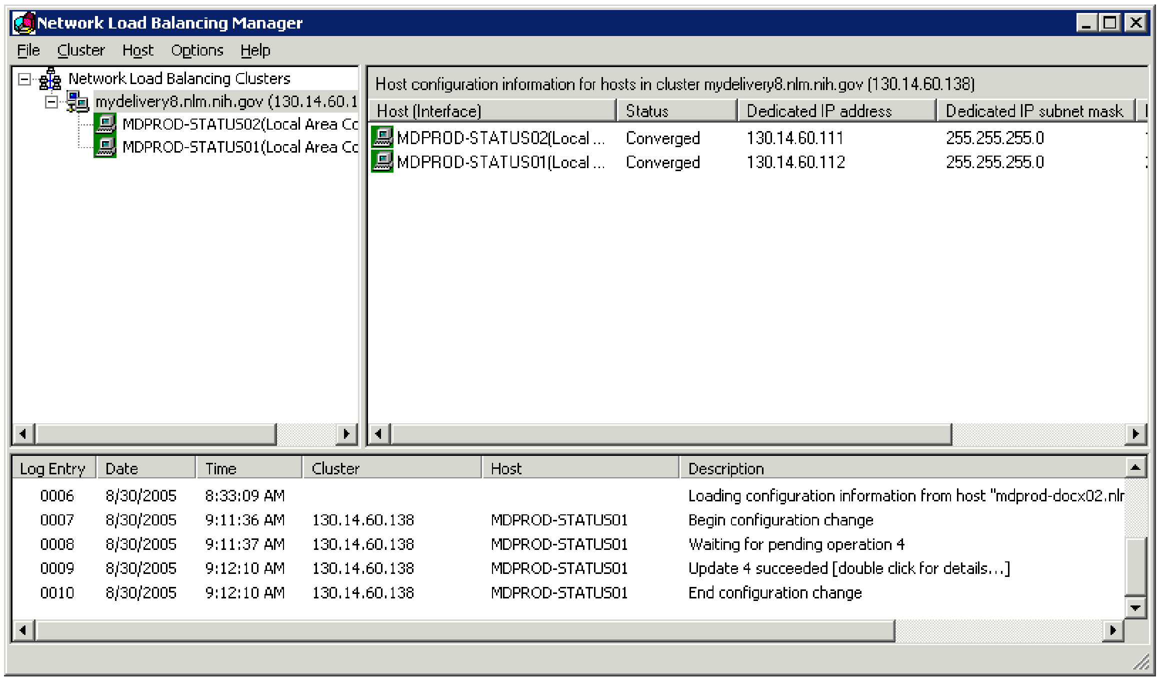
Interface
Local Area Connection

Priority (unique host identifier): 2

Dedicated IP configuration
IP address: 130 . 14 . 60 . 112
Subnet mask: 255 . 255 . 255 . 0

Initial host state
Default state: Started
☐ Retain suspended state after computer restarts

- After a couple of minutes, the NLB Manager dialog box will show the convergence of the second host into the cluster:

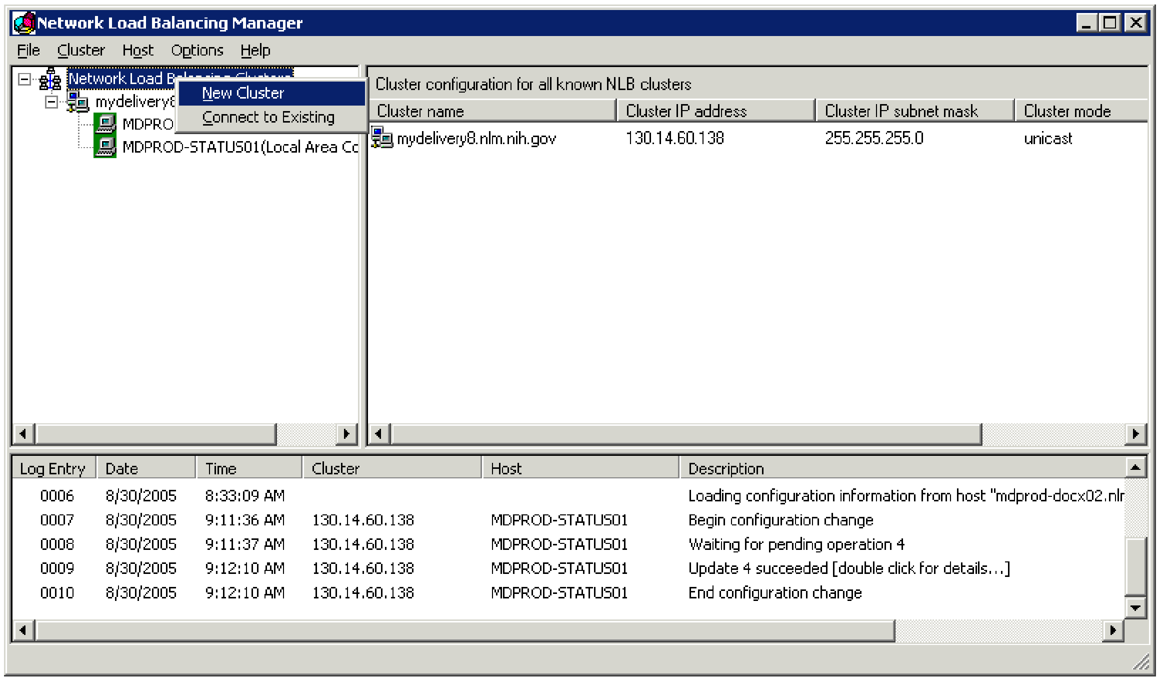

Network Load Balancing Manager
File Cluster Host Options Help
Network Load Balancing Clusters
mydelivery8.nlm.nih.gov (130.14.60.1
MDPROD-STATUS02(Local Area Cc
MDPROD-STATUS01(Local Area Cc

Host configuration information for hosts in cluster mydelivery8.nlm.nih.gov (130.14.60.138)

Host (Interface)	Status	Dedicated IP address	Dedicated IP subnet mask
MDPROD-STATUS02(Local ...	Converged	130.14.60.111	255.255.255.0
MDPROD-STATUS01(Local ...	Converged	130.14.60.112	255.255.255.0

Log Entry	Date	Time	Cluster	Host	Description
0006	8/30/2005	8:33:09 AM			Loading configuration information from host "mdprod-docx02.nlr
0007	8/30/2005	9:11:36 AM	130.14.60.138	MDPROD-STATUS01	Begin configuration change
0008	8/30/2005	9:11:37 AM	130.14.60.138	MDPROD-STATUS01	Waiting for pending operation 4
0009	8/30/2005	9:12:10 AM	130.14.60.138	MDPROD-STATUS01	Update 4 succeeded [double click for details...]
0010	8/30/2005	9:12:10 AM	130.14.60.138	MDPROD-STATUS01	End configuration change

- The procedure for creating a new cluster is easy: just right-click on “Network Load Balancing Clusters” and select New Cluster. Then follow the above procedure for adding hosts to the new cluster.


Network Load Balancing Manager
File Cluster Host Options Help
Network Load Balancing Clusters
mydelivery8
MDPRO
MDPROD-STATUS01(Local Area Cc
New Cluster
Connect to Existing

Cluster configuration for all known NLB clusters

Cluster name	Cluster IP address	Cluster IP subnet mask	Cluster mode
mydelivery8.nlm.nih.gov	130.14.60.138	255.255.255.0	unicast

Log Entry	Date	Time	Cluster	Host	Description
0006	8/30/2005	8:33:09 AM			Loading configuration information from host "mdprod-docx02.nlr
0007	8/30/2005	9:11:36 AM	130.14.60.138	MDPROD-STATUS01	Begin configuration change
0008	8/30/2005	9:11:37 AM	130.14.60.138	MDPROD-STATUS01	Waiting for pending operation 4
0009	8/30/2005	9:12:10 AM	130.14.60.138	MDPROD-STATUS01	Update 4 succeeded [double click for details...]
0010	8/30/2005	9:12:10 AM	130.14.60.138	MDPROD-STATUS01	End configuration change

- On each Front-End Processor, create a c:\mdservices folder containing all MyDelivery web services. Under the c:\mdservices folder will be additional folders, each containing components of the web services:

    1. CacheData – This is a web service component of mdFlusher that is used to store/retrieve items from cache memory.
    2. mdFlusher – This is the web service for CacheData.
    3. ClientList – This is a web service component of mdClientListService that keeps track of MyDelivery clients that are online (currently running).
    4. mdClientListService – This is the web service for ClientList.
    5. FileXFER – This is the web service component that allows external processes to store or retrieve data on the server.
    6. mdFileXFERService – This is the web service for FileXFER.
    7. mdMonitor – This is a web service that monitors current conditions on the Front-End Processor.
    8. Pinger – This is a web service component for mdPinger. It is the equivalent of a “ping” service, which allows other machines to determine whether the machine running mdPinger is online and itself running.
    9. mdPinger – This is the web service for Pinger.
    10. mdClientManager - This is a web service that monitors the entire MyDelivery server system and all online clients. In a multi-cpu server system, a Primary mdClientManager runs on its own separate machine (Manager). A Secondary (backup) mdClientManager runs on one of the Front-End Processors.

- Set up all Windows services in the MyDelivery Server system to run using a service account that has a non-expiring password. This includes all md services: mdClientManager, mdClientListService, mdFileXFERService, mdFlusher, mdMonitor, and mdPinger. Go to MyComputer/manage.   Under Groups, click Administrators. Add the service account to the Administrators group.

- In the web.config files on all Front-End Processors, set up the identity element:

```
<!--  used for setting the identity of ASP.NET for contacting the SQL Server

 -->
 <identity impersonate="true"
userName="registry:HKLM\SOFTWARE\MyDelivery\identity\ASPNET_SETREG,userName"
password="registry:HKLM\SOFTWARE\MyDelivery\identity\ASPNET_SETREG,password"/>
```

This allows MyDelivery SOAP services to access the SQL Server through the username and password that will be stored encrypted in the registry in the next step.

- On all Front-End Processors, run the aspnet_setreg utility. This allows us to encrypt the username and password associated with the MyDelivery service account and store it in the registry.
- Give the Network Service rights to read the registry key entries for the username and password created in the last step. Using regedt32.exe, go to HKEY_LOCAL_MACHINE\SOFTWARE\MyDelivery key. Right-click ASPNET_SETREG, and click Permissions. Add the Network Service, giving it read permissions for the username and password subkeys.
- Install and run all MyDelivery web services. This is done by running a batch file, mdsetup.bat, located in c:\bin. The contents of mdsetup.bat are as follows:

```
@REM Batch file for setting up services on a Front-End Processor
c:
net stop mdFileXFERService
cd \MDServices\mdFileXFERService\bin
c:\bin\installutil -uninstall mdFileXFERService.exe
c:\bin\installutil mdFileXFERService.exe
net start mdFileXFERService

net stop mdClientListService
cd \MDServices\mdClientListService\bin
c:\bin\installutil -uninstall mdClientListService.exe
c:\bin\installutil mdClientListService.exe
net start mdClientListService

net stop mdFlusher
cd \MDServices\mdFlusher\bin
c:\bin\installutil -uninstall mdFlusher.exe
c:\bin\installutil mdFlusher.exe
net start mdFlusher

net stop mdMonitor
cd \MDServices\mdMonitor\bin
c:\bin\installutil -uninstall mdMonitor.exe
c:\bin\installutil mdMonitor.exe
net start mdMonitor

net stop mdpinger
```

```
cd \mdservices\mdpinger\bin
c:\bin\installutil -uninstall mdpinger.exe
c:\bin\installutil mdpinger.exe
net start mdpinger

@REM Install a primary mdClientManager only once
@REM For a multi-cpu system, install a secondary mdClientManager only once
net stop mdClientManager
cd \MDServices\mdClientManager\bin
c:\bin\installutil -uninstall mdClientManager.exe
c:\bin\installutil mdClientManager.exe
net start mdClientManager

cd \bin
pause
```

- Finally, run the Services executable found under Administrative Tools. You may need to Stop each service, configure the proper UserID and login to be used for each of the services, and restart the services. Typically the best UserID to be used for a service would be an administrators account that does not have an expiring password.

5.0 MyDelivery SQL Server Database Tables

The following database tables are used for the MyDelivery database residing on the MyDelivery Database (SQL) Server:

1. ClientsOnline – Keeps track of which clients are currently online and ready for communication.
2. Deliveries – Keeps track of all deliveries made by the system.
3. IDs – A table containing a list of all valid MyDelivery ID's. These are the same as the Handle in the Users table. This table provides a fast look-up of a MyDelivery ID to see if it exists.
4. InvalidLogin – Used to permit no more than 3 incorrect logins in a 60 minute period.
5. PendingSenders – A list of all receiver MyDelivery ID / sender MyDelivery ID pairs for each sender that is pending to send to the receiver.
6. Processors – Keeps track of functionality and states of all processors used in the MyDelivery server.
7. RegCodes – Prevents hackers from generating thousands of requests for user registration.
8. SystemInfo – Keeps track of key characteristic of the Server system.
9. Users – Keeps track of user registrations. Includes user name, address, email address, date of registration, and other information.

5.1 ClientsOnline

The ClientsOnline table is used by several server processes to maintain a list of all users whose clients are online.

Kev →

Column Name	Data Type	Length	Allow Nulls	Description
UserID	int	4		Unique ID of the user
RandomID	int	4		Randomly-generated ID used to identify this user
DoUpdateStatus	bit	1		TRUE if client should execute UpdateStatus to find out changes in deliveries.
PendingSender	bit	1		TRUE if another client wants to send to this client, but is not in this client's address book.

5.2 Deliveries

The Deliveries table keeps track of the status of each delivery.

Kev →

Column Name	Data Type	Length	Allow Nulls	Description
DeliveryNumber	bigint	8		Unique delivery number
bCleanedUp	bit	1		0 indicates this delivery is not cleaned up; 1 means it is cleaned up
StartDate	datetime	8		Date the delivery was initiated
MostRecentDateUsed	datetime	8		Last time the sender or receiver uploaded or downloaded anything
Attachments	int	4		Number of attachments in this delivery
TotalFiles	int	4		Total number of files being sent in this delivery
TotalSize	bigint	8		Total size of the delivery in bytes
TotalTime	int	4		Time (in seconds) of this delivery
SenderHandle	nvarchar	15		MyDelivery ID of the sending client
ReceiverHandle	nvarchar	15		MyDelivery ID of the receiving client
UploadStatus	int	4		Status of the uploading client delivery: 0 = Pending 1st upload, 1 = At least one attachment uploaded, 2 = All attachments uploaded, 3 = Delivery terminated, 4 = Job removed by server
DownloadStatus	int	4		Status of the download delivery 0 = Pending, 2 = Success 3 = Job abandoned by receiver or removed by server
LastUploadedAttachment	int	4		Most recent attachment uploaded successfully
LastDownloadedAttachment	int	4		Most recent attachment downloaded successfully
Resync	bit	1		0 indicates no recync required; 1 indicates whether a sending clients needs to resync with the receiving; Download Request contains requested data
DownloadRequest	int	8		Used to resync a sending client with a receiving client. If > 0, this indicates the next attachment to

				be sent, or if 0, this indicates the sender should send the DeliveryHeader through StartDeliveryUpload.
Processor	varchar	64		Full path to computer, i.e. proc1.mydelivery.com
bDirectionUp	Bit	1		True indicates the UpDelay is increasing; False indicates it is decreasing
UpDelay	Int	4		Most recent delay for UploadAttachment
bDirectionDown	Bit	1		True indicates the DownDelay is increasing; False indicates it is decreasing
DownDelay	Int	4		Most recent delay for DownloadAttachment

5.3 IDs

The IDs table keeps track of user IDs currently in use. This table is used primarily for registration of new users, to determine whether an ID is currently in use. The MyDeliveryID is the same as the “Handle” in the Users table.

	Column Name	Data Type	Length	Allow Nulls	Description
Key →	MyDeliveryID	Nvarchar	15		MyDelivery ID of this user

5.4 InvalidLogin

The InvalidLogin table is used to keep track of the most recent time a user attempted, but failed, to log into the system. It is used to prevent hackers from guessing a user's password. Users are allowed only 4 invalid logins within a 60 minute period. A record is placed into this table for each invalid login. The mdClientManager process periodically cleans up the table by removing old records.

Column Name	Data Type	Length	Allow Nulls	Description
Handle	Nvarchar	15		MyDelivery ID of this user. Same as the "Handle" in the users table.
InvalidLoginTime	datetime	8		Date and Time the user failed to log into MyDelivery

5.5 PendingSenders

The PendingSenders table is used to inform a person of other users who wish to want to communicate with the person, but who cannot because the person does not have the others in his address book. The table maintains the receiver/sender relationship for these pairs of users who could potentially communicate with each other.

Column Name	Data Type	Length	Allow Nulls	Description
Number	Int	4		Unique record number
ReceiverHandle	nvarchar	15		Receiver MyDelivery ID
SenderHandle	nvarchar	15		Sender MyDelivery ID

5.6 Processors

The Processors table keeps track of all Front-End Processors used in the system, and the status of each.

Column Name	Data Type	Length	Allow Nulls	Description
Processor	char	64		Full path to computer, e.g. proc1.mydelivery.com
Status	int	4		Processor Status: 0 = Down; 1 = Up; 2 = Going Down; 3 = Coming Up
Sick	int	4		1 = SICK; 0 = WELL; updated by mdClientManager. If processor is SICK, no other processor is allowed to access it except mdClientManager or the processor itself.
PercentUtilization	Int	4		Percentage of processor utilization.
HeartBeat	Int	4		Local HeartBeat for this processor (Time period for CSP)

5.7 RegCodes

The RegCodes table is used for new user registration to prevent hackers from randomly generating user registrations. It stores a unique “Code” that is displayed on the user registration form.

Column Name	Data Type	Length	Allow Nulls	Description
Code	Nvarchar	7		Unique registration code
RegistrationDate	datetime	8		Date this code was registered

5.8 SystemInfo

The SystemInfo table is used for keeping track of the overall system.

Column Name	Data Type	Length	Allow Nulls	Description
Number	Int	4		Unique record number
ClientVersionNumber	char	10		Version number of the MyDelivery Client software
ClientURL	char	256		URL where the client installation software package can be downloaded
SetupEXE	Char	256		Physical location of the client setup package

ServerUp	Bit	1		TRUE = System is Up;
				FALSE = System is Down

Window	Int	4		Number of minutes a client can be offline, and a delivery can stay on the server
Day2Update	Int	4		Used by mdClientManager for updating the system once a day (Number of day from 1 to 366)
ClientManagerAccess	Datetime	8	Yes	The most recent access time by the Primary mdClientManager

5.9 Users

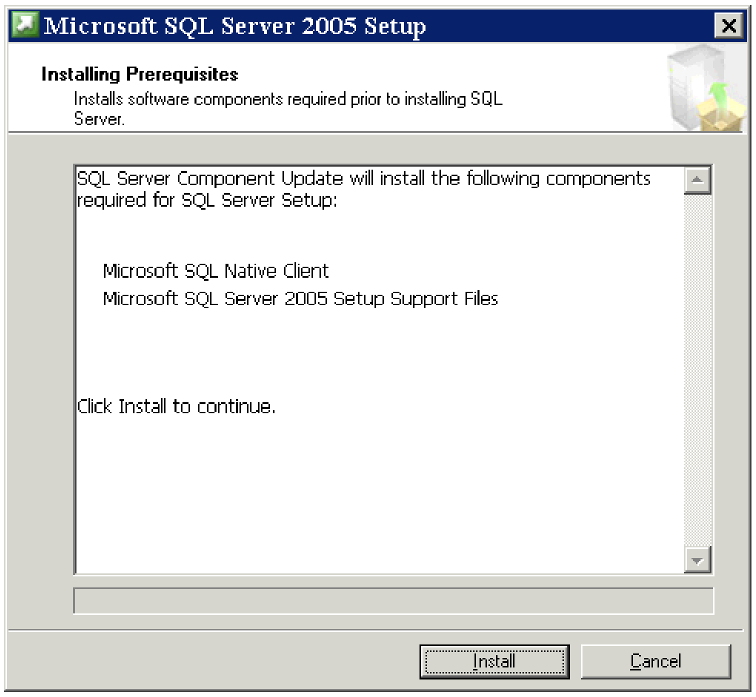
The Users table keeps track of all registered users: their registration information, number of deliveries sent and received, address book and disk space on their most recently used computer.

Column Name	Data Type	Length	Allow Nulls	Description
UserID	int	4		Unique record number
FirstName	nvarchar	22		First name
LastName	nvarchar	22		Last name
Org	nvarchar	50	Yes	Organization
Address1	nvarchar	50		First line of address
Address2	nvarchar	50	Yes	Second line of address
City	nvarchar	30	Yes	City
State	nvarchar	20		State
Country	nvarchar	20		Country (United States)
Zip	nvarchar	10		Zip Code
Handle	nvarchar	15		User's MyDelivery ID
Password	nvarchar	30		User's password
RegistrationDate	Datetime	8		Date registered
MostRecentDateUsed	datetime	8		Date the user most recently used MyDelivery
TotalTransmissions	int	4		Total # of transmissions by this user
TotalReceptions	int	4		Total # of receptions by this user
SpamControl	int	4	Yes	Controls spam. 0 = BlockAll, 1 = AddressBook
LargestFile	bigint	8		Largest file that can fit on the hard drive of the user's computer
FreeDiskSpace	bigint	8		Amount of free disk space available on the client computer for deliveries
AddressBook	ntext	-	Yes	Address book

6.0 SQL Server Database Configuration using SQL Server Express

The MyDelivery database server may be configured to run on a freely available SQL Server 2005 Express or SQL Server 2008 Express edition. MyDelivery has been tested and works well on both of these servers. The Express Edition server is free, but it has some limitations. First, the database size is limited to 4 gigabytes, which is not a problem for MyDelivery, since its database is small (typically 50 megabytes maximum). Second, the Express edition limits memory usage of the SQL Server to 1 gigabyte and one processor (again, not really a problem for small MyDelivery server systems.) Third, backups must be done manually, which is usually not a problem. Forth, there is no database mirroring with automatic failover protection. This means that if the SQL Server Express goes down, then the MyDelivery server system is down until manually restored to service. Implementations should not use SQL Server Express if high reliability is absolutely critical, because it may not always be possible for a human operator to respond quickly enough to restore service if the server should go down. For implementations where minimizing cost is an important factor, usage of a SQL Server Express edition should be quite suitable.

Installation of SQL Server Express is quite straightforward; most of the defaults can be selected during installation. The individual running the installation should have administrator rights. Various flavors of SQL Server Express are available from Microsoft. For the example here, SQL Server 2005 Express edition with Advanced Services (including Management Studio) was used. In the picture below, the Microsoft SQL Native Client and Setup Support files will be installed. Click Install as shown.


Microsoft SQL Server 2005 Setup

Installing Prerequisites

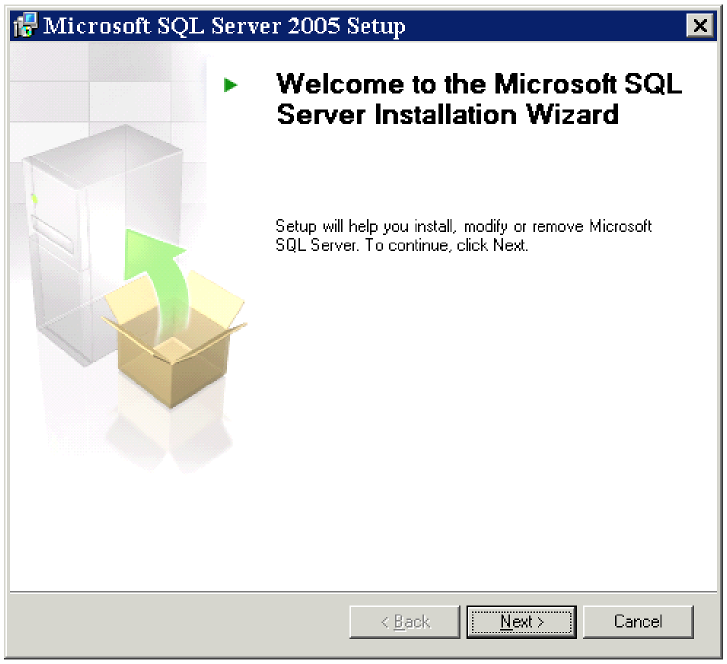
Installs software components required prior to installing SQL Server.

SQL Server Component Update will install the following components required for SQL Server Setup:

Microsoft SQL Native Client
Microsoft SQL Server 2005 Setup Support Files

Click Install to continue.

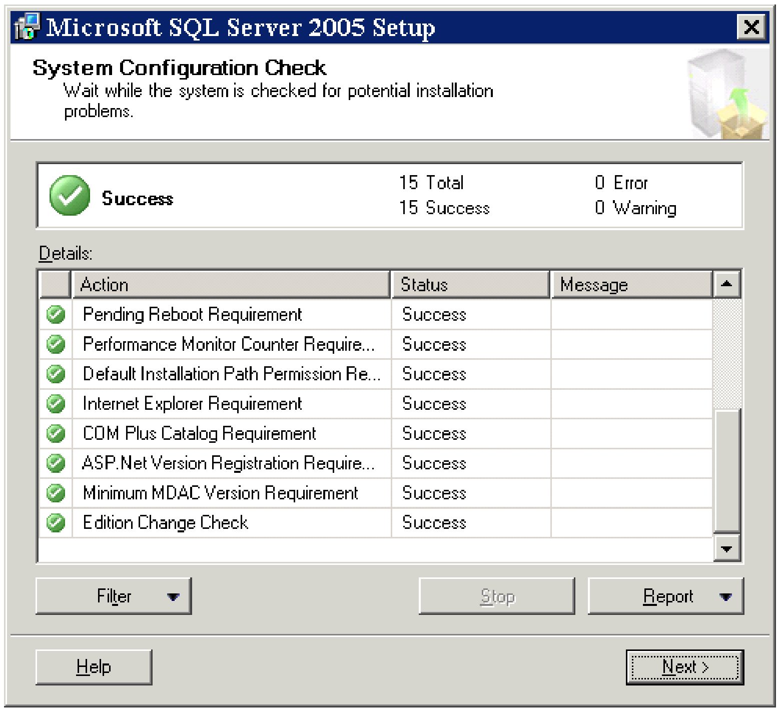
Click Next to begin the server installation:



Welcome to the Microsoft SQL Server Installation Wizard

Setup will help you install, modify or remove Microsoft SQL Server. To continue, click Next.

If all went well, click Next:



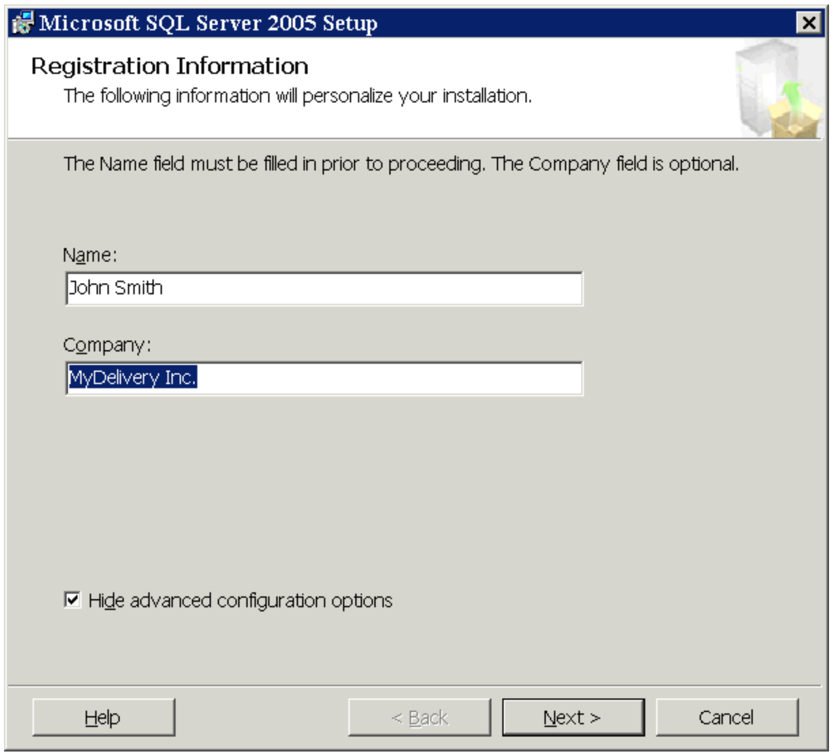
System Configuration Check
Wait while the system is checked for potential installation problems.

Success — 15 Total, 15 Success, 0 Error, 0 Warning

Details:

Action	Status	Message
Pending Reboot Requirement	Success	
Performance Monitor Counter Require...	Success	
Default Installation Path Permission Re...	Success	
Internet Explorer Requirement	Success	
COM Plus Catalog Requirement	Success	
ASP.Net Version Registration Require...	Success	
Minimum MDAC Version Requirement	Success	
Edition Change Check	Success	

Fill in your name and company, then click Next:



Registration Information

The following information will personalize your installation.

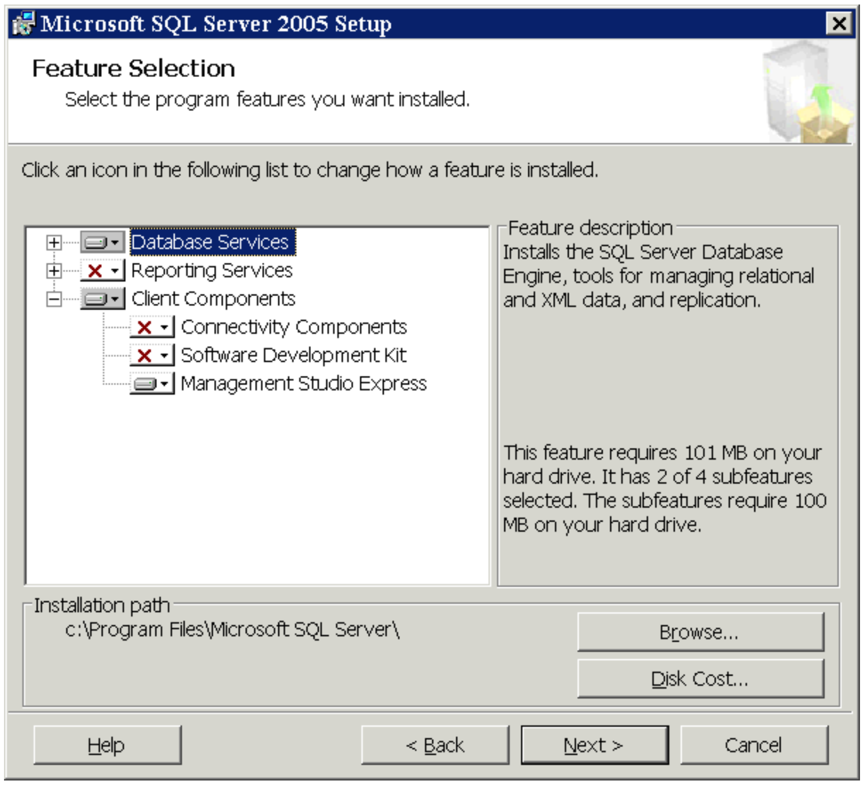
The Name field must be filled in prior to proceeding. The Company field is optional.

Name:
John Smith

Company:
MyDelivery Inc.

☑ Hide advanced configuration options

Ask for Database Service, Client Components, and Management Studio Express:



Feature Selection

Select the program features you want installed.

Click an icon in the following list to change how a feature is installed.

- Database Services
- ✗ Reporting Services
- Client Components
  - ✗ Connectivity Components
  - ✗ Software Development Kit
  - Management Studio Express

Feature description

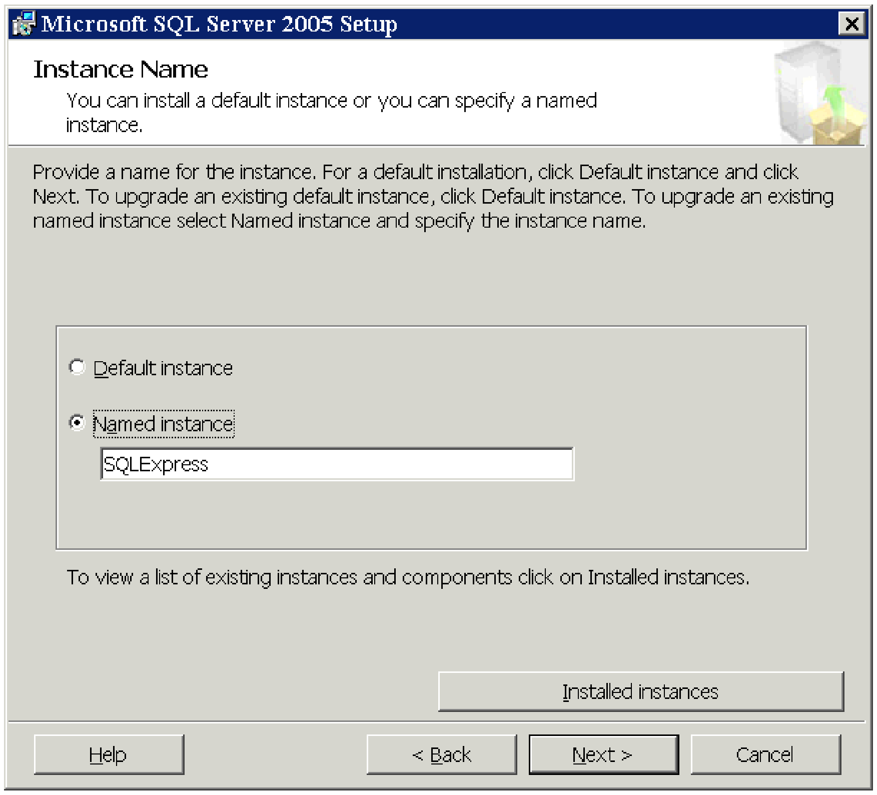
Installs the SQL Server Database Engine, tools for managing relational and XML data, and replication.

This feature requires 101 MB on your hard drive. It has 2 of 4 subfeatures selected. The subfeatures require 100 MB on your hard drive.

Installation path

c:\Program Files\Microsoft SQL Server\

Select the default named instance, SQLExpress:



Microsoft SQL Server 2005 Setup

Instance Name

You can install a default instance or you can specify a named instance.

Provide a name for the instance. For a default installation, click Default instance and click Next. To upgrade an existing default instance, click Default instance. To upgrade an existing named instance select Named instance and specify the instance name.

○ Default instance

● Named instance

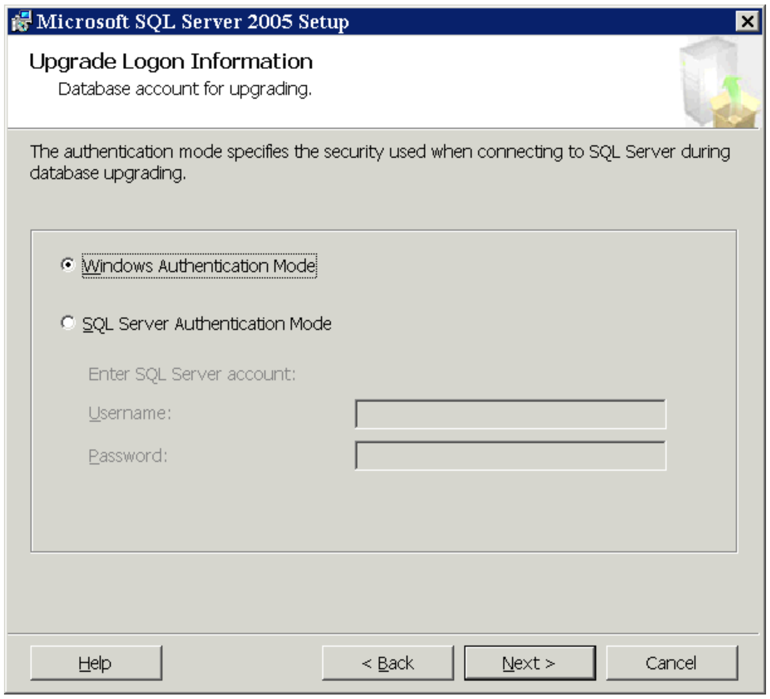
SQLExpress

To view a list of existing instances and components click on Installed instances.

Installed instances

Help | < Back | Next > | Cancel

Select Windows Authentification Mode to create a more secure access to the SQL Server:



Microsoft SQL Server 2005 Setup

Upgrade Logon Information

Database account for upgrading.

The authentication mode specifies the security used when connecting to SQL Server during database upgrading.

● Windows Authentication Mode

○ SQL Server Authentication Mode

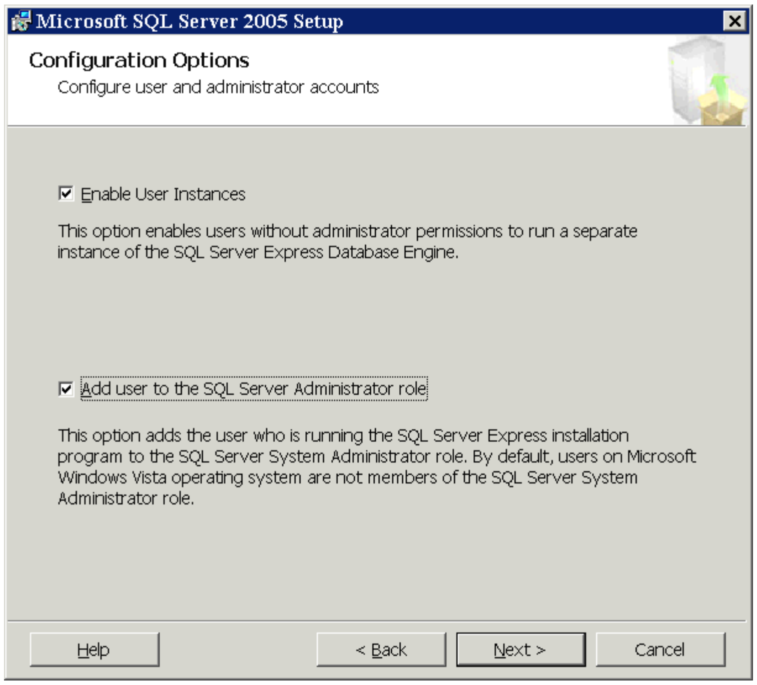
Enter SQL Server account:

Username:

Password:

Help | < Back | Next > | Cancel

Enable the user instance. It usually is not necessary to add the user to the SQL Server administrator role.



Configuration Options

Configure user and administrator accounts

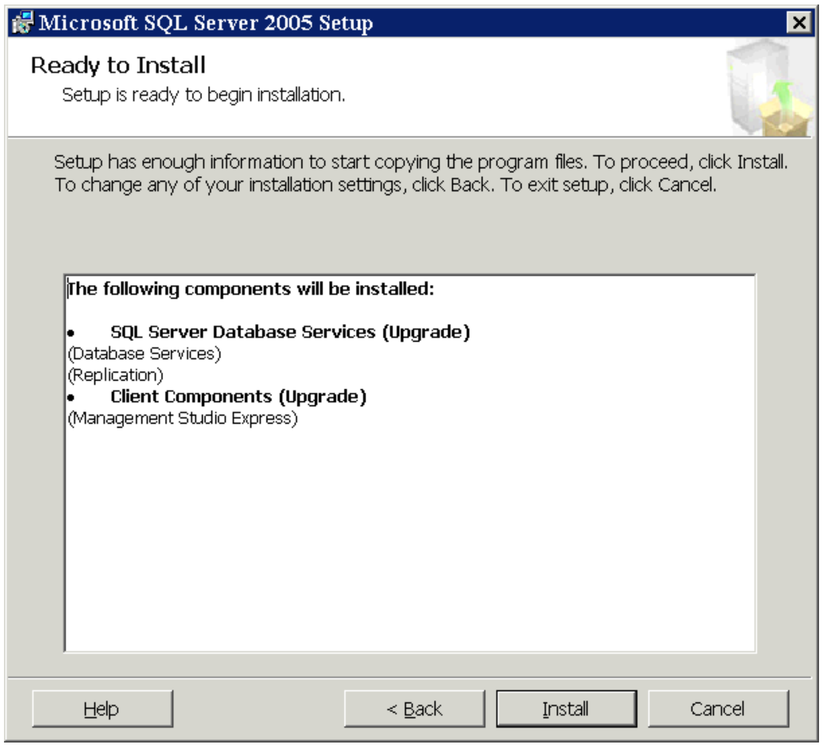
[✓] Enable User Instances

This option enables users without administrator permissions to run a separate instance of the SQL Server Express Database Engine.

[✓] Add user to the SQL Server Administrator role

This option adds the user who is running the SQL Server Express installation program to the SQL Server System Administrator role. By default, users on Microsoft Windows Vista operating system are not members of the SQL Server System Administrator role.

Click Install to continue:



Ready to Install

Setup is ready to begin installation.

Setup has enough information to start copying the program files. To proceed, click Install. To change any of your installation settings, click Back. To exit setup, click Cancel.

The following components will be installed:

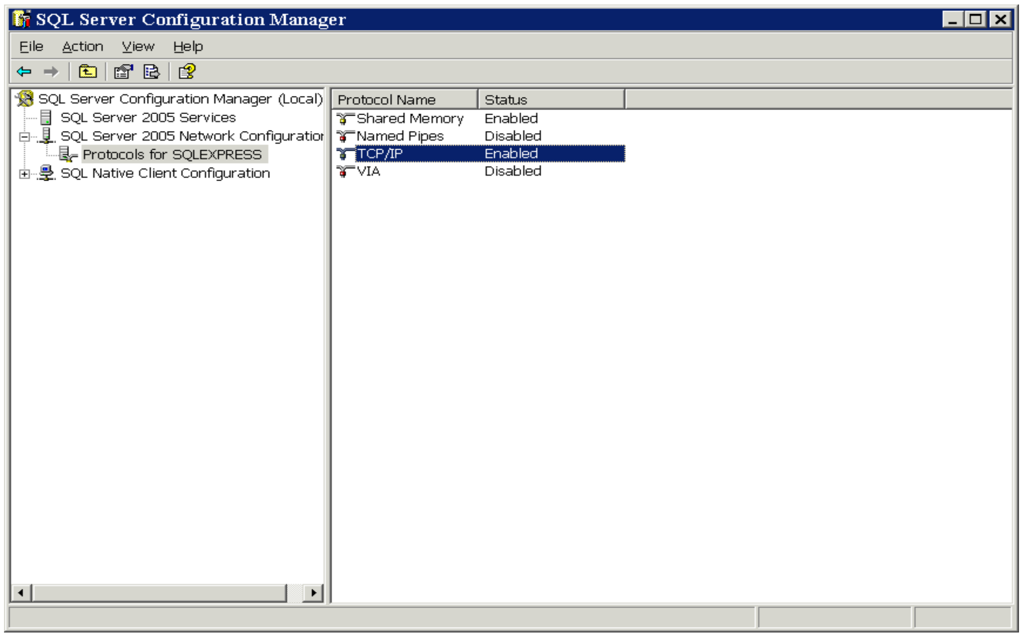
- **SQL Server Database Services (Upgrade)**

(Database Services)
(Replication)

- **Client Components (Upgrade)**

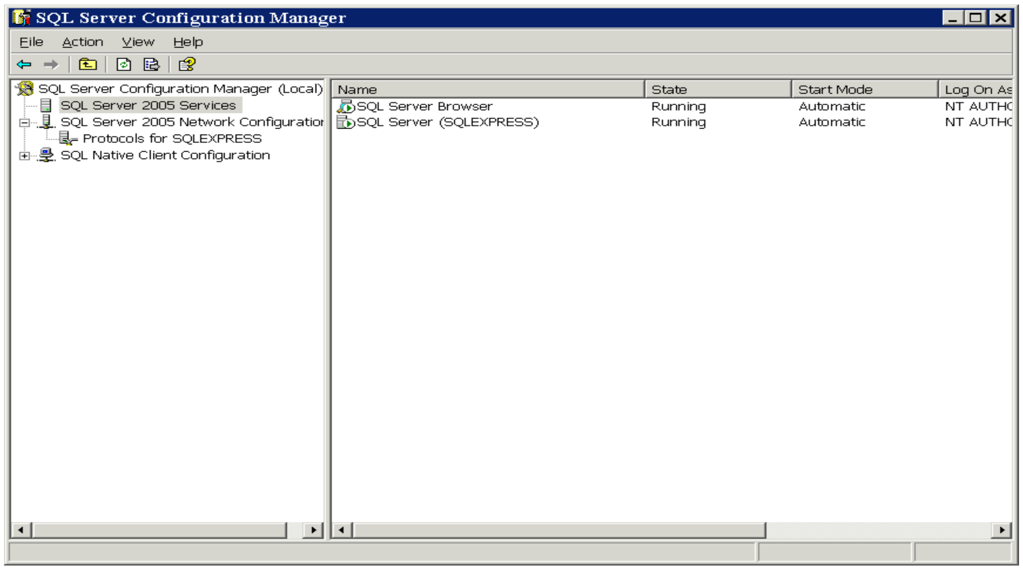
(Management Studio Express)

After installation is complete, go to Administrative Tools and run Services. For the SQL Server Browser service, make this service run Automatic, then go ahead and start the service. Next, run the SQL Server Configuration Manager. Under SQL Server Network Configuration, select Protocols. Enable the TCP/IP protocol:


SQL Server Configuration Manager
File Action View Help
SQL Server Configuration Manager (Local)
SQL Server 2005 Services
SQL Server 2005 Network Configuratior
Protocols for SQLEXPRESS
SQL Native Client Configuration

Protocol Name	Status
Shared Memory	Enabled
Named Pipes	Disabled
TCP/IP	Enabled
VIA	Disabled

Next, go to SQL Server 2005 Services and ensure that both that SQL Server Browser and SQL Server (Express) are both running:

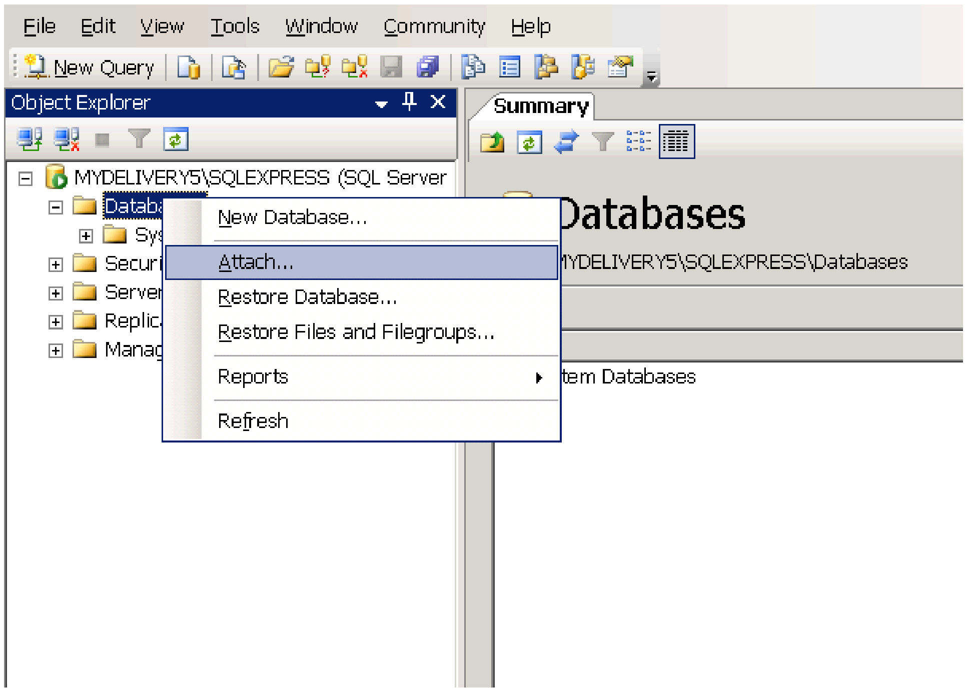

SQL Server Configuration Manager
File Action View Help
SQL Server Configuration Manager (Local)
SQL Server 2005 Services
SQL Server 2005 Network Configuratior
Protocols for SQLEXPRESS
SQL Native Client Configuration

Name	State	Start Mode	Log On As
SQL Server Browser	Running	Automatic	NT AUTHC
SQL Server (SQLEXPRESS)	Running	Automatic	NT AUTHC

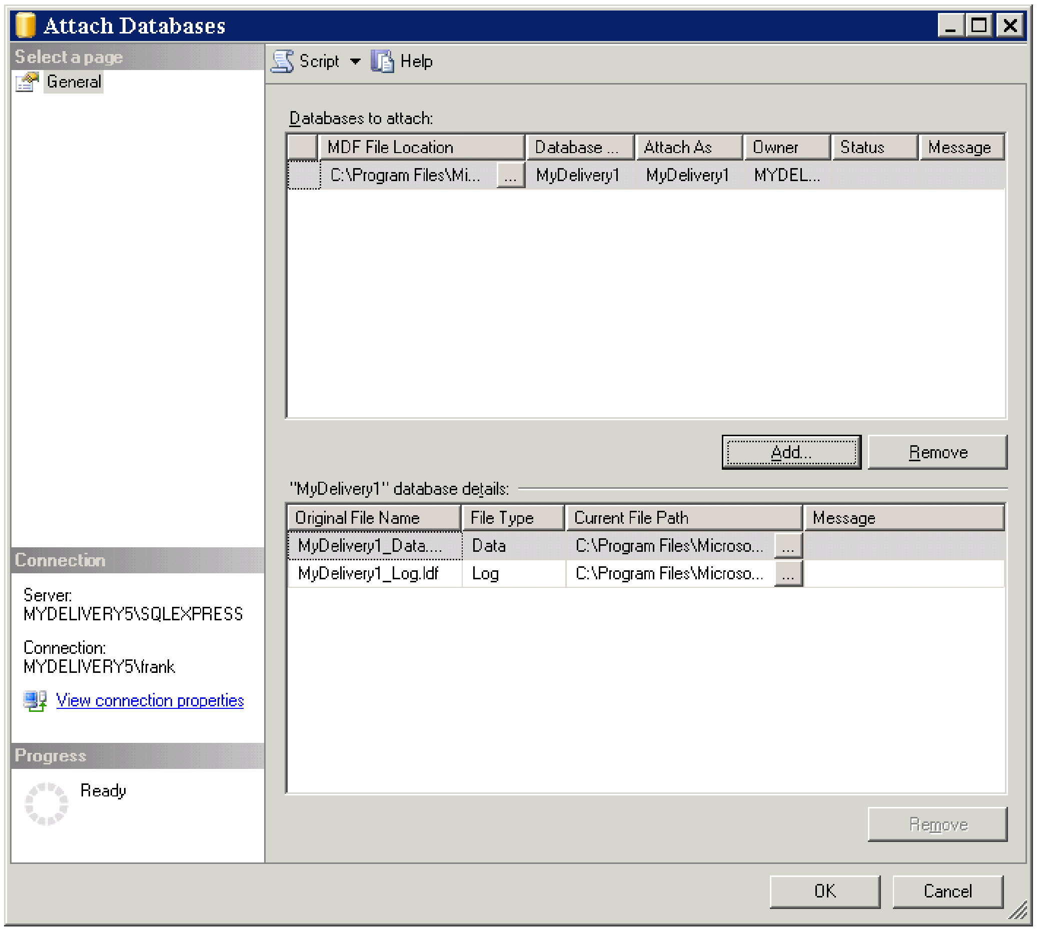
Next, run SQL Server Management Studio Express. It is time to install the MyDelivery database and its transaction log file, named MyDelivery1_data.mdf and MyDelivery1_log.ldr, respectively. Copy these two files to the default folder for SQL Server databases:

C:\program files\Microsoft SQL Server\MSSQL.1\data.

In Management Studio, right-click databases, and select Attach:



Next, click Add to add the database just copied to hard disk:



Databases to attach:

	MDF File Location	Database ...	Attach As	Owner	Status	Message
	C:\Program Files\Mi...	MyDelivery1	MyDelivery1	MYDEL...		

"MyDelivery1" database details:

Original File Name	File Type	Current File Path	Message
MyDelivery1_Data....	Data	C:\Program Files\Microso...	
MyDelivery1_Log.ldf	Log	C:\Program Files\Microso...	

Connection

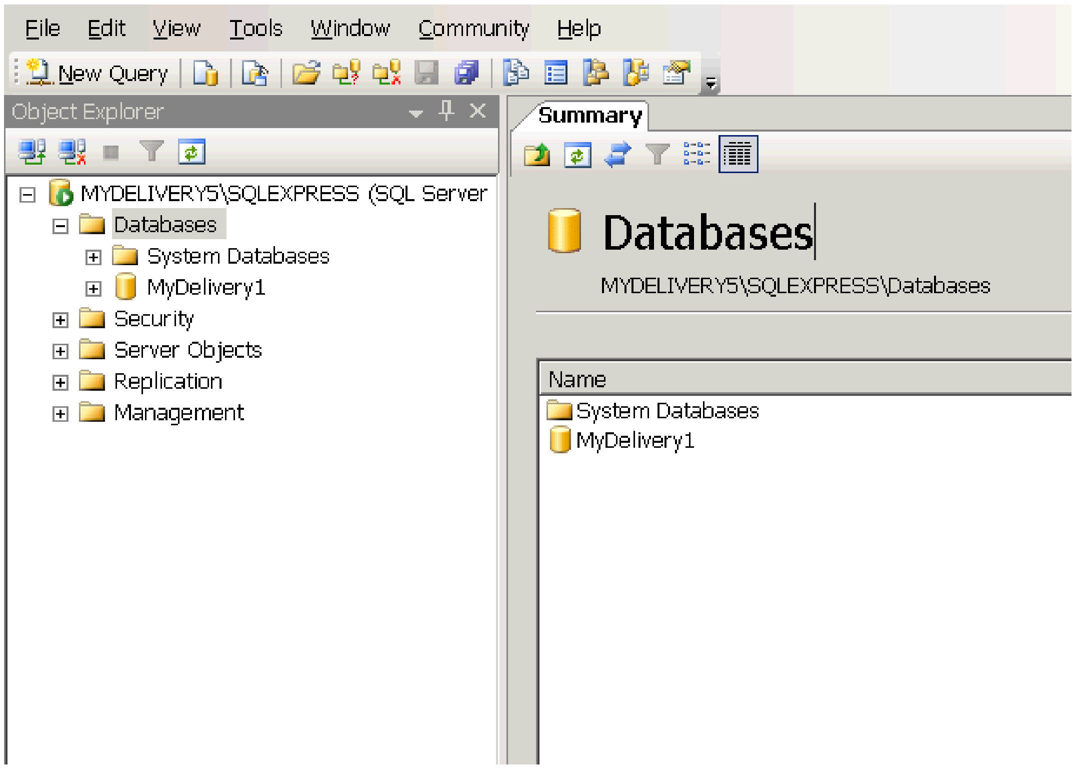
Server:
MYDELIVERY5\SQLEXPRESS

Connection:
MYDELIVERY5\frank

Progress

Ready

At this point the MyDelivery1 database has been added, and can be viewed or updated through Management Express. This completes the installation of SQL Server Express and the MyDelivery database.

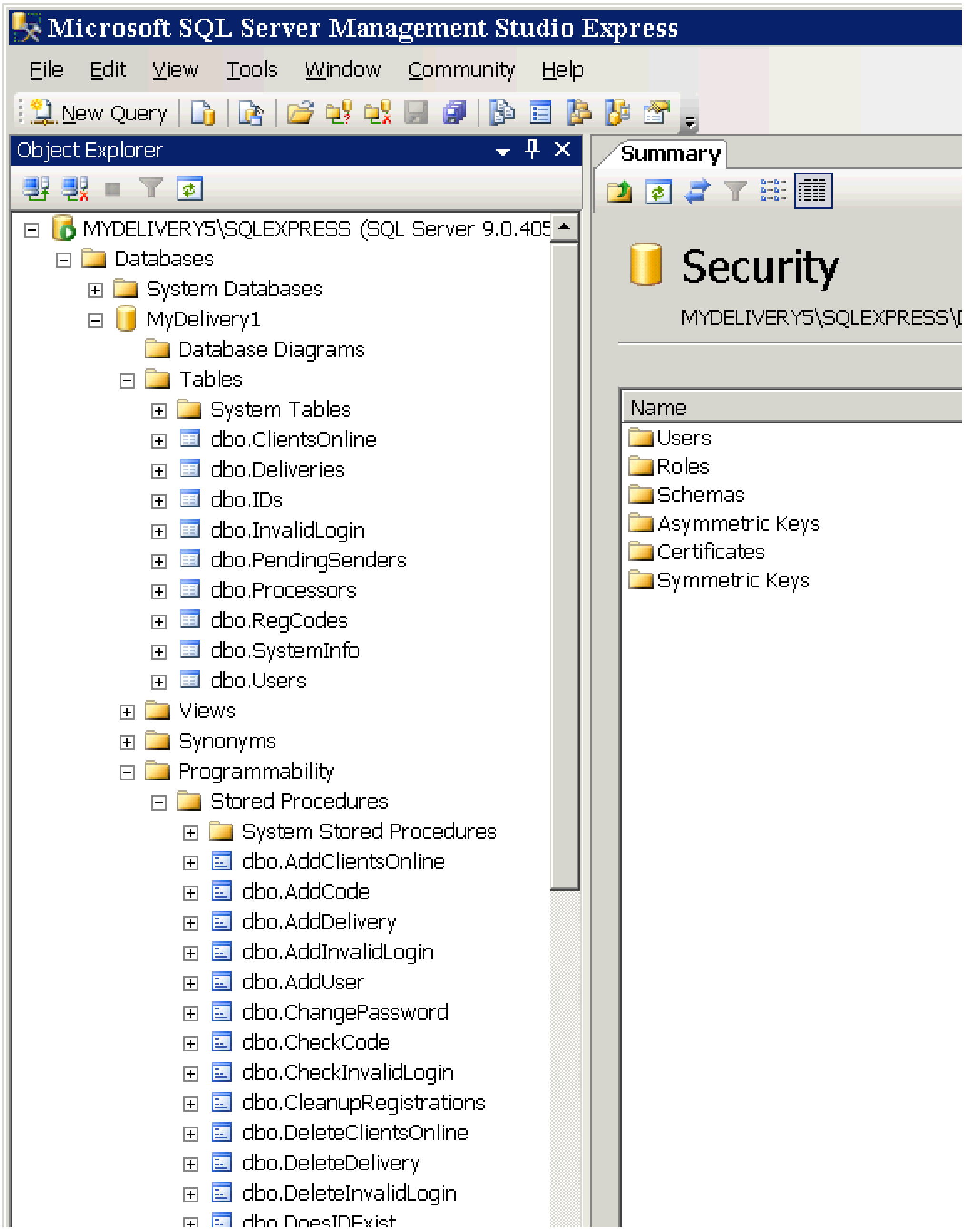


MYDELIVERY5\SQLEXPRESS (SQL Server
Databases
System Databases
MyDelivery1
Security
Server Objects
Replication
Management

Databases
MYDELIVERY5\SQLEXPRESS\Databases

Name
System Databases
MyDelivery1

The next figure shows the two important sections of the MyDelivery1 database: the tables and the stored procedures. Descriptions of the tables are given in the next section of this document.


Microsoft SQL Server Management Studio Express
File Edit View Tools Window Community Help
New Query
Object Explorer
Summary

MYDELIVERY5\SQLEXPRESS (SQL Server 9.0.405
Databases
System Databases
MyDelivery1
Database Diagrams
Tables
System Tables
dbo.ClientsOnline
dbo.Deliveries
dbo.IDs
dbo.InvalidLogin
dbo.PendingSenders
dbo.Processors
dbo.RegCodes
dbo.SystemInfo
dbo.Users
Views
Synonyms
Programmability
Stored Procedures
System Stored Procedures
dbo.AddClientsOnline
dbo.AddCode
dbo.AddDelivery
dbo.AddInvalidLogin
dbo.AddUser
dbo.ChangePassword
dbo.CheckCode
dbo.CheckInvalidLogin
dbo.CleanupRegistrations
dbo.DeleteClientsOnline
dbo.DeleteDelivery
dbo.DeleteInvalidLogin
dbo.DoesIDExist

Security

MYDELIVERY5\SQLEXPRESS\

Name
Users
Roles
Schemas
Asymmetric Keys
Certificates
Symmetric Keys

7.0 Implementation of Database Mirroring via SQL Server 2005

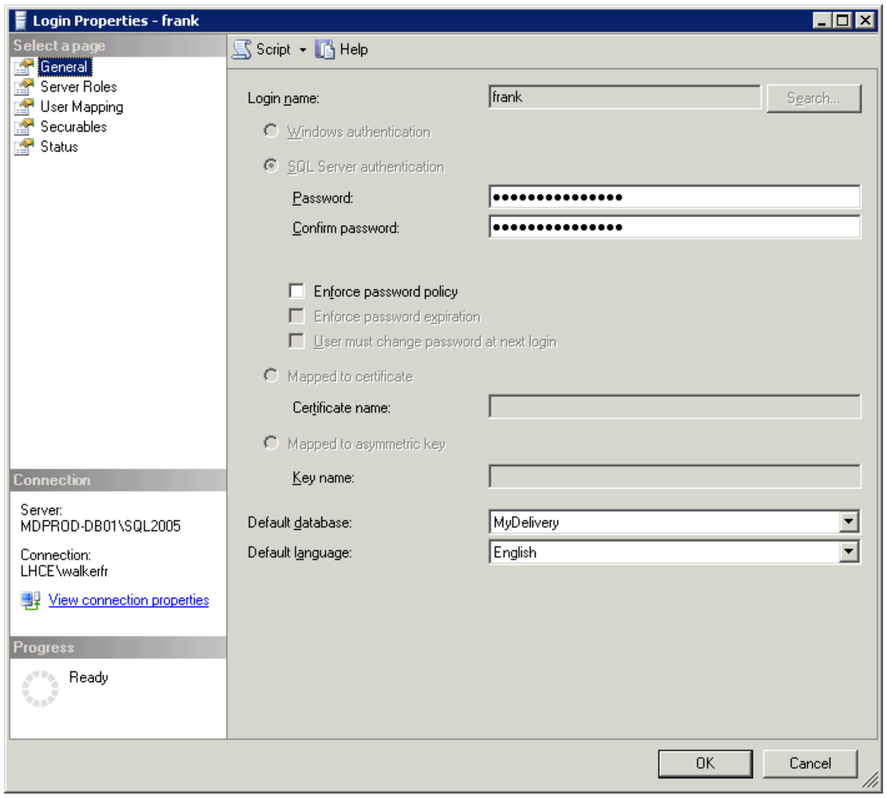
While MyDelivery will run well using just one database server, its reliability will increase by using two database servers with mirroring. This section shows how to set up two SQL Server 2005 Standard edition database servers for database mirroring. For explanatory purposes here these two machines will be named mdprod-db01 and mdprod-db02. For your own installation, you can name your machines whatever you wish. For database mirroring, there is a Principal (initially, mdprod-db01) and a mirror (initially, mdprod-db02). There is a third computer called the Witness. This will be installed and running under SQL Server 2005 Express edition on a third machine named mdprod-manager, on which also runs the mdClientManager service. The witness, principal and mirror will be configured to have High Safety with Automatic Failover. If mdprod-db01 (and its SQL Server) shall ever go down, then all database clients (running on the Front-End Processors) will automatically switch to the SQL server running on mdprod-db02. The latter becomes the principal server. Once mdprod-db01 is restored to operational status, it becomes the mirror. When the two databases are both operational, they are fully synchronized at all times. In other words, one MyDelivery database is a precise copy of the other. It is possible for one of the three computers (mdprod-db01, mdprod-db02 and mdprod-manager) to be down at a time, and the "database server system" is still fully functional. Although the name of the database distributed with the MyDelivery source code is 'MyDelivery1', this section refers to the database as 'MyDelivery'. The screen captures were made at an earlier date when the original database name was 'MyDelivery'. Wherever a screen capture shows the MyDelivery database, it should be MyDelivery1.

7.1 Setting up Security

There should be a common service account for administering all three machines: mdprod-db01, mdprod-db02 and mdprod-manager. Let's name this service account svc_mydelivery. When logged in as the database service account, svc_mydelivery, install SQL Server 2005 Standard edition on both mdprod-db01 and mdprod-db02. Similarly, log in as svc_mydelivery and install SQL Server 2005 Express edition on mdprod-manager. Also, install SQL Server Management Studio Express on mdprod-manager. The SQL Server services shall all be run under the svc_mydelivery account on all three machines.

Run Microsoft SQL Server Management Studio on both mdprod-db01 and mdprod-db02. In the Security settings for the SQL Server, make sure that there is a svc_mydelivery user. The svc_mydelivery user is used by all the Front-End Processors and mdprod-manager to access the MyDelivery database on the Principal machine. The next several screen captures show the default settings for the svc_mydelivery user.

The next picture shows the "General" tab settings for the svc_mydelivery user.



Login name: frank

( ) Windows authentication

(●) SQL Server authentication

Password: •••••••••••••••

Confirm password: •••••••••••••••

[ ] Enforce password policy

[ ] Enforce password expiration

[ ] User must change password at next login

( ) Mapped to certificate

Certificate name:

( ) Mapped to asymmetric key

Key name:

Default database: MyDelivery

Default language: English

Connection

Server:
MDPROD-DB01\SQL2005

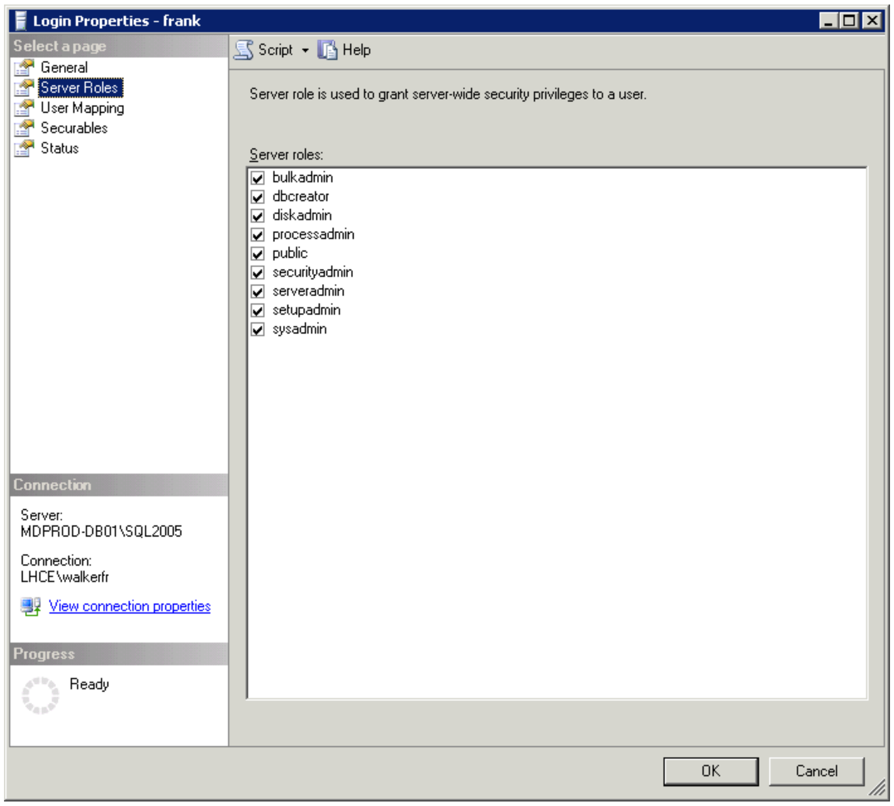
Connection:
LHCE\walkerfr

View connection properties

Progress

Ready

The next picture show the “Server Roles” tab settings for the “svc_mydelivery” user.



Server role is used to grant server-wide security privileges to a user.

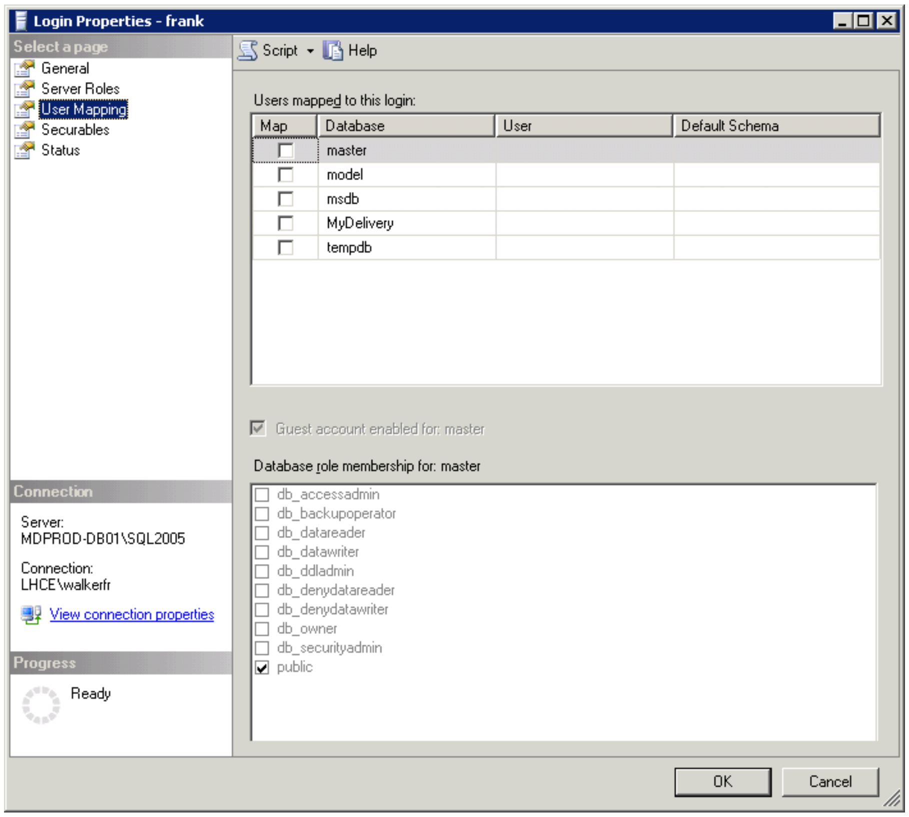
Server roles:

- [✓] bulkadmin
- [✓] dbcreator
- [✓] diskadmin
- [✓] processadmin
- [✓] public
- [✓] securityadmin
- [✓] serveradmin
- [✓] setupadmin
- [✓] sysadmin

Server:
MDPROD-DB01\SQL2005

Connection:
LHCE\walkerfr

This shows the “User Mapping” for the “svc_mydelivery” user.



Users mapped to this login:

Map	Database	User	Default Schema
☐	master		
☐	model		
☐	msdb		
☐	MyDelivery		
☐	tempdb		

[✓] Guest account enabled for: master

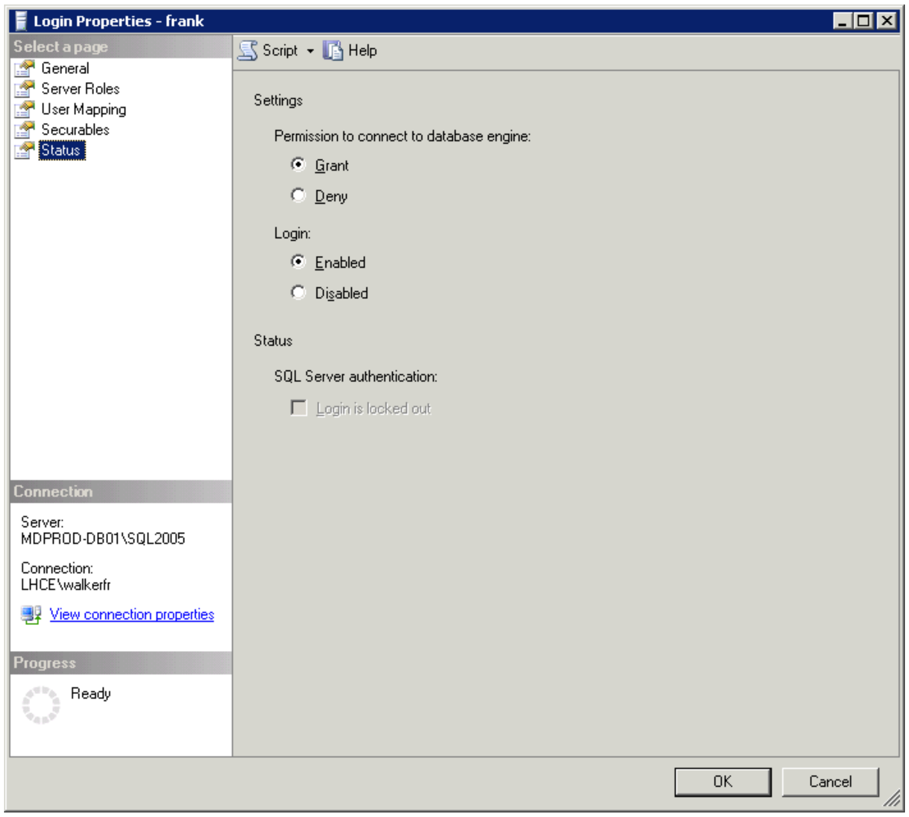
Database role membership for: master

- [ ] db_accessadmin
- [ ] db_backupoperator
- [ ] db_datareader
- [ ] db_datawriter
- [ ] db_ddladmin
- [ ] db_denydatareader
- [ ] db_denydatawriter
- [ ] db_owner
- [ ] db_securityadmin
- [✓] public

Server:
MDPROD-DB01\SQL2005

Connection:
LHCE\walkerfr

The next picture shows the “Status” for the “svc_mydelivery” user.


Login Properties - frank

Select a page: General, Server Roles, User Mapping, Securables, Status

Settings

Permission to connect to database engine:

[● Grant] ○ Deny

Login:

[● Enabled] ○ Disabled

Status

SQL Server authentication:

☐ Login is locked out

Connection

Server:
MDPROD-DB01\SQL2005

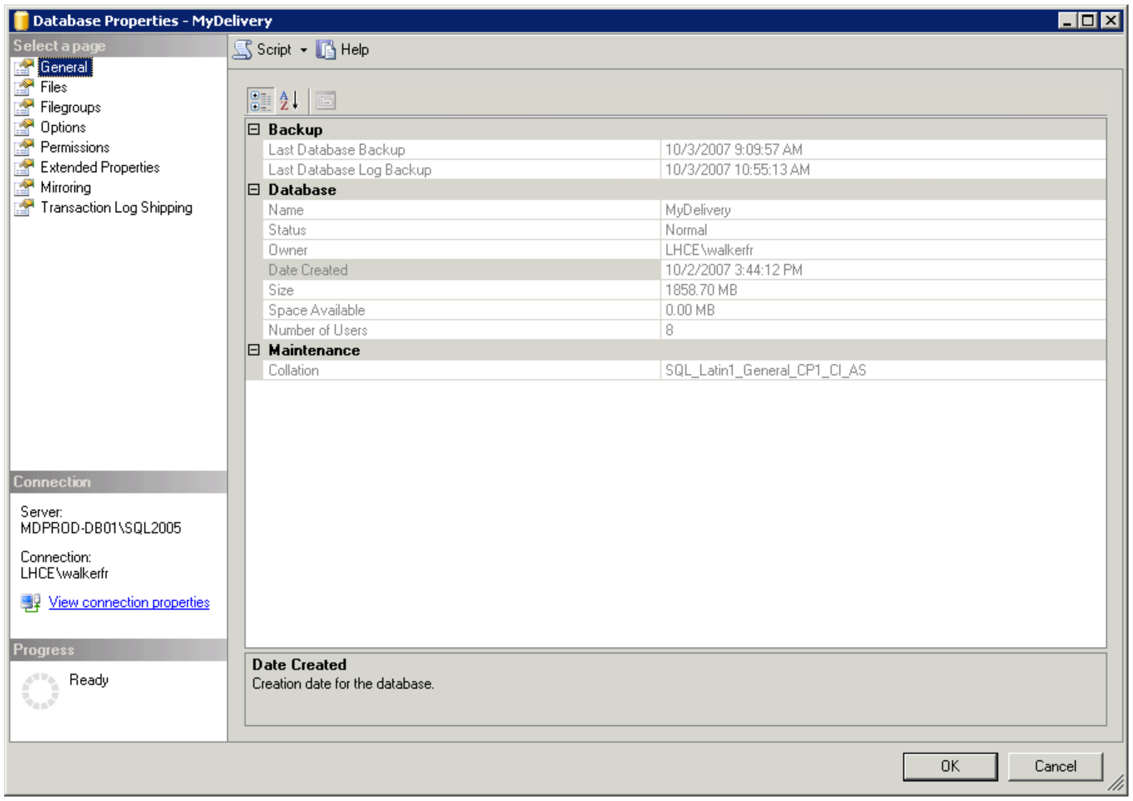
Connection:
LHCE\walkerfr

View connection properties

Progress

Ready

Next we show the properties of the MyDelivery database that we install on the mdprod-db01 server. These are the “General” properties.



Database Properties - MyDelivery

Select a page: General, Files, Filegroups, Options, Permissions, Extended Properties, Mirroring, Transaction Log Shipping

Script Help

Backup	
Last Database Backup	10/3/2007 9:09:57 AM
Last Database Log Backup	10/3/2007 10:55:13 AM
Database	
Name	MyDelivery
Status	Normal
Owner	LHCE\walkerfr
Date Created	10/2/2007 3:44:12 PM
Size	1858.70 MB
Space Available	0.00 MB
Number of Users	8
Maintenance	
Collation	SQL_Latin1_General_CP1_CI_AS

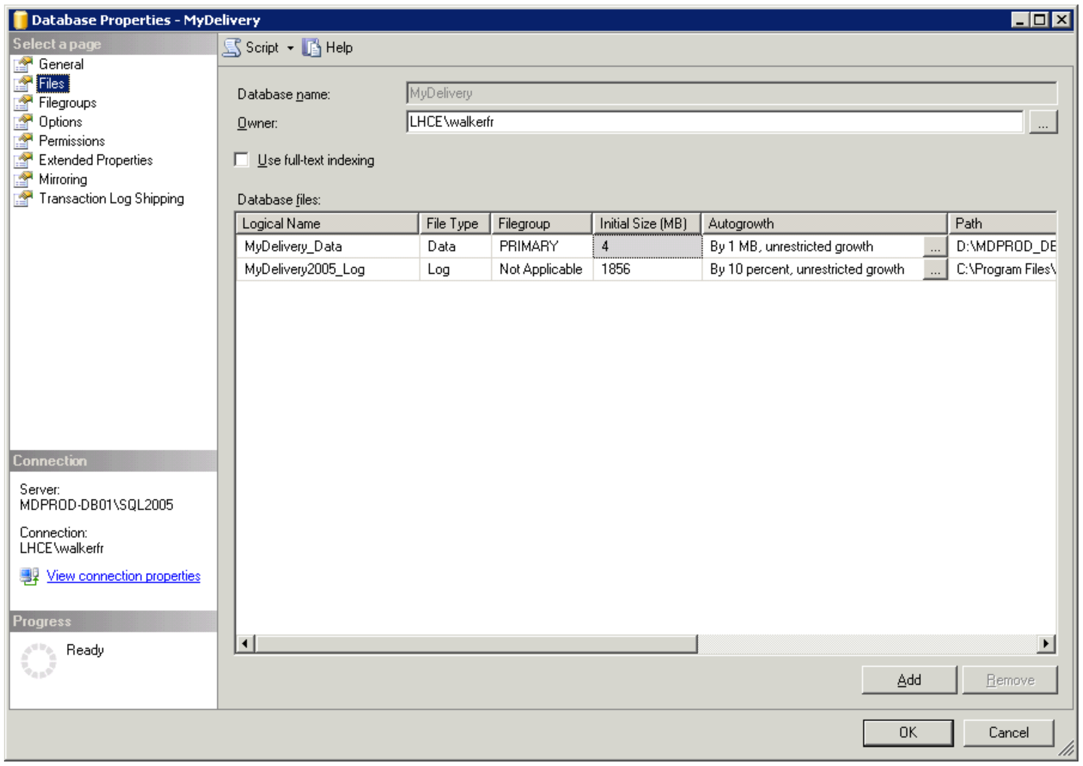
Date Created
Creation date for the database.

Connection
Server:
MDPROD-DB01\SQL2005
Connection:
LHCE\walkerfr
View connection properties

Progress
Ready

OK Cancel

These are the left portion of the “Files” properties.



Database Properties - MyDelivery

Select a page: General, Files, Filegroups, Options, Permissions, Extended Properties, Mirroring, Transaction Log Shipping

Script Help

Database name: MyDelivery
Owner: LHCE\walkerfr
Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth	Path
MyDelivery_Data	Data	PRIMARY	4	By 1 MB, unrestricted growth	D:\MDPROD_DE
MyDelivery2005_Log	Log	Not Applicable	1856	By 10 percent, unrestricted growth	C:\Program Files\

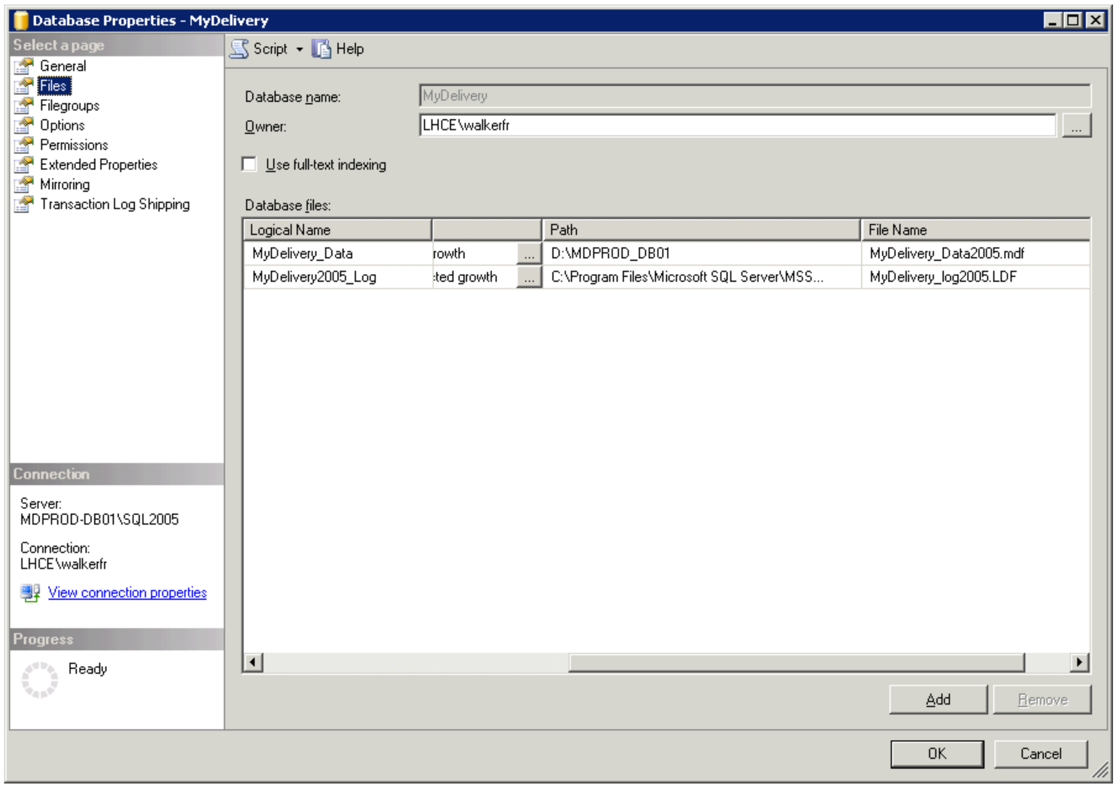
Add Remove

Connection
Server:
MDPROD-DB01\SQL2005
Connection:
LHCE\walkerfr
View connection properties

Progress
Ready

OK Cancel

These are the right portion of the “Files” properties.



Database name: MyDelivery

Owner: LHCE\walkerfr

Use full-text indexing

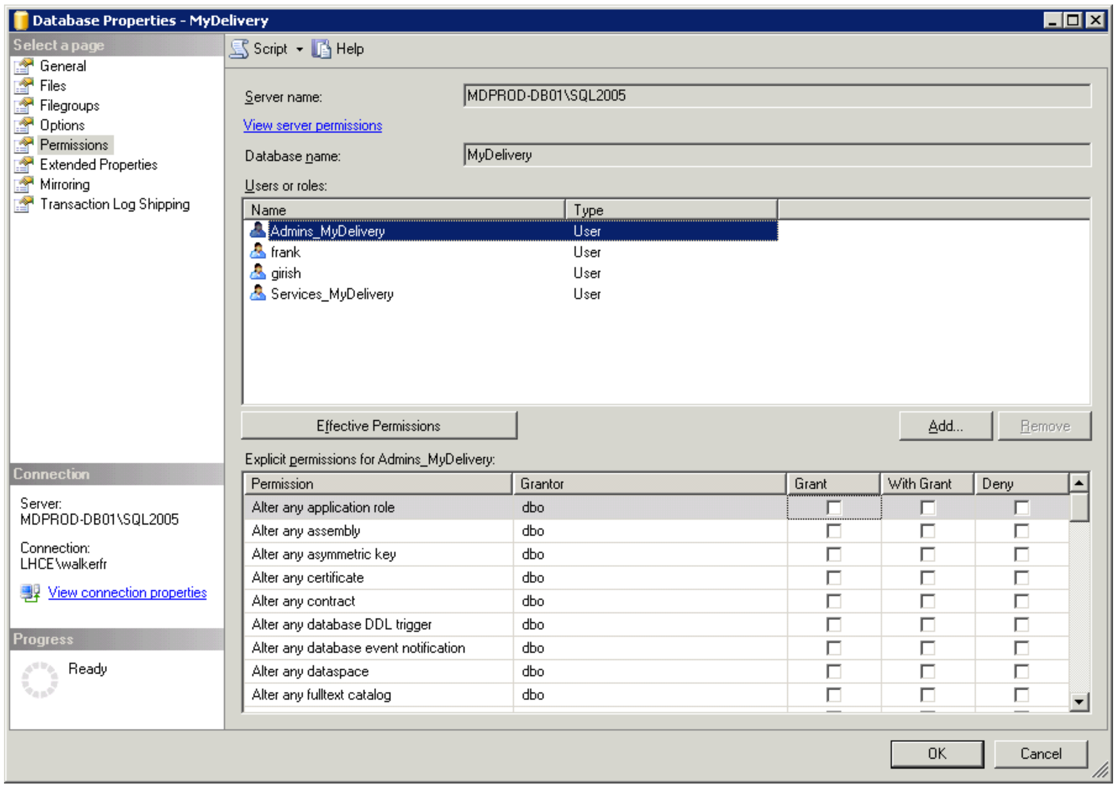
Server: MDPROD-DB01\SQL2005

Connection: LHCE\walkerfr

Database files:

Logical Name		Path	File Name
MyDelivery_Data	rowth	D:\MDPROD_DB01	MyDelivery_Data2005.mdf
MyDelivery2005_Log	ted growth	C:\Program Files\Microsoft SQL Server\MSS...	MyDelivery_log2005.LDF

These are the “Permissions.”



Database Properties - MyDelivery

Server name: MDPROD-DB01\SQL2005

View server permissions

Database name: MyDelivery

Users or roles:

Name	Type
Admins_MyDelivery	User
frank	User
girish	User
Services_MyDelivery	User

Explicit permissions for Admins_MyDelivery:

Permission	Grantor	Grant	With Grant	Deny
Alter any application role	dbo	☐	☐	☐
Alter any assembly	dbo	☐	☐	☐
Alter any asymmetric key	dbo	☐	☐	☐
Alter any certificate	dbo	☐	☐	☐
Alter any contract	dbo	☐	☐	☐
Alter any database DDL trigger	dbo	☐	☐	☐
Alter any database event notification	dbo	☐	☐	☐
Alter any dataspace	dbo	☐	☐	☐
Alter any fulltext catalog	dbo	☐	☐	☐

Connection

Server:
MDPROD-DB01\SQL2005

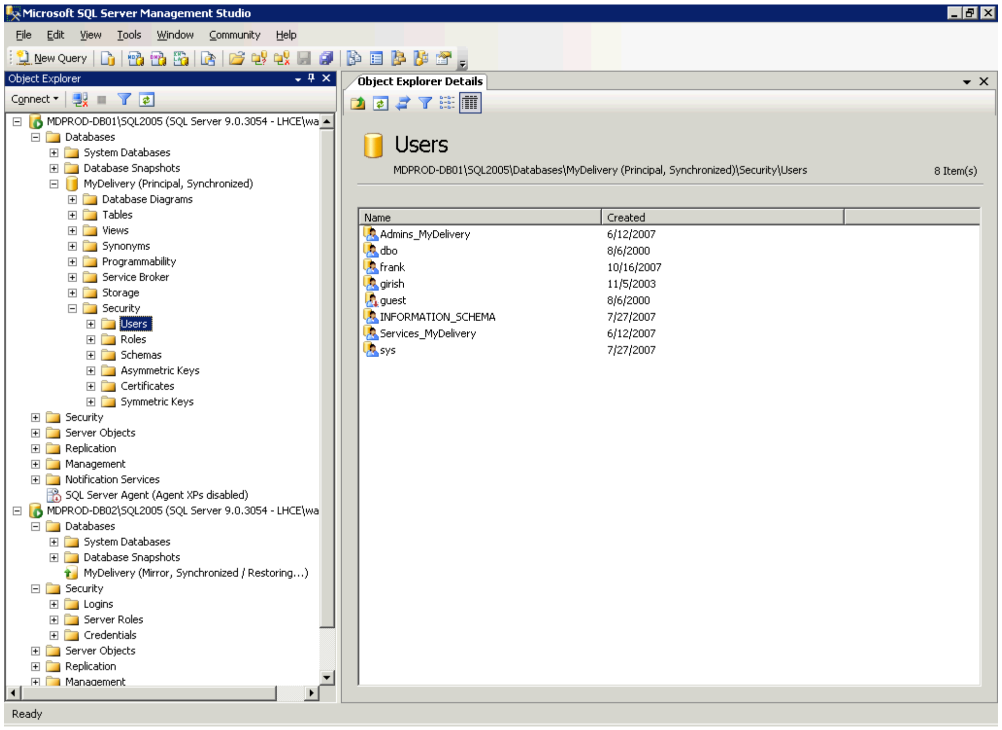
Connection:
LHCE\walkerfr

View connection properties

Progress

Ready

Under “Security” for the MyDelivery database, these are the users:



Microsoft SQL Server Management Studio

Object Explorer

```
MDPROD-DB01\SQL2005 (SQL Server 9.0.3054 - LHCE\wa
  Databases
    System Databases
    Database Snapshots
    MyDelivery (Principal, Synchronized)
      Database Diagrams
      Tables
      Views
      Synonyms
      Programmability
      Service Broker
      Storage
      Security
        Users
        Roles
        Schemas
        Asymmetric Keys
        Certificates
        Symmetric Keys
  Security
  Server Objects
  Replication
  Management
  Notification Services
  SQL Server Agent (Agent XPs disabled)
MDPROD-DB02\SQL2005 (SQL Server 9.0.3054 - LHCE\wa
  Databases
    System Databases
    Database Snapshots
    MyDelivery (Mirror, Synchronized / Restoring...)
  Security
    Logins
    Server Roles
    Credentials
  Server Objects
  Replication
  Management
```

Object Explorer Details

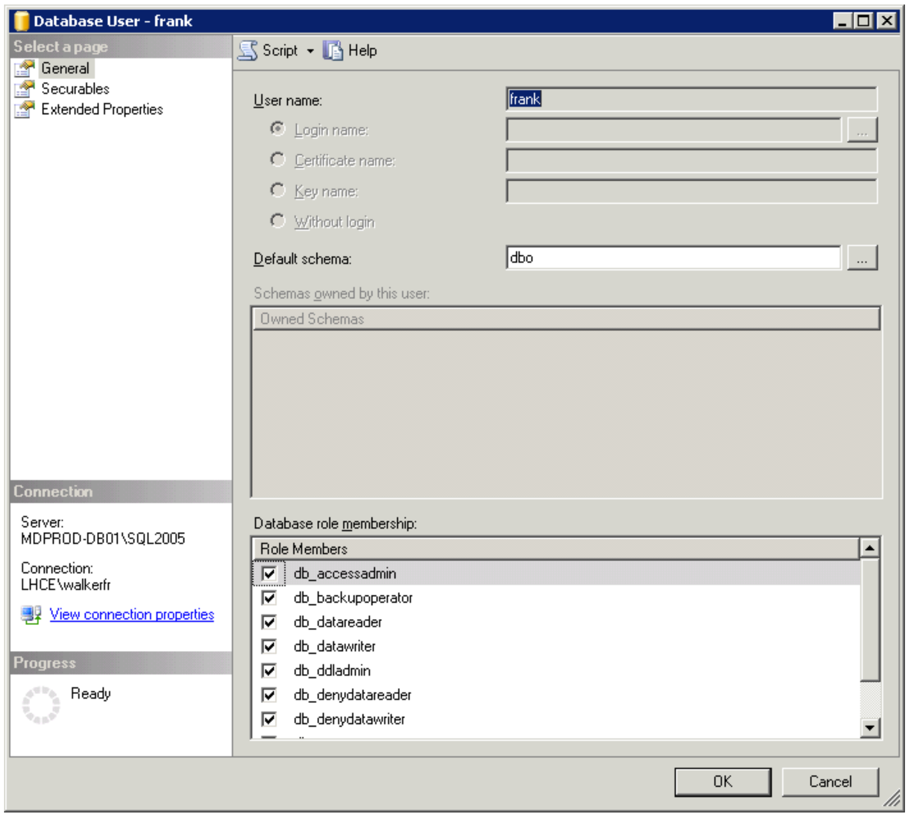
Users

MDPROD-DB01\SQL2005\Databases\MyDelivery (Principal, Synchronized)\Security\Users — 8 Item(s)

Name	Created
Admins_MyDelivery	6/12/2007
dbo	8/6/2000
frank	10/16/2007
girish	11/5/2003
guest	8/6/2000
INFORMATION_SCHEMA	7/27/2007
Services_MyDelivery	6/12/2007
sys	7/27/2007

Ready

Right-click the "svc_mydelivery" user. These are the "General" properties of "svc_mydelivery." Note that all roles are checked.



Server:
MDPROD-DB01\SQL2005

Connection:
LHCE\walkerfr

User name: frank

Login name:

Certificate name:

Key name:

Without login

Default schema: dbo

Schemas owned by this user:

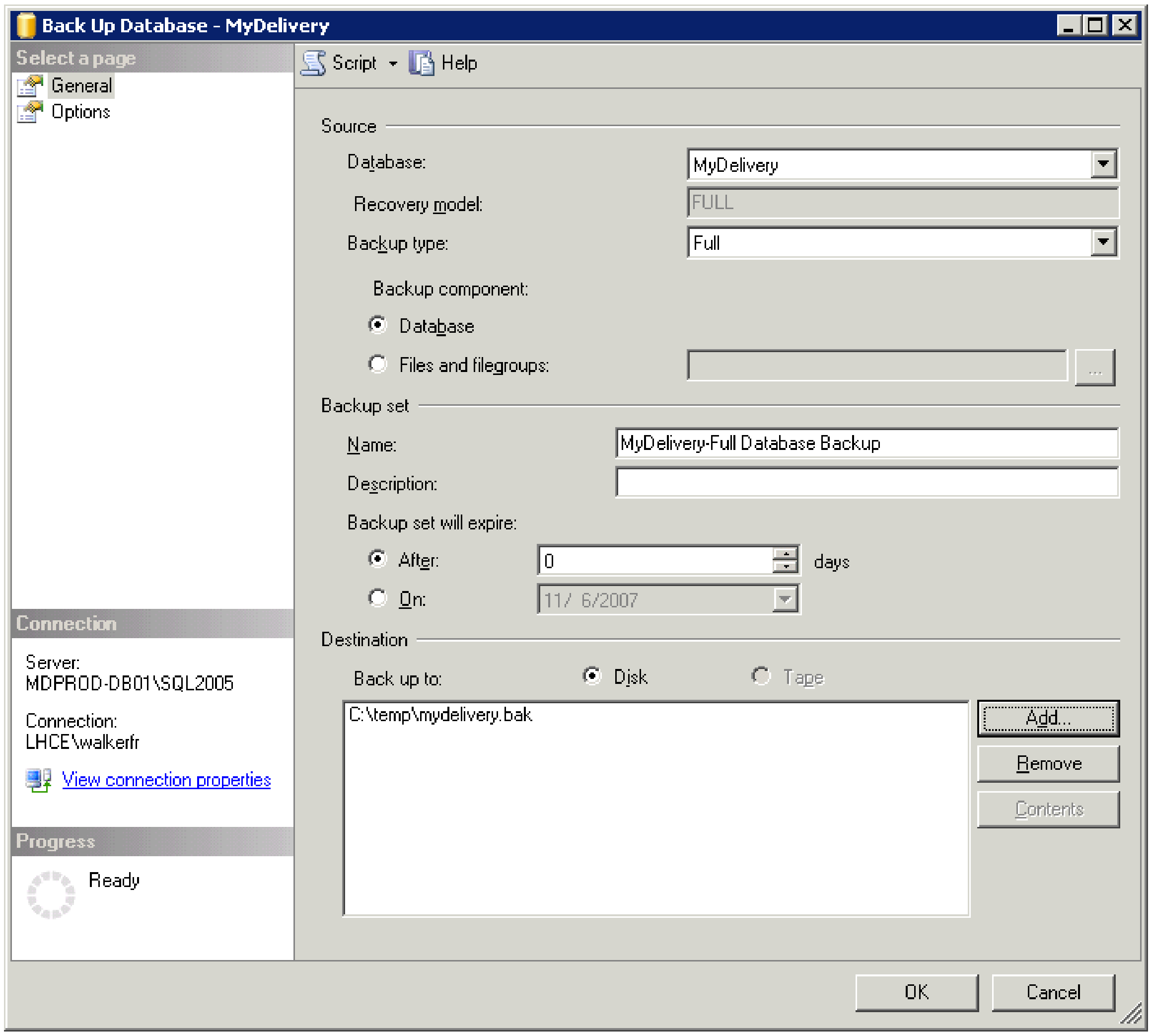
Owned Schemas

Database role membership:

Role Members
☑ db_accessadmin
☑ db_backupoperator
☑ db_datareader
☑ db_datawriter
☑ db_ddladmin
☑ db_denydatareader
☑ db_denydatawriter

7.2 Backup MyDelivery

Next, truncate both the MyDelivery database and its log file on mdprod-db01. Then, backup both the MyDelivery database on mdprod-db01, along with its log transaction file. See below for the General tab choices.


Back Up Database - MyDelivery

Select a page: General, Options

Source
Database: MyDelivery
Recovery model: FULL
Backup type: Full
Backup component: [✓ Database] Files and filegroups:

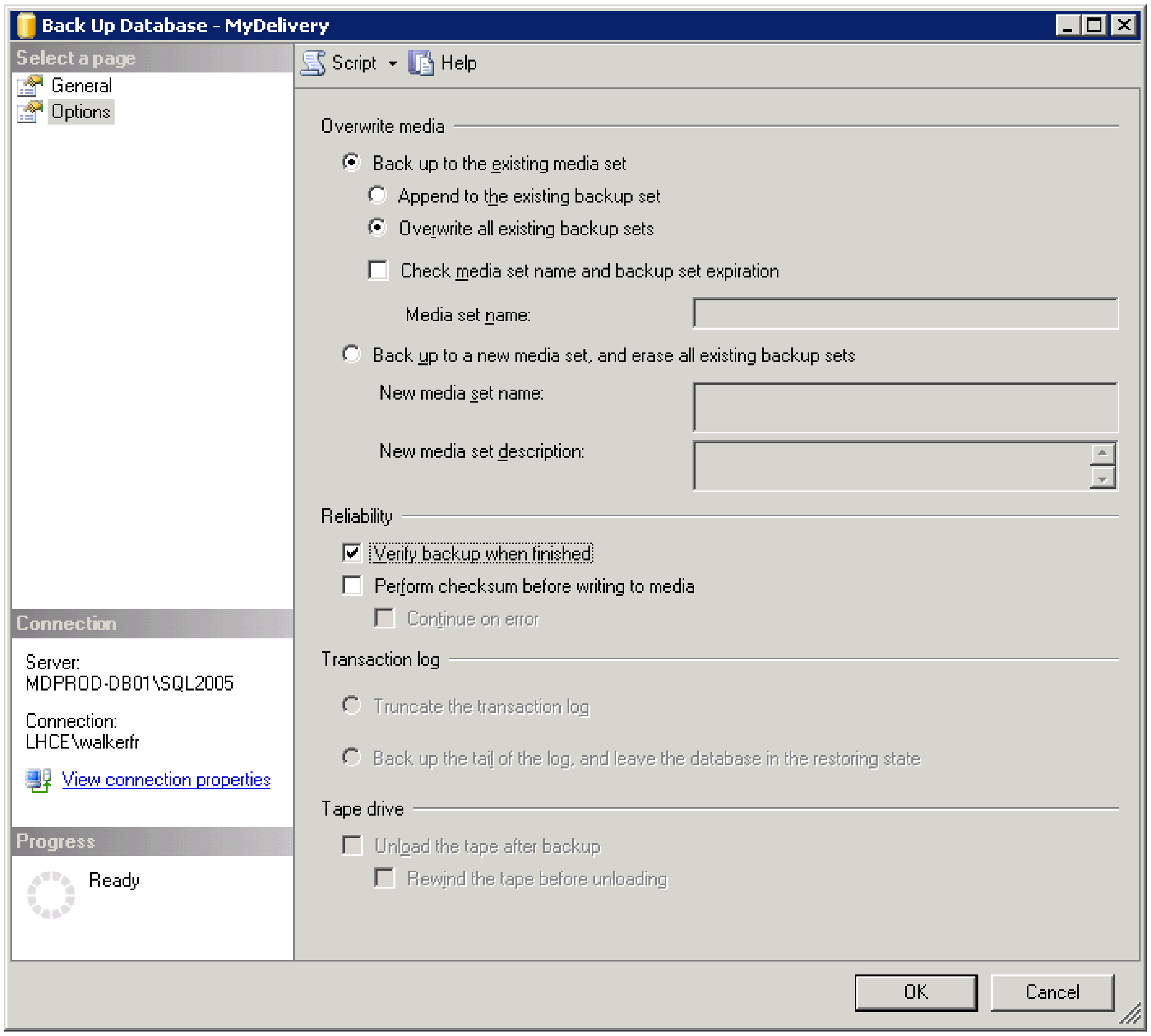
Backup set
Name: MyDelivery-Full Database Backup
Description:
Backup set will expire:
[✓ After: 0 days]
On: 11/ 6/2007

Destination
Back up to: [✓ Disk] Tape
C:\temp\mydelivery.bak

Connection
Server: MDPROD-DB01\SQL2005
Connection: LHCE\walkerfr

Progress
Ready

These are the additional options for backing up MyDelivery. Click OK. You will get a dialog box showing the backup was successful.


Back Up Database - MyDelivery

Select a page
- General
- Options

Script ▾ Help

Overwrite media

- (●) Back up to the existing media set
  - ( ) Append to the existing backup set
  - (●) Overwrite all existing backup sets
  - [ ] Check media set name and backup set expiration
  - Media set name:
- ( ) Back up to a new media set, and erase all existing backup sets
  - New media set name:
  - New media set description:

Reliability

- [✓] Verify backup when finished
- [ ] Perform checksum before writing to media
  - [ ] Continue on error

Transaction log

- ( ) Truncate the transaction log
- ( ) Back up the tail of the log, and leave the database in the restoring state

Tape drive

- [ ] Unload the tape after backup
  - [ ] Rewind the tape before unloading

Connection

Server:
MDPROD-DB01\SQL2005

Connection:
LHCE\walkerfr

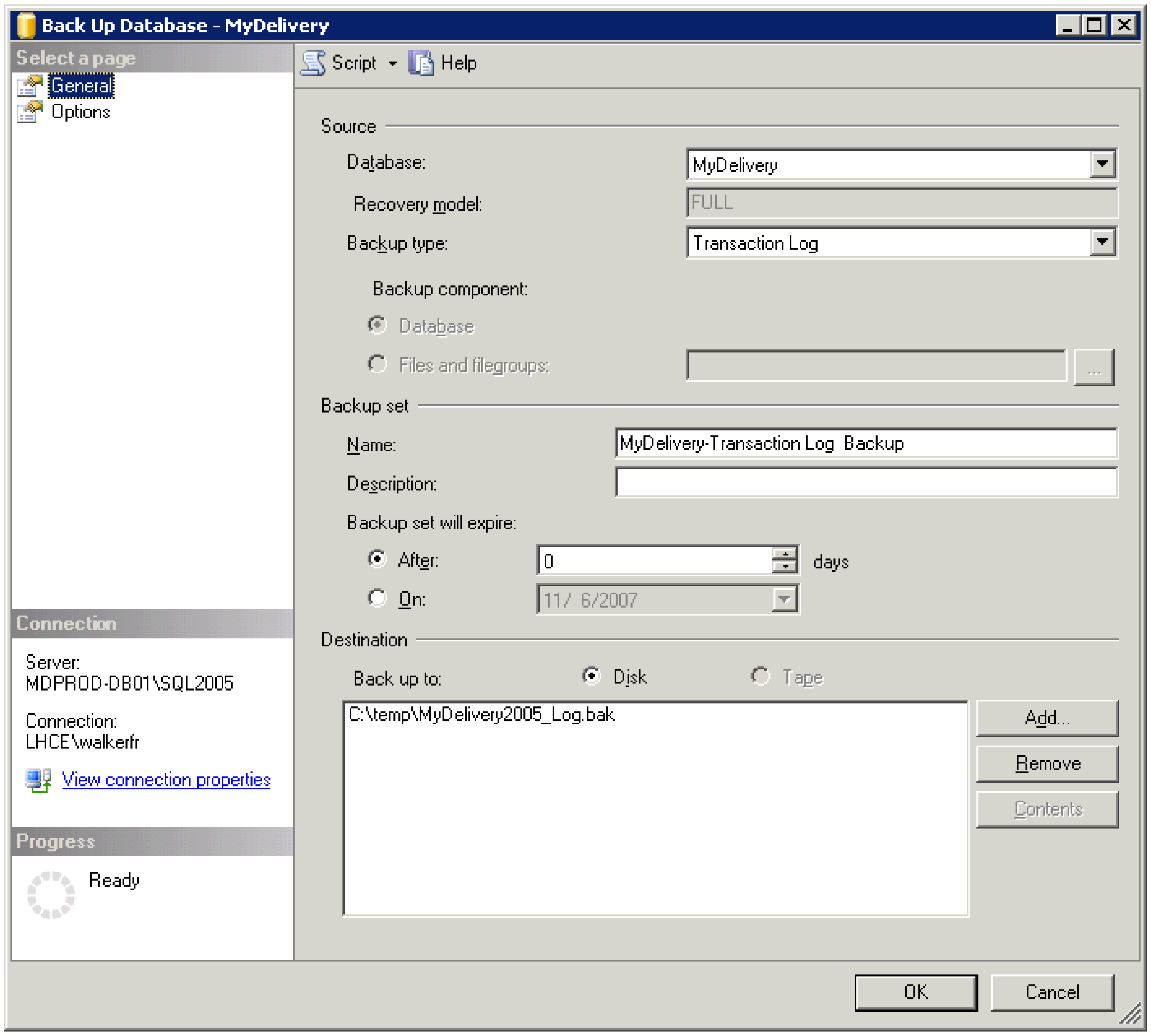
View connection properties

Progress

Ready

OK Cancel

Next, backup the transaction log file for MyDelivery. Use the same choice of Options as for backing up the database (see last picture). Click OK.



Source

Database: MyDelivery

Recovery model: FULL

Backup type: Transaction Log

Backup component: [✓ Database] Files and filegroups:

Backup set

Name: MyDelivery-Transaction Log Backup

Description:

Backup set will expire:

[✓ After:] 0 days

On: 11/ 6/2007

Destination

Back up to: [✓ Disk] Tape

C:\temp\MyDelivery2005_Log.bak

Server:
MDPROD-DB01\SQL2005

Connection:
LHCE\walkerfr

View connection properties

Ready

Once the database and log file are backed up, copy them to mdprod-db02, where they will be restored.

7.3 Detach, Attach and Restore MyDelivery

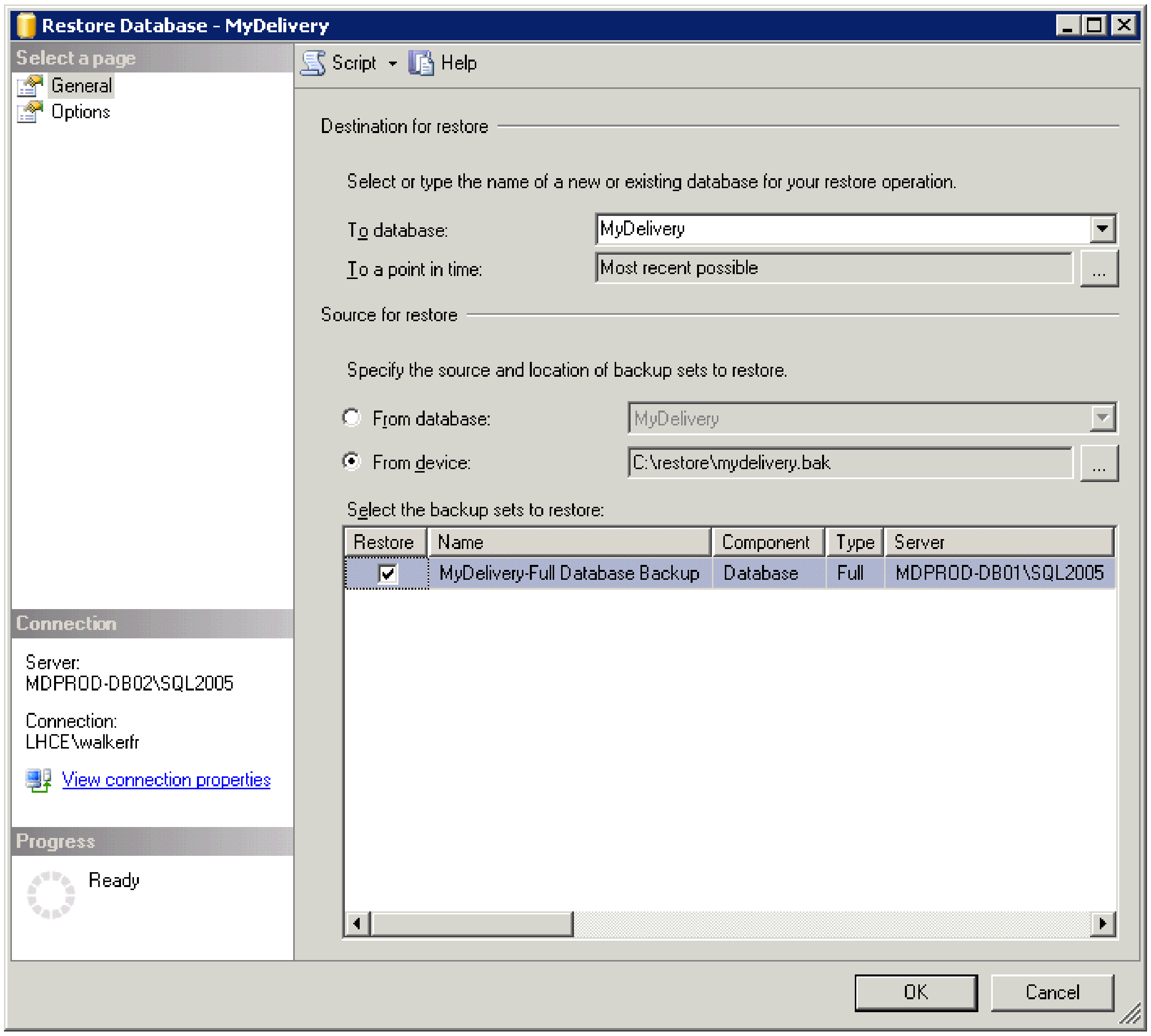
Next, detach the MyDelivery database on mdprod-db01. Copy both the database and its transaction log file to mdprod-db02 to the folders that correspond on mdprod-db01.

Path to database on mdprod-db01: D:\MDPROD_DB01\mydelivery_data2005.mdf

Path to log file on mdprod-db01: c:\program files\microsoft sql server\mssql.1\data\mydelivery_log2005.ldf.

On mdprod-db02, attach the MyDelivery database and its log file, assuming there is no database already attached. NOTE: If there was a previous MyDelivery database (in restoring mode), then it must first be restored (with Recovery), then detached before copying and attaching the database and log files from mdprod-db01.

On mdprod-db02, select the MyDelivery database, and choose Restore Database under tasks.



Destination for restore

Select or type the name of a new or existing database for your restore operation.

To database: MyDelivery

To a point in time: Most recent possible

Source for restore

Specify the source and location of backup sets to restore.

From database: MyDelivery

From device: C:\restore\mydelivery.bak

Select the backup sets to restore:

Restore	Name	Component	Type	Server
✓	MyDelivery-Full Database Backup	Database	Full	MDPROD-DB01\SQL2005

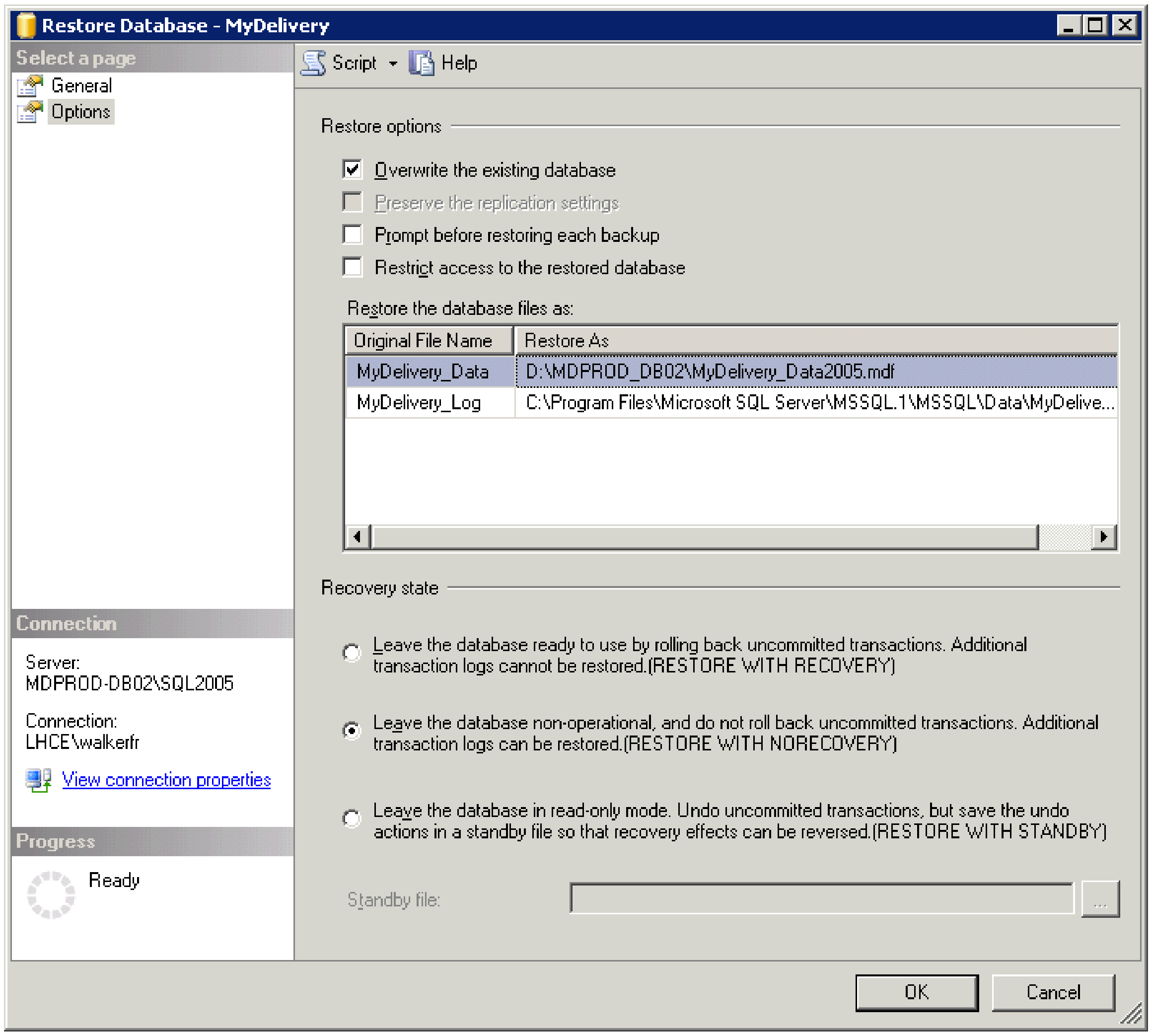
Server:
MDPROD-DB02\SQL2005

Connection:
LHCE\walkerfr

View connection properties

Ready

This is the Options panel for restoring the database. Be sure to select RESTORE WITH NORECOVERY.



Restore options

[✓] Overwrite the existing database
[ ] Preserve the replication settings
[ ] Prompt before restoring each backup
[ ] Restrict access to the restored database

Restore the database files as:

Original File Name	Restore As
MyDelivery_Data	D:\MDPROD_DB02\MyDelivery_Data2005.mdf
MyDelivery_Log	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\MyDelive...

Recovery state

( ) Leave the database ready to use by rolling back uncommitted transactions. Additional transaction logs cannot be restored.(RESTORE WITH RECOVERY)

(●) Leave the database non-operational, and do not roll back uncommitted transactions. Additional transaction logs can be restored.(RESTORE WITH NORECOVERY)

( ) Leave the database in read-only mode. Undo uncommitted transactions, but save the undo actions in a standby file so that recovery effects can be reversed.(RESTORE WITH STANDBY)

Standby file:

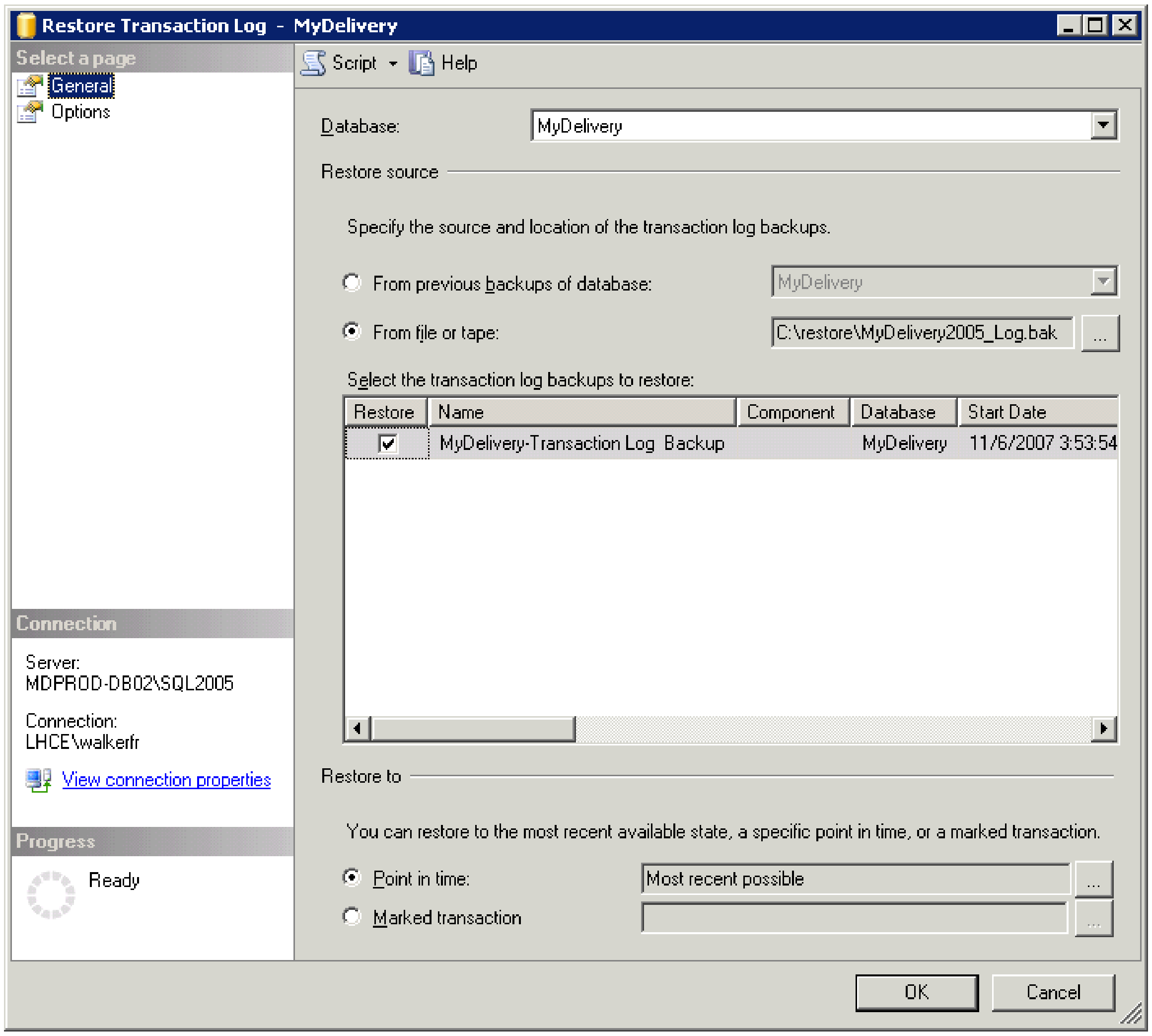
Server:
MDPROD-DB02\SQL2005

Connection:
LHCE\walkerfr

View connection properties

Ready

Next, restore the transaction log file.



Restore Transaction Log - MyDelivery

Database: MyDelivery

Restore source

Specify the source and location of the transaction log backups.

( ) From previous backups of database: MyDelivery

(•) From file or tape: C:\restore\MyDelivery2005_Log.bak

Select the transaction log backups to restore:

Restore	Name	Component	Database	Start Date
✓	MyDelivery-Transaction Log Backup		MyDelivery	11/6/2007 3:53:54

Restore to

You can restore to the most recent available state, a specific point in time, or a marked transaction.

(•) Point in time: Most recent possible

( ) Marked transaction

Connection

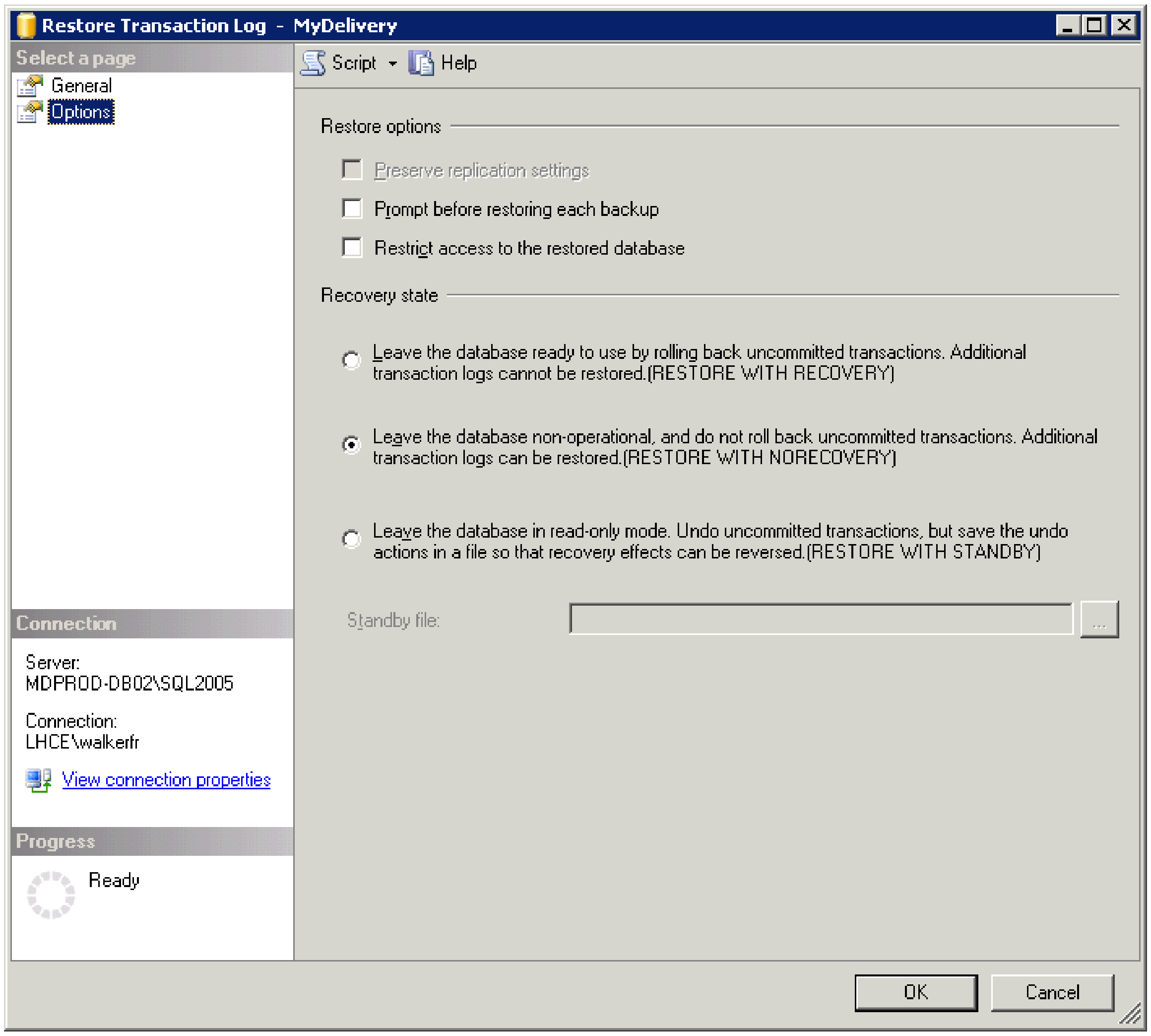
Server:
MDPROD-DB02\SQL2005

Connection:
LHCE\walkerfr

Progress

Ready

Here is the Options tab for restoring the transaction log file. . Be sure to select RESTORE WITH NORECOVERY. Click OK.



Restore options

- [ ] Preserve replication settings
- [ ] Prompt before restoring each backup
- [ ] Restrict access to the restored database

Recovery state

- ○ Leave the database ready to use by rolling back uncommitted transactions. Additional transaction logs cannot be restored.(RESTORE WITH RECOVERY)
- ● Leave the database non-operational, and do not roll back uncommitted transactions. Additional transaction logs can be restored.(RESTORE WITH NORECOVERY)
- ○ Leave the database in read-only mode. Undo uncommitted transactions, but save the undo actions in a file so that recovery effects can be reversed.(RESTORE WITH STANDBY)

Standby file:

Server:
MDPROD-DB02\SQL2005

Connection:
LHCE\walkerfr

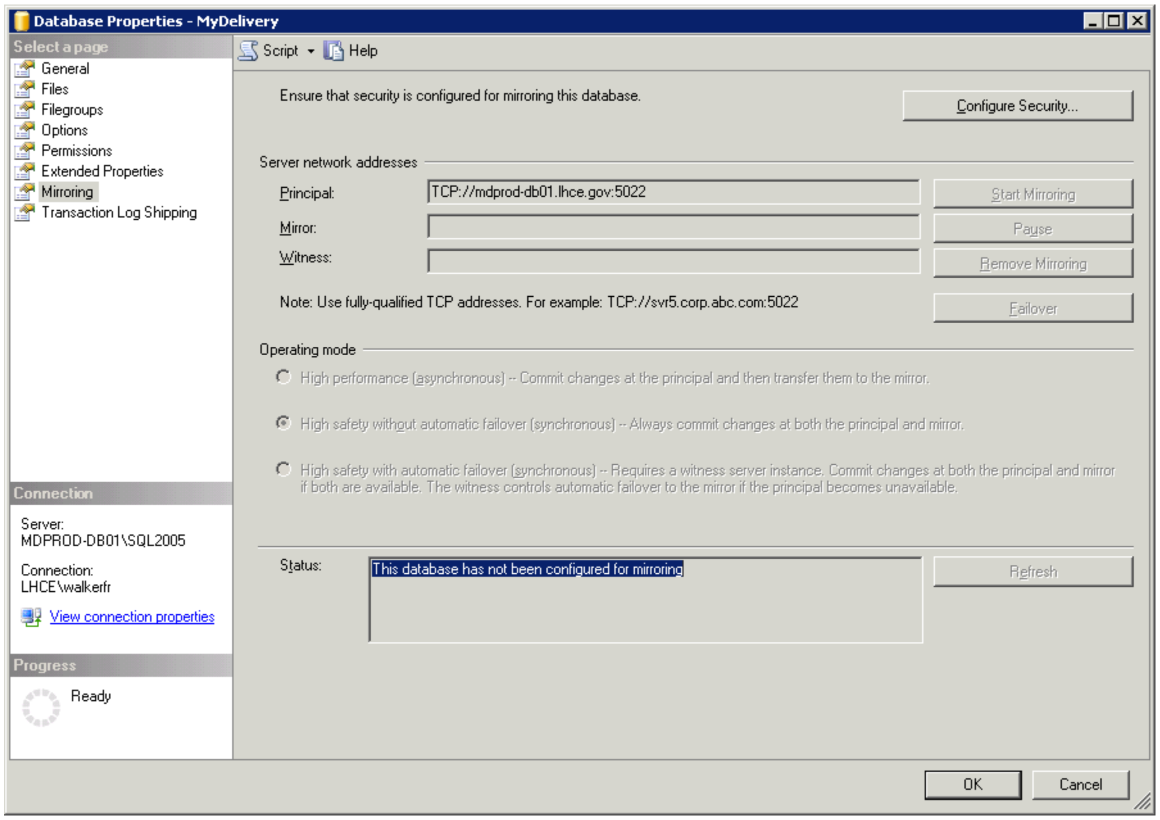
View connection properties

Ready

At this point, the MyDelivery database on mdprod-db02 is in a restoring state. Its tables cannot be opened.

7.4 Configure Mirroring

On mdprod-db01, under Tasks for the MyDelivery database, click Mirror. This brings up the following dialog box.



Ensure that security is configured for mirroring this database.

Server network addresses

Principal: TCP://mdprod-db01.lhce.gov:5022

Mirror:

Witness:

Note: Use fully-qualified TCP addresses. For example: TCP://svr5.corp.abc.com:5022

Operating mode

( ) High performance (asynchronous) -- Commit changes at the principal and then transfer them to the mirror.

(•) High safety without automatic failover (synchronous) -- Always commit changes at both the principal and mirror.

( ) High safety with automatic failover (synchronous) -- Requires a witness server instance. Commit changes at both the principal and mirror if both are available. The witness controls automatic failover to the mirror if the principal becomes unavailable.

Status: This database has not been configured for mirroring

Connection

Server:
MDPROD-DB01\SQL2005

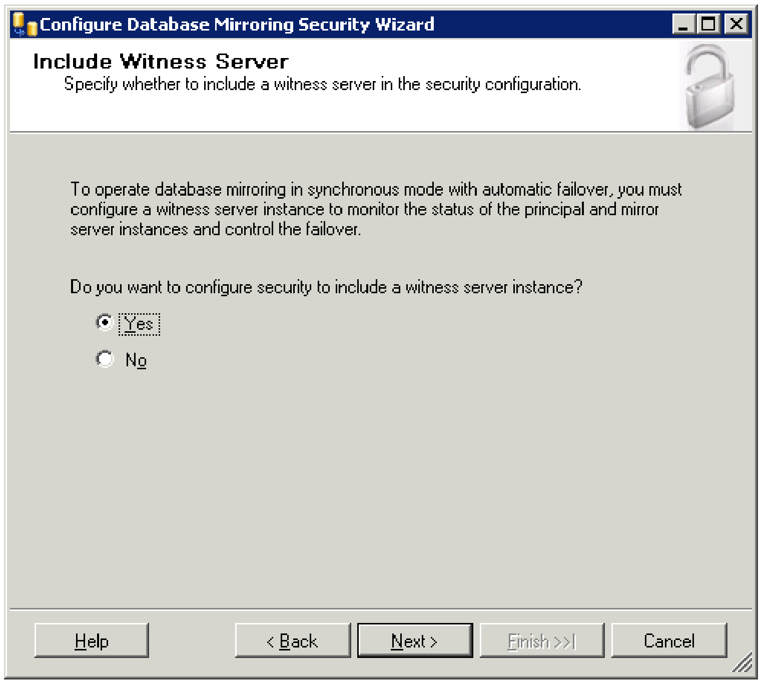
Connection:
LHCE\walkerfr

View connection properties

Progress

Ready

Click “Configure Security.” This brings up a dialog box about whether to include a Witness Server. Select Yes, then click Next.


Configure Database Mirroring Security Wizard

Include Witness Server

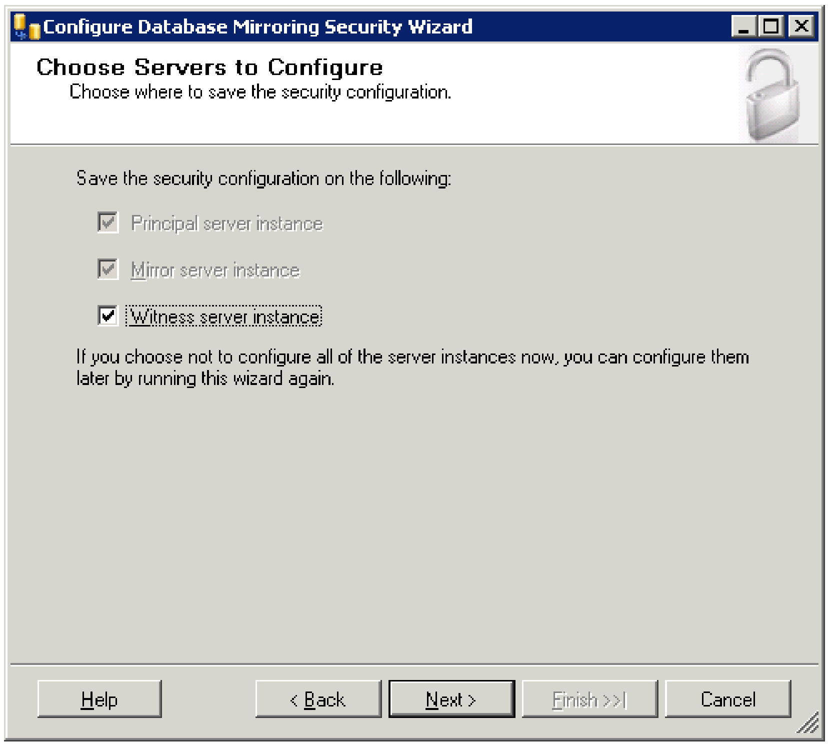
Specify whether to include a witness server in the security configuration.

To operate database mirroring in synchronous mode with automatic failover, you must configure a witness server instance to monitor the status of the principal and mirror server instances and control the failover.

Do you want to configure security to include a witness server instance?

[✓ Yes] No

This brings up the following dialog box. Click Next.


Configure Database Mirroring Security Wizard

Choose Servers to Configure

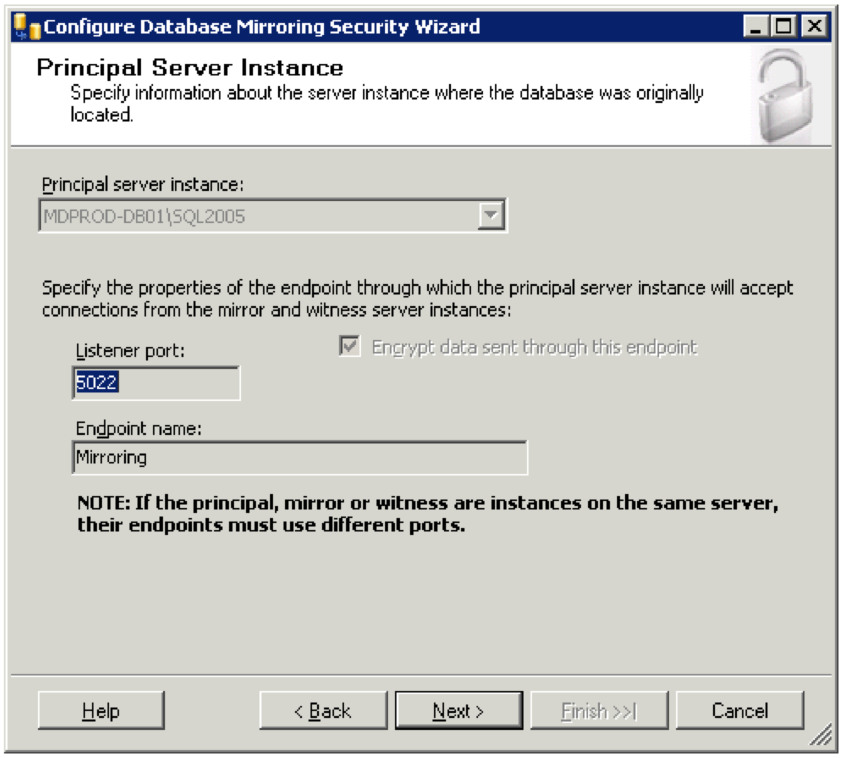
Choose where to save the security configuration.

Save the security configuration on the following:

- ✓ Principal server instance
- ✓ Mirror server instance
- ✓ Witness server instance

If you choose not to configure all of the server instances now, you can configure them later by running this wizard again.

This brings up the following dialog box. Click Next.


Configure Database Mirroring Security Wizard

Principal Server Instance

Specify information about the server instance where the database was originally located.

Principal server instance:
MDPROD-DB01\SQL2005

Specify the properties of the endpoint through which the principal server instance will accept connections from the mirror and witness server instances:

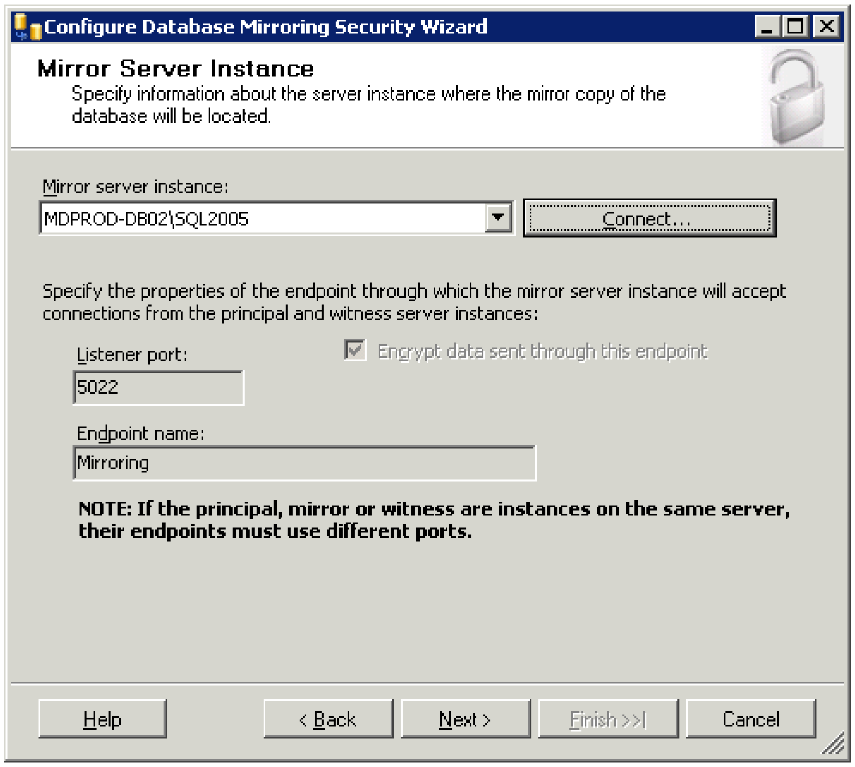
Listener port:
5022

[✓] Encrypt data sent through this endpoint

Endpoint name:
Mirroring

NOTE: If the principal, mirror or witness are instances on the same server, their endpoints must use different ports.

This brings up the next dialog box. Connect to SQL2005 running on mdprod-db02. Then click Next.



Mirror Server Instance

Specify information about the server instance where the mirror copy of the database will be located.

Mirror server instance: MDPROD-DB02\SQL2005

Specify the properties of the endpoint through which the mirror server instance will accept connections from the principal and witness server instances:

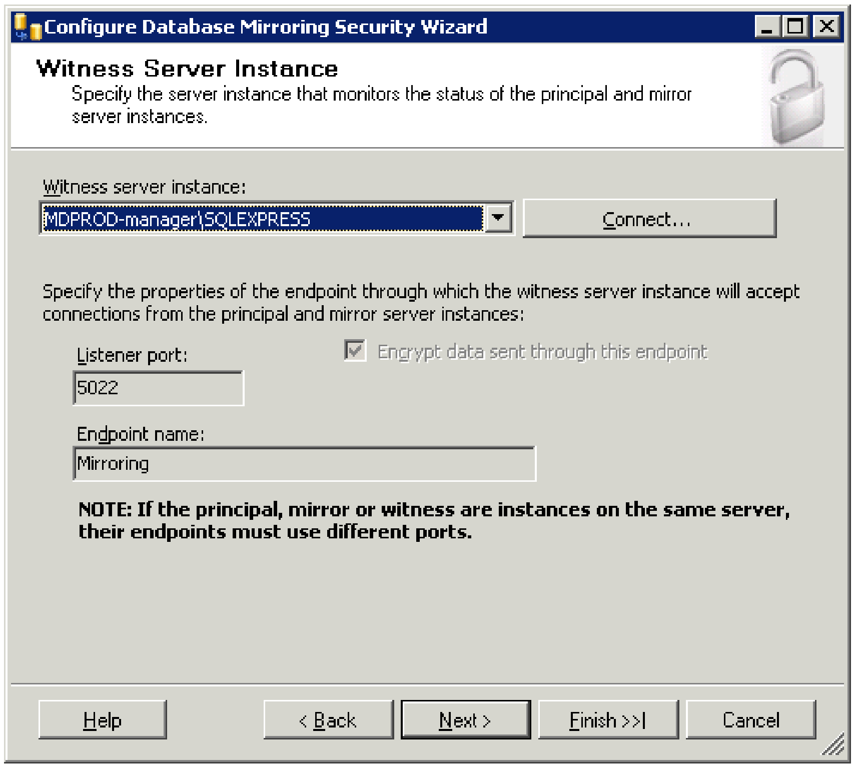
Listener port: 5022

[✓] Encrypt data sent through this endpoint

Endpoint name: Mirroring

NOTE: If the principal, mirror or witness are instances on the same server, their endpoints must use different ports.

This brings up the witness dialog box. Connect to SQLExpress running on mdprod-manager. Then click Next.



Witness Server Instance

Specify the server instance that monitors the status of the principal and mirror server instances.

Witness server instance: MDPROD-manager\SQLEXPRESS

Specify the properties of the endpoint through which the witness server instance will accept connections from the principal and mirror server instances:

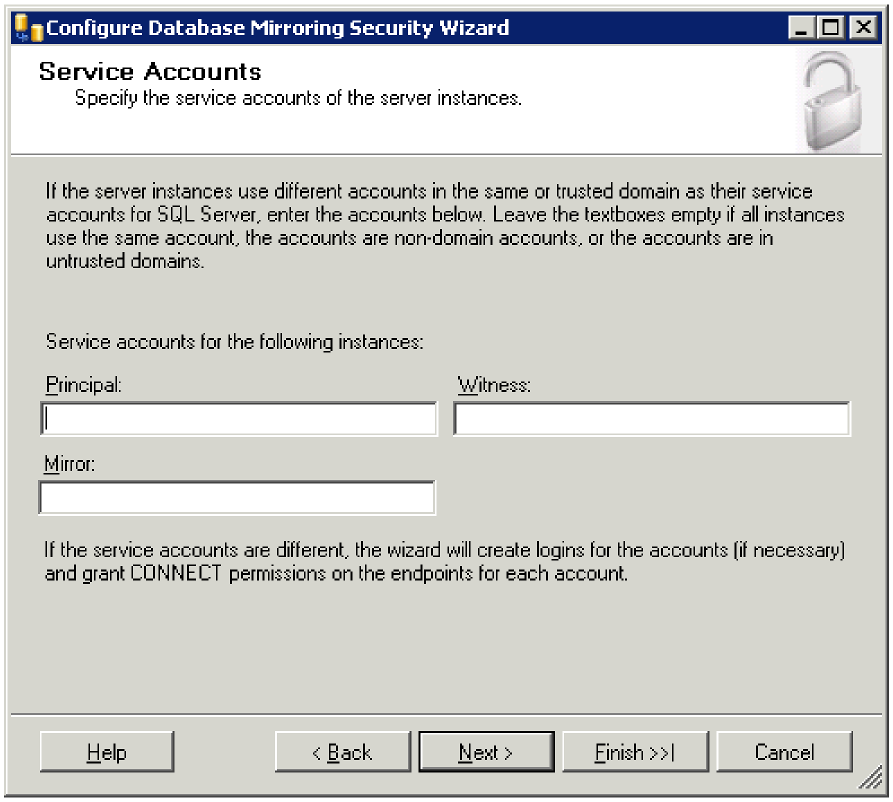
Listener port: 5022

[✓] Encrypt data sent through this endpoint

Endpoint name: Mirroring

NOTE: If the principal, mirror or witness are instances on the same server, their endpoints must use different ports.

This brings up the service accounts dialog box. Leave everything here blank, and click Next.



Service Accounts

Specify the service accounts of the server instances.

If the server instances use different accounts in the same or trusted domain as their service accounts for SQL Server, enter the accounts below. Leave the textboxes empty if all instances use the same account, the accounts are non-domain accounts, or the accounts are in untrusted domains.

Service accounts for the following instances:

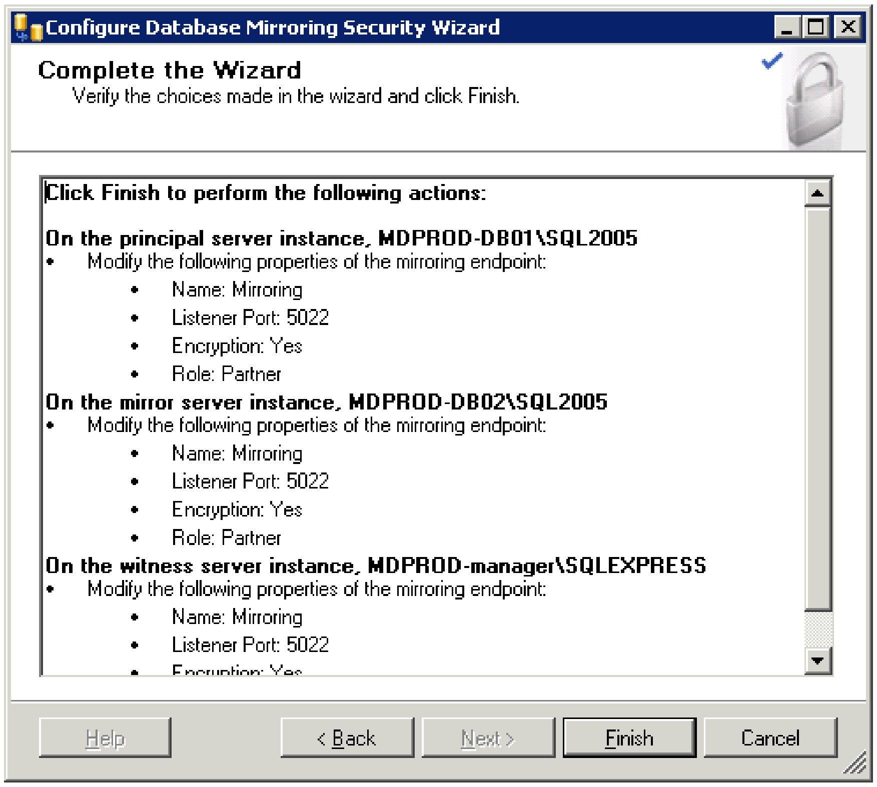
Principal:

Witness:

Mirror:

If the service accounts are different, the wizard will create logins for the accounts (if necessary) and grant CONNECT permissions on the endpoints for each account.

This gives us the final dialog box. Click Finish.



Complete the Wizard

Verify the choices made in the wizard and click Finish.

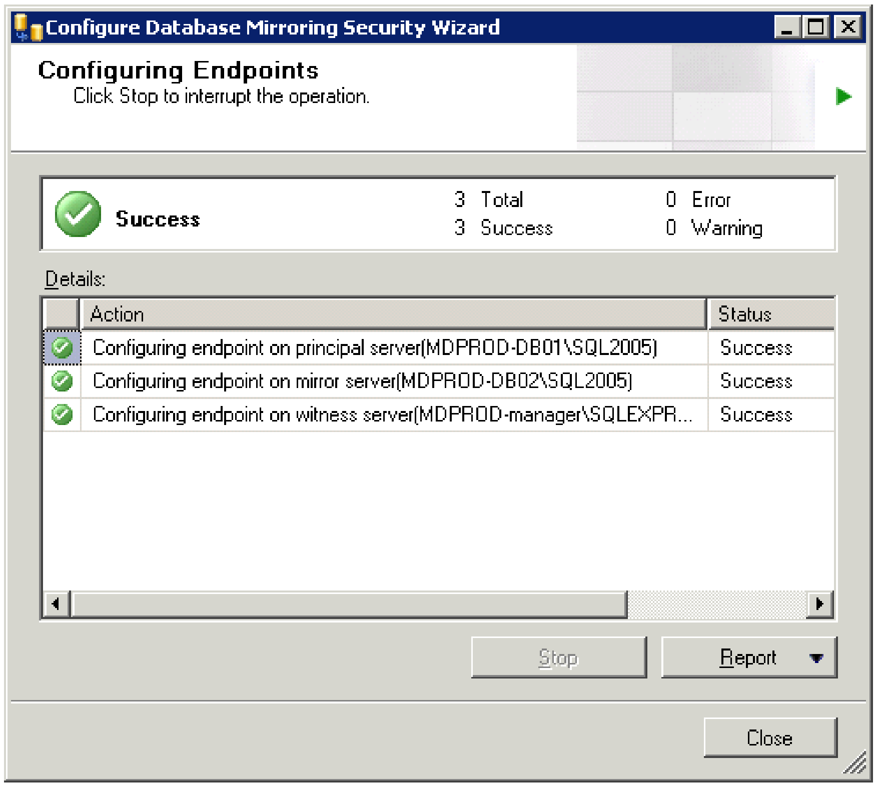
Click Finish to perform the following actions:

On the principal server instance, MDPROD-DB01\SQL2005
- Modify the following properties of the mirroring endpoint:
    - Name: Mirroring
    - Listener Port: 5022
    - Encryption: Yes
    - Role: Partner

On the mirror server instance, MDPROD-DB02\SQL2005
- Modify the following properties of the mirroring endpoint:
    - Name: Mirroring
    - Listener Port: 5022
    - Encryption: Yes
    - Role: Partner

On the witness server instance, MDPROD-manager\SQLEXPRESS
- Modify the following properties of the mirroring endpoint:
    - Name: Mirroring
    - Listener Port: 5022
    - Encryption: Yes

If successful, we should have this dialog box. Click Close.



Configuring Endpoints

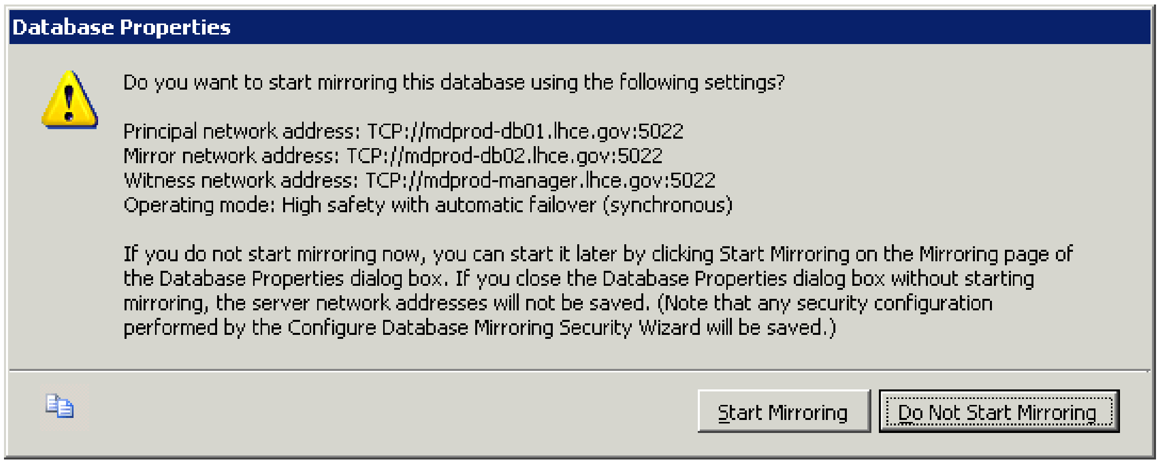
Click Stop to interrupt the operation.

Success	3 Total	0 Error
	3 Success	0 Warning

Details:

Action	Status
Configuring endpoint on principal server(MDPROD-DB01\SQL2005)	Success
Configuring endpoint on mirror server(MDPROD-DB02\SQL2005)	Success
Configuring endpoint on witness server(MDPROD-manager\SQLEXPR...	Success

Next we get the dialog box to start mirroring. Click Start Mirroring.

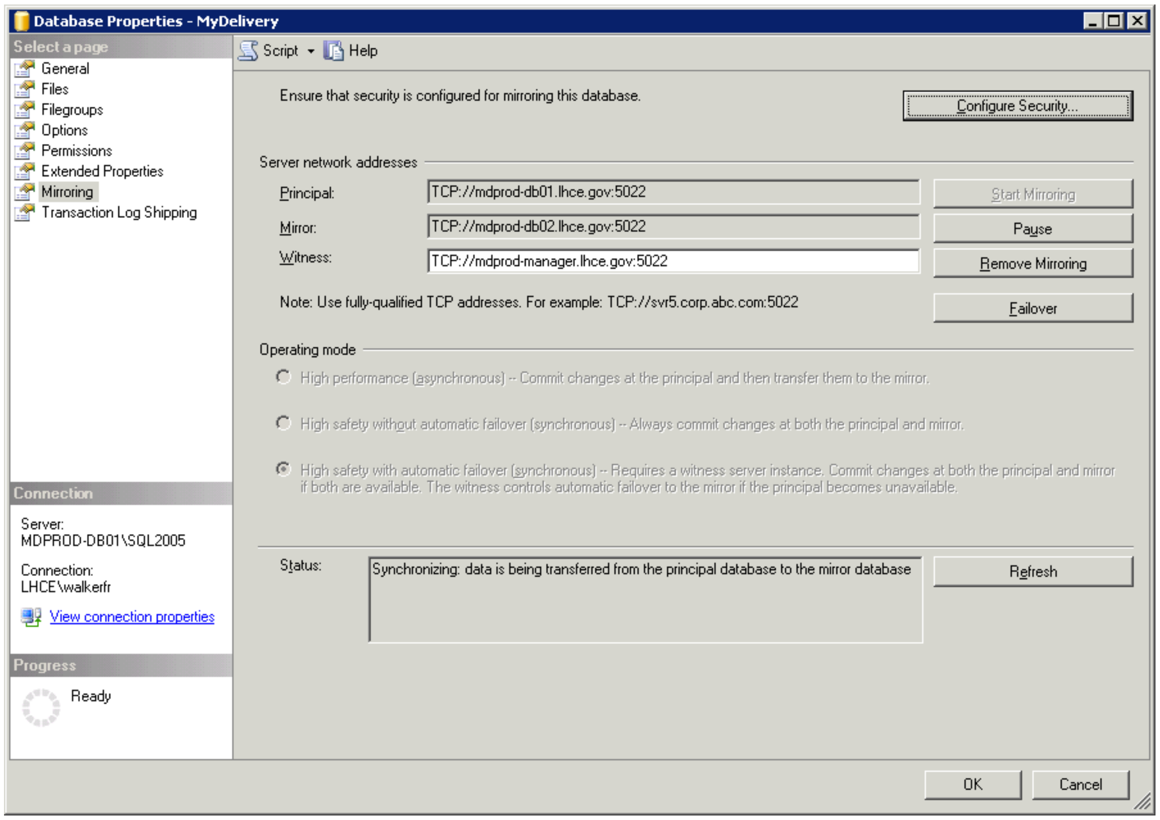


Do you want to start mirroring this database using the following settings?

Principal network address: TCP://mdprod-db01.lhce.gov:5022
Mirror network address: TCP://mdprod-db02.lhce.gov:5022
Witness network address: TCP://mdprod-manager.lhce.gov:5022
Operating mode: High safety with automatic failover (synchronous)

If you do not start mirroring now, you can start it later by clicking Start Mirroring on the Mirroring page of the Database Properties dialog box. If you close the Database Properties dialog box without starting mirroring, the server network addresses will not be saved. (Note that any security configuration performed by the Configure Database Mirroring Security Wizard will be saved.)

If all went well, and mirroring successfully started, we end up here. Click OK.



Ensure that security is configured for mirroring this database.

Server network addresses

Principal: TCP://mdprod-db01.lhce.gov:5022

Mirror: TCP://mdprod-db02.lhce.gov:5022

Witness: TCP://mdprod-manager.lhce.gov:5022

Note: Use fully-qualified TCP addresses. For example: TCP://svr5.corp.abc.com:5022

Operating mode

( ) High performance (asynchronous) -- Commit changes at the principal and then transfer them to the mirror.

( ) High safety without automatic failover (synchronous) -- Always commit changes at both the principal and mirror.

[✓ High safety with automatic failover (synchronous) -- Requires a witness server instance. Commit changes at both the principal and mirror if both are available. The witness controls automatic failover to the mirror if the principal becomes unavailable.]

Status: Synchronizing: data is being transferred from the principal database to the mirror database

Connection

Server: MDPROD-DB01\SQL2005

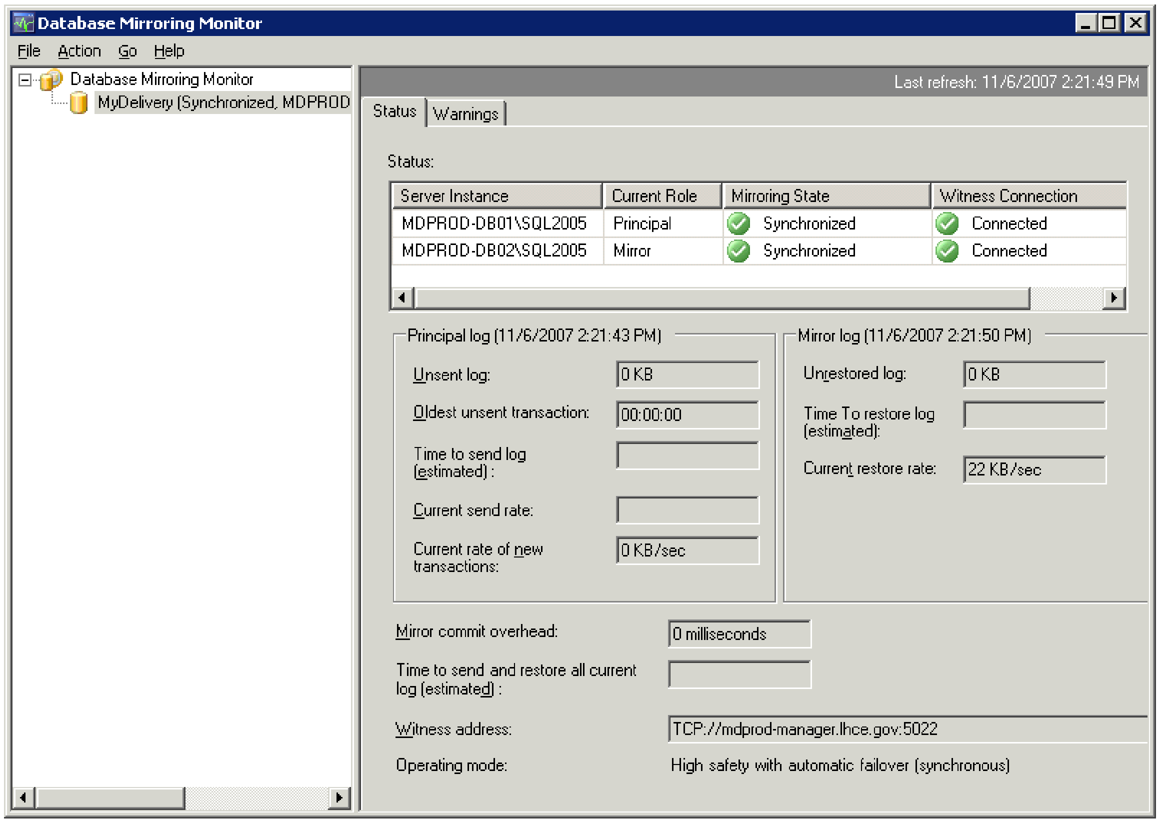
Connection: LHCE\walkerfr

View connection properties

Progress

Ready

To check on the status of database mirroring, use the Database Mirroring Monitor (under Tasks for the MyDelivery database on mdprod-db01.


Database Mirroring Monitor
File Action Go Help
Database Mirroring Monitor
MyDelivery (Synchronized, MDPROD
Last refresh: 11/6/2007 2:21:49 PM
Status
Warnings

Status:

Server Instance	Current Role	Mirroring State	Witness Connection
MDPROD-DB01\SQL2005	Principal	✓ Synchronized	✓ Connected
MDPROD-DB02\SQL2005	Mirror	✓ Synchronized	✓ Connected

Principal log (11/6/2007 2:21:43 PM)

Unsent log:	0 KB
Oldest unsent transaction:	00:00:00
Time to send log (estimated) :	
Current send rate:	
Current rate of new transactions:	0 KB/sec

Mirror log (11/6/2007 2:21:50 PM)

Unrestored log:	0 KB
Time To restore log (estimated):	
Current restore rate:	22 KB/sec

Mirror commit overhead:	0 milliseconds
Time to send and restore all current log (estimated) :	
Witness address:	TCP://mdprod-manager.lhce.gov:5022
Operating mode:	High safety with automatic failover (synchronous)