Performance

Microsoft Dynamics CRM 3.0

Optimizing Performance

White Paper

Date: January 10, 2007

http://go.microsoft.com/fwlink/?LinkId=80916

Microsoft Dynamics

Contents

Summary

This white paper discusses some of the ways that you can optimize the performance of the Microsoft Dynamics™ CRM 3.0 system. The paper begins with several optimizations that can potentially increase performance the most and are also fairly easy to perform. The optimizations later in this paper can also improve performance, but are more complex, and require more advanced knowledge to finish.

This paper also provides links to Microsoft® Knowledge Base articles and related Microsoft Dynamics CRM 3.0 documentation that can help improve performance.

To enhance performance of Microsoft Dynamics CRM the most, try these optimizations in the following order:

1. Reindex the Microsoft Dynamics CRM database. For more information, see "Re-Indexing the Database" on page 7.

2. Apply Microsoft Dynamics CRM performance enhancement updates. For more information, see "Downloading Software Updates for Performance Enhancement and Security" on page 27.

3. Configure Internet Information Services (IIS) on the server that is running Microsoft Dynamics CRM Server. For more information, see "Optimizing Internet Information Services and WAN Performance" on page 44.

4. Recommended: Move the Microsoft SQL Reporting Server for Microsoft Dynamics CRM to another dedicated server. For more information, see "Dedicated Report Server" on page 35.

5. Set up performance monitoring of the server that is running Microsoft Dynamics CRM. For more information, see "Monitoring and Optimizing Microsoft Dynamics CRM System Performance" on page 49.

6. Recommended: Increase the number of file groups and files for the tempDB database on Microsoft SQL Server™. For more information, see "Configuring the tempdb Database" on page 22.

Important   These optimizations are based on a standard Microsoft Dynamics CRM 3.0 configuration. Because of the many types of customized configurations available to Microsoft Dynamics CRM 3.0 customers, the optimizations may not work with all configurations.

Warning   Before you perform any of the following optimization procedures, back up your databases. These include the system databases and Microsoft Windows® Active Directory®. If you do not back up these items, you risk losing the information that is contained in them.

Maintaining Databases and SQL Server

The database architecture for your organization's implementation of Microsoft Dynamics CRM 3.0 includes Microsoft SQL Server and also the databases that contain your organization's records. Ongoing maintenance of your databases and SQL Server is one of the most important ways that you can optimize the performance of Microsoft Dynamics CRM.

Understanding How Data Size Affects Performance

The data in your Microsoft Dynamics CRM database can have a significant effect on the performance of the Microsoft Dynamics CRM system. This is true of both data that was imported into the database as part of a migration from another customer relationship management system and data that users in your organization enter in the system on a day-to-day basis. The Microsoft SQL Server that contains Microsoft Dynamics CRM data is a very important component of any system operating at optimal efficiency.

Several factors affect the total size of the data in your Microsoft Dynamics CRM database:

- The total number of records (both for each entity and over the whole system).
- The number of users in the system.
- Metadata about those users.
- The data that each user will own.
- The business use cases that each user will run.
- The frequency at which those use cases occur.

Because these factors can combine in an infinite number of ways, each Microsoft Dynamics CRM deployment truly is unique. Therefore, it is difficult to compare two deployments side by side, even if they had about the same number of users or data size.

Carefully consider all the business requirements and other unique deployment details that shape your deployment.

Installing Microsoft SQL Server in Clusters

Microsoft Dynamics CRM 3.0 servers can be installed in a clustered Microsoft SQL Server environment. A cluster of computers that are running SQL Server can reduce system down time. If one of the servers in the cluster fails, there will be an automatic fail-over to another computer that is running SQL Server.

You can configure the SQL Server cluster so that the Microsoft Dynamics CRM databases will fail over to the second node in the SQL Server cluster. Only a Single Instance (Active/Passive) cluster is supported. This includes:

- Microsoft Dynamics CRM Database
- Microsoft Dynamics CRM Metadata
- Microsoft Dynamics CRM SQL Reporting Services (SRS)

   Note: SRS can reside on a dedicated SRS server outside the cluster.

For more information about how to implement Microsoft Dynamics CRM 3.0 in a SQL Server clustered environment, see the article "Install Microsoft Dynamics CRM 3.0 Server using a Microsoft SQL Server cluster environment" on Microsoft.com:

- http://www.microsoft.com/dynamics/crm/using/deploy/clusteringSQLservers.mspx

Maintaining the Database

To guarantee optimal performance of the Microsoft Dynamics CRM database on SQL Server, you should have SQL Server maintenance plans running regularly, for example, every night, or every week. These plans should include the following best practice recommendations for file groups:

- Updating statistics

- Reducing the size of the database table

- Removing records deleted from Microsoft Dynamics CRM

- Creating and managing indexes

Re-Indexing the Database

The DBCC DBREINDEX database maintenance command is helpful for de-fragmenting indexes in SQL Server. However, DBCC DBREINDEX puts a shared lock on the tables it is operating on for the duration of the operation for non-clustered indexes, and puts an exclusive table lock on the table for rebuilding clustered indexes. Therefore, the Microsoft Dynamics CRM system may be unavailable to users during the execution of this command. This can be minimized by running the DBCC DBREINDEX statement after regular business hours.

You can use the DBCC DBREINDEX command to rebuild one or more indexes for a specific table. When you use DBCC DBREINDEX, you do not have to know anything about the underlying table structure or any PRIMARY KEY or UNIQUE constraints; these are preserved automatically during the rebuilding. DBCC DBREINDEX completely rebuilds the indexes. Therefore, it restores the page density levels to the original fill factor (the default method), or you can select another target value for the page density. Internally, running DBCC DBREINDEX resembles using Transact-SQL statements to drop and re-create the indexes manually.

Note: If you have to create new indexes or manage existing indexes, see "Creating and Managing Indexes" on page 17.

Advantages of Running DBCC DBREINDEX

For large or small data sets, the advantages of running DBCC DBREINDEX are that you can:

- Rebuild statistics automatically during the rebuilding of the indexes. This can have significant improvements on workload performance.

- Take advantage of multiple-processor computers. Rebuilding large or heavily fragmented indexes using DBCC DBREINDEX can be significantly faster on a multi-processor computer.

All work done by DBCC DBREINDEX occurs as a single, automatic transaction. The new indexes must be present and must be completely built before the old index pages are released. Performing the rebuild requires sufficient free space in the data file or files. Without sufficient free space in the data files, DBCC DBREINDEX may be unable to rebuild the indexes, or the indexes may be rebuilt with logical fragmentation values greater than zero. The free space that is needed varies and depends on the number of indexes being created in the transaction. For clustered indexes, a good guideline is:

Required free space = $1.2 \times$ (*average rowsize*) $\times$ (*number of rows*)

For non-clustered indexes, you can predict free space necessary by calculating the average size of each row in the non-clustered index--length of the non-clustered key plus the length of clustering key or row ID. Then multiply that value by the number of rows. If you rebuild indexes for a complete table, you will need sufficient free space to

7

build the clustered index and all non-clustered indexes. Similarly, if you rebuild a non-unique clustered index, you will also need free space for both the clustered and non-clustered indexes. The non-clustered indexes are implicitly rebuilt because SQL Server must generate new unique identifiers for the rows.

Tip: When you use DBCC DBREINDEX, it is good practice to specify the index you want to de-fragment. This gives you more control over the operations being performed and can help avoid unnecessary work.

Use the REINDEX script to re-index all the indexes in a particular database using a fill factor of 90, as shown in the following procedure.

If you want to estimate the free space that is needed for the AccountBase table, use the following SQL statement. This statement calculates the space that is needed to re-index both the clustered indexes and non-clustered indexes:

```
select sum(xmaxlen) as 'Max_NonClusteredIndexLength', Sum(xmaxlen)*rows/1024 as
'Estimated Index Rebuild Size in KB'
from sysindexes
where id in (select Object_id('AccountBase'))
And rows > 0
group by rows
```

If you are using SQL Server 2005, use the ALTER INDEX 'TABLENAME' REBUILD command instead of the DBCC DBREINDEX command.

For more information about ALTER INDEX, see the "ALTER INDEX (Transact-SQL)" entry in the Transact-SQL section of the SQL Server Language Reference on MSDN:

- http://msdn2.microsoft.com/en-us/library/ms188388.aspx

Note: With Microsoft Dynamics CRM 3.0, you can use two SQL Server jobs to de-fragment and rebuild the indexes. These SQL Server jobs are installed with Microsoft Dynamics CRM Server on the Microsoft SQL Server and will work with both SQL Server 2000 and SQL Server 2005. On Microsoft SQL Server 2005, you can find and run these jobs from SQL Server Management Studio.

1. Click **Start**, point to **All Programs**, point to **Microsoft SQL Server 2005**, and then click **SQL Server Management Studio**.

2. Open Object Explorer, click **Connect Database Engine**, and then connect to the Microsoft Dynamics CRM SQL Server, if it is not already connected.

3. Open **SQL Server Agent**, and then click **Jobs** under the SQL Server that contains the Microsoft Dynamics CRM database. SQL Server Agent displays all jobs for this SQL Server.

   - The **MSCRM Index Defragmentation** job de-fragments Microsoft Dynamics CRM indexes. By default, the de-fragmentation job is scheduled to run one time per day.

   - The **MSCRM Index Reindexing** job rebuilds the Microsoft Dynamics CRM indexes. By default, the re-indexing job is scheduled to run one time per week.

     Important: Both of these jobs only take action on default Microsoft Dynamics CRM indexes. The jobs ignore any other indexes. However, the sample scripts provided in this white paper de-fragment and re-index *all* Microsoft Dynamics CRM indexes. If you decide to use the sample scripts in this white paper, you can safely disable these two default jobs as long as you create new jobs that include the sample scripts.

     Tip: You can manually run these jobs by right-clicking the job name and selecting **Start Job at Step**.

On Microsoft SQL Server 2000, you can find and run these jobs from SQL Server 2000 Enterprise Manager.

1. Click **Start**, point to **All Programs**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.

2. Expand the **SQL Server Group** and expand the SQL Server that the Microsoft Dynamics CRM databases have been installed to.

3. Expand **Management** > **SQL Server Agent** > **Jobs**.

   - The **MSCRM Index Defragmentation** job de-fragments Microsoft Dynamics CRM indexes. By default, the de-fragmentation job is scheduled to run one time per day.

   - The **MSCRM Index Reindexing** job rebuilds the Microsoft Dynamics CRM indexes. By default, the re-indexing job is scheduled to run one time per week.

     Important: Both of these jobs only take action on default Microsoft Dynamics CRM indexes. The jobs ignore any other indexes. However, the sample scripts provided in this white paper de-fragment and re-index *all* Microsoft Dynamics CRM indexes. If you decide to use the sample scripts in this white paper, you can safely disable these two default jobs as long as you create new jobs that include the sample scripts.

     Tip: You can manually run these jobs by right-clicking the job name and selecting **Start Job**.

You can see the contents of the SQL queries for these jobs in the stored procedures, **p_DefragIndexes** and **p_Reindex** in the Microsoft Dynamics CRM database. These stored procedures should not be modified.

You must update the statistics after all the indexes in a database have been de-fragmented, re-indexed, or both. How to update statistics is discussed later in this paper in the section "Updating Table Statistics."

Note If you have a large data set, such as more than 100,000 records in any one table, updating the statistics for all tables with fullscan may require both significant time and resources.

```
sp_MSForEachTable  "UPDATE STATISTICS ? with fullscan"
```

Re-Indexing all Tables with a Script

This section includes a sample script that re-indexes all the tables in the Microsoft Dynamics CRM database.

Notes:

- You can use the script in the following procedure, without modification, with both SQL Server 2000 and SQL Server 2005.

- Perform this procedure after regular business hours because the table or tables will be locked during the re-indexing process.

To re-index all the indexes in a SQL Server 2000 or a SQL Server 2005 database:

1. Connect to SQL Server.

   For SQL Server 2000:

   a) Click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Query Analyzer**.

   b) In the **Connect to SQL Server** dialog box, click **OK**.

9

c) On the **Query** menu, click **Change Database**.

d) In the **Select Database of <ServerName>** dialog box, click the Microsoft Dynamics CRM database that you want to work on, and then click **OK**.

e) In the **Query** window, type the commands listed later in this step.

For SQL Server 2005:

a) Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2005**, and then click **SQL Server Management Studio**.

b) In the **Connect to Server** dialog box enter the server name and select **Windows authentication credentials**, and then click **OK**.

c) In the **Database Selection** menu, select the Microsoft Dynamics CRM database that you want to run this script against.

d) In the **Query** window, type the following commands:

```
SET NOCOUNT ON
-- DB & OS Version control START
DECLARE @SQLVersionMaj dec(4,2), @OSVersionMaj dec(4,2)
DECLARE @SQLVersionMin dec(4), @OSVersionMin dec(4)
CREATE TABLE #Version (IndexId int NOT NULL
                    ,Name varchar(60)
                    ,Internal_Value int
                    ,Character_Value varchar(255))
INSERT #Version exec master.dbo.xp_msver
SELECT @SQLVersionMaj = CONVERT(dec(4,2),SUBSTRING(Character_Value,1,4))
, @SQLVersionMin = CONVERT(dec(4),SUBSTRING(Character_Value,6,4))
FROM #Version
WHERE Name = 'ProductVersion'
SELECT @OSVersionMaj = CONVERT(dec(4,2),SUBSTRING(Character_Value,1,4))
, @OSVersionMin = CONVERT(dec(4),SUBSTRING(Character_Value,6,4))
FROM #Version
WHERE Name = 'WindowsVersion'
--PRINT @SQLVersionMaj
DROP TABLE #Version
-- DB & OS Version control END
IF @SQLVersionMaj = 9.00
BEGIN
        --PRINT 'SQL 2005'
        --INSERT SQL 2005 Specific code here

        EXEC Sp_MSForEachTable @command1 = "PRINT 'ALTER INDEX ALL ON TABLE ?
REBUILD' ", @command2="ALTER INDEX ALL ON ? REBUILD WITH (FILLFACTOR = 80, ONLINE =
OFF,SORT_IN_TEMPDB = ON, STATISTICS_NORECOMPUTE = OFF)"

END
ELSE
        --PRINT 'SQL 2000'
        --INSERT SQL 2000 specific code here

        EXEC Sp_MSForEachTable @command1="Print 'DBCC DBREINDEX ON TABLE ?' DBCC
DBREINDEX ([?],'',80)"
```

2. On the toolbar, click **Execute Query**. The results appear in the results pane.

Note: You must update the statistics after all the indexes in a database have been de-fragmented, re-indexed, or both. How to update statistics is discussed in "Updating Table Statistics" on page 13.

De-Fragmenting all Tables with a Script

This section describes how to de-fragment all the tables in the Microsoft Dynamics CRM database.

When to Use DBCC INDEXDEFRAG and SHOWCONTIG

For large datasets of 5,000 entities or more, you may also have to run DBCC INDEXDEFRAG to repair fragmentation of the database. The DBCC INDEXDEFRAG

command can de-fragment clustered and non-clustered indexes on tables and views. DBCC INDEXDEFRAG de-fragments the leaf level of an index so that the physical order of the pages matches the left-to-right logical order of the leaf nodes. This improves index-scanning performance.

The DBCC SHOWCONTIG script identifies the extent of the fragmentation, and then DBCC INDEXDEFRAG de-fragments the database. For an illustration of how to use DBCC SHOWCONTIG and DBCC INDEXDEFRAG to de-fragment the indexes in a database, see Example E in the DBCC SHOWCONTIG topic in the SQL Server 2000 section of the MSDN Library.

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_dbcc_46cn.asp

For more information about DBCC INDEXDEFRAG, see the DBCC INDEXDEFRAG topic in the same section of the MSDN Library.

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_dbcc_94mw.asp

De-Fragmenting Indexes with a Script

Note: You can use the script in the following procedure, without modification, with both SQL Server 2000 and SQL Server 2005.

To de-fragment all the indexes in a SQL Server 2000 or a SQL Server 2005 database:

1. Connect to SQL Server.

   For SQL Server 2000:

   a) Click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Query Analyzer**.

   b) In the **Connect to SQL Server** dialog box, click **OK**.

   c) On the **Query** menu, click **Change Database**.

   d) In the **Select Database of <ServerName>** dialog box, click the Microsoft Dynamics CRM database that you want to work on, and then click **OK**.

   e) In the **Query** window, type the commands listed later in this step.

   For SQL Server 2005:

   a) Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2005**, and then click **SQL Server Management Studio**.

   b) In the **Connect to Server** dialog box enter the server name and select **Windows authentication credentials**, and then click **OK**.

   c) In the **Database Selection** menu, select the Microsoft Dynamics CRM database that you want to run this script against.

   d) In the **Query** window, type the following commands:

```
SET NOCOUNT ON
-- DB & OS Version control START
DECLARE @SQLVersionMaj dec(4,2), @OSVersionMaj dec(4,2)
DECLARE @SQLVersionMin dec(4), @OSVersionMin dec(4)
CREATE TABLE #Version (IndexId int NOT NULL
                ,Name varchar(60)
                ,Internal_Value int
                ,Character_Value varchar(255))
INSERT #Version exec master.dbo.xp_msver
SELECT @SQLVersionMaj = CONVERT(dec(4,2),SUBSTRING(Character_Value,1,4))
, @SQLVersionMin = CONVERT(dec(4),SUBSTRING(Character_Value,6,4))
```

```
FROM #Version
WHERE Name = 'ProductVersion'
SELECT @OSVersionMaj = CONVERT(dec(4,2),SUBSTRING(Character_Value,1,4))
, @OSVersionMin = CONVERT(dec(4),SUBSTRING(Character_Value,6,4))
FROM #Version
WHERE Name = 'WindowsVersion'
--PRINT @SQLVersionMaj
DROP TABLE #Version
-- DB & OS Version control END
IF @SQLVersionMaj = 9.00
BEGIN
        --PRINT 'SQL 2005'
        --INSERT SQL 2005 Specific code here

        EXEC Sp_MSForEachTable @command1 = "PRINT 'ALTER INDEX ALL ON TABLE ? WITH
Reorganize'", @command2= "ALTER INDEX ALL ON ? REORGANIZE"

END
ELSE
BEGIN
        --PRINT 'SQL 2000'
        --INSERT SQL 2000 specific code here
        -- Defragment indexes with > 10% logical fragmentation
        -- Declare variables.
        SET NOCOUNT ON
        DECLARE @tablename varchar (128)
        DECLARE @execstr        varchar (255)
        DECLARE @objectid       int
        DECLARE @indexid        int
        DECLARE @frag           decimal
        DECLARE @maxfrag      decimal
        -- Decide on the maximum fragmentation to allow.
        SELECT @maxfrag = 10.0 -- per BOL recommendation
        -- Declare cursor.
        DECLARE tables CURSOR FOR
            SELECT TABLE_NAME
            FROM INFORMATION_SCHEMA.TABLES
            WHERE TABLE_TYPE = 'BASE TABLE'
        -- Create the table.
        CREATE TABLE #fraglist (
            ObjectName char (255),
            ObjectId int,
            IndexName char (255),
            IndexId int,
            Lvl int,
            CountPages int,
            CountRows int,
            MinRecSize int,
            MaxRecSize int,
            AvgRecSize int,
            ForRecCount int,
            Extents int,
            ExtentSwitches int,
            AvgFreeBytes int,
            AvgPageDensity int,
            ScanDensity decimal,
            BestCount int,
            ActualCount int,
            LogicalFrag decimal,
            ExtentFrag decimal)
        -- Open the cursor.
        OPEN tables
        -- Loop through all the tables in the database.
        FETCH NEXT
            FROM tables
            INTO @tablename
        WHILE @@FETCH_STATUS = 0
        BEGIN
        -- Run the DBCC SHOWCONTIG command to view
        -- fragmentation information about all the table's indexes.
            INSERT INTO #fraglist
            EXEC ('DBCC SHOWCONTIG ('' + @tablename + ''')
                WITH FAST, TABLERESULTS, ALL_INDEXES, NO_INFOMSGS')
            FETCH NEXT
                FROM tables
```

```
              INTO @tablename
        END
        -- Close and deallocate the cursor.
        CLOSE tables
        DEALLOCATE tables
        -- Declare cursor for list of indexes to be defragmented.
        DECLARE indexes CURSOR FOR
            SELECT ObjectName, ObjectId, IndexId, LogicalFrag
            FROM #fraglist
            WHERE LogicalFrag >= @maxfrag
                AND INDEXPROPERTY (ObjectId, IndexName, 'IndexDepth') > 0
        -- Open the cursor.
        OPEN indexes
        -- Loop through the indexes.
        FETCH NEXT
            FROM indexes
            INTO @tablename, @objectid, @indexid, @frag
        WHILE @@FETCH_STATUS = 0
        BEGIN
            PRINT 'Executing DBCC INDEXDEFRAG (0, ' + RTRIM(@tablename) + ',
                ' + RTRIM(@indexid) + ') - fragmentation currently '
                + RTRIM(CONVERT(varchar(15),@frag)) + '%'
            SELECT @execstr = 'DBCC INDEXDEFRAG (0, ' + RTRIM(@objectid) + ',
                ' + RTRIM(@indexid) + ')'
            EXEC (@execstr)
            FETCH NEXT
                FROM indexes
                INTO @tablename, @objectid, @indexid, @frag
        END
        -- Close and deallocate the cursor.
        CLOSE indexes
        DEALLOCATE indexes
        -- Delete the temporary table.
DROP TABLE #fraglist
END
GO
```

Updating Table Statistics

Updating statistics on the various tables in the Microsoft Dynamics CRM database enables SQL Query Analyzer to select an optimal query execution plan.

To use the UPDATE STATISTICS command to update statistics:

Note You can use the script listed later in the following procedure with SQL Server 2000 or SQL Server 2005 without modification.

1. Connect to SQL Server.

    For SQL Server 2000:

    a) Click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Query Analyzer**.

    b) In the **Connect to SQL Server** dialog box, click **OK**.

    c) On the **Query** menu, click **Change Database**.

    d) In the **Select Database of <ServerName>** dialog box, click the Microsoft Dynamics CRM database that you want to work on, and then click **OK**.

    e) In the **Query** window, type the commands listed later in this step.

    For SQL Server 2005:

    a) Click **Start**, point to **Programs**, point to **Microsoft SQL Server 2005**, and then click **SQL Server Management Studio**.

    b) In the **Connect to Server** dialog box enter the server name and select **Windows authentication credentials**, and then click **OK**.

c) In the **Database Selection** menu, select the Microsoft Dynamics CRM database that you want to run this script against.

d) In the **Query** window, type the following commands:

```
Update Statistics QueueItemBase With FullScan
Update Statistics LeadBase With FullScan
Update Statistics ActivityBase with FullScan
```

2. Click the **Execute Query** button on the toolbar. "The command(s) completed successfully." appears in the results pane of the **Query** window.

Alternatively, you can run the following script in SQL Query Analyzer to update the statistics for every table in the database:

```
sp_MSForEachTable  "UPDATE STATISTICS ? with fullscan"
```

Note: Updating statistics with fullscan on large tables (tables with 500,000 to a million records or more) may take lots of time to run (from 10 minutes to an hour or more) and should be run after regular business hours. The time this takes to update statistics depends on both the number of records in the Microsoft Dynamics CRM database and also on the speed and disk configuration of the Microsoft SQL Server that contains the Microsoft Dynamics CRM database.

Removing Workflow Log Records

After Microsoft Dynamics CRM has been running for some time, Microsoft Dynamics CRM workflow log tables can grow to a very large size and can adversely affect performance. The following script should be run against the Microsoft Dynamics CRM database (*OrganizationName*_MSCRM) in SQL Query Analyzer or SQL Server Management Studio. This script can also be added as a SQL Server job to run at scheduled times, such as one time per week or during non-business hours to minimize any additional decrease in performance for users. The script later in this section depends on the speed of the Microsoft SQL Server, but can delete up to 1,000,000 workflow log records in less than one minute. Microsoft Dynamics CRM does not clean up successful workflow log events and this table can become large enough to affect performance for Microsoft Dynamics CRM users if not cleaned up.

Note: Running the following script removes all records from the workflow log tables for workflows that have successfully completed. These records will no longer exist. Therefore, you will no longer be able to see them in the Microsoft Dynamics CRM Workflow Monitor tool.

Important: After you run either of the scripts provided later in this section, we recommend that you also defragment the indexes, especially for the Microsoft Dynamics CRM Workflow tables. For more information about how to determine fragmenting and defragmenting, see the Defragmenting Script for All Tables section in this white paper.

Removing Workflow Log Records on SQL Server 2000

Use the following script to remove workflow log records on Microsoft SQL Server 2000:

```
SET NOCOUNT ON
DECLARE @SQL as nvarchar(4000)

WHILE ( 1=1 )
BEGIN
        -- RETRIEVE THE RECORDS TO DELETE AND STORE THEM IN TEMP TABLES
        -- BY USING A 1,000 RECORD BATCH, IF THE SCRIPT RUNS FOR AN
        -- EXTENDED PERIOD OF TIME THIS WILL ONLY BLOCK SELECTS FOR
        -- A SHORTER PERIOD OF TIME
SET RowCount 1000
    SET @SQL = 'SELECT WFRuleLogId INTO wfRuleDelete FROM WFRuleLog  INNER JOIN
WFProcessInstance
            ON WFRuleLog.ProcessInstanceId = WFProcessInstance.ProcessInstanceId
```

```
                AND WFProcessInstance.StateCode IN (4, 5)'
        exec sp_executesql @SQL

        SET @SQL = 'SELECT WFActionLogId INTO wfActionDelete FROM WFActionLog INNER
JOIN WFProcessInstance
                ON WFActionLog.ProcessInstanceId = WFProcessInstance.ProcessInstanceId
                AND WFProcessInstance.StateCode IN (4, 5)'

        exec sp_executesql @SQL

        SET @SQL = 'SELECT ProcessInstanceId INTO wfProcessInstanceDelete FROM
WFProcessInstance
                WHERE WFProcessInstance.StateCode IN (4, 5)'
        exec sp_executesql @SQL

--CHECK IF THE TABLES ARE EMPTY, IF SO, THEN JUMP OUT OF THE LOOP
        IF NOT EXISTS(SELECT TOP 1 wfActionLogId FROM wfActionDelete)
                AND NOT EXISTS(SELECT TOP 1 wfRuleLogId FROM wfRuleDelete)
                AND NOT EXISTS(SELECT TOP 1 processInstanceId FROM
wfProcessInstanceDelete)
        BEGIN
                IF  EXISTS (SELECT name FROM sysobjects WHERE name = 'wfRuleDelete'
                                AND type = ('U')) DROP TABLE [dbo].[wfRuleDelete]
                IF  EXISTS (SELECT name FROM sysobjects WHERE name = 'wfActionDelete'
                                AND type = ('U')) DROP TABLE [dbo].wfActionDelete
                IF  EXISTS (SELECT name FROM sysobjects WHERE name =
'wfProcessInstanceDelete'
                                AND type = ('U')) DROP TABLE
[dbo].[wfProcessInstanceDelete]
                BREAK
        END

--TRANSACT THE DELETES
        BEGIN TRAN
-- BATCH DELETE THE 1000 ARCHIVED WFRULE ENTRIES
        PRINT 'Delete rule logs'

        SET @SQL='DELETE WFRuleLog FROM wfRuleDelete twf JOIN WFRuleLog wf
                ON twf.WFRuleLogId = wf.WFRuleLogId'
        exec sp_executesql @SQL

-- BATCH DELETE THE 1000 ARCHIVED WFACTION LOG ENTRIES

        PRINT 'Delete action logs'
        SET @SQL='DELETE WFActionLog FROM wfActionDelete twf JOIN WFActionLog wf
                ON twf.WFActionLogId = wf.WFActionLogId'
        exec sp_executesql @SQL

-- BATCH DELETE THE 1000 ARCHIVED WFPROCESSINSTANCES
-- THE EXCEPTIONS ARE IF SOME PROCESS INSTANCES STILL HAVE LOG OR RULE ENTRIES

        PRINT 'Delete wfprocessinstances'

        SET @SQL='DELETE wfProcessInstance FROM wfProcessInstanceDelete twf
                        JOIN wfProcessInstance wf
                                ON twf.processInstanceId = wf.processInstanceId
                        LEFT Outer JOIN WFRuleLog wfr
                                ON wfr.processInstanceId = wf.processInstanceId
                        LEFT Outer JOIN WFActionLog wfa
                                ON wfa.processInstanceId = wf.processInstanceId
                WHERE wfr.processInstanceId IS NULL AND wfa.processInstanceId IS NULL'
        exec sp_executesql @SQL

--CLEANUP TEMP TABLES
        IF(@@ERROR <> 0)
                BEGIN
                        PRINT 'Rolling back transaction, Error#: ' + CAST(@@error as
varchar(50))
                        ROLLBACK TRAN
                        BREAK
                END
        ELSE
                COMMIT TRAN

set rowcount 0
```

15

```
        PRINT 'Dropping temp tables'
--CLEANUP TABLES IF APPLICABLE

        IF  EXISTS (SELECT name FROM sysobjects WHERE name = 'wfRuleDelete'
                AND type = ('U')) DROP TABLE [dbo].[wfRuleDelete]
        IF  EXISTS (SELECT name FROM sysobjects WHERE name = 'wfActionDelete'
                AND type = ('U')) DROP TABLE [dbo].[wfActionDelete]
        IF  EXISTS (SELECT name FROM sysobjects WHERE name =
'wfProcessInstanceDelete'
                AND type = ('U')) DROP TABLE [dbo].[wfProcessInstanceDelete]
END
```

Removing Workflow Log Records on SQL Server 2005

Use the following script to remove workflow log records on Microsoft SQL Server 2005:

```
SET NOCOUNT ON
DECLARE @SQL as nvarchar(4000)

WHILE ( 1=1 )
BEGIN
        -- Retrieve the records to delete and store them in temp tables
        -- by using a 1,000 record batch, if the script runs for an
        -- extended period of time this will only block selects for
        -- a shorter period of time
        SET @SQL = 'Select TOP 1000 WFRuleLogId INTO wfRuleDelete from WFRuleLog
inner join WFProcessInstance
                on WFRuleLog.ProcessInstanceId = WFProcessInstance.ProcessInstanceId
                and WFProcessInstance.StateCode in (4, 5)'
        exec sp_executesql @SQL

        SET @SQL = 'Select TOP 1000 WFActionLogId INTO wfActionDelete from
WFActionLog inner join WFProcessInstance
                on WFActionLog.ProcessInstanceId = WFProcessInstance.ProcessInstanceId
                and WFProcessInstance.StateCode in (4, 5)'

        exec sp_executesql @SQL

        SET @SQL = 'Select TOP 1000 ProcessInstanceId INTO wfProcessInstanceDelete
FROM WFProcessInstance
                where WFProcessInstance.StateCode in (4, 5)'
        exec sp_executesql @SQL

        --check if the tables are empty, if so, then jump out of the loop
        IF NOT EXISTS(Select TOP 1 wfActionLogId FROM wfActionDelete)
                AND NOT EXISTS(Select TOP 1 wfRuleLogId FROM wfRuleDelete)
                AND NOT EXISTS(Select TOP 1 processInstanceId FROM
wfProcessInstanceDelete)
        BEGIN
                IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[wfRuleDelete]')
                                AND type in (N'U')) DROP TABLE
[dbo].[wfRuleDelete]
                IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[wfActionDelete]')
                                AND type in (N'U')) DROP TABLE
[dbo].[wfActionDelete]
                IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[wfProcessInstanceDelete]')
                                AND type in (N'U')) DROP TABLE
[dbo].[wfProcessInstanceDelete]
                break;
        END

        --transact the deletes
        BEGIN TRAN
        -- Batch delete the 1000 archived wfrule entries
        PRINT 'delete rule logs'

        SET @SQL='Delete WFRuleLog FROM wfRuleDelete twf JOIN WFRuleLog wf
                ON twf.WFRuleLogId = wf.WFRuleLogId'
        exec sp_executesql @SQL

        -- Batch delete the 1000 archived wfaction log entries
```

```
        PRINT 'delete action logs'
        SET @SQL='Delete WFActionLog FROM wfActionDelete twf JOIN WFActionLog wf
            ON twf.WFActionLogId = wf.WFActionLogId'
        exec sp_executesql @SQL

        -- Batch delete the 1000 archived wfProcessInstances
        -- the exceptions are if some process instances still have log or rule
entries
        PRINT 'delete wfprocessinstances'

        SET @SQL='Delete wfProcessInstance FROM wfProcessInstanceDelete twf
                        JOIN wfProcessInstance wf
                            ON twf.processInstanceId = wf.processInstanceId
                        Left Outer JOIN WFRuleLog wfr
                            ON wfr.processInstanceId = wf.processInstanceId
                        Left Outer JOIN WFActionLog wfa
                            ON wfa.processInstanceId = wf.processInstanceId
            WHERE wfr.processInstanceId is null AND wfa.processInstanceId is null'
        exec sp_executesql @SQL

        --Cleanup temp tables
        IF(@@ERROR <> 0)
            BEGIN
                print 'Rolling back transaction, Error#: ' + CAST(@@error as
varchar(50))
                ROLLBACK TRAN
                break;
            END
    ELSE
            COMMIT TRAN;

        PRINT 'dropping temp tables'
        --cleanup tables if applicable

        IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[wfRuleDelete]')
                AND type in (N'U')) DROP TABLE [dbo].[wfRuleDelete]
        IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[wfActionDelete]')
                AND type in (N'U')) DROP TABLE [dbo].[wfActionDelete]
        IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[wfProcessInstanceDelete]')
                AND type in (N'U')) DROP TABLE [dbo].[wfProcessInstanceDelete]
END
```

Creating and Managing Indexes

A common problem that affects Microsoft Dynamics CRM performance is searching the SQL Server database for information in a specific view without having a corresponding index that adequately satisfies that particular query. For example, if you query for leads ordered by company name, but do not have an index set up that gathers leads by company name, SQL Server must search through every record in the **Leads** table, looking for the company name that you requested. When SQL Server uses an index, it goes directly to the records that match the query.

Important: Creating indexes and managing indexes are advanced tasks. Make sure that you have the necessary knowledge and experience before you try the tasks described in this section.

For more information, see the "Indexes" topic in the SQL Server 2000 section of the MSDN Library:

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/createdb/cm_8_des_05_30s5.asp

Optimizing Indexes with the Index Tuning Wizard

If you have a large database, determining what indexes to create can be a complex task. The Index Tuning Wizard enables you to select and create an optimal set of indexes and statistics for a Microsoft SQL Server 2000 database without requiring an expert understanding of the structure of the database, the workload, or the internals of SQL Server.

The Index Tuning Wizard can perform the following actions:

- Recommend the best mix of indexes for a database given a workload, by using SQL Query Analyzer to analyze the queries in the workload.

- Analyze the effects of the proposed changes. These include index usage, distribution of queries among tables, and performance of queries in the workload.

- Recommend ways to tune the database for a small set of problem queries.

- Enable you to customize the recommendation by specifying advanced options such as disk space constraints.

A recommendation consists of SQL statements that can be executed to create new, more effective indexes and, if it is necessary, remove existing indexes that are ineffective. We recommend indexed views on platforms that support their use. After the Index Tuning Wizard has suggested a recommendation, you can then use this information to determine how you want to handle the recommended change:

- Implement the change immediately.

- Schedule the change to be implemented later by creating a SQL Server job that executes an SQL script.

- Save the change to an SQL script, which you can execute manually later, or on a different server.

For more information about index views, see the white paper "Improving Performance with SQL Server 2000 Indexed Views", available in the MSDN Library:

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql2k/html/indexedviews1.asp

The Index Tuning Wizard **does not** recommend indexes for:

- Tables that are referenced by cross-database queries that do not exist in the selected database.

- System tables.

- PRIMARY KEY constraints and unique indexes.

For more information about index tuning, see the article "Index Tuning Wizard," available in the MSDN Library:

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/createdb/cm_8_des_05_0cbo.asp

Warning: Before you perform any of the following optimization procedures, back up your databases and Active Directory. If you do not back up these items, you risk losing the information that is contained in them.

Determine what indexes to create

The first step in using the Index Tuning Wizard to optimize your indexes is to determine what indexes must be created.

To determine what indexes to create:

1. Click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.

2. Expand **SQL Server Group**, and then expand the server that contains the Microsoft Dynamics CRM database for which you want to create an index.

3. On the **Tools** menu, click **Wizards**.

4. Expand **Management**.

5. Double-click **Index Tuning Wizard**.

6. Complete the steps in the wizard.

Create an index using the Create Index Wizard

If you want to add indexes without using the Index Tuning Wizard, you can use the SQL Server Enterprise Manager to create the indexes that you must have by following the steps described in this section.

For more information, see the "[Creating an Index](#)" topic in the SQL Server 2000 section of the MSDN Library:

* [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/createdb/cm_8_des_05_8185.asp](#)

To create an index using the Create Index Wizard:

1. In SQL Server Enterprise Manager, expand a server group, and then expand the server in which to create the index.

2. On the **Tools** menu, click **Wizards**.

3. Expand **Database**.

4. Double-click **Create Index Wizard**.

5. Complete the steps in the wizard.

Optimizing Indexes with the Database Engine Tuning Advisor

If you are using SQL Server 2005 you can use the Database Engine Tuning Advisor instead of the SQL Server 2000 Index Tuning Wizard. The latest version of this tool includes many improvements, and frequently displays good recommendations for indexes to use in the Microsoft Dynamics CRM database.

Note: Additional indexes may affect performance both negatively and positively. Therefore, backups should be performed before you add any indexes or change any indexes. Default Microsoft CRM indexes should not be modified or deleted.

For more information about how to use the Database Engine Tuning Advisor to optimize your indexes, see the topic "[How to: Tune a Database by Using Database Engine Tuning Advisor](#)" in the SQL Server 2005 section of the MSDN Library:

* [http://msdn2.microsoft.com/en-us/library/ms186354.aspx](#)

Finding Missing Indexes with SQL Server 2005 Database Management View

You can use SQL Server 2005 Database Management View (DMV) to find any indexes that may be missing.

This script will only work using SQL Server 2005. The script recommends and generates CREATE INDEX statements based on the last five hundred SELECT statements that were run since the SQL Server 2005 server was last started. It bases these indexes on the worst-performing queries that are missing indexes.

Notes:

- The Create Index statements may not have the index columns in the correct order. In addition, to receive more accurate results that take into account UPDATE and DELETE statements, you should run the SQL Server Database Engine Tuning Advisor with a representative profiler trace workload when Microsoft Dynamics CRM is being accessed by users.

- You may not find a business need for every new index that DMV suggests. Because too many indexes can cause performance problems, it is important to test and validate each set of results before you implement a new index in your production environment.

1. Before you run the script at the end of this procedure:

    - Change the format of the results to text or a file.

    - Change the database context to the *OrganizationName*_MSCRM database.

      For example, if you were working with the Adventure Works Cycle Microsoft Dynamics CRM database, you would type the following commands:

```
USE Adventure_Works_Cycle_MSCRM
GO
```

2. To create the indexes recommended by DMV, copy the CREATE INDEX statements from the results into a new SQL Management Studio window and run them on the Microsoft Dynamics CRM database.

```
    set nocount on
    set ansi_warnings on
    set ansi_padding on
    set arithabort on
    set concat_null_yields_null on
    set numeric_roundabort off

    declare @exec_stmt nvarchar(4000),
    @table_name nvarchar(521),
    @column_name sysname,
    @column_usage varchar(20),
    @column_id smallint,
    @index_handle int,
    @database_id int,
    @object_id int

    declare ms_cri_tnames cursor local static for
    Select Top 5
                mid.database_id,
                mid.object_id,
                mid.statement as table_name,
                mig.index_handle as index_handle
    from
    (select (user_seeks+user_scans) * avg_total_user_cost * (avg_user_impact *
0.01) as index_advantage,
                migs.*        from sys.dm_db_missing_index_group_stats migs)
as migs_adv,
    sys.dm_db_missing_index_groups mig, sys.dm_db_missing_index_details mid
```

```sql
        where migs_adv.group_handle = mig.index_group_handle and
        mig.index_handle = mid.index_handle and migs_adv.index_advantage > 10
        order by migs_adv.index_advantage DESC

        create table #tablenametab
        ( table_name  nvarchar(521) collate database_default )
        create table indexList
        ( execstmt  nvarchar(521) collate database_default )
        --DISCLAIMER:
                insert INTO indexList (execstmt) values ('--THESE INDEXES ARE ONLY
BASED ON THE LAST 500 SQL SERVER QUERIES EXECUTED')
                insert INTO indexList (execstmt) values ('--THESE INDEXES SHOULD BE
TESTED AND VERFIED BEFORE THEY ARE PUT INTO A PRODUCTION ENVIRONMENT')
        truncate table #tablenametab
        open ms_cri_tnames
        fetch next from ms_cri_tnames into @database_id, @object_id, @table_name,
@index_handle

        while (@@fetch_status <> -1)
                begin
                if (@table_name not in (select table_name from #tablenametab ))
                        begin
                        declare ms_cri_cnames cursor local for
                                select column_id, quotename(column_name,'['), column_usage
                                from sys.dm_db_missing_index_columns(@index_handle)

                        open ms_cri_cnames
                        fetch next from ms_cri_cnames into @column_id, @column_name,
@column_usage
                        declare @index_name sysname
                        declare @include_column_list nvarchar(517)
                        declare @key_list nvarchar(517)
                        select @index_name = '_MS_Sys'
                        select @key_list = ''
                        select @include_column_list = ''
                        declare @num_keys smallint
                        declare @num_include_columns smallint
                        select @num_keys = 0
                        select @num_include_columns = 0

                        while @@fetch_status >= 0
                        begin
                                if (@column_usage = 'INCLUDE')
                                        begin
                                        if (@num_include_columns = 0)
                                                SET @include_column_list = @column_name
                                        else
                                                SET @include_column_list = @include_column_list
+ ', ' +@column_name

                                        SET @num_include_columns = @num_include_columns + 1
                                        end
                                else
                                        begin
                                        if (@num_keys = 0)
                                                SET @key_list = @column_name
                                        else
                                                SET @key_list = @key_list + ', ' +@column_name

                                        SET @num_keys = @num_keys + 1
                                        SET @index_name = @index_name + '_'+cast ( @column_id
as nvarchar(10))
                                        end

                                fetch next from ms_cri_cnames into @column_id,
@column_name, @column_usage
                        end
                        close ms_cri_cnames
                        deallocate ms_cri_cnames

                        if (@num_include_columns > 0)
                                SET @exec_stmt = 'CREATE INDEX ' +  @index_name  + ' ON ' +
@table_name + '(' + @key_list + ') INCLUDE ('+ @include_column_list + ')'-- WITH
(ONLINE = ON)'
                        else
```

21

```
                          SET @exec_stmt = 'CREATE INDEX ' + @index_name + ' ON ' +
@table_name + '(' + @key_list + ')'-- WITH (ONLINE = ON)'
                    insert INTO indexList (execstmt) values (@exec_stmt)
            end

            fetch next from ms_cri_tnames into @database_id, @object_id,
@table_name, @index_handle
      end
      deallocate ms_cri_tnames
      drop table #tablenametab
      select * FROM indexList
      drop table indexList
set nocount off
```

Manipulating File Groups

File groups on the computer that is running SQL Server consist of named collections of one or more files that form single units of allocation or that are used for the administration of a database. You can improve the performance of Microsoft Dynamics CRM installed on the same computer as SQL Server by following the recommendations described in this section.

Configuring the tempdb Database

We recommend that you configure the **tempdb** database with multiple files, equal to the number of processors that are available to SQL Server.

Note: In this context, processors can be either logical processors (if hyperthreading is available and enabled) or processor cores (if the database server user multi-core technology).

Allocate each **tempdb** data file with enough space, so that auto-grow does not occur. The amount of space you allocate to **tempdb** depends on the unique requirements of your Microsoft Dynamics CRM implementation. However, as a general guideline, the total size of all **tempdb** data files should be at least 10% of the size of the user databases that exist on SQL Server.

For example, if the combined size of all Microsoft Dynamics CRM databases is 200 GB, then the size of the **tempdb** database should be 20 GB. If the SQL Server instance is running on a four-processor server, you would divide 20 GB evenly divided into four files of 5 GB each.

Storing Logs and Databases on Devices Separate from the Data

You can improve performance by putting the database logs and databases on a physical disk that is separate from the main data device. Because data modifications are written to the log and to the database (and also to the **tempdb** database, if temporary tables are used), having three different locations on different disk controllers provides significant benefits. This section describes how to move the **MSDB**, **Master**, **model**, and **tempdb** databases.

Moving the MSDB Database

Note If you are using this procedure when you also move the **MSDB** and **model** databases, the order of reattachment must be **model** first and then **MSDB**. If **MSDB** is reattached first, it must be detached and not reattached until after **model** has been attached.

In SQL Server 2000, system databases cannot be detached using the **sp_detach_db** stored procedure. Executing sp_detach_db 'msdb' will fail with the following message:

```
"Server: Msg 7940, Level 16, State 1, Line 1
System databases master, model, msdb, and tempdb cannot be detached."
```

To move the MSDB database on the computer that is running SQL Server 2000:

1. In SQL Server Enterprise Manager, right-click the server name and then click **Properties**.

2. On the **General** tab, click **Startup Parameters**.

3. Add a new parameter as **-T3608**.

4. Click **Add**, and then click **OK** two times to close the dialog boxes.

5. Stop SQL Server and then restart SQL Server.

6. Make sure that the SQL Server Agent service is currently not running.

7. Run the following script in SQL Query Analyzer to detach the **MSDB** database:

```
use master
go
sp_detach_db 'msdb'
go
```

8. Move the Msdbdata.mdf and Msdblog.ldf files from the current location (for example, D:\MSSQL\Data) to the new location (for example, E:\MSSQL\Data).

9. Remove the -T3608 trace flag from the startup parameters box in Enterprise Manager.

10. Stop and then restart SQL Server again.

11. Reattach the MSDB database by using the following script:

```
use master
go
sp_attach_db 'msdb','E:\Mssql\Data\msdbdata.mdf','E:\Mssql\Data\msdblog.ldf'
go
```

Note   If you try to reattach the **MSDB** database by starting SQL Server by using trace flag -T3608, you receive the following error:

```
"Server: Msg 615, Level 21, State 1, Line 1
Could not find database table ID 3, name 'model'."
```

Moving the Master Database

In order to move the **Master** database, you must change the path for the data and log files for the **Master** database in SQL Server Enterprise Manager by following these steps:

Note You can also change the location of the error log here.

1. Right-click the server in Enterprise Manager and then click **Properties**.

2. Click the **Startup Parameters** button, and you will see the following entries:

```
-dD:\MSSQL\data\master.mdf
-eD:\MSSQL\log\ErrorLog
-lD:\MSSQL\data\mastlog.ldf
```

- **-d** is the fully qualified path for the master database data file.

- **-e** is the fully qualified path for the error log file.

- **-l** is the fully qualified path for the master database log file.

3. Remove the current entries for the Master.mdf and Mastlog.ldf files, and add new entries specifying the new location:

```
-dE:\MSSQL\DATA\master.mdf
-lE:\MSSQL\DATA\mastlog.ldf
```

4. Stop SQL Server.

5. Copy the Master.mdf and Mastlog.ldf files to the new location (for example, E:\MSSql\data).

23

6. Restart SQL Server.

Moving the Model Database

To move the **model** database, SQL Server must be started with trace flag 3608 so that it does not recover any database except the **Master**.

Note   You will be unable to access any user databases during this process. When you are using this trace flag, you must not perform any operations other than the steps described in this section.

To add trace flag 3608 as a SQL Server startup parameter:

1. In SQL Server Enterprise Manager, right-click the server name, and then click **Properties**.

2. On the **General** tab, click **Startup Parameters**.

3. Add a new parameter as **-T3608**.

4. Stop and then restart SQL Server.

5. Run the following script in SQL Query Analyzer to detach the **model** database as follows:

```
use master
go
sp_detach_db 'model'
go
```
6. Move the Model.mdf and Modellog.ldf files from, for example, D:\Mssql\Data to E:\MSSql\data.

7. Run the following script in SQL Query Analyzer to reattach the **model** database as follows:

```
use master
go
sp_attach_db
'model','E:\MSSql\data\model.mdf','E:\MSSql\data\modellog.ldf'
go
```
8. Remove the **-T3608** trace flag from the startup parameters box in Enterprise Manager.

9. Stop and restart SQL Server.

   You can verify the change in file locations using **sp_helpfile**:

```
use model
go
sp_helpfile
go
```

Moving the tempdb Database

You can move **tempdb** files by using the ALTER DATABASE statement, as shown in the following procedure:

1. Run the following script in SQL Query Analyzer to determine the logical file names for the **tempdb** database by using **sp_helpfile** as follows:

```
use tempdb
go
sp_helpfile
go
```
2. The **name** column contains the logical name for each file. This example uses the default file names of **tempdev** and **templog**.

3. Use the **ALTER DATABASE** statement, specifying the logical file name as follows:

```
use master
```

```
go
Alter database tempdb modify file (name = tempdev, filename =
'E:\MSSql\data\tempdb.mdf')
go
Alter database tempdb modify file (name = templog, filename =
'E:\MSSql\data\templog.ldf')
go
```
You should receive the following messages confirming the change:

```
"File 'tempdev' modified in sysaltfiles. Delete old file after restarting SQL
Server."

"File 'templog' modified in sysaltfiles. Delete old file after restarting SQL
Server."
```
Note: Using **sp_helpfile** in **tempdb** will not confirm these changes until you restart SQL Server.

4. Stop and restart SQL Server.

For more information, see the following Knowledge Base articles on the Microsoft Help and Support Web site:

- PRB: Troubleshooting orphaned users topic in Books Online is incomplete.

- How To: Transfer logins and passwords between instances of SQL Server.

- PRB: User logon and/or permission errors after restoring dump.

Choosing an Appropriate RAID Configuration

For a database server, you should select hardware-level RAID instead of software RAID. Software RAID is immediately available through Microsoft Windows functionality and requires no additional hardware or software, but software RAID uses CPU cycles. If CPU usage is a bottleneck for you, SQL Server may not perform optimally.

Two core RAID levels are of value for a database server:

- Striping with parity (RAID 5)

- Striped mirror (RAID 0+1)

When you select a RAID level, you have to consider your cost, performance, and availability requirements. RAID 5 is less expensive than RAID 0+1, and RAID 5 performs better for read operations than write operations. Compared to software RAID, RAID 0+1 may require additional hardware and software, but RAID 0+1 performs better for write-intensive operations and for accessing the **tempdb** database.

For more information about other deployment considerations and file group manipulation, see "Improving SQL Server Performance" in the MSDN Library:

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenetchapt14.asp

Removing Messages Stored in the Microsoft Dynamics CRM Connector for Microsoft Dynamics GP Integration Database

The size of the integration database used by the Microsoft Dynamics CRM Connector for Microsoft Dynamics GP can affect the performance of Microsoft Dynamics CRM 3.0.

For more information about maintaining the size of this integration database, see Chapter 9 in the *Microsoft Dynamics CRM 3.0 Connector for Microsoft Dynamics GP Guide*. This chapter discusses how to maintain the connector. The chapter includes details on how to use the int_DatabaseCleanup stored procedure to clean up old messages in the integration database to reduce its size. By decreasing the number of old messages stored in this database, the Integration Monitor can perform better, because it will not have to query as much data in the following tables:

25

- IntMessageBodyHeader
- IntMessageBodyData
- IntMessageComponentLogs

The latest version of the connector includes many improvements for both performance and data integrity. For more information about the Microsoft Dynamics CRM 3.0 Connector for Microsoft Dynamics GP, see:

- http://www.microsoft.com/dynamics/crm/using/downloads/crmconnectorforgp.mspx

Downloading Software Updates for Performance Enhancement and Security

This section provides information about the software updates you can download to improve the performance of your organization's Microsoft Dynamics CRM 3.0 system.

You can see a complete list of all the software updates available for Microsoft Dynamics CRM 3.0 in the following Knowledge Base article, "Microsoft Dynamics CRM 3.0 updates and hotfixes":

- http://support.microsoft.com/kb/908951

Update Rollup 1 for Microsoft Dynamics CRM 3.0 provides a single download for many of the software updates described in this section:

- http://support.microsoft.com/kb/922815

Performance-Related Hotfixes

The hotfixes specifically related to performance are listed in the following table.

Note: Because of differences in how organizations use Microsoft Dynamics CRM, not all fixes described here may apply to your implementation. If you are not sure about whether a hotfix is applicable or not, the hotfix Knowledge Base article provide information about what the update fixes and the area of Microsoft Dynamics CRM to which it applies.

Hotfix Article Title	Knowledge Base Link	In Rollup
System performance decreases regardless of the file size of the dataset in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/default.aspx?scid=kb;EN-US;912210	Yes
An entities grid is populated slower than you expect when you move to a custom entity in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/default.aspx?scid=kb;EN-US;913462	Yes
Error message in Microsoft Dynamics CRM 3.0: "SQL Server timeout expired"	http://support.microsoft.com/default.aspx?scid=kb;EN-US;913802	Yes
Error message when you use Microsoft Dynamics CRM 3.0: "Buffer overrun detected!"	http://support.microsoft.com/default.aspx?scid=kb;EN-US;915343	Yes
You experience slower performance than typical in Microsoft Dynamics CRM 3.0 when the MSCRM Stored Procedure Priming job is running	http://support.microsoft.com/default.aspx?scid=kb;EN-US;915722	No
An activity that has a due date in the past may be found when you search for upcoming activities in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/default.aspx?scid=kb;EN-US;917822	Yes
A list update occurs when you click an item in the List Web Part for Microsoft Dynamics CRM 3.0	http://support.microsoft.com/default.aspx?scid=kb;EN-US;917830	No
Internet Information Services stops responding on a Microsoft Dynamics CRM 3.0 server	http://support.microsoft.com/default.aspx?scid=kb;EN-US;921065	No

Hotfix Article Title	Knowledge Base Link	In Rollup
You experience timeouts or slow performance when you reassign records from one user to another user in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/default.aspx?scid=kb;EN-US;921235	No
The Microsoft Dynamics CRM client for Outlook stops responding when you promote an e-mail message	http://support.microsoft.com/default.aspx?scid=kb;EN-US;921653	Yes
You experience slow performance when you try to load forms in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/kb/927854	No
1) Rendering the detail form for an existing service via Settings | Services is slow and 2) Rendering the Resource Tree when finding available resources via Service Calendar | Schedule is slow	http://support.microsoft.com/kb/924425	No
The "Available Times" Web page is slowly populated when you click "Find Available Times" on the "Schedule Service Activity" Web page in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/kb/924425	No
You can view the data of all system users when you click "Find Available Time" in the "Schedule Service Activity" dialog box in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/kb/925473	No
The address book provider for the Microsoft Dynamics CRM client for Outlook may take several hours to synchronize if there are many records in entities in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/kb/925780/	No
You experience timeouts or slow performance after you change a security role in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/kb/918990	No
You can view the data of all system users when you click "Find Available Time" in the "Schedule Service Activity" dialog box in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/kb/925473	No

Microsoft SQL Server Updates

The updates for Microsoft SQL Server 2000 applicable to Microsoft Dynamics CRM are listed in the following table:

Hotfix Article Title	Knowledge Base Link
Not all memory is available when AWE is enabled on a computer that is running a 32-bit version of SQL Server 2000 SP4[1]	http://support.microsoft.com/default.aspx?kbid=899761

---

[1] The KB article indicates that it is applicable if AWE is enabled, but performance can be improved in SQL Server by applying this hotfix to SQL Server 2000 SP4 systems even if AWE is not being used.

OPTIMIZING THE PERFORMANCE OF MICROSOFT DYNAMICS CRM 3.0

Configuring the Microsoft Dynamics CRM Web Application

With a basic change to the configuration of Microsoft Dynamics CRM, such as changing the default view, you can improve performance of the application.

If the database contains many records and you want to view all customer records or invoices every time that you open Microsoft Dynamics CRM, you may find that Microsoft Dynamics CRM performance deteriorates significantly. From the **Settings** page in Microsoft Dynamics CRM, you can change the default view for all kinds of records. For example, instead of displaying all the active accounts for your whole organization, you can display only the active accounts owned by the user.

Changing the Default View for Accounts

1. In Microsoft Dynamics CRM, on the **Home** page, click the **Settings** tab.

2. On the **Settings** page, click **System Customization**.

3. Under **Customize Entities**, double-click **Account**.

4. In the **Entity: Account** left navigation area, click **Forms and Views**.

   The Default view is marked with a star and lists its type as **Default Public View**.

5. Select the view that you want to set as the default view, such as **My Active Accounts**.

6. Click **More Actions** on the action toolbar and select "**Set Default**" to set the **My Active Accounts** as the default view.

7. On the **File** menu, click **Save**.

8. On the **Actions** menu, click **Publish**.

Modifying Quick Find Search Columns

You can also improve Microsoft Dynamics CRM performance by modifying the columns that the Quick Find feature searches.

Quick find views search a predefined list of fields. You can add fields that you want to search. However, you should only add fields that are appropriate for the way your organization uses the Quick Find feature. Whenever possible, these fields should be fields that exist in a SQL Server index (assuming that the Quick Find view is used frequently by the Microsoft Dynamics CRM users in your organization).

The following example shows how to check the default search columns and change them if necessary:

1. In the Microsoft Dynamics CRM **Navigation Pane**, click **Settings**.

2. On the **Settings** page, click **System Customization**.

3. Under **Customize Entities**, double-click **Account**.

4. Under **Entity: Account**, click **Forms and Views**.

5. Select the **Quick Find Active Accounts** view.

6. Under Common Tasks, click **Add Find Columns**.

7. In the **Quick find view: Accounts** form, select check boxes for the columns that you want to include in your Quick Find search and clear any check boxes for columns you want to remove.

8. On the **File** menu, click **Save**.

9. On the **Actions** menu, click **Publish**.

The **Add Find Columns** dialog box displays all fields available for the Account entity. Any fields that have the check box selected are fields that are searched when users use the Quick find feature.

For example, if **Account Name** and **Account Number** are selected, the Quick Find feature searches both Account Name and Account Number columns to find that text.

Important: To minimize the performance effect on the Microsoft SQL Server that is used for Microsoft Dynamics CRM, whenever possible, we recommend that you select three or fewer columns. For example, if 1,000,000 or more Account records exist, each additional Quick Find column greatly increases the load on the Microsoft SQL Server every time that a Microsoft Dynamics CRM user uses the Account Quick Find. We recommend that you select columns in the **Add Find Columns** dialog box that are indexed and that are appropriate columns on which to create a SQL index, such as **Address1: Street 1**. You can improve performance if you add a non-clustered index to search columns such as **Address1:ZIP/Postal Code**.

Improving Report Performance

The best way to improve report performance is by limiting the data that is used in each report. You can do this by adding parameters that users must specify when they run the report, such as date ranges or owner, or by adding additional selection criteria to a report. This section describes how to add an **Owner** parameter that enables users to select either only records they own, or all records.

If there is a noticeable delay (5 seconds or more) in displaying data in a Microsoft Dynamics CRM list, or when opening a Microsoft Dynamics CRM form on a local area network, this delay may be caused by configuration problems in the SQL Reporting Server (SRS). Typically, this only occurs when Microsoft Dynamics CRM is installed on a separate server from the SRS server.

Reasons for this delay can include the following:

- Incorrect SRS locations in the SQLRSServerURL registry key in HKEY_LOCAL_MACHINE_SOFTWARE | Microsoft | MSCRM

- DNS problems

- Missing or invalid Service Principal Names (SPNs) for the Host name or Microsoft SQL Server (both the SQL Server that contains the Microsoft Dynamics CRM databases and the SRS Microsoft SQL Server)

- Incorrect permissions on the computer that is running Microsoft SQL Server for SRS

  Note: Incorrect permissions can cause problems accessing Microsoft Dynamics CRM reports. This can prevent Microsoft Dynamics CRM from displaying the list of valid reports in lists and forms.

Troubleshooting SRS Problems

Verify that the issue is caused by SRS

a) Set the registry key HKEY_LOCAL_MACHINE | Software | Microsoft | MSCRM| SQLRSServerURL to **http://0.0.0.0**. This will cause Microsoft CRM to fail immediately when it tries to retrieve reports.

   Note: You must back up or note the value in this registry key before changing it. Changing this value to **http://0.0.0.0** breaks all Microsoft Dynamics CRM reporting functionality.

b) Try to reproduce the slow performance by taking actions such as opening the Microsoft Dynamics CRM Web application and navigating to a list, such as the **Accounts** list, or opening a Microsoft Dynamics CRM form, such as an Account record. If the list page or record form opens in less than a second or two, this indicates that the problem is with the SQL SRS server or with the network connection from the server that is running Microsoft Dynamics CRM to the Microsoft SQL SRS server.

Resolving SRS Problems

If you verify that the SQL SRS server has a problem, the following items should be checked and verified:

1. Verify that the **SQLRSServerURL** location can be accessed from Microsoft Internet Explorer by a Microsoft Dynamics CRM Administrator from the Microsoft Dynamics CRM Server. Type the address into a new Internet Explorer address bar. If the Microsoft SQL Reporting Server is working correctly, you should see a Web page that has a directory that is the same name as the Microsoft Dynamics CRM organization

name, such as "Adventure_Works_Cycle_MSCRM". Clicking the link should display all the reports available for Microsoft Dynamics CRM. If the account you are using to access this Web page has the privileges necessary to view Microsoft Dynamics CRM reports, clicking a report name, such as User Summary, should display the corresponding report.

2. If you cannot access the **SQLRSServerURL** location in Internet Explorer on the Microsoft Dynamics CRM server, the following items should be verified:

- Verify that the SQL Reporting Services service is started on the Microsoft SQL Server specified in the **SQLRSServerURL** registry key. You can check the service by running Services.Msc on the SQL Server and verifying that the service named "SQL Server Reporting Services (MSSQLSERVER)" is started and set to start automatically.

- Verify that the SQL SRS Server can be accessed over the network. Open a command prompt and type **NSLOOKUP** *SRS_SERVERNAME* where *SRS_SERVERNAME* is the name of the SRS Server specified in the **SQLRSServerURL** location. No errors should be returned and you should see the correct DNS name and IP address. If either of these is incorrect, you can use network troubleshooting tools such as NetDiag and DCDiag to analyze and find network and DNS configuration problems. You can download DCDiag from: http://www.microsoft.com/downloads/details.aspx?familyid=23870A87-8422-408C-9375-2D9AAF939FA3&displaylang=en.

  You can download NetDiag from: http://www.microsoft.com/downloads/details.aspx?familyid=1EA70814-7E6C-46E5-8C8C-3C439A732E9F&displaylang=en.

  You can also use the MPSRPT_Network.exe Microsoft Customer Support reporting tool to analyze the network configuration on the Microsoft Dynamics CRM server and on the Microsoft SQL Reporting Services server. You can download this tool from: http://www.microsoft.com/downloads/details.aspx?familyid=cebf3c7c-7ca5-408f-88b7-f9c79b7306c0&displaylang=en.

- If you have installed Reporting Services on a different server, verify that the additional setup tasks have been performed. For more information about these additional tasks, see "Microsoft CRM 3.0: Additional Setup Tasks Required if Reporting Services is Installed on Different Server": http://www.microsoft.com/downloads/details.aspx?FamilyID=51bf9f20-bd00-4759-8378-b38eefda7b99&DisplayLang=en.

  For additional troubleshooting information about Trust for Delegation and Kerberos Double-Hop authentication problems, see the article "Troubleshooting Kerberos Delegation": http://go.microsoft.com/fwlink/?LinkId=57546.

- If you receive the error "An error has occurred. For more information, contact your system administrator" or "The request failed with HTTP status 401: Unauthorized" when you try to access reports in Microsoft Dynamics CRM 3.0, see the Knowledge Base article "Error message when you try to access Reports in Microsoft CRM 3.0: "Server Error in '/' Application"": http://support.microsoft.com/default.aspx?scid=kb;EN-US;916168.

- If you receive the error "Reports.config has invalid schema, and could not be loaded." when you access a report, see the Knowledge Base article "You receive an error message when you access Calendar or Reports in Microsoft CRM":

http://support.microsoft.com/default.aspx?scid=kb;[LN];916163.

You may see this error if you have installed Microsoft .Net Framework 2.0 on the Microsoft Dynamics CRM Server or on SQL Server 2000 with Reporting Services, and the ReportServer virtual directory and the Microsoft Dynamics CRM Web application are set to use ASP.Net version 2.0.50727 instead of 1.1.4322.

Note: SQL Server 2005 Reporting Services can use the ASP.Net version 2.0.50727 without errors.

- Slow performance opening Microsoft CRM lists or forms can also indicate invalid permissions for SRS reports. You can correct this by republishing the Microsoft Dynamics CRM SRS reports. Microsoft Dynamics CRM Reports can be republished by a user who has Microsoft CRM Administrative rights. To do this, follow these steps:

a) Open the Microsoft Dynamics CRM Web application.

b) In the Navigation Pane, under Workplace, click Reports.

c) Select the report you want to republish, and then on the **More Actions** menu click **Edit Report**.

d) Click **Browse** in the report form, navigate to the "<Microsoft CRM Installation Dir>\Reports\MSCRM" directory, and then select the report file that you opened. In other words, if you were editing the Account Distribution Report, select the **Account Distribution.rdl** file.

e) Click **Save and Close** to publish the report again.

f) Verify that each Microsoft Dynamics CRM user is a member of the Reporting Group in Active Directory in the Microsoft Dynamics CRM organizational unit (OU). This enables a user to access Microsoft Dynamics CRM reports.

Report Scheduling Wizard

Ad hoc reports performed during peak hours can cause decreases in performance. The suggestions described in this section may also help you avoid performance issues caused by long-running reports, custom reports, or reporting on large datasets.

Warning: Before You Perform any of the following optimization procedures, back up your databases and Active Directory. If you do not back up these items, you risk losing the information that is contained in them.

- For long-running reports, use the Report Scheduling Wizard.

- As soon as it is installed, the Report Scheduling Wizard is available from the Reports area of Microsoft Dynamics CRM. It can be used by any user who has the Manage Reports privilege to schedule any Reporting Services report to run on a daily, weekly, or monthly schedule, and to create a snapshot of a report that includes data from a specific time point. You can do this to schedule the reports to run during non-business hours to offload processing to a time when most Microsoft Dynamics CRM users are not using the system.

Snapshots created by using the wizard can be made available to the user running the wizard, or can be shared with other users. For more information, see the documentation for the Report Scheduling Wizard, available as part of the following download:

http://www.microsoft.com/downloads/details.aspx?FamilyID=a101d4d9-6463-4a45-899b-3c3ee979c4d0&DisplayLang=en

- When you create custom SRS reports, review the "Performance Issues" section in the Report Writers Guide section of the Microsoft CRM 3.0 SDK, available on Microsoft.com:

  http://www.microsoft.com/downloads/details.aspx?FamilyID=9c178b68-3a06-4898-bc83-bd14b74308c5&DisplayLang=en

Dedicated Report Server

If your organization has a large data set, you may benefit by setting up a dedicated server for Microsoft Dynamics CRM reports. For more information, see the "Improving Performance of Microsoft Dynamics CRM 3.0 by Using a Dedicated Report Server" document, available to download on Microsoft.com:

- http://www.microsoft.com/downloads/details.aspx?FamilyID=c82dfbe2-db8f-4a78-92b2-7c866057cde6&DisplayLang=en

If you decide to set up a dedicated report server, there are additional tasks you must perform. For more information about these tasks, see the "Additional Setup Tasks Required if Reporting Services Is Installed on Different Server" document, available to download on Microsoft.com:

- http://www.microsoft.com/downloads/details.aspx?FamilyID=51bf9f20-bd00-4759-8378-b38eefda7b99&DisplayLang=en

Preparing to Add Parameters to Your Reports

Tip 1: Create a report in 15 minutes or less

For basic information about how to create reports for Microsoft Dynamics CRM, refer to the following article on Microsoft.com

- http://www.microsoft.com/dynamics/crm/using/customizing/reporttutorial.mspx

Tip 2: Make the report pre-filterable

When you create a report, you can configure it to have a default filter that each user can edit before they run the report. This process is referred to as making the report pre-filterable. There are two advantages to making a report pre-filterable:

- The default filter prevents users from unintentionally running the report on all records. By default, this filter selects active records that were modified in the last 30 days. If you have the Manage Reports privilege, you can define specific default criteria for the default filter for each report.

- Users can edit the filter to find exactly the data that they need on the first try.

For Microsoft Dynamics CRM to make a report pre-filterable, you must specify the CRMAF_ prefix in your SQL query when you create your report in Report Designer. When you add this prefix to at least one filtered view in the query, Microsoft Dynamics CRM adds a default filter to the report. For each filtered view that has this prefix in the query, users can edit filter criteria. For example, if your query includes the FilteredAccount and FilteredContact views, and your SQL query uses CRMAF_FilteredAccount and FilteredContact, the report will have a default filter. Users will be able to edit criteria related to accounts, but will be unable to edit criteria related to contacts.

For more information about pre-filtering reports, see the "Using Filters in a Report" section of the Microsoft Dynamics CRM 3.0 *Report Writer's Guide*, available in the MSDN Library:

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/CrmSdk3_0/htm/v3d0usingfiltersinareport.asp

Dynamic Excel or Filtered View queries

The article "2 ways to see combined account and order data" on Microsoft.com describes how to create a query to combine data from two related record types, display that data in Microsoft Office Excel, and make it available as a report in Microsoft Dynamics CRM:

- http://www.microsoft.com/dynamics/crm/using/reporting/multentity.mspx

To limit the number of records a report returns if you are using a dynamic Excel worksheet or using a Filtered View query (this includes FilteredView queries in custom Microsoft Dynamics CRM SQL Reporting Services reports), consider making it more restrictive. If a field in the WHERE clause is used frequently, verify that a non-clustered index exists on that field.

```
SELECT    a.name,
          a.accountcategorycodename,
          a.industrycodename,
          a.customersizecodename,
          o.totalamount,
          o.submitdate
FROM      FilteredAccount AS a INNER JOIN
          FilteredSalesOrder AS o
ON        a.accountid = o.accountid
WHERE     (a.statecode = 0)
ORDER BY a.name
--more restrictive by adding condition to WHERE Clause with a.City = 'Fargo'
SELECT    a.name,
          a.accountcategorycodename,
          a.industrycodename,
          a.customersizecodename,
          o.totalamount,
          o.submitdate
FROM      FilteredAccount AS a INNER JOIN
          FilteredSalesOrder AS o
ON        a.accountid = o.accountid
WHERE     (a.statecode = 0)
AND       (a.City = 'Fargo')
ORDER BY a.name
```

Optimizing Microsoft Dynamics CRM Outlook and Laptop Client Performance

Microsoft Dynamics CRM Client for Microsoft Office Outlook

The requirements for Microsoft Dynamics CRM 3.0 client for Microsoft Office Outlook are documented in the *Microsoft Dynamics CRM 3.0 Implementation Guide* (section 5-3). However, these requirements are *minimum* requirements. To boost performance, system specifications beyond these minimum requirements are required.

Microsoft Dynamics CRM Laptop Client

Because the Microsoft Dynamics CRM 3.0 laptop client for Microsoft Office Outlook has offline functionality, more powerful hardware client requirements for the client are typically required. We recommend for the laptop client to have *at least* the following requirements to adequately meet performance expectations:

Microsoft Windows Edition	Processors	RAM	Network
Windows XP or a later version	1 single processor or 1 dual-core processor	1 GB or more	100 mbps or more

For software requirements and hardware minimum requirements (what it takes for the laptop client to run), see the *Microsoft Dynamics CRM 3.0 Implementation Guide*:

- http://www.microsoft.com/downloads/details.aspx?familyid=1ff067f8-4f77-40f0-ae9c-68ada7d4f16a&displaylang=en

The following list includes several practices we recommend for optimizing laptop client offline synchronization. This can affect performance on the computer that is running Microsoft SQL Server, in addition to performance of your overall Microsoft Dynamics CRM deployment.

- Assign all users roles with the *minimum* access levels and permissions that are required for them to do their jobs. This will help guarantee that data synchronization to the laptop client's performance is optimized.

- Optimize local data filtering for each laptop client. In other words, limit the data any user takes offline. Offline data filtering is available by using the Microsoft Outlook toolbar after the laptop client is installed (on the **CRM** menu, click **Local Data**).

- Have an approved security data plan (data access levels and rights within Microsoft Dynamics CRM) for the organization's implementation.

- Install high-speed connections at remote branches to improve offline synchronization performance for the laptop client.

Tip: If you have questions on other ways to increase synchronization performance, contact your regional Microsoft Customer Support Services office.

Other processes and applications can also adversely affect Microsoft Dynamics CRM laptop client performance. Turning off any of these processes that are not critical to your business may be necessary to achieve your target performance levels. The following items may affect Microsoft Dynamics CRM laptop client performance (this is not an all-inclusive list):

- Any other third-party Microsoft Office Outlook add-ins

- Third-party software on the client computer that is not critical to a user's business needs, such as games and music programs
- Disk fragmentation

More Information

"Hitting the road with Microsoft Dynamics CRM laptop client for Outlook":

- http://www.microsoft.com/dynamics/crm/using/sales/localdatagroup.mspx

"Speed up data synchronization"

- http://www.microsoft.com/dynamics/crm/using/configure/improvesync.mspx

Limit the record types you synchronize

If you must synchronize contacts, tasks, and appointments, limit the record types you synchronize to those that you must have offline.

1. In Microsoft Dynamics CRM client for Outlook, on the **CRM** menu, click **Options**.

2. In the **Set Personal Options** dialog box, on the **Synchronization** tab, select only the record types you must synchronize.

Deactivate local data groups that you do not use

Deactivate local data groups for areas that you do not use in Microsoft Dynamics CRM client for Outlook.

1. On the **Data Groups** tab, select one or more data groups.

2. On the toolbar, click **Deactivate**. This deactivates all the local data groups, and then moves them to the **Inactive Data Groups** tab.

Reduce the number of records to synchronize

Modify local data groups for areas that you do use in Microsoft Dynamics CRM client for Outlook.

1. In the Microsoft Dynamics CRM client for Outlook, on the **CRM** menu, click **Local Data**.

2. Reduce the number of records that you own or share. This includes templates, contracts, and sales literature. To view the owner of a record, on the **Details** tab, see the **Owner** field.

After you implement any of these suggestions, synchronize the data manually when you are still online. The first synchronization will be slower than later synchronizations because Microsoft Dynamics CRM must remove records from the local data store.

Optimizing Performance of Microsoft Dynamics CRM Customizations

Whether you are creating custom entities or users are querying for them, you can optimize the way that Microsoft Dynamics CRM works with your organization-specific custom entities.

Optimizing the Creation of Custom Entity

Creating custom entities can be a time-consuming process. However, if the custom entities that you are creating or importing do not require activities or notes, you can reduce the time that the system takes to create or import them by turning off activities and notes.

Optimizing Queries for Custom Entity

When new columns are added to an entity within the Microsoft Dynamics CRM database, they are added to an extension table, not to the entity's base table. For example, when customizing the SalesOrder entity, new columns are not added to SalesOrderBase but to SalesOrderExtensionBase. The purpose of this approach is to help manage and to reduce excessive growth in row length to the based tables in the database schema.

The exact location of a column (in either the base or extension base table) is not important because access to these tables and their related tables occurs through views. For example, a query to the SalesOrder View can reference columns from both SalesOrderBase and SalesOrderExtensionBase in any part of the query's structure:

- select list
- WHERE clause
- ORDER BY clause

Determining the Table that Contains a Specific Column

In some cases, for performance reasons, you must determine which table contains a specific column. The following query provides an example:

```
select top 51
salesorder.CustomerId as 'customerid',
salesorder.CustomerIdName as 'customeridname',
salesorder.CustomerIdType as 'customeridtype', salesorder.CustomerIdDsc as
'customeriddsc',
salesorder.ContactId as 'contactid',
salesorder.AccountId as 'accountid',
salesorder.New_Total as 'new_total',
salesorder.SubmitDate as 'submitdate',
salesorder.New_PONumber as 'new_ponumber',
salesorder.New_JobNumber as 'new_jobnumber',
salesorder.New_OrderNumber as 'new_ordernumber',
salesorder.SalesOrderId as 'salesorderid'
from SalesOrder as salesorder
where salesorder.DeletionStateCode in (0)
and (salesorder.StateCode = @P1)
AND (((salesorder.OwningUser is null) OR(salesorder.OwningUser = @P2)
 OR (salesorder.OwningBusinessUnit in
(              select biz.SubBusinessId from BusinessUnitMap as biz
where biz.BusinessId = @P3))
 OR ( salesorder.SalesOrderId in (
select POA.ObjectId from PrincipalObjectAccess POA
join SystemUserPrincipals sup
on   POA.PrincipalId = sup.PrincipalId
      where sup.SystemUserId = @P4
and   POA.ObjectTypeCode = @P5 and
((POA.AccessRightsMask|POA.InheritedAccessRightsMask) & 1)=1))))
```

```
order by salesorder.New_OrderNumber desc ,
salesorder.SalesOrderId asc
```

The *top 51* operator is highlighted in the previous select statement. The *top n* operator is used to limit the number of rows produced by a query to the number that was specified. In the previous example query, *top 51* is specified to limit the number of rows returned to what will fill a single page.

The query's *ORDER BY* clause is also highlighted. The **New_OrderNumber** column is a custom column added to SalesOrderExtensionBase, and the **SalesOrderID** column is a standard column from the SalesOrderBase table. If the *ORDER BY* clause references columns from both the base and extension base tables, the execution plan for the query will include a sort operator and the complete result of the query must be sorted before the *TOP* operator is evaluated. If the number of rows in the underlying table is large or the criteria that was specified in the *WHERE* clause is not highly selective, or both, query performance can be poor.

Note: There are no indexing strategies that can counteract this behavior.

The previous query's execution plan displays the following output (abridged):

```
Rows         Executes     StmtText
-----------  -----------  ---------------------------------------------------------------
-----------
51           1            select top 51 salesorder.CustomerId as
'customerid',salesorder.Custom
51           1              |--Top(51)
51           1                   |--Compute Scalar(DEFINE: ([Expr1029]=If
([SalesOrderBase].[Acc
51           1                        |--Nested Loops(Left Outer Join, OUTER
REFERENCES: ([Sales
51           1                             |--Nested Loops(Left Outer Join, OUTER
REFERENCES: ([
51           1                             |    |--Sort(ORDER
BY: ([SalesOrderExtensionBase].[Ne
2069544      1                             |    |    |--Nested Loops(Left Outer Join,
OUTER REF
2069544      1                             |    |         |--
Filter(WHERE: ((([SalesOrderBase].[
4324415      1                             |    |         |    |--Nested Loops(Left
Semi Join,
4324415      1                             |    |         |         |--Nested
Loops(Left Semi J
4324415      1                             |    |         |         |    |--Clustered
Index Sca
2069544      4324415                       |    |         |         |    |--Row Count
Spool
955909       1913039                       |    |         |         |         |--
Clustered Inde
0            2254871                       |    |         |         |--Nested
Loops(Inner Join,
187885       2254871                       |    |         |              |--Clustered
Index See
0            187885                        |    |         |              |--Clustered
Index See
2069544      2069544                       |    |         |--Clustered Index
Seek(OBJECT: ([ACME
0            51                            |         |--Clustered Index
Seek(OBJECT: ([ACME_MSCRM].[d
51           51                            |--Clustered Index
Seek([ACME_MSCRM].[dbo].[ContactB
```

The elapsed time for this query is three minutes.

The solution to this problem is to guarantee that all columns on the *ORDER BY* clause derive from a single table, and that we build an index in order to satisfy the *ORDER BY* requirements and as much of the query's *WHERE* clause selection criteria as possible.

Determining this ideal index is likely to be an iterative process. However, when you implement it correctly, the performance benefit can be very significant.

Changing the ORDER BY on a Microsoft Dynamics CRM View

The following steps change the ORDER BY value for the SalesOrder entity's Active Orders view.

1. In Microsoft Dynamics CRM, on the **Home** page, click the **Settings** tab.

2. On the **Settings** page, click **System Customization**.

3. Under **Customize Entities**, double-click **Order**.

4. In the **Entity: Order** left navigation area, click **Forms and Views**.

5. Select the **Active Orders** view.

6. Click **Configure Sorting** in Common Tasks.

7. Select a column name that exists in the primary SalesOrderBase table instead of any custom columns that will be contained in the SalesOrderExtensionBase table.

   Note: In the earlier sample query and view, you would do this by changing the sort from the **NEW_OrderNumber** column to the **SalesOrderId** column (assuming that this is a column that exists in the View definition)

8. In the **Configure Sort Order** dialog box, click **OK**.

9. On the **File** menu, click **Save**.

10. In the **Entity: Order** form, on the **Actions** menu, click **Publish**.

To illustrate the effect the proposed solution has on the example query described to this point, the first *ORDER BY* column is removed from that clause of the query. This leaves a single *ORDER BY* column, **SalesOrderId**, from a single table, SalesOrderBase, in the view's definition. With suitable indexing, the sort operator can then be eliminated from the query's execution plan, and the *TOP 51* rows are returned very quickly.

```
select top 51
        salesorder.CustomerId as 'customerid',
        salesorder.CustomerIdName as 'customeridname',
        salesorder.CustomerIdType as 'customeridtype', salesorder.CustomerIdDsc as
'customeriddsc',
        salesorder.ContactId as 'contactid',
        salesorder.AccountId as 'accountid',
        salesorder.New_Total as 'new_total',
        salesorder.SubmitDate as 'submitdate',
        salesorder.New_PONumber as 'new_ponumber',
        salesorder.New_JobNumber as 'new_jobnumber',
        salesorder.New_OrderNumber as 'new_ordernumber',
        salesorder.SalesOrderId as 'salesorderid'
from SalesOrder as salesorder
where salesorder.DeletionStateCode in (0)
and (salesorder.StateCode = @P1)
AND (((salesorder.OwningUser is null) OR(salesorder.OwningUser = @P2)
 OR (salesorder.OwningBusinessUnit in
(       select biz.SubBusinessId from BusinessUnitMap as biz
        where biz.BusinessId = @P3))
 OR ( salesorder.SalesOrderId in (
        select POA.ObjectId from PrincipalObjectAccess POA
        join SystemUserPrincipals sup
        on   POA.PrincipalId = sup.PrincipalId
        where sup.SystemUserId = @P4
        and  POA.ObjectTypeCode = @P5 and
((POA.AccessRightsMask|POA.InheritedAccessRightsMask) & 1)=1))))
order by salesorder.SalesOrderId asc
```

When you examine the revised query's execution plan, it is clear that it has significantly changed:

- The sort operator is gone, as predicted.

- From the rows and executions columns of the query plan, it is clear that the query accesses far fewer rows from the tables included in the query:

```
Rows        Executes    StmtText
----------- ----------- ------------------------------------------------------------
----------
51          1           select top 51 salesorder.CustomerId as
'customerid',salesorder.Custom
51          1             |--Top(51)
51          1                  |--Compute Scalar(DEFINE: ([Expr1029]=If
([SalesOrderBase].[Acc
51          1                       |--Nested Loops(Left Outer Join, OUTER
REFERENCES: ([Sales
51          1                            |--Nested Loops(Left Outer Join, OUTER
REFERENCES: ([
51          1                            |    |--Nested Loops(Left Outer Join,
OUTER REFERENC
51          1                            |    |    |--
Filter(WHERE: ((([SalesOrderBase].[Ownin
104         1                            |    |    |    |--Nested Loops(Left Semi
Join, WHERE
104         1                            |    |    |         |--Nested Loops(Left
Semi Join,
104         1                            |    |    |         |    |--Clustered
Index Scan(OBJ
51          104                          |    |    |         |    |--Row Count
Spool
24          47                           |    |    |         |         |--Clustered
Index See
0           53                           |    |    |         |--Nested Loops(Inner
Join, OUTE
53          53                           |    |    |              |--Clustered
Index Seek(OBJ
0           53                           |    |    |              |--Clustered
Index Seek(OBJ
51          51                           |    |    |--Clustered Index
Seek(OBJECT: ([ACME_MSCR
0           51                           |    |--Clustered Index
Seek(OBJECT: ([ACME_MSCRM].[d
51          51                           |--Clustered Index
Seek(OBJECT: ([ACME_MSCRM].[dbo].[
```

Compared to the original version of the query described earlier, which took three minutes to execute, the elapsed time for this revised query is 260 milliseconds.

There are important considerations that must be made when you consider the solution described in this section:

- There may be important business reasons for the choice of order in the query's result. You cannot remove or change a column in the *ORDER BY* clause arbitrarily without considering the effect it may have on your organization's business. Improving a query's performance by diminishing its business value is not a good tradeoff.

- Determining the correct index to improve the performance of this and other queries can be an iterative process. Test each index. Testing should include a variety of selection criteria that may be common for the specific query. One set of criteria may yield the performance increase you expect for an index, but different criteria may have no effect.

Optimizing Performance for Custom Microsoft Dynamics CRM SDK Applications

This section describes several techniques that you can use to optimize the performance of the custom application, plug-in, or add-in that you have developed by using the Microsoft Dynamics CRM 3.0 SDK.

For complete details about how to develop custom applications for Microsoft Dynamics CRM 3.0, refer to the SDK, available from the MSDN Library:

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/CrmSdk3_0/htm/v3d0microsoftcrmv3d0sdk.asp

Retrieve only needed columns and rows

Retrieve only the columns you actually need to achieve your application's business goals. To restrict the data that the FetchXML and ConditionExpressions queries return, use *Condition* attributes. This becomes even more important when Microsoft Dynamics CRM users access the data from a Wide Area Network (WAN) with higher network latencies.

Similarly, you can use paging to restrict the number of rows your custom application returns.

Test your application in an integrated environment

If Microsoft Dynamics CRM is integrated with other systems, consider the performance of Microsoft Dynamics CRM when it is integrated. Performance on test systems may not reflect that of production Microsoft Dynamics CRM servers if the test server does not have the same integrations to other systems as the production Microsoft Dynamics CRM Server. Also, performance results may differ if the test system does not have the same amount of data or similar data compared to the production Microsoft Dynamics CRM servers.

Optimizing Internet Information Services and WAN Performance

Configuring Microsoft Internet Information Services (IIS) settings on the server that is running Microsoft Dynamics CRM can benefit both Microsoft Dynamics CRM itself and any custom applications, plug-ins, or add-ins that you may have developed by using the Microsoft CRM 3.0 SDK. This section describes changes that you can make to IIS settings that can improve performance.

For general information about how to configure IIS to improve performance of Web service calls from ASPX pages, see the "At Your Service: Performance Considerations for Making Web Service Calls from ASPX Pages" article on MSDN:

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservice/html/service07222003.asp

Working around the HTTP Specification's Two-Connection Limit

The HTTP specification indicates that an HTTP client should make a maximum of two simultaneous TCP connections to any single server. This keeps a single browser from overloading a server that has connection requests when it browses to a page that has lots of images, such as 120 embedded thumbnail images. Instead of creating 120 TCP connections and sending an HTTP request on each, the browser will only create 2 connections and then start to send the 120 HTTP requests for the thumbnail images on those two connections.

The problem with this approach becomes clear when you consider an example with 50 simultaneous users. If you had to make a Mappoint Web Service call for each of those users, you would have 48 users sitting around waiting for one of those two connections to become available.

You may be able to discover the source of a performance bottleneck by manipulating the two-connection limit. Because the two-connection limit is part of the HTTP specification, we do not recommend making this change permanently on a production server.

The default two-connection limit for connecting to a Web resource can be controlled by using a configuration element called **connectionManagement**. The **connectionManagement** setting enables you to add the names of sites where you want a connection limit that differs from the default. The following code can be added to a typical Web.config file to increase the default value for all servers with which you are connecting, to a connection limit of 40:

```
<configuration>
  <system.net>
    <connectionManagement>
      <add address="*" maxconnection="40" />
    </connectionManagement>
  </system.net>

  <system.web>
```

Note: There is never a limit to the number of connections that you can make to a local computer. Therefore, if you are connecting to *localhost*, this setting has no effect.

Configuring Microsoft .NET ThreadPool Settings

For more information about how to configure Microsoft .NET ThreadPool settings, see the Knowledge Base article "Contention, poor performance, and deadlocks when you make Web service requests from ASP.NET applications":

- http://support.microsoft.com/kb/821268

You can tune the parameters in your Machine.config file to best fit your situation. However, if you are making one Web service call to a single IP address from each ASPX page, we recommend that you tune the parameters in the Machine.config file so that they use the following settings:

- Set the values of the *maxWorkerThreads* parameter and the *maxIoThreads* parameter to **100**.

- Set the value of the *maxconnection* parameter to **12*N** (where *N* is the number of CPUs that you have).

- Set the values of the *minFreeThreads* parameter to **88*N** and the *minLocalRequestFreeThreads* parameter to **76*N**.

- Set the value of *minWorkerThreads* to **50**.

    Important: By default, *minWorkerThreads* is not in the configuration file. You must add it.

Several of these recommendations include a formula that calculates the number of CPUs on a server. The variable that represents the number of CPUs in the formulas is *N*. For these settings, if you have hyperthreading enabled, you must use the number of logical CPUs instead of the number of physical CPUs. For example, if you have a four-processor server for which hyperthreading has been enabled, the value of *N* in the formulas will be **8** instead of **4**.

Note: When you use this configuration, you can execute a maximum of 12 ASP.NET requests per CPU at the same time because **100-88=12**. Therefore, at least 88*N worker threads and 88*N completion port threads are available for other uses (such as Web service callbacks).

For example, you have a server that has four processors and hyperthreading enabled. Based on these formulas, you would use the following values for the configuration settings that are mentioned in this section.

```
<processModel maxWorkerThreads="100" maxIoThreads="100"
minWorkerThreads="50"><httpRuntime minFreeThreads="704"
minLocalRequestFreeThreads="608"><connectionManagement><add
address="[ProvideIPHere]" maxconnection="96"/></connectionManagement>
```

For more information, see "Improving ASP.Net Performance" in the "Improving .NET Application Performance and Scalability" section of the MSDN Library:

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenetchapt06.asp

Configuring the Memory Limit

Configuring and tuning the memory limit is very important for the cache to perform optimally. The ASP.NET cache starts trimming the cache based on a least recently used page (LRU) algorithm and the **CacheItemPriority** enumerated value assigned to the item after memory consumption is within 20% of the configured memory limit. If the memory limit is set too high, the process can be recycled unexpectedly. The application might also experience out-of-memory exceptions. If the memory limit is set too low, it

45

could increase the time that must be spent performing garbage collections. This decreases overall performance.

Empirical testing shows that the possibility of receiving out-of-memory exceptions increases when private bytes exceed 800 MB. A good rule to follow when you are determining when to increase or decrease this number is that 800 MB is only relevant for the .NET Framework 1.0. If you have .NET Framework 1.1 and if you use the /3 GB switch, you can increase this number to 1,800 MB.

When you use the ASP.NET process model, you configure the memory limit in the Machine.config file as follows.

Note: If this limit is set too low, the Microsoft Dynamics CRMAppPool will recycle too frequently and will prevent some larger processes from completing correctly. By default this is set to a value of 60 with Microsoft .Net Framework 1.1.

```
<processModel memoryLimit="50">
```

This value controls the percentage of physical memory that the worker process can consume. The process is recycled if this value is exceeded. In the previous sample, if there are 2 GB of RAM on the server, the process recycles after the total available physical RAM falls below 50% of the RAM; in this case 1 GB. In other words, the process recycles if the memory used by the worker process goes beyond 1 GB. You monitor the worker process memory by using the **process** performance counter object and the **private bytes** counter.

More Information

For more information about how to tune the memory limit and about the /3 GB switch, see "Configure the Memory Limit" and "/3GB Switch" in "Tuning .NET Application Performance" in the "Improving .NET Application Performance and Scalability" section of the MSDN Library:

*   http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenetchapt17.asp

Configuring Web Gardens

By default, ASP.NET uses all CPUs available. In Web garden mode, ASP.NET creates one worker process for each CPU. Each process creates an affinity to a single CPU. Web gardens offer an addition layer of reliability and robustness. If a process crashes, there are other processes that can still service incoming requests.

Web gardens may perform better under the following scenarios:

*   The application uses STA objects heavily.

*   The application accesses a pool of resources that are bound by the number of processes. For example, a single process is restricted to using a particular number of resources.

To determine the effectiveness of Web gardens for your application, run performance tests, and then compare your results with and without Web gardens. Typically, in the two scenarios that are described in this section, you are likely to notice a larger benefit with servers that contain four or eight CPUs.

Note Do not use the in-process session state store or any technique that causes process affinity if Web gardens are enabled.

IIS 6.0 vs. the ASP.NET Process Model

By default, the ASP.NET Process Model is not enabled in IIS 6.0. If you enable Web gardens, you may adversely affect the performance of the garbage collector, which makes sure that unused memory is released. The performance of the garbage collector may be affected because the server version of the garbage collector is still used when bound to a single CPU. The disadvantage is that this creates one worker process per CPU. Because there is a worker process for each CPU, additional system resources are consumed.

Enabling Web Gardens by Using IIS 6.0

You can enable Web gardens in IIS 6.0 by using the Internet Information Services Manager. This is the recommended method for enabling Web gardens for Microsoft Dynamics CRM.

To enable Web gardens:

1. Right-click the application pool, **CRMAppPool**, for which you want enable Web gardening, and then click **Properties**.

2. Click the **Performance** tab.

3. In the **Web garden** section, specify the number of worker processes that you want to use.

Enabling Web Gardens by Using the ASP.NET Process Model

In the **<processModel>** section of the Machine.config file, set the **webGarden** attribute to **true**, and then configure the **cpuMask** attribute as follows.

```
<processModel webGarden="true" cpuMask="0xffffffff" />
```

Configuring the cpuMask Attribute

The **cpuMask** attribute specifies the CPUs on a multiprocessor server that are eligible to run ASP.NET processes. By default, all CPUs are enabled and ASP.NET creates one process for each CPU. If the **webGarden** attribute is set to **false**, the **cpuMask** attribute is ignored, and only one worker process runs. The value of the **cpuMask** attribute specifies a bit pattern that indicates the CPUs that are eligible to run ASP.NET threads. The following table includes several examples.

CPUs	Hexadecimal	Bit Pattern	Results
2	0x3	11	2 processes, uses CPU 0 and 1.
4	0xF	1111	4 processes, uses CPU 0, 1, 2, and 3.
4	0xC	1100	2 processes, uses CPU 2 and 3.
4	0xD	1101	3 processes, uses CPU 0, 2 and 3.
8	0xFF	11111111	8 processes, uses CPU 0, 1, 2, 3, 4, 5, 6, and 7.
8	0xF0	11110000	4 processes, uses CPU 4, 5, 6, and 7.

More Information

For more information about how to use ASP.NET Web gardens, see the Knowledge Base article "How to restrict ASP.NET to specific processors in a multiprocessor system":

- http://support.microsoft.com/default.aspx?scid=kb;en-us;815156

Disabling Tracing and Debugging

Tracing and debugging may cause performance issues. We do not recommend that you use tracing and debugging when the application is running in a production environment. This is not typically enabled in a Microsoft Dynamics CRM Server environment, but can be disabled using the following steps.

Disable tracing and debugging in the Machine.config and Web.config files, as shown in the following sample:

```
<configuration>
  <system.Web>
    <trace enabled="false" pageOutput="false" />
      <compilation debug="false" />
  </system.Web>
</configuration>
```

Disabling Microsoft Dynamics CRM Server Platform Tracing

If you are not using Microsoft Dynamics CRM 3.0 Server platform tracing for troubleshooting, it should be disabled. Platform tracing can affect the performance of a production server.

To enable Microsoft Dynamics CRM 3.0 Server platform tracing:

1. Start REGEDIT.

2. Open HKEY_LOCAL_MACHINE | Software | Microsoft | MSCRM

3. If Platform tracing is enabled, you will see the following registry keys:

   - TraceDirectory

   - TraceRefresh

   - TraceSchedule

   - TraceCategories

   - TraceCallStack

   - TraceEnabled

4. Double-click the **TraceEnabled** registry key and set it to a value of 0.

5. Double-click the **TraceRefresh** registry key

6. Increase the value of the **TraceRefresh** key by 1, and then save it.

Working with Microsoft Windows Server Terminal Services

Users in your organization may be able to use Microsoft Windows Server Terminal Services to optimize their performance in any of the following circumstances:

- Users are connecting over a WAN or VPN using the Microsoft Dynamics CRM Web application or Microsoft Dynamics CRM client for Outlook.

  Note: The Microsoft Dynamics CRM laptop client is not supported with Terminal Services with Microsoft Dynamics CRM 3.0.

- Round-trip client/server latency does not meet your expectations.

- Remote users are connecting with a finite (or budgeted) amount of bandwidth. Terminal Services can enable more users to use that connection with better results (faster, with reduced latency) because more work is performed on the Terminal Services computer instead of the remote clients.

Monitoring and Optimizing Microsoft Dynamics CRM System Performance

Performance monitoring checks systems to ensure that your organization is making optimal use of the hardware and software resources at your disposal and that the resources are meeting your performance goals. Through effective monitoring, you can determine whether you are meeting performance goals. If you are not, you can determine the areas that are causing problems. Over time, you can also use performance monitoring to generate data that can be used in trend analysis. This enables you to predict possible performance and availability issues in the future and helps you solve problems before they occur.

To effectively monitor the performance of the Microsoft Dynamics CRM system, you should examine performance monitor counters on each server that makes up the environment.

This section provides details about the following information:

- How to analyze some of the specific counters that have been found to be most useful when monitoring a Microsoft Dynamics CRM environment.

- How to tune the Microsoft .NET Framework and common language runtime (CLR).

- How to use performance counters to help determine CLR bottlenecks.

Frequently, this analysis will indicate the changes to make to optimize Microsoft Dynamics CRM. You can use the information here to help determine which, if any, hardware upgrades are necessary together with any operational practices to help improve the performance of Microsoft Dynamics CRM.

Warning   Before you perform any of the following optimization procedures, back up your databases and Active Directory. If you do not back up these items, you risk losing the information that is contained in them.

Monitoring Performance on Windows 2000 and Windows Server 2003

The Windows 2000 and Windows Server 2003 operating systems include System Monitor for analyzing the performance of the system. System Monitor consists of Performance Monitor and Network Monitor. You can add objects and counters to these monitors. For example, when you add SQL Server, Exchange 2000 Server, or Exchange Server 2003 to that environment, additional objects and counters are installed. These can prove very useful in determining the overall health of Microsoft Dynamics CRM.

Note: Remote monitoring is usually better than self-monitoring, because performance is not tainted by the load caused by monitoring. For more information about remote monitoring, see the following Knowledge Base articles:

Knowledge Base Article Title	Knowledge Base Link
Creating a Log File to Send to Customers for Remote Monitoring	http://support.microsoft.com/kb/243283
Log Is Not Started When You Try to Start a Log with Remote Counters in System Monitor.	http://support.microsoft.com/kb/240389

Creating a Baseline with System Monitor

If you use System Monitor to collect many Performance Counters, together with other tools to collect server-specific information, how can you know what numbers to expect?

In some cases, there are specific figures to look for. In many more the answer will depend on several factors, such as the specifics of the hardware that you have, the network environment in which it exists, and the functionality of the application.

To help you understand what figures to expect for your environment, you should use System Monitor to generate a baseline. You do this by measuring counters in a functional environment that works well. You can measure a baseline in your test environment. However, in your test environment, you should make sure that you are effectively simulating potential real-world use in your production environment.

As you collect your baseline figures, be aware that in typical use, the Microsoft Dynamics CRM environment will face different stresses at different times of the day. For example, there may be more stress on the system at the start of the work day, during database backup, or when reports are being run. It can be very useful to combine logs over 24 hour periods with more intensive logging during stressful periods as you collect your baseline.

Many organizations are also seasonal in nature. Your organization may, for example, have more CRM activity immediately before a holiday shopping season, or at the end of a financial year. You should continue to update baseline figures to guarantee that they accurately reflect the usage of Microsoft Dynamics CRM in your environment.

Heavily used Windows 2000 and Windows Server 2003 servers may have bottlenecks in several areas. Monitoring only the applications that are running on Windows 2000 and Windows Server 2003 will not give you information about the condition of the server itself. You should also monitor for bottlenecks in the Disk Subsystem, Memory, Processor, and Network Subsystem. Frequently, there will be multiple instances of disks and processors. Therefore, make sure that you monitor all instances (that is, each disk or each processor).

You should measure the counters shown in the following table for all servers in the Microsoft Dynamics CRM environment.

Note When monitoring disk counters, you must use the **diskperf –y** command to enable them to start on startup.

Object	Counter	Comments
Logical Disk	% Free Disk Space	Especially important on computers that are running Exchange Server and SQL Server, as databases and transaction logs may fill disk space. This results in loss of availability.
Physical Disk	Disk Reads/sec	The main reason for variation in this value is variation in the usage of your environment. If you are experiencing performance problems and these figures are still low, this counter may help provide evidence of the problem.
Physical Disk	Disk Writes/sec	The main reason for variation in this value is variation in the usage of your environment. If you are experiencing performance problems and these figures are still low, this counter may help provide evidence of the problem.
Physical Disk	Current Disk Queue Length	Generally, this should be at or near zero. On computers that are running SQL Server 2000, this counter can spike at high values but should not stay high for more than 30 seconds. Any longer indicates a potential bottleneck.

Object	Counter	Comments
Physical Disk	Avg secs per read	Generally similar to published disk speed.
Physical Disk	Avg secs per write	Generally similar to published disk speed or 1-2 milliseconds (ms) if you have write-back caching enabled on your RAID controller.
Memory	Pages/sec	Exchange 2000 and Exchange 2003 servers make heavy use of a pagefile. On an Exchange server lots of paging is not in itself an indication of a problem. For computers that are running SQL Server 2000, any paging is a detriment to performance. This number should stay fairly low. For other servers, measure your paging against your baseline.
Memory	Page Reads/sec	This value should generally be less than 100. If the value is consistently high, you may have to increase system memory.
Memory	Page Writes/sec	This value should generally be less than 100. If the value is consistently high, you may have to increase system memory.
Paging File	% Usage	You may have to increase the size of your pagefile for Exchange Server. Try to keep this counter under 70%.
Process	Page Faults/sec	A page fault can be either a cache fault or a hard disk fault. For the true number of hard disk faults, subtract the number of Cache Faults/sec from the Page Faults/sec value. For the SQLSERVR instance, this value should be at or near zero. Any SQL Server paging beyond this indicates a bottleneck.
Processor	Interrupts/sec	Will vary depending on usage in your environment.
Processor	%Processor Time	Measure for a specific processor instance. When you are using the _TOTAL instance, the total percentage can be 100 times the number of processors. When 100% of the available processor time is being used for an extended period, this indicates a need for more processors. Also see the Processor Queue Length counter for this processor.
Process	%Process Time	On Exchange servers and Microsoft Dynamics CRM servers, measure inetinfo (IIS). On domain controllers, measure lsass (security system including Active Directory), and on SQL Server computers, measure the SQLSERVR instance.
System	Processor Queue Length	This is a cumulative value for all processors. A sustained value of more than double the number of processors indicates a processor bottleneck.
Network Segment	% Net Utilization	Will vary depending on usage in your environment.

Object	Counter	Comments
Redirector	Bytes Total/sec	Will vary depending on usage in your environment.
Redirector	Network Errors/sec	A high figure will generally indicate the Redirector and one or more servers having communication difficulties.
Server	Bytes Total/sec	Will vary depending on usage in your environment.
Server	Pool Paged Peak	Indicates the correct sizes of the page files and physical memory.
Server Work Queues	Queue Length	A sustained queue length of more than four may indicate processor congestion.

For more information about how to monitor Windows 2000 objects, see the Windows 2000 Server Resource Kit:

- http://www.microsoft.com/windows2000/techinfo/reskit/

For information about performance parameters and settings for Windows Server 2003, see the "Performance Tuning Guidelines for Windows Server 2003" document:

- http://www.microsoft.com/windowsserver2003/evaluation/performance/tuning.mspx

Monitoring the Performance of the Server That Runs IIS

Microsoft Dynamics CRM server is basically an Internet Information Services (IIS) server that runs a Microsoft.NET-connected application. To monitor the overall health of the servers, you should collect information about the Windows 2000 and Windows Server 2003 counters mentioned in the previous section. One of the key counters to be measured against a baseline is the %Process Time for the inetinfo (IIS). Generally, if the Microsoft Dynamics CRM server meets the recommended hardware requirements and does not perform any other tasks, you should find no performance issues on this server.

Monitoring the Performance of Exchange 2000 and Exchange 2003

Because Microsoft Dynamics CRM uses the Exchange implementation of Simple Mail Transfer Protocol (SMTP), you must monitor the SMTP Server object. Specifically, the Microsoft Dynamics CRM-Exchange E-Mail Router (the Router) is implemented as a transport event sink that occurs on the pre-categorization event. Therefore, you should monitor counters that refer to the message categorizer. This is in addition to the Windows 2000 and Windows Server 2003 counters shown in the previous table.

The following table shows the most important counters to monitor.

Object	Counter	Comments
SMTP Server	Bytes Received/sec	The rate bytes are received by the SMTP server.
SMTP Server	Cat: Address Lookups/sec	Number of Address Lookups sent to the Active Directory per second.
SMTP Server	Cat: Categorization Completed/sec	The total number of messages submitted to the categorizer that have been categorized.
SMTP Server	Cat: LDAP Searches/sec	LDAP searches successfully dispatched per second.
SMTP Server	Cat: Messages	The total number of messages submitted to the

Object	Counter	Comments
	Submitted/sec	categorizer.
SMTP Server	Message Bytes Received/sec	The rate that bytes are received in messages.
SMTP Server	Messages Delivered/sec	The rate messages are delivered to local mailboxes (this refers to Exchange mailboxes).
SMTP Server	Messages Received/sec	The rate at which incoming messages are received.
SMTP Server	DNS Queries/sec	The rate of DNS lookups on the server.

All the counters listed in this table will vary depending on how busy the server is. Frequently, this depends on how heavily the Exchange server is being used for Exchange e-mail purposes. However, monitoring these counters will enable you to see which Exchange servers are less heavily used. You may then decide to put the E-mail Router on one of these servers.

On the Exchange server itself, you may want to use the Monitoring and Status tool. This will enable you to monitor items such as the SMTP Queue Growth and issue notifications if they continue to grow for longer than a specified length of time.

Monitoring the Performance of SQL Server 2000

Microsoft Dynamics CRM depends heavily on Microsoft SQL Server. You should make sure that you measure the Windows 2000 and Windows Server 2003 counters discussed in earlier sections. However, you should also monitor the SQL Server counters on the computer that is running SQL Server.

Use the performance counters listed in the following table to help determine performance problems with Microsoft SQL Server:

Object	Counter	Comments
SQLServer:Access Methods	Full Scans/sec	When the number of full scans is significantly more than a baseline comparison, it may indicate index statistics are out of date.
SQLServer:Buffer Manager	Buffer Cache Hit Ratio	If this value is less than 80%, the system may need additional memory resource for SQL Server. Ideally, this value is at or near 100%. When this percentage is near 100%, the server is operating at optimal efficiency (as far as disk I/O is concerned).
SQLServer:Databases	Log Growths (run against the application database instance)	Log files growing during times of heavy system usage will result in poor performance.
SQLServer:Databases Application Database	Percent Log Used (run against the application database instance)	If the percentage of log space that is used approaches 100%, transaction log backups should be performed more frequently, or the transaction log file sizes should be increased.
SQLServer:Databases Application Database	Transactions/sec (run against the	The number of transactions started for the database.

Object	Counter	Comments
	application database instance)	
SQLServer:Locks	Lock Waits/sec	Although blocking locks are unavoidable, a value significantly more than a baseline comparison that appears for a long time indicates a performance penalty caused by blocking locks. Blocking locks occur when read operations block write operations, writes block reads, or writes block other writes.
SQLServer:Locks	Number of Deadlocks/sec	Although deadlocks are unavoidable, a value significantly more than a baseline comparison that appears for a long time indicates a performance bottleneck. Deadlocks occur when operations each want a resource the other has locked. If the operations both involve writes, SQL Server must select one of the transactions and roll it back for the other transaction to continue. The undo and redo operations are the causes of less than optimal performance.
SQLServer:Memory Manager	Memory Grants Pending	The current number of processes waiting for a workspace memory grant. This counter, together with Buffer Cache Hit Ratio, can confirm a memory resource bottleneck.

On the computer that is running SQL Server, you should also consider using alerts. This will enable you to send notifications to an administrator if a particular state is reached on that computer.

Optimizing Deletion Service Performance

The Microsoft Dynamics CRM Deletion service deletes records from the Microsoft Dynamics CRM database. By default, this service runs every 4 hours. This can have adverse effects on performance. To reduce this as a risk during the day, the following steps can be performed to disable the service and run it manually during a time when users are not heavily using the system.

To schedule the deletion service as a Windows job:

1. Click **Start**, click **Run**, type Services.msc, and then press Enter.

2. In the Service console, find the Microsoft Dynamics CRM Deletion Service.

3. Right-click **Microsoft Dynamics CRM Deletion Service** and then click **Properties**.

4. Change the selection in the **Startup type** list to "Disabled", and then click **OK**.

   Note: Now that the service is disabled, it must be run manually by using the Windows task scheduler

5. On the Microsoft Dynamics CRM server, click **Start**, and then open **Control Panel**.

6. In Control Panel, double-click **Schedule Tasks**.

7. Double-click **Add Scheduled Task**.

8. In the Scheduled Task Wizard, locate your CRMDeletionService.exe file (usually located in C:\program files\Microsoft Dynamics CRM\Server\Bin).

9. Select a schedule that fits your organization's usage patterns. For example: Run daily at 1am Every Day.

10. After you configure the schedule of the task, configure the user to execute the task in the wizard. This user should be an administrative user who also has administrative access to Microsoft SQL Server.

11. On the final screen of the Scheduled Task Wizard, select the **Open advanced properties for this task when I click finish** check box.

12. In the **Run** box, add the *RunOnce* argument to the task.

    For example:

    - If your Run command was:

      "C:\Program Files\Microsoft Dynamics CRM\Server\bin\CrmDeletionService.exe"

    - It should now read:

      "C:\Program Files\Microsoft Dynamics CRM\Server\bin\CrmDeletionService.exe" –runonce

13. Repeat these steps to create another job to create indexes for new entities. Step 12 should use the argument of –runindexonce.

Load Balancing

You can install multiple Microsoft Dynamics CRM 3.0 Servers in order to balance the processing load across several servers. With multiple servers, you can also implement departmental Microsoft Dynamics CRM systems that still have access to the same Microsoft Dynamics CRM database.

Network Load Balancing (NLB) technology is designed to spread the load between the different nodes of a cluster. With NLB, administrators can add another server to the node as traffic increases (referred to as "scaling out").

For more information about how to implement Microsoft Dynamics CRM 3.0 in an NLB environment, refer to the following article on Microsoft.com:

- http://www.microsoft.com/dynamics/crm/using/deploy/clusteringmscrmservers.mspx.

Optimizing the Microsoft .NET Framework

To tune the .NET Framework, you must tune the common language runtime (CLR). Tuning the CLR affects all managed code, regardless of the implementation technology. Next, you tune the relevant .NET Framework technology, depending on the nature of the application. For example, tuning the relevant technology might include tuning ASP.NET-connected applications or Web services, Enterprise Services, and ADO.NET code. You can also use performance counters to identify CLR bottlenecks. The following sections address CLR tuning and how to use counters to identify bottlenecks.

Tuning the Common Language Runtime

Common language runtime (CLR) tuning is mostly achieved by designing and then optimizing your code to enable the CLR to perform its tasks efficiently. Your design must enable efficient garbage collection (for example, when you use the Dispose pattern and considering object lifetime correctly).

The main CLR-related bottlenecks are caused by contention for resources, inefficient resource cleanup, misuse of the thread pool, and resource leaks. For more information about how to optimize your code for efficient CLR processing, see "Improving Managed Code Performance":

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenetchapt05.asp

Use the performance counters shown in the following table to help identify CLR bottlenecks.

Area	Counter
Memory	Process\Private Bytes .NET CLR Memory\% Time in GC .NET CLR Memory\# Bytes in all Heaps .NET CLR Memory\# Gen 0 Collections .NET CLR Memory\# Gen 1 Collections .NET CLR Memory\# Gen 2 Collections .NET CLR Memory\# of Pinned Objects .NET CLR Memory\Large Object Heap size
Working Set	Process\Working Set
Exceptions	.NET CLR Exceptions\# of Exceps Thrown /sec
Contention	.NET CLR LocksAndThreads\Contention Rate /sec .NET CLR LocksAndThreads\Current Queue Length
Threading	.NET CLR LocksAndThreads\# of current physical threads Thread\% Processor Time Thread\Context Switches/sec Thread\Thread State
Code Access Security	.NET CLR Security\Total Runtime Checks .NET CLR Security\Stack Walk Depth

For more information about how to measure these counters, their thresholds, and their significance, see "ASP.NET Chapter 15, Measuring .NET Application Performance":

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenetchapt15.asp

Identifying Common Bottlenecks

The following list describes several common bottlenecks that occur in applications written using managed code and explains how you identify them using system counters.

- **Excessive memory consumption**: Excessive memory consumption can result from poor managed or unmanaged memory management. To identify this symptom, watch the following performance counters:

  - Process\Private Bytes

  - .NET CLR Memory\# Bytes in all Heaps

  - Process\Working Set

  - .NET CLR Memory\Large Object Heap size

An increase in **Private Bytes** when the **# of Bytes in all Heaps** counter remains the same indicates unmanaged memory consumption. An increase in both counters indicates managed memory consumption.

- **Large working set size**. The working set is the set of memory pages currently loaded in RAM. This is measured by **Process\Working Set**. A high value might indicate that you have loaded several assemblies. Unlike other counters, **Process\Working Set** has no specific threshold value to watch, although a high or fluctuating value can indicate a memory shortage. A high or fluctuating value accompanied by a high rate of page faults clearly indicates that the server has insufficient memory.

- **Fragmented large object heap**. Objects larger than 83 KB are allocated in the large object heap. This is measured by **.NET CLR Memory\Large Object Heap size**. Frequently, these objects are buffers (large strings, byte arrays, and so on) used for I/O operations (for example, creating a **BinaryReader** to read an uploaded image). Such large allocations can fragment the large object heap. You should consider recycling those buffers to avoid fragmentation.

- **High CPU usage**. High CPU usage is usually caused by poorly written managed code, such as code that does the following:

  - Causes excessive garbage collection. This is measured by **% Time in GC**.

  - Throws many exceptions. This is measured by **.NET CLR Exceptions\# of Exceps Thrown /sec**.

  - Creates many threads. This causes the CPU to spend large amounts of time switching between threads instead of performing real work. This is measured by **Thread\Context Switches/sec**.

- **Thread contention**: Thread contention occurs when multiple threads try to access a shared resource. To identify this symptom, watch the following performance counters:

  - .NET CLR LocksAndThreads\Contention Rate / sec

  - .NET CLR LocksAndThreads\Total # of Contentions

    To reduce the contention rate, identify and fix the code that accesses shared resources or uses synchronization mechanisms.

For more information, see the "Improving .NET Application Performance and Scalability" article in the .NET Performance section of the MSDN Library:

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenet.asp.

57

Optimizing Performance in Wide Area Network Environments

Optimizing the performance of Microsoft Dynamics CRM in wide area network (WAN) or low-bandwidth/high-latency environments requires special considerations.

No files, including graphics, icons, and static IIS content, are cached if Microsoft Dynamics CRM is installed on a site that is using SSL. If SSL is used, other ways of WAN optimization should be considered, such as using Terminal Server access for users to access Microsoft CRM through a Web browser session on a Terminal Server that is located in the same local area network as the Microsoft CRM server to eliminate any latency delays.

Configuring Content Expiration

Microsoft Dynamics CRM uses content expiration to control the Web objects cache for the clients accessing Microsoft Dynamics CRM. The original configuration of content expiration is set to 3 days. For customers using Microsoft Dynamics CRM over a slower link connection (50-200 ms latency) you may benefit from increasing the content expiration value to 15 days. This would enable a client computer that uses the Microsoft Dynamics CRM Web application or the Microsoft Dynamics CRM client for Outlook to download items into their temporary Internet files without refreshing them for 15 days. This configuration change will have the most effect when combining it with the client-side Web browser settings configuration.

To configure content expiration:

1. Open Internet Information Services (IIS) Manager from the Administrative Tools area on the Microsoft Dynamics CRM Server.

2. Right-click **Microsoft Dynamics CRM v3.0 Web Site**, and then click **Properties**.

3. Open the **HTTP Headers** tab.

4. Change the content expiration "expires after" selection to 15 days, and then click **OK**.

    This change in settings will take effect on client systems after their current content expires (in less than 72 hours).

Configuring Anonymous Access Settings

In the default configuration of Microsoft Dynamics CRM, all items within the Microsoft Dynamics CRM 3.0 Web site are protected by Windows integrated authentication. For security reasons, all static content, graphics, and dynamic content require authentication.

Authenticated elements require an additional request from the client to the server to access information for a particular object. An anonymous request is always sent first, and if the page requires authentication, the server then returns an IIS 401 error indicating that it needs authentication from the client. This in turn returns an IIS 200 success message after authentication. Turning off authentication for static elements can improve performance by reducing the number of requests and responses between the client and server, especially in a WAN environment in which there is low network bandwidth or high network latencies between the Microsoft Dynamics CRM server and the client workstation.

If you decide that some static content and graphics do not require this level of security, you can set their anonymous access flag, and allow client systems to download the content without any authentication.

Important: In most cases, turning off authentication for static content does not pose a security risk. However, you should carefully consider the security implications and potential business impact of each object for which you are considering turning off authentication.

To turn off security authentication for an object:

1. Open IIS Manager from the Administrative Tools area on the Microsoft Dynamics CRM Server.

2. Expand **Microsoft Dynamics CRM Web Site** and find the static content file for which you want to disable authentication.

    Caution: To prevent unknown consequences for your business and for your Microsoft Dynamics CRM users, do *not* disable authentication for any ASPX files or dynamic content files.

3. Right-click the object and then click **Properties**.

4. Click the **File Security** tab.

5. In the **Authentication and Access Control** area, click **Edit**.

6. In the **Authentication Methods** dialog box select the **Enable Anonymous Access** check box, and then click **OK**.

7. In the object properties window, click **OK** to return to IIS Manager.

    The object no longer requires authentication when it is downloaded.

Modifying the 401.1 and 401.2 Error Pages

The size of the default 401.1 and 401.2 error pages can have an adverse effect on Microsoft Dynamics CRM performance. You can replace these default pages with much smaller files. This improves performance.

To replace the default 401.1 and 401.2 error pages:

1. On the server that is running Microsoft Dynamics CRM and IIS, start Notepad.

2. Enter the following text:

```
<html ><body>ERROR: 401.1</body></html >
```

3. On the **File** menu, click **Save As**.

4. In the **Save As** dialog box open the following path:

    C:\Windows\Help\iisHelp\common\

5. From the **Save as type** list, select "All Files," and in the **File name** box, type **401-1_custom.htm**.

6. Click **Save**.

7. Start Notepad again.

8. Enter the following text:

```
<html ><body>ERROR: 401.2</body></html >
```

9. On the **File** menu, click **Save As**.

10. In the **Save As** dialog box open the following path:

    C:\Windows\Help\iisHelp\common\

11. From the **Save as type** list, select "All Files," and in the **File name** box, type **401-2_custom.htm**.

12. Click **Save**.

13. Open IIS Manager from the Administrative Tools area on the Microsoft Dynamics CRM Server.

14. Right-click **Microsoft Dynamics CRM Web Site** and then click **Properties**.

15. Click the **Custom Errors** tab and change the 401.1 and 401.2 error pages to the custom pages that you created earlier in this procedure.

Configuring Client-Side Browser Settings

The client-side browser settings can greatly affect the users' experience over a slower connection (connections with a latency of 50–200 milliseconds). These settings will reduce client-side resource usage and will also reduce the calls that the client must make to the server. The effect can be significant. For example, without these settings the edit form for the Account record type can make up to 115 round trips between the server and the client. After these settings have been configured, this same form can be opened in fewer than 20 round trips between the server and the client.

To configure client-side browser settings:

1. Start Internet Explorer on the client.

2. In Internet Explorer, click the **Tools** menu, and then click **Internet Options**.

3. Configure the temporary Internet files to "Automatically check for newer versions of pages" and use between 200 and 300 megabytes of disk space for temporary Internet files.

4. Click **OK**.

   The next time Microsoft Dynamics CRM loads, it may take several moments longer. However, later page loads will then use the new settings and will cache many of the Microsoft Dynamics CRM Web pages.

Software Updates for WAN Environments

This section includes information about how to improve Microsoft Dynamics CRM 3.0 performance in WAN environments by applying hotfixes, performance enhancement updates, and security updates that are currently available.

All available Microsoft Dynamics CRM 3.0 hotfixes can be seen with the following KB article, "Microsoft Dynamics CRM 3.0 updates and hotfixes":

* http://support.microsoft.com/kb/908951

The hotfixes specifically related to WAN environments are listed in the following table.

Hotfix Article Title	Knowledge Base Link
You experience slow performance when you try to load forms in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/kb/927854
An entities grid is populated slower than you expect when you move to a custom entity in Microsoft Dynamics CRM 3.0	http://support.microsoft.com/default.aspx?scid=kb;EN-US;913462

Deploying Microsoft Dynamics CRM to Virtual Servers

Deploying Microsoft Dynamics CRM in a virtual environment is not supported in production. You can install Microsoft Dynamics CRM on computers that are running either Microsoft Virtual Server 2005 or Microsoft Virtual PC 2004 (or latest version). However, note the following support conditions:

- Because of decreased performance, *do not use* Microsoft Dynamics CRM running on a virtual server as your main production business environment.

- Microsoft Customer Support Services will consider Collaboration Requests and hotfix investigations for issues that involve Microsoft Dynamics CRM and Microsoft Virtual Server 2005 or Microsoft Virtual PC 2004 only in test, development, and demonstration systems.

Additional Resources

Additional Reading

More Information about Microsoft Dynamics CRM

Microsoft Dynamics CRM 3.0 Performance and Stress Testing Toolkit:

*   http://www.microsoft.com/downloads/details.aspx?familyid=1a25db7c-5060-417c-86db-6377a84ee650&displaylang=en

Microsoft Dynamics CRM 3.0 Suggested Hardware for Deployments up to 250 Concurrent Users:

*   http://www.microsoft.com/downloads/details.aspx?familyid=E835C764-AE63-4876-87DA-786464EE7269&displaylang=en

More Information about SQL Server

Microsoft SQL Server 2000 Index Defragmentation Best Practices

*   http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/ss2kidbp.mspx

SQL Server 2000 – Maintain

*   http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/default.mspx

Top 30 Features of SQL Server 2005

*   http://www.microsoft.com/sql/prodinfo/features/top30features.mspx

Troubleshooting Performance Problems in SQL Server 2005

*   http://www.microsoft.com/technet/prodtechnol/sql/2005/tsprfprb.mspx

Advantages of a 64-bit environment:

*   http://www.microsoft.com/sql/techinfo/whitepapers/advantages-64bit-environment.mspx

Note: Microsoft Dynamics CRM Server can have its databases installed on 64-bit Microsoft SQL Server 2005. However, the Microsoft Dynamics CRM application server cannot run or be installed on a 64-bit version of Windows (Windows Server 2003 or Windows Server 2003R2)

Services

North America

Business Systems Architecture Services

The North America Microsoft Dynamics Business Systems Architecture Team provides prescriptive guidance on deployment infrastructure and hardware to partners and customers for Microsoft Dynamics deployments. Specific packaged services include Business Systems Architecture Assessments (includes hardware sizing), Health Checks and Onsite System Performance Workshops.

Note: These types of services are outside the scope of the Microsoft Dynamics CRM Technical Presales Advisory Group (TPAG) resource:

*   https://partner.microsoft.com/global/40023009

For pricing information and availability, contact MBSProfessionalServices@microsoft.com today.

Visit PartnerSource to learn more about the services that are provided by the Microsoft Dynamics Business Systems Architecture Team:

- https://partner.microsoft.com/US/40029785

United Kingdom

UK Microsoft Dynamics Consulting

The UK Microsoft Dynamics Consulting team provides a full range of service offerings that address the complete project life cycle. For help with complex sizing, performance tuning, or load testing scenarios, a series of workshops, assistance, and quality assurance offerings are available. For pricing information and availability, contact ukcrmc@microsoft.com today.

Services in Other Regions

For services in other regions, contact your local Microsoft Support Services or Microsoft Consulting Services (MCS) office to request hardware sizing or architecture services (availability may vary).

Microsoft Dynamics is a line of integrated, adaptable business management solutions that enables you and your people to make business decisions with greater confidence. Microsoft Dynamics works like and with familiar Microsoft software, automating and streamlining financial, customer relationship and supply chain processes in a way that helps you drive business success.

U.S. and Canada Toll Free 1-888-477-7989

Worldwide +1-701-281-6500

www.microsoft.com/dynamics

Microsoft®

OPTIMIZING THE PERFORMANCE OF MICROSOFT DYNAMICS CRM 3.0