Microsoft Dynamics® AX 2012

Performance data collection and analysis process

White Paper

This document outlines the core processes, techniques, and procedures that the Microsoft Dynamics AX product team uses during its fact-finding and analysis phase. The document lists the expected results from each data collection phase and describes the process for bringing the data together to create actionable steps.

June 2014

www.microsoft.com/dynamics/ax

Table of Contents

This document outlines the core processes, techniques, and procedures that the Microsoft Dynamics AX product team uses during its fact-finding and analysis phase. The document lists the expected results from each data collection phase and describes the process for bringing the data together to create actionable steps.

No matter whether you are working before or after go-live, performance tuning is an iterative process.

Tools overview

The following tools are recommended for monitoring performance:

- **Performance Monitor** – Performance Monitor is the basic tool for obtaining an overview of performance. You can review the CPU, disk, and memory counters to find performance issues. After you have identified a potential time frame when your system is suffering from performance problems, you can add more specific counters to understand the problem in greater detail. For more information, see Collect AX 2012 event traces with Windows Performance Monitor.
- **System Center Operations Manager management packs** – The Management Pack for Microsoft Dynamics AX 2012 can be used to monitor your core system. It can automatically discover the servers in your environment, monitor server availability, and find setup best practice violations.

  The Management Pack for Microsoft Dynamics AX Retail 2012 R3 automatically discovers the Retail components in your environment, and monitors the configuration and availability of those components. It provides early warnings that an operator can use to proactively identify issues that can affect the availability of the Microsoft Dynamics AX Retail system.

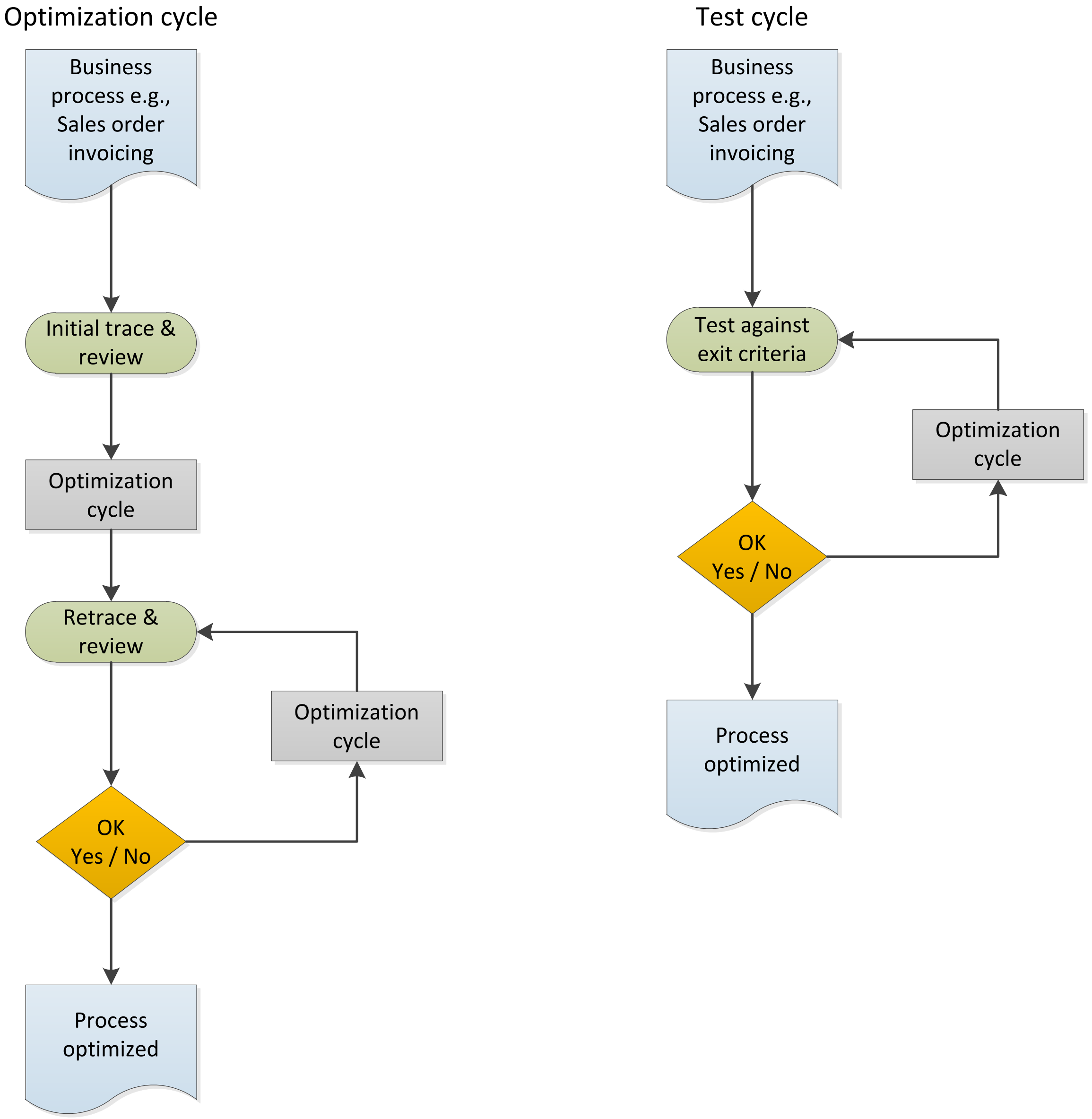
  For more information, see What's new: System Center Operations Manager Management Packs for Microsoft Dynamics AX 2012.
- **Lifecycle Services System diagnostic service** – The System diagnostic service helps administrators monitor and understand the health of one or more Microsoft Dynamics AX environments. It is a cloud-based tool that has a locally installed component that can be configured to perform the following tasks: discover on-premises Microsoft Dynamics AX environments, collect data from the environments that were discovered, run rules on the collected data, report rule violations on a dashboard, and provide reports. For more information, see System diagnostic service.
- **Performance Analyzer for Microsoft Dynamics AX (DynamicsPerf)** – DynamicsPerf can be used to collect information from Microsoft Dynamics AX and Microsoft SQL Server. It consists of a database and a collection of scripts that are used to collect information from SQL Server and Microsoft Dynamics AX. Based on this information, you can find issues such as expensive queries, locking/blocking, and, after you gain some experience, inefficient code or business processes. For more information and to download the tool, see http://Dynamicsperf.codeplex.com.
- **Intelligent Data Management Framework (IDMF)** – IDMF provides similar functionality to DynamicsPerf and also includes a user interface. It also provides tools for data management activities, such as archiving and purging. For more information, see Intelligent Data Management Framework for Microsoft Dynamics AX 2012.
- **Microsoft Dynamics AX 2012 Trace Parser** – Trace Parser consolidates information from multiple sources, such as RPC and SQL, to provide an integrated view of application performance at run time. For more information, see Walk through major features of Microsoft Dynamics AX 2012 Trace Parser (Part 1) and Walk through major features of Microsoft Dynamics AX 2012 Trace Parser (Part 2).
- **Microsoft Visual Studio Profiling Tools** – The Visual Studio Profiling Tools help you identify performance issues in source code and compare the performance of possible solutions. For more information, see Find Application Bottlenecks with Visual Studio Profiler and Beginners Guide to Performance Profiling.

Performance optimization and testing

This section provides an overview of performance optimization and testing, and describes the testing that you should perform both before and after go-live.

Performance optimization and testing cycle

The following diagram illustrates typical performance optimization and test cycles.



Performance monitoring and tuning before go-live

The goals of performance testing and monitoring before go-live include:

- Ensuring good performance of your core business processes before you move to a production environment.
- Finding and optimizing expensive code patterns.
- Finding missing indexes and configurations.

Preparation

Identify all your core scenarios. Many times, you will find 5 to 10 processes that are essential for your business. If you find other specific things, such as reports, add them to the list for performance testing.

Examples of core scenarios might include sales order creation and processing, production order creation and processing, and transfer order creation and processing.

Testing

There are three primary ways to test performance before go-live. Define how to test your scenarios by using one of the following methods:

- **Tracing** – You can use either Trace Parser or the Visual Studio Profiling Tools to get an overview of how your processes perform. You must use the Visual Studio Profiling Tools if X++ code is traversing into Microsoft Compiled Intermediate Language (CIL) – for example, if you use the **RunAs()** command, run a batch process, or work with services. Through repeated tracing of your core processes, you can find up to 90 percent of all performance issues in your pre-production system.
- **User acceptance tests** – Projects usually have multiple user acceptance tests. During these tests, you can use DynamicsPerf to find bottlenecks. You can also monitor the system by using Performance Monitor. Interviewing users can also help you find bottlenecks.
- **Benchmarks** – You can use the Microsoft Dynamics AX 2012 Performance Benchmark SDK or the Benchmark toolkit for Microsoft Dynamics AX 2009 to simulate the behavior of your system under a load.

Performance monitoring and tuning after go-live

The goals of performance testing and monitoring after go-live include:

- During the first few weeks after you go into production, monitoring performance closely to ensure that everything is running as expected.
- As time passes, and the amount of data increases, monitoring to ensure that performance continues as expected. Larger amounts of data might cause SQL Server to use different execution plans. Also, previously fast-performing code, such as while loops, might become slower, because there is more data to iterate through.

Preparation

Plan to trace all your important business processes to double-check that there are no obvious problems.

Plan a test in which as many users as possible execute their normal work, and analyze the results by using DynamicsPerf.

Test the efficiency of your infrastructure. If you have offshore offices, check the latency between them and the data center.

Double-check that all your systems are set up and configured according to best practices.

At go-live and during the first few weeks

Monitor your system closely for any potential performance problems by using Performance Monitor or the System Center Management Packs for AX 2012.

Collect DynamicsPerf snapshots on a daily basis if you are finding problems.

In production

There are two important steps for understanding the performance of your production system: constantly monitoring its performance and ensuring that you have a solid maintenance strategy in place.

Monitoring

We recommend that you use the following monitoring tools in the production environment:

- Performance Monitor
- The System Center Operations Manager Monitoring Pack for Microsoft Dynamics AX 2012

No matter whether you are working before or after go-live, performance tuning is an iterative process. For example, if you identify a slow process at any point, you can go back, and then trace and optimize by using either the Visual Studio Profiling Tools or Trace Parser.

Regular maintenance activities

Regular maintenance activities that you should consider performing in your production environment include:

- **Index defragmentation** – You can defragment indexes from either SQL Server Management Studio or IDMF.
- **Updating SQL Server statistics from SQL Server Management Studio** – Updating statistics manually, in addition to running automatic statistics updates, might become more important as the size of your database increases, because automatic statistics updates are less likely to be completed on large data sets.
- **Database size reduction** – You can use IDMF to maintain a small production database size and make your database operations more efficient. For example, you can purge or archive data that is not needed in your production system.

Resources

We recommend that you bookmark and refer to the following resources:

- **Microsoft Dynamics AX Performance Team Blog** – The Microsoft Dynamics AX Performance Team Blog provides helpful tips and tricks about performance tuning, links to the latest benchmark reports, and Trace Parser versions that can be used for Microsoft Dynamics AX 2009 and AX 2012.
- **Lifecycle Services** – Lifecycle Services hosts services and tools, including the System diagnostic service.
- **InformationSource** – InformationSource hosts services and tools.

Conclusion

Good performance requires good and continuous monitoring. If you find problematic processes early, you can step in and optimize them with little impact on your end users. Keep in mind that performance tuning is an iterative process. Often, you need more than one round of tuning to improve performance as much as possible.

Outcomes of performance analysis

We use the performance data from the Performance Monitor counters and dynamic management views to produce direct optimizations for SQL Server and other servers.

This step is an iterative process that requires us to go through multiple cycles of optimization for top issues. After the top issues in one cycle are fixed, we move to the next set of issues, and so on, until most issues are fixed.

Some issues might require more analysis before fixes can be defined. Examples of this type of issue include the following:

- Code with too many fetches or too many executions of a particular SQL statement require an application fix.
- We must understand how a report is used and how best to schedule it.

By using the event trace logs, we will do the following:

- Walk through each business process on a code level.
- Determine who uses each business process and how frequently, and then determine potential optimization points.
- Analyze the following:
  - Communication between the client and Application Object Server (AOS) in terms of the number of remote procedure calls
  - The speed of client-side code execution compared to server-side code execution
  - Communication between AOS and the database server for set-based operations
  - The locking model, potential table scans, and index scans

Outcomes might include potential optimizations for business logic, display methods, and caching.

We request that you repeat the initial traces after each set of fixes, to determine whether any additional optimization is required to improve performance.

In certain cases, you will need to take traces in the production environments to understand the effect of the load on a given business process.

Appendix: Data collection tools

Performance Counters and monitoring

A set of Performance Counter objects needs to be collected on each component of the architecture: AOS, the database server, and Microsoft Internet Information Services (IIS).

Recommended collection interval: 30 seconds

File format: CSV or binary file

File name: We recommend that you include the date in your file names to prevent files from being overwritten.

Recommend objects to monitor

Performance Counter objects		Database server	AOS	IIS	Terminal Services
Memory	Available Mbytes	Yes	Yes	Yes	Yes
Paging File	%Usage	Yes	Yes	Yes	Yes
Process (sqlservr, Ax32Serv, w3wp)	%Processor Time (All instances), %User Time (All instances), %Privileged Time (All instances)	Yes	Yes	Yes	Yes
Processor	%Processor Time (All instances), %User Time (All instances), %Privileged Time (All instances)	Yes	Yes	Yes	Yes
Network	(All instances)	Yes	Yes	Yes	Yes
PhysicalDisk	Avg. Disk sec/Read	Yes	No	Yes	Yes
PhysicalDisk	Avg. Disk sec/Write	Yes	No	Yes	Yes
SQLServer:Access Methods	Forwarded Records/sec	Yes	No	No	No
SQLServer:Access Methods	FreeSpace Scans/sec	Yes	No	No	No
SQLServer:Access Methods	Full Scans/sec	Yes	No	No	No
SQLServer:Access Methods	Workfiles Created/sec	Yes	No	No	No
SQLServer:Access Methods	Worktables Created/sec	Yes	No	No	No
SQL Server:Buffer Manager	Buffer Cache hit ratio	Yes	No	No	No
SQL Server:Buffer Manager	Free list stalls/sec	Yes	No	No	No
SQL Server:Buffer Manager	Lazy Writes/sec	Yes	No	No	No
SQL Server:Buffer Manager	Page Life Expectancy	Yes	No	No	No
SQLServer:Buffer Manager	Page lookups/sec	Yes	No	No	No
SQL Server:Locks	Lock Requests/sec	Yes	No	No	No
SQLServer:SQL Statistics	SQL Compilations/sec	Yes	No	No	No
SQLServer:SQL Statistics	SQL Re-Compilations/sec	Yes	No	No	No
SQLServer:SQL Statistics	Batch Requests/sec	Yes	No	No	No

Performance Analyzer (DynamicsPerf)

The DynamicsPerf tool can be downloaded at http://Dynamicsperf.codeplex.com. The download package includes documentation about how to install and use the tool.

1. After you install the tool, but before you begin testing, perform the following tasks:
   - Collect AOS table data information.
   - Capture Microsoft Dynamics AX performance data.
   - Capture Application Object Tree (AOT) metadata.
   - Collect locking/blocking and other information.
   - Ensure that the blocked process threshold for SQL Server is set to at least **2**.
   - Capture a SQL Trace (Server Side Profiler Trace). The profiler job can run every 24 hours, and then the trace is always on for the duration of the test.

   Note: There is an error in the documentation. The @TRACE_DETAILED parameter is set to **Y**. If you execute the profiler trace in all cases, set @TRACE_DETAILED to **N** unless you are asked to do otherwise.
2. Enable the statement trace log for every user of Microsoft Dynamics AX.

   By default, the interval is set to 2 seconds (**2000**). Set the interval to 1 second (**1000**) instead.
3. Configure data collection for single-user tests and load tests.

   Execute the following statement before each load test to free the SQL cache:

   ```
   DBCC freeproccache()
   ```
4. Collect data after every benchmark, or after every user transaction if you are directed to do this.

Trace Parser

There are two versions of Trace Parser: one for AX 2012 and one for AX 2009. This toolset allows for in-depth tracing of business process flows across client-side, server-side, database, and inter-tier communication.

Using Trace Parser for AX 2012

The procedures in this section apply only to the AX 2012 version.

Preparation

The following instructions are intended for short-term tracing only.

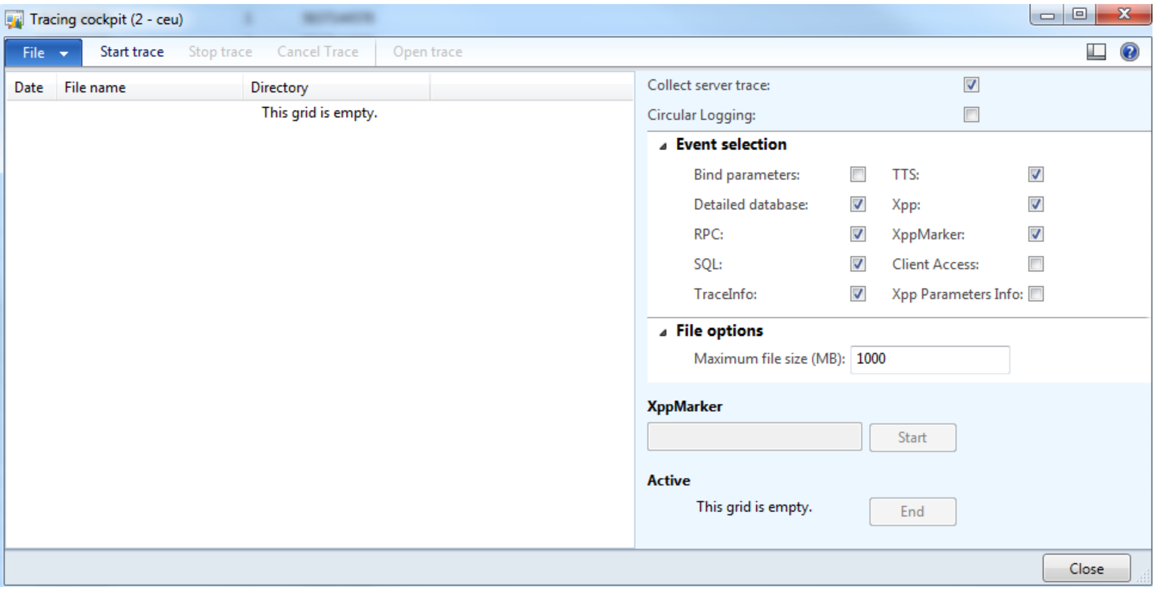
1. Ensure that the AOS service is running with an Administrator account.
2. On the **Start** menu, right-click **Microsoft Dynamics AX 2012**, and then click **Run as Administrator**.
3. Open a Development Workspace.
4. Ensure that drive C on the AOS server has at least 10 GB of free space.

   If the server does not have 10 GB free, you will need to locate the log directory in the registry and change the location to a space with 10 GB free. If there is not enough free space, the trace will not run.

Record traces

For each critical process in which you have experienced performance issues, complete the following steps.

1. Execute the process once, and then prepare to execute it a second time.
2. Open the Tracing Cockpit. (Click **Tools** > **Tracing Cockpit**.)
3. Click **Start trace**.
4. Execute the process again.
5. After the process is completed, click **Stop trace**.



Date	File name	Directory
		This grid is empty.

Collect server trace: ☑

Circular Logging: □

Event selection

Bind parameters:	□	TTS:	☑
Detailed database:	☑	Xpp:	☑
RPC:	☑	XppMarker:	☑
SQL:	☑	Client Access:	□
TraceInfo:	☑	Xpp Parameters Info:	□

File options

Maximum file size (MB): 1000

XppMarker

Active

This grid is empty.

Send the trace files

When you transfer the trace files to Microsoft, also upload the application version that the traces are taken from.

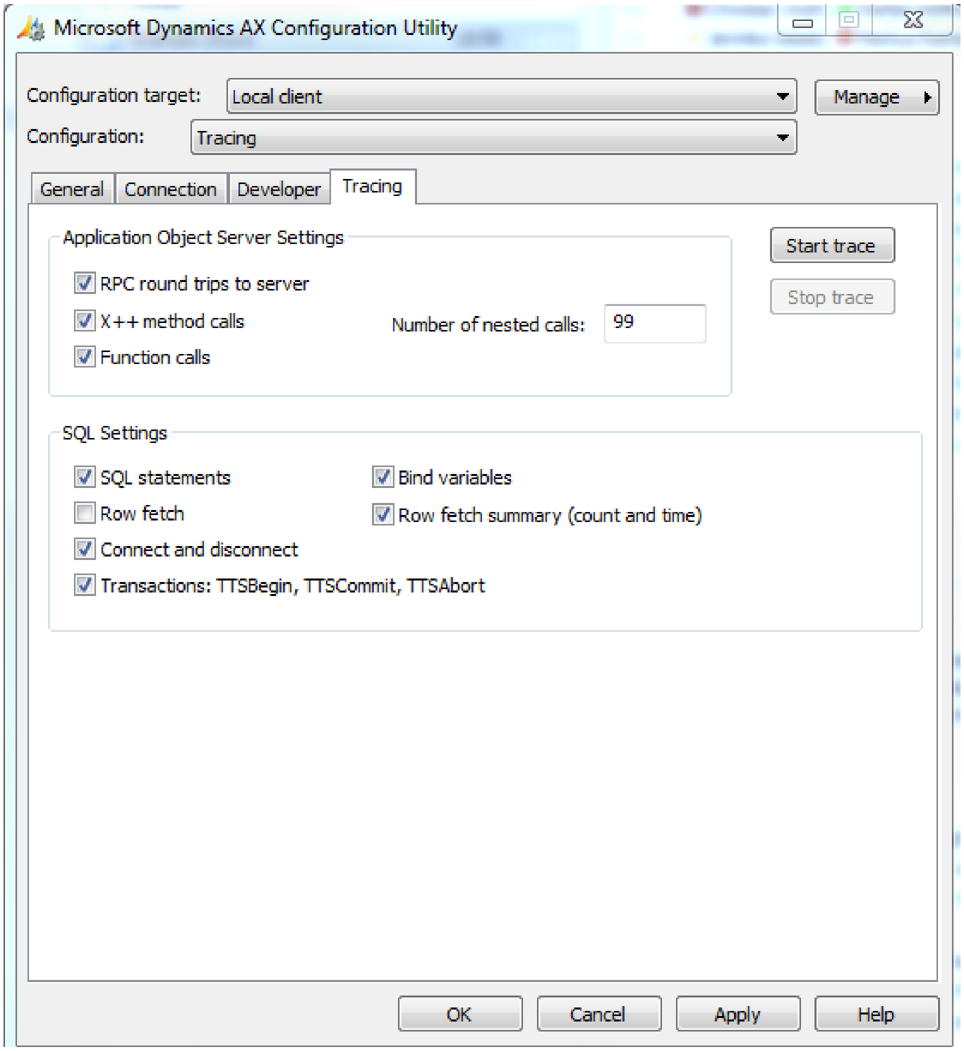
Using Trace Parser for AX 2009

The procedures in this section apply only to the AX 2009 version.

Preparation

The following instructions are intended for short-term tracing only.

1. On one client that can be used for remote access, set the following options in the Client Configuration Utility.



Microsoft Dynamics AX Configuration Utility

Configuration target: Local client

Configuration: Tracing

Application Object Server Settings

☑ RPC round trips to server

☑ X++ method calls

Number of nested calls: 99

☑ Function calls

SQL Settings

☑ SQL statements

☑ Bind variables

□ Row fetch

☑ Row fetch summary (count and time)

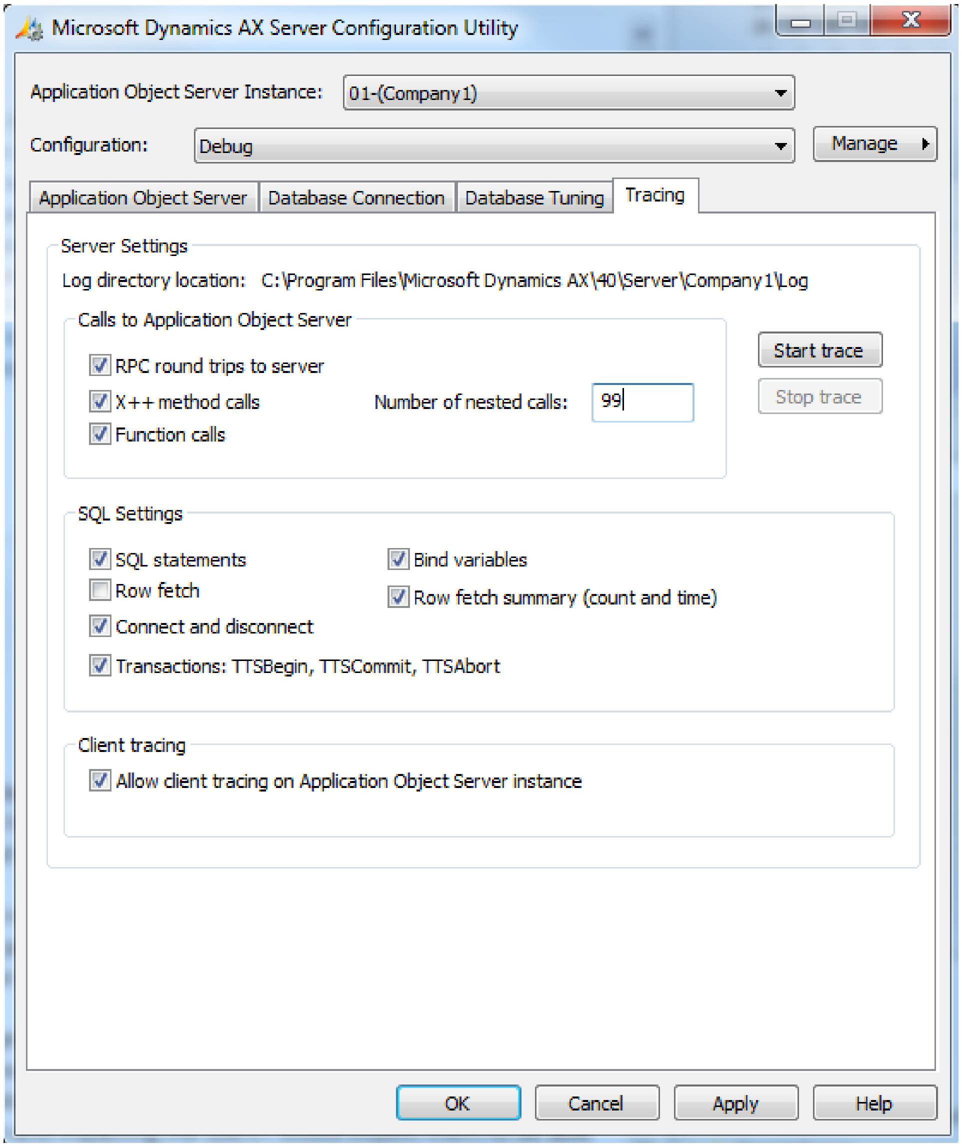
☑ Connect and disconnect

☑ Transactions: TTSBegin, TTSCommit, TTSAbort

2. Ensure that drive C on the AOS server has at least 10 GB of free space.

   If the server does not have 10 GB free, you will need to locate the log directory in the registry and change the location to a space with 10 GB free. If there is not enough free space, the trace will not run.

3. In the Server Configuration Utility, set the following options.



Microsoft Dynamics AX Server Configuration Utility

Application Object Server Instance: 01-(Company1)

Configuration: Debug

Server Settings

Log directory location: C:\Program Files\Microsoft Dynamics AX\40\Server\Company1\Log

Calls to Application Object Server

☑ RPC round trips to server

☑ X++ method calls Number of nested calls: 99

☑ Function calls

SQL Settings

☑ SQL statements

□ Row fetch

☑ Connect and disconnect

☑ Transactions: TTSBegin, TTSCommit, TTSAbort

☑ Bind variables

☑ Row fetch summary (count and time)

Client tracing

☑ Allow client tracing on Application Object Server instance

4. Restart AOS.

Record traces

For each critical process in which you have experienced performance issues, complete the following steps.

1. Execute the process once, and then prepare to execute it a second time.
2. Start the trace, and rerun the process.
3. When the process is completed, stop the trace.

   If you are unsure whether the trace started, you can double-check by opening a Command Prompt window and entering the following command:

   ```
   logman query - ets
   ```

Send the trace files

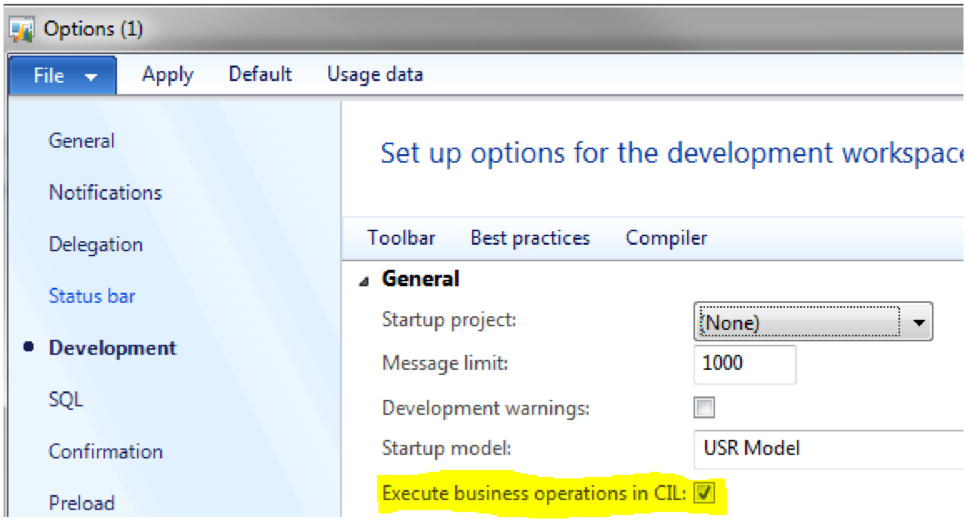
When you transfer the trace files to Microsoft, also upload the application version that the traces are taken from.

Using the Visual Studio Profiling Tools to trace CIL (AX 2012)

Trace Parser cannot effectively trace batch processes, services, and code that traverses CIL by using the **RunAs()** command. Instead, you must use the Visual Studio Profiling Tools. First, you should disable code that is running in CIL.

Disable code running in CIL

In the Microsoft Dynamics AX client, on the **File** menu, click **Options**. On the **Development** tab, clear the **Execute business operations in CIL** check box.



Startup project: (None)

Message limit: 1000

Development warnings: ☐

Startup model: USR Model

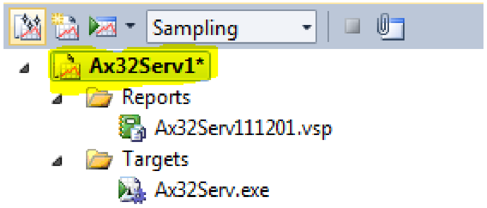
Execute business operations in CIL: ☑

Run the Visual Studio Profiling Tools

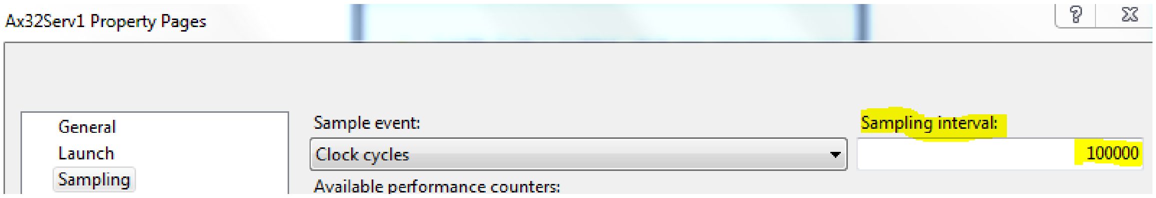
If you run a service that is called outside AX 2012, run a service in the AxClient service group from inside Microsoft Dynamics AX, or run a service in a batch job, this workaround will not help you get meaningful trace data. You must use the Visual Studio Profiling Tools instead.

1. Start Microsoft Visual Studio 2010 Ultimate.
2. Navigate to **Analyze** > **Launch Performance Wizard**.
3. In the wizard, complete the following steps:
    1. Select **CPU Sampling**, and then click **Next**.
    2. Click **An executable (.EXE file)**, and then click **Next**.
    3. Browse to the location of AX32Serv.exe, and then click **Next**.
    4. Clear the **Launch profiling after the wizard finishes** check box, and then click **Finish**.

4. In the **Performance Explorer** pane, right-click your new project, and then click **Properties**.



5. Click **Sampling**, and then remove two zeros (00) from the sampling interval.

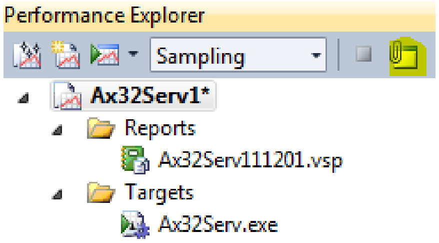


Sample event: Clock cycles

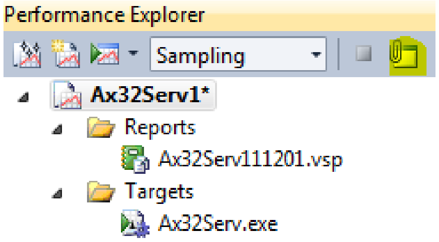
Sampling interval: 100000

Available performance counters:

6. In Microsoft Windows, prepare the process that you want to trace, and execute it once to ensure that it is in a warm state.
7. Prepare the process to be executed again.
8. In Visual Studio, click the **Attach** button.



9. Select the AOS process, select the **Show processes from all users** and **Show processes in all sessions** check boxes, and then click **Attach**.
10. Execute the process.
11. Click the **Attach** button, and then click **Detach**.



12. Send the report (.vsp) file and the .psess file to your contact at Microsoft.

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

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. Microsoft makes no warranties, express or implied, in this document.

Microsoft