# COM+ Application Guidelines for Visual Basic Development

Edward A. Jezierski
Microsoft Corporation

May 2000

**Summary:** Presents guidelines for designing, building, and deploying COM+ applications, with an emphasis on Visual Basic. (80 printed pages)

**Contents**

## Introduction

This document presents guidelines that will help you succeed designing, building, and deploying great Microsoft® Windows® DNA and COM+ 1.0 applications using Microsoft Visual Basic® 6.0 Service Pack 3. References to Microsoft SQL Server™ are for SQL Server 7.0 Service Pack 1 unless otherwise noted.

If you are designing, building, or deploying COM+ applications, or thinking about doing it, then this document is for you. It provides code styles, general design patterns, code snippets, and workflow procedures that will help you become successful with these technologies.

This paper includes many scenarios and configurations developers have used in distributed projects through the lifetime of Microsoft Transaction Services (MTS). Even though COM+ itself is essentially new, many of the principles and experiences in making Windows DNA applications still hold.

This white paper is based on a previous paper, MTS Application Guidelines. If you have used the MTS paper, you'll see that good MTS practices make good COM+ practices. In addition, there are some references to MTS documents, rather than COM+. These references were screened for COM+ compliance and are included for your convenience.

There are many topics discussed here. You will find this document more useful if you are already familiar with MTS/COM+ and Visual Basic programming. Keep in mind that guidelines are provided for related technologies that you might not use (such as COM+ Events, or ASP)—feel free to skip the sections that do not apply to you.

COM+ drastically expands the scenarios a distributed application can leverage by providing plumbing services that are easy to use. Queued Components, Loosely Coupled Events, Object Constructors, and other new technologies bring interesting questions. How should one use this feature? When should one use this feature? What's the smartest way to leverage this feature? Will the application have to be drastically changed to leverage them? These questions are addressed where appropriate.

### What is COM+ 1.0?

COM+ 1.0 means different things to different people. You have heard—and will continue to hear—diverse definitions for COM+.

COM+ is the current step in the evolution of the software distribution services that Microsoft provides. COM+ is an evolution of COM and MTS. COM+ is a set of services that you can leverage to make more scalable, flexible, fast, and available applications. COM+ is about spinning amber balls and all the cool stuff you can do with them (overheard at a Redmond foosball table).

However, what does it mean to Windows DNA developers? You can picture COM+ 1.0 as:

- The environment where your objects live, with rules of interaction for these objects and other resources, such as concurrency issues, transaction flow, security boundaries, and more.
- A set of services that your objects can use to behave in novel ways, making it easy to develop functionality that was hard or tedious to do, and opening the door to more complex application architectures without forcing you to write plumbing code.

Both items above imply a programming model or style that will let your application leverage the environment and services COM+ provides. Therefore, in a sense, COM+ is also a programming model.

If you want a fast summary of the main benefits COM+ gives you, check out this Knowledge Base article.

INFO: What's New with COM+ 1.0 (Q253669) at
http://support.microsoft.com/support/kb/articles/q253/6/69.asp

## Migration to Windows 2000 and COM+

You have the Windows 2000 CD and now you're wondering how to start using COM+. You probably want to maintain interoperability with some developers' computers, or maybe you want to upgrade a server and wonder how to do it.

This section is targeted at helping you during the transition to Windows 2000 as a server and development environment. COM+ brings significant improvements over the previous technologies, so migration is worth the effort. Read this section through and work out a migration plan that will work for you and your team.

## Consider Installing Windows 2000 from Scratch

When installing Windows 2000, you can choose to either install the system from scratch or to upgrade the current configuration. In the latter case, the Windows 2000 installer will actually migrate the preexisting configuration to the new installation, including transforming MTS Packages into COM+ Applications. Installing Windows 2000 on a clean system and recreating the packages manually is recommended. This will prevent further propagation of preexisting configuration glitches.

### Why

Our experience with upgrades suggests that the time saved (by not reconfiguring the server) is not worth the guarantee that you are not carrying any spurious state (broken or dirty object registrations, clogged directories, and so on) into the new environment.

There are exceptions, of course. Depending on the role of the machine, you may decide that you prefer to preserve state through the upgrade—if you are upgrading a primary domain controller for example. However, application servers usually don't maintain much local state and therefore are good targets for "fresh installs."

### More Information

When planning an upgrade, you should always check the *Windows 2000 Planning and Design Guide* in the Windows 2000 Server Resource Kit. The deployment guide lists the supported upgrade paths for Windows 2000 Professional, Windows 2000 Server, and Windows 2000 Advanced Server, and describes other factors that may affect your decision to upgrade rather than to perform a clean install. (Keep in mind that the term "upgrade" means to replace the operating system using the upgrade features of Windows 2000. A clean install refers to formatting the disk and reinstalling the operating system and all applications.) It then discusses tools and procedures for automating the operating system upgrade process.

### References

Windows 2000 Server Resource Kit (http://mspress.microsoft.com/Prod/books/1394.htm)

Deployment Planning Guide (http://msdn.microsoft.com/isapi/gomscom.asp?Target=/WINDOWS2000/library/resources/reskit/dpg/default.asp)

Step-by-Step Guide to Remote OS Installation (http://msdn.microsoft.com/isapi/gomscom.asp?Target=/windows2000/library/planning/management/remotesteps.asp)

*Upgrading MSMQ 1.0 Enterprise to a Message Queuing Deployment in a Windows 2000 Domain* (This document can be downloaded at ftp://ftp.microsoft.com/bussys/distapps/msmq/win2000/migration/migration3.doc.)

## Create New Packages and Verify All Properties

If you installed Windows 2000 from scratch, your MTS packages were not migrated to COM+ applications. You have two choices to migrate your MTS packages to COM+ applications:

1. If you have the PAK file for the MTS package, you can install the PAK files for the MTS packages in Windows 2000 to convert the MTS application into a COM+ Application. However converting MTS

applications into COM+ applications may be more error prone and carry over too much information or information that has changed (such as role members). Further, all the DLL file dependencies required by those applications may not be correctly installed on the server.

2. Create, install, and configure completely new COM+ applications based on your existing MTS applications. This is the only way you can ensure that your application is correctly installed and configured in the new environment.

### How To

Before installing Windows 2000, make sure the documentation on your application is up to date, especially regarding the configuration of your components under MTS. Make sure your application includes:

- Components installed under what packages.
- If the packages are Server or Library.
- Authentication levels, roles, and security settings.
- Application identities.
- Distributed Transaction Coordinator (DTC) log size, timeout.
- Transactional attributes.

Record configuration information that is not MTS-specific such as:

- DCOM Authentication and security settings.
- The privileges for the identities of your MTS Packages and services.
- SQL Server connection configuration settings.
- Any services you may have installed (such as Microsoft Message Queue Server (MSMQ) listeners).
- Registry keys.
- Network settings.
- Manually added Hostname resolution entries (in LMHOSTS or HOSTS).

After installing Windows 2000, follow these steps:

1. Run the installers for your components to ensure that all the dependencies required by these components are in place.
2. Create the required users, if any, for the Application identities (remember, MTS Packages are called Applications in COM+) and grant these users the required privileges.
3. Create the COM+ Applications you need. You must now make sure that the application, package, and interface properties match the settings you had in your MTS packages.
4. Add the roles to your COM+ application.
5. Add the components to the COM+ Applications, configuring them as appropriate.
6. Set the right roles to your objects and interfaces (and methods).

### References

See the links in the Consider Installing Windows 2000 from Scratch section for more information on Windows 2000 deployment. Check this Knowledge Base article for more information.

HOWTO: Convert MTS Packages to COM+ Applications (Q252400) at
http://msdn.microsoft.com/isapi/gosupport.asp?Target=/support/kb/articles/q252/4/00.asp

## Keep Your Build Environment in Sync with Deployment Target

When you are compiling objects intended to run under Windows 2000, compile them and package them for deployment on a Windows 2000 computer.

### Why

Many system DLLs, Type Libraries, and a number of other resources your application depends on may have changed from Windows NT® to Windows 2000. As a rule, your system should not be at risk if you install a component compiled in Windows NT 4.0 in Windows 2000, but why risk it? By keeping the build environment consistent with the test and deployment environments, you may save yourself a couple problems hard to track down or solve.

### How To

If you will be deploying your components in Windows 2000, make your build machine a Windows 2000 computer.

## Clean Interface Forwarding

You might want to leverage the opportunity for a fresh start (Hey, it's not a new millennium but it's a new OS after all) and clean up your object interfaces. If your build practices have been less than rigorous, you may have ended with an issue called *interface forwarding*. Interface forwarding happens when you tell VB you wanted to maintain Binary Compatibility after adding a method or a parameter to a component. You should try to avoid interface forwarding.

Keep in mind that if your clients use early binding, then you will need to rebuild them with a reference to the new DLL or Type Library. Depending on your distribution strategy, the deployment costs may be hard to justify in terms of "clean interfaces."

### How To

The simplest way to remove interface forwarding is:

1. Store a copy of your current DLL somewhere safe (or rename it).
2. Set 'no compatibility' in Project..Properties..Component and recompile the DL.
3. Set to 'Binary Compatibility' and point to your saved DLL; recompile the DLL again.

### Why

MTS did not work well, if at all, with components that have interface forwarding, and this feature is seldom tested, if at all. It also adds unneeded complexity to the registration and some uninstallers get dizzy when hitting this feature.

### More Information

When you set Binary Compatibility on your project and later decide to add a method to an object, you get the choice of breaking or maintaining compatibility. If you choose to maintain it (try not to), VB uses a special feature of COM registration called *interface forwarding*. This means that your old interface IDs will be registered, but with forward information to the new interfaces.

Adding a method to an interface, or an optional parameter to a function, absolutely does require a new IID for your interface. It all boils down to something simple—your IID identifies an interface (a thing) and what it exposes, not an arrangement on how a client uses it. Therefore, the IID must change if the combination of all you could do on the interface changes. You can compare it to adding a key to a keyboard. You can assume previous users won't press the key. You can even decide not to reprint the manuals. However, the SKU number for the keyboard must change.

### References

Maintaining Binary Compatibility (http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbcontypelibraryversions.htm)

PRB: This Interface Has Been Forwarded Error in Visual Basic with MTS (Q222634) at http://msdn.microsoft.com/isapi/gosupport.asp?Target=/support/kb/articles/Q222/6/34.ASP

Building, Versioning, and Maintaining Visual Basic Components (http://msdn.microsoft.com/library/techart/msdn_bldvbcom.htm)

## Use the Windows 2000 Type Libraries and ProgIDs for COM+

Your components probably have references to the Transaction Server objects, Shared Property Manager, and so forth. The Type Library names have changed in Windows 2000, so double check to see that you have the right references and that you can create the objects you need.

When taking a DLL or EXE over to Windows 2000, you will see it can still create the **Administration** and **Shared Property** objects—whether using the **New** keyword or using **CreateObject** and **ProgIDs**.

### More Information

The following table summarizes some of the most important changes. If your project references MTS libraries, these references will be updated when opening it on a Windows 2000 computer, as summarized in this table.

| MTS | COM+ |
|---|---|
| Microsoft Transaction Server Type Library<br>(MTXAS.DLL) | COM+ Services Type Library<br>(COMSVCS.DLL) |
| Shared Property Manager Type Library<br>(MTXSPM.DLL) | COM+ Services Type Library<br>(COMSVCS.DLL) |
| MTS 2.0 Admin Type Library<br>(MTXADMIN.DLL) | MTS 2.0 Admin Type Library<br>(MTSADMIN.TLB) |

The MTS Admin DLL reference was converted to a MTS TLB reference to allow for backwards compatibility. The COM+ Admin reference is a separate item. If you are planning to use the COM+ **Admin** objects, however, you will want to research the documentation for them since there have been changes in the collections, properties and methods they support.

### How To

As mentioned above, taking a working DLL and putting it in Windows 2000 will not break it, at least not when creating MTS objects. You may see some issues when you are compiling in Windows 2000 for the first time due to the new Type Library names. The easiest solution is to remove the Type Library qualifiers from the variable declarations. Use following checklist to help you make the changes.

If you have variables declared as:

`MTxSpm.SharedProperty[…],`
*c*hange them to:

`COMSVCSLib.SharedProperty[…]`
or remove the TypeLib qualifier alltogether:

`SharedProperty[…]`
If you have variables declared as:

```
MTxAS.ObjectContext (or MtxAS.[…]),
```
change them to:

```
ObjectContext
```
The **ProgIDs MTSAdmin.Catalog, MTSAdmin.Catalog.1,** and **MTxSpm.SharedProperty[…].1** will work properly under Windows 2000.

## Become Familiar with the New Configuration Options and Terms

Become familiar with all the new configuration options COM+ makes available for your objects. The article listed below gives you a bird's view of the new COM+ features. However, you can get a good idea of what new options are available by browsing the COM+ Component Services Manager trees and property sheets for Computer, Application, Objects, Interfaces, Methods, and so on.

It is much more efficient to read about these settings before fiddling with them while learning. The Knowledge Base article below is a good place to start. You can also check out MSDN and take a couple of minutes to read the main documentation. The fun will start later when that spark goes off in your mind when you get it. The following Knowledge Base article provides useful links and may help you decide what is more relevant to you.

### References

INFO: What's New with COM+ 1.0 (Q253669) at
http://msdn.microsoft.com/isapi/gosupport.asp?Target=/support/kb/articles/q253/6/69.asp

## Build an Efficient Management Console

Windows 2000 is incredibly efficient for management tasks, but some of the tools from Windows NT 4.0 have been moved into better locations. Try to locate the utilities you use, and group them in your own MMC console to have everything handy.

### More Information

The following lists out some common tools MTS/COM+ developers and administrators use, with their new location.

- Transaction Server Explorer: This has moved to the Component Services (or Component Services Manager) snap-in. You can access it directly by clicking on the **Start** button, click **Programs**, **Administrative Tools**, and **Component Services**.
- MSMQ Explorer: This has gone into the Computer Management snap-in, under [Services And Applications]...[Message Queuing].
- ODBC: This was moved into Administrative Tools...Data Sources [ODBC].
- Local Security & Auditing: These are found in Administrative Tools/Local Security Policy, and the Local Users & Groups snap-in.
- Event Viewer: This is now a snap-in.
- Services: This is also a snap-in now.

### How To

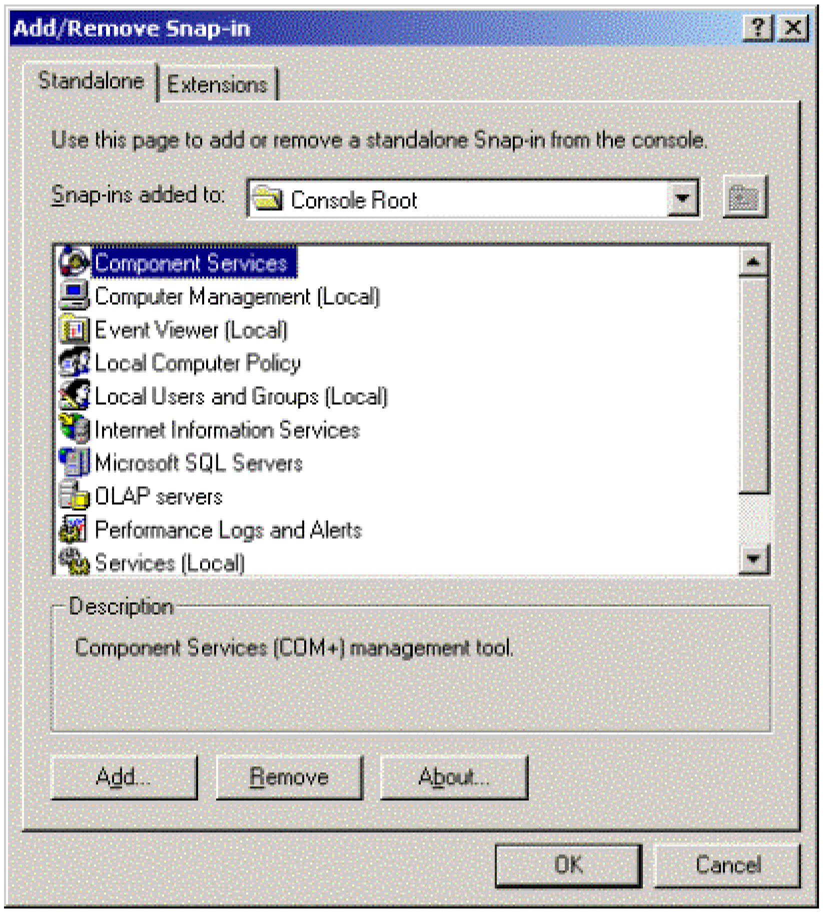
You can save your own console by following these steps:

- Click on the **Start** button, then **Run**, and type *MMC.*
- In the **Console** Menu, go to **Add/Remove Snap-in**.
- In the Add/Remove Snap-in dialog box (Figure 1 below), click the **Add** button and select the snap-ins you want.

These may come in handy:

  - Component Services
  - Event Viewer
  - SQL Server Enterprise Manager
  - Internet Information Services
  - Computer Management
  - Local Users and Groups
  - Performance Logs and Alerts
  - Services
  - System Information

- You can also add "Links to Web Addresses" to the Microsoft Knowledge Base or MSDN search, intranet sites, or add links to your most used file shares. If you think your console has too many snap-ins, try organizing them in folders.
- Go to Console and select Options. You may select User Mode to prevent further changes. Choose an appropriate title and icon for it and whether to save the changes or not.
- Save your console to a MSC file where you can easily get to it.
- Drag the MSC file to the Start button to create a fast shortcut to it. You can even assign a shortcut key sequence to it.



Use this page to add or remove a standalone Snap-in from the console.

Snap-ins added to: Console Root

Component Services
Computer Management (Local)
Event Viewer (Local)
Local Computer Policy
Local Users and Groups (Local)
Internet Information Services
Microsoft SQL Servers
OLAP servers
Performance Logs and Alerts
Services (Local)

Description

Component Services (COM+) management tool.

**Figure 1. Handy MMC Snap-ins**

You might also want to enable Terminal Services for remote administration. This will allow you to use another computer as a terminal into your server, eliminating the limitations you find in using standard administration tools remotely.

Check the links below for more information on how to do this.

#### References

Windows 2000 Management Services (http://msdn.microsoft.com/isapi/gomscom.asp?Target=/windows2000/library/technologies/management/default.asp)

Using Terminal Services for Remote Administration of the Windows 2000 Server Family (http://msdn.microsoft.com/isapi/gomscom.asp?Target=/windows2000/library/operations/terminal/tsremote.asp)

### Review the Windows 2000 Server Application Guidelines

Microsoft has published a series of guidelines regarding the behavior and resource interaction that applications have to attain to receive Designed for Windows 2000 status. Even if you are not interested in achieving the official certification, it is good to review the practices outlined to see the official Microsoft definition of a "well-behaved" Windows 2000 application.

#### References

Application Specification for Windows 2000 for Desktop Applications (http://msdn.microsoft.com/library/specs/w2kcli.htm)

Application Specification for Windows 2000 Server (http://msdn.microsoft.com/library/specs/w2kserve.htm)

## Selecting the Technologies

Selecting the technologies sounds broad, but it is a critical part of the application lifetime. You know what it has to do, and you have a specification of how, but the question of what will let your application run in the fastest, most reliable, and maintainable way remains. What technologies can you count on to make an investment in education and resources? Which language, what middleware, and what client to choose are other questions. All of these questions are important.

### Visual Basic or Visual C++?

Don't assume that redeveloping Visual Basic components in Visual C++® will increase throughput or give any performance or scalability increase. Take, for example, an object function that takes in some parameters, executes a stored procedure, and then retrieves data from a database. How much of the code in the whole activity is the VB or VC++ code you wrote versus all the infrastructure code required to create objects, establish database connections, parse result sets, connect through the network, verify security, and so forth? A very small proportion of a typical OLTP backend is actually implemented in the language of choice.

It is worth mentioning that in many cases, components submitted to third parties to benchmark Microsoft server products and application architectures are written in Visual Basic, including record-breaking cases.

However, using Visual C++ may have benefits including opening the door to other threading models or allowing designs to get out of the standard behavior for COM objects in order to optimize specific actions. However, much more technical knowledge is required to do it right. It is easier to do a bad VC++ component than a bad VB component.

A good example of this situation is object pooling. COM+ provides the ability to pool objects. However, only components with certain threading models can be pooled—threading models not supported by Visual Basic 6. If you choose to implement your objects in Visual C++ to take advantage of pooling, you will have to manually enlist and de-enlist database connections, forcing you to use OLE DB directly and not ActiveX® Data Objects (ADO). As you can see, the translation stops being linear and becomes more complex the more you take advantage of functionality.

If you have developers with the skill set to make Visual C++ components, assign them to build specific portions of your application—those "sweet spot" functionality areas where a performance improvement (obtained through the more involved use of the system facilities) has a huge impact.

## Use Queued Components (or MSMQ) for Asynchronous Communications

Consider Queued Components for everything you would normally do with a callback or other notification facility. MSMQ is fast, reliable, scalable, and easy to use. Queued Components builds on this foundation and hides all the MSMQ mechanics to you. As a developer, sending and receiving asynchronous operations is no different from making and receiving regular COM calls.

### How To

Queued Components isolates you from most development decisions regarding MSMQ. All you have to do is install MSMQ, design your components to support write-only interfaces, create them with the **GetObject** method and a queue moniker, and make an exception class. You can then use your components in a traditional fashion. See the Using Queued Components section for more information.

Using MSMQ entails installing the service and using the MSMQ objects to compose, send, and receive messages. The MSMQ object model allows for complete control over MSMQ options regarding queue naming, certificate usage, authentication options, and more. There is a lot of documentation and samples available from the links below.

### More Information

Whenever you have to do an asynchronous task, you may be tempted to implement a complex callback-making, state-managing, thread-locking scheme. Consider QC/MSMQ instead.

In the vast majority of cases, using Queued Components or MSMQ makes far more sense than days and days of custom coding and lost project time over synchronization issues that may never be solved. Defaulting to MSMQ isolates the project from the subtle differences in thread and object lock management that occurs with service packs. Unfortunately, Asynchronous COM calls (available with COM+) are not an option for Visual Basic 6.0 SP3 developers. Using Queued Components is the best alternative in most cases.

When should you use Queued Components and when should you use MSMQ? Use Queued Components when:

- You don't care about the contents of the message—(Queued Components will build and parse the message for you)—all you want is something to go through.
- You don't have complex requirements for a Queue listener.
- Transactional messages work fine for you.
- Queued Component's retry/dead-letter failure handling scheme (and its variations) suits your application.
- You don't want to implement yet another API in your application.

Use MSMQ when:

- You need control over the contents of the messages, integration purposes for example.
- You need to manage your own queues.
- You need full control over particular aspects of MSMQ, such as certificates.
- You have the skill set to develop queue listener services, usually in C++.
- You want to handle "bad" messages and handling errors in unique ways.

- You can manage the maintenance cost increase due to having more APIs in your projects.

Given these short checklists, and knowing Queued Component's flexibility in both handling many MSMQ options and exception handling, somebody would have to provide a very compelling argument for using MSMQ and not QC.

#### References

Queued Components Reference (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_queuedcomponents_2vhv.htm)

All Queued Component Knowledge Base Articles (http://search.support.microsoft.com/kb/psssearch.asp?)

MSDN MSMQ Reference (http://msdn.microsoft.com/library/psdk/msmq/msmq_overview_4ilh.htm)

Programming Best Practices with Microsoft Message Queuing Services (http://msdn.microsoft.com/library/backgrnd/html/msmqbest.htm)

### Use Components Intended for Server-side Development

Some object models your application could use are designed to run on the server, some are not. The main conditions that an object model must have to work on the server are:

- The ability to work without displaying interface or interacting with a user. This includes doing its work, reporting errors, and being able to start without having a user logged on at the machine.
- Being thread-safe.
- Not leaking memory, handles, or any other resource.

On occasion, certain functionality is required from objects models that is almost impossible to acquire any other way.

#### More Information

Some object models were not designed to be used on the server and will bring instability to your application. It is best to find alternative ways to obtain the service, or to regulate access to it and isolate your application from failures it may cause.

#### How To

Read the documentation of the component you are about to use to see if it has any reference to thread-safety or if it uses any external service that is probably not intended for server side use. You can also check on newsgroups or call directly to the vendor to see if they support your usage of the component.

If you absolutely must use resources not intended for the server, you may evaluate serializing access to it by specifying a wrapper object to be single-thread or by using Queued Components MSMQ. Try to isolate the resource's failures from your applications as much as possible by calling them from components in Server Applications, and provide the means to gracefully fail and retry operations.

### Use CDO for Windows 2000 When Possible

Microsoft has upgraded the Collaboration Data Objects (CDO) library. On Windows 2000, you now have access to CDO for Windows 2000 (CDOSYS), a messaging object model aimed at server applications. This is the object model of choice, so use this library instead of CDO, which carries more overhead, or CDO-NTS, which has less functionality.

#### How To

Review the documentation below to get familiar with this component. In Visual Basic, you will need to add a reference to Microsoft CDO for Windows 2000 Library (CDOSYS.DLL) to use these objects.

### More Information

CDO for Windows 2000 is a library designed for Windows 2000. This component is not MAPI-based and is aimed at server-side development. It is more scalable and offers better support for MIME body parts and international character sets. It is also better tested than its predecessors. It interacts with the SMTP or NNTP services (or whatever service shares their drop folders). It is also transparently upgraded when installing Microsoft Exchange 2000 so applications continue to work with its superset, CDO for Exchange.

MSDN suggests the following as typical uses for CDO for Windows 2000:

- Create and send messages using the SMTP and NNTP protocols.
- Append disclaimers or other notices to e-mail sent through your server.
- Create ASP applications with messaging capabilities.
- Detect and discard unsolicited bulk mailings.
- Detect and discard inappropriate newsgroup postings.
- Check inbound messages for viruses.
- Forward and filter messages automatically.

CDO also allows an application to interface with Exchange Server and other messaging solutions with a richer exposure of functionality. It has been designed and tested with a limited server orientation in mind. Keep in mind that if you need to interact with Exchange, CDO for Exchange is the object model of choice.

### References

This document provides an overview on the different CDO versions and their relationship.

Collaboration Data Objects Roadmap (http://msdn.microsoft.com/library/techart/cdo_roadmap.htm)

Check out these links for more in-depth information.

Exchange Server Developer Center (http://msdn.microsoft.com/isapi/gomsdn.asp?Target=/exchange)

About CDO for Windows 2000
(http://msdn.microsoft.com/library/psdk/cdosys/_cdosys_about_cdo_for_windows_2000.htm)

INFO: Relationship between 1.x CDO Libraries and CDOSYS.DLL (Q195683) at
http://msdn.microsoft.com/isapi/gosupport.asp?Target=/support/kb/articles/q195/6/83.asp

HOWTO: Send HTML Formatted Mail Over SMTP Using CDONTS (Q189945) at
http://msdn.microsoft.com/isapi/gosupport.asp?Target=/support/kb/articles/q189/9/45.asp

INFO: What is the Difference Between CDO 1.2 and CDONTS? (Q177850) at
http://msdn.microsoft.com/isapi/gosupport.asp?Target=/support/kb/articles/q177/8/50.asp

## Rely on a Database for State Storage

Consider using a relational database such as SQL Server for application state management. Flat files, registry entries, and other storage mechanisms fall short in terms of reliability, performance, and scalability. However, in some cases you might want to rely on other specialized storage mechanisms, such as Lightweight Directory Access Protocol (LDAP), to solve specific problems.

For most needs, a relational database is the best way to go. In fact, there is no easier way to achieve full recovery and failover for a user's session than a common sense approach to handling state management through the facilities of a relational database. Be sure to choose your relational database wisely. Make sure that its drivers or providers can be used from multiple threads, that transactions can span more than one connection, and that your database can easily scale to accommodate your evolving and growing business.

A brief mention of Jet databases. Jet is a database solution not intended for enterprise use. Due to the number of excellent alternatives, such as MSDE/SQL Server 7.0, there is no reason to use Jet either on the server or on the desktop. MSDE and SQL Server 7 are more suitable for small databases than Jet. Also, MSDE is redistributable with your solution and it can be managed through Access 2000.

### References

Getting Started with SQL Server 7.0 (http://msdn.microsoft.com/library/psdk/sql/8_gs_00.htm)

SQL Server: An Overview of Transaction Processing Concepts and the MS DTC (http://msdn.microsoft.com/library/backgrnd/html/msdn_dtcwp.htm)

Technical FAQ: Deploying MSDE Solutions (http://msdn.microsoft.com/isapi/gomsdn.asp?Target=/vstudio/MSDE/deploying.asp)

Active Directory (http://msdn.microsoft.com/library/psdk/adsi/glns2(1)_5kit.htm)

## Architecting the Solution

You know what your application will be built on. You know which languages to use, and the general shape of the services and their interactions. However, there are still some outstanding issues. How much of the infrastructure should you build? Which pieces of implementation will be shared, how will they be distributed, and how will you tackle special communication cases? The answers involve engineering and technical knowledge alike, but there's no need to reinvent the wheel.

Visit the link below for some good design information.

General Design Tips for Using COM+
(http://msdn.microsoft.com/library/psdk/cossdk/pgdistributed_design_81pb.htm)

Here is some good reading on app design and software distribution issues.

Top Windows DNA Performance Mistakes and How to Prevent Them
(http://msdn.microsoft.com/library/techart/windnamistakes.htm)

A Blueprint for Building Web Sites Using the Microsoft Windows DNA Platform
(http://msdn.microsoft.com/library/techart/dnablueprint.htm)

### Let Us Do the Plumbing

Sometimes you wish that the platform/runtime/data access mechanism would just work this way or that way, and it doesn't. On the other hand, maybe it does, but you haven't figured out how to make it behave just the way you think you want it. In those cases, it's sometimes tempting to say "Oh well. I know how to code, I'll do my own (database connection pool, object dispenser, asynchronous message dispenser), and that's it!"

It's tempting to take off with some idea regarding the way the infrastructure should be done, but be careful. Many attempts to home-grow an infrastructure design for state management, cache, or pooling have resulted in a performance or scalability hindrance rather than a benefit.

At that point in time, you may have gone through complex design decisions regarding flexibility, implementation, and scalability of your infrastructure. You probably will have spent precious time implementing and testing it.

Although Visual Basic has evolved as the enterprise language of choice, it is best suited to build business objects and not plumbing or infrastructure objects where complex technical issues, such as thread management, interface marshalling, and concurrent access, have to be dealt with in more involved ways.

#### More Information

This issue applies not only to the way objects are managed, but also to the common pitfalls encountered when one tries to optimize the infrastructure with a tool not designed for it. Some examples are:

- "I can keep objects in memory, in a global variable or collection, so they will be available as required." This common misconception does not take into account that there will be a copy of the global variable per thread, but also that objects can execute on any thread they like. VB6 objects are apartment threaded, so every method call that executes on an instance has to be on the thread that created it. Making these collections defeats MTS thread pooling and makes the project very unstable.
- "If I store properties in my objects across calls, there will be less network traffic." This usually results in stateful objects, which burn more resources such as transaction locks, memory, and network traffic. Unless you're planning to call dozens of methods that require all the object data unchanged, the stateless design will be more efficient. Establishing a network connection is expensive and from then on, sending more or less data makes almost no difference.

The application design should be linear and simple. Store transient shareable state such as connection strings (strings – not objects!) in the SPM, store persistent state such as customer info in the database, and store activity-wide state such as temporal calculation results in the appropriate function-scoped variables.

## Adopt a Stateless Programming Model on the Server

This design issue has been debated thoroughly in many channels and has been approached in different ways. Although there are many reasons to go for stateless designs, stateful design has its niche. However, correct software distribution methodologies suggest that for OLTP server operations (a shared resource/concurrent isolated activity environment) the stateless approach is optimal in most cases. They allow for an interdependent, easily cloned, design of services with transparent failover, easier load balancing, less resource contention, and easier management of transactional data.

In addition, a stateless approach is more linear in design and implementation than a stateful approach. This means that it reduces dependency on time-variant factors or method call sequencing, producing activities that are more isolated and providing a finer granularity for load balancing. It is easier to develop software from a single-user activity standpoint than it would be to deal with all the issues involved in state management.

### More Information

Exceptional reasons to maintain a user/session/instance-specific state in a server object is when all of the below hold to a strong degree:

- The data being kept is expensive or impossible to reacquire.
- It has undergone a transformation that is too costly or impossible to perform on the client machine.
- It has undergone a transformation that is too costly or impossible to redo on the client machine.
- It is too expensive or impossible to transmit the state data to the client.
- The data is not duplicated in an alternate transactional store.

If most of the above holds true, you have a strong candidate for a stateful server-side programming model. This might improve your performance but it will drastically reduce your concurrency and maybe your scalability.

At this point, you must ask yourself if this piece of data belongs in an OLTP-ish TP environment. You may evaluate alternate state management methods (storing to specialized databases) or transport mechanisms. Also, evaluate the benefits of asynchronous operations when dealing with this type of information.

### How To

Make sure you have good state management approach in your server:

- You do not have global variables that cache information in Recordsets, Collections, or Dictionaries. These will cause bottlenecks and failures.
- Your classes have no class-level members (Private or Public variables) that have a lifetime beyond that of one method call. Use parameters on the functions instead of properties (network roundtrips are more expensive and network packets seldom have to be wasted because of larger numbers of parameters). In general, the only Private variables you will have are those initialized from constructor strings.
- Your methods open connections to resources and close them immediately. All ADO objects that have been opened are closed, and all objects created are explicitly released by setting them to nothing.
- You call **ObjectContext.SetComplete/SetAbort** before returning from the method.

### References

Microsoft Systems Journal (MSJ): Q&A ActiveX/COM by Don Box (http://msdn.microsoft.com/library/periodic/period98/activex0398.htm)

Holding State in Objects (http://msdn.microsoft.com/library/psdk/mts20sp1/building_5xmb.htm)

## Avoid Storing Objects in the Shared Property Manager

The Shared Property Manager (SPM) is designed to store small pieces of information intended for single-write, multiple-read scenarios. Objects stored in the SPM are subject to multithreaded access and will cause a serializing bottleneck, and it will force an object to be specifically associated with the MTS thread that created it. This means it will not be able to take advantage of the thread pool MTS uses to achieve high scalability.

### More Information

The SPM can be used to store items such as connections strings, and other packages or applications. That is, specific information useful to have at hand.

Remember that under this category, one does not find transaction-related pieces of information, such as category lists or small lookup tables implemented as an array. Current RDBMS, such as SQL Server 7.0, optimize memory usage by data at a finer granularity, and with current technology, it usually is a better decision to let the RDBMS' heuristics determine what memory needs to be cached based on dynamic criteria.

### How To

If you store objects in the SPM and determine that the information belongs there, you can add methods to the object that let it load itself from a string, which will in turn be stored in the SPM or to somehow have its *state* stored in the SPM, but not its *instance*. Take into account that the SPM allows for sets of atomic (Properties with Names) or array-like attributes (Properties with Positions) and this namespace probably could be accessed directly without adding an extra string-parsing burden.

If you don't have to write information to SPM more than once, or update any more than a very few non-critical elements (such as page hits), then you will probably find SPM very valuable. Before you store anything in SPM, ensure that the item you will be accessing will never require failover to another machine and any data you lose if the process goes down is disposable.

### References

COM+ Shared Property Manager Reference (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_spm_8awi.htm)

Application Design Notes: Sharing State by Using the Shared Property Manager (http://msdn.microsoft.com/library/psdk/mts20sp1/mtxpg08vb_5o55.htm)

## Avoid Passing COM+ Object References Around

Usually, a COM object will call other objects it needs to get its job done. All the status it needs can be gathered after the call itself. In a synchronous server environment, having to receive a callback from a callee, or having to notify your caller about something mid-function is usually a symptom of an overly complicated design.

If you keep to a stateless design, you'll not run into this issue. However, if you feel that an object should notify one of its callers (either on the client or on the server itself) of something other than on function return, you may be tempted to implement a callback. This, besides bringing interesting questions about the atomicity of the operation of the parent function, may lead to complicated synchronization problems.

**Note** You will also have to make sure all your object references are being set to Nothing at the right time.

There are many other important issues to take into account, such as transaction flow, security, and so on. If an object you are calling into must call you to complete its work, since your state won't change in the meantime, why not gather all the data the child object needs beforehand and make one function call? This will result in a cleaner, straightforward design.

### How To

If you want to pass object references around, you don't need to use the **SafeRef** function to get a Safe Reference to your own instance. COM+ will make sure that your object references are safe (they are pointing to the right context wrapper) and **SafeRef** is just maintained for backward compatibility.

#### References

Understanding COM+ Contexts (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_contexts_8dwz.htm)

### Do Not Raise Events from Your COM+ Objects

If you need to notify a client about something, do not use events (Declaring an Event in your class and raising it with **RaiseEvent**). This will make the method not thread-safe under some scenarios, and will add complexity to your security configuration.

#### More Information

The use of events from COM+ components not only introduces these technical problems but also raises a fundamental question regarding the transaction's dependency on a nontransacted and fallible resource—remote communications. How "atomic" is your transaction's operations if they require going out and telling some caller that a certain operation is going on? If you absolutely need client notification schemes, think about making the notification asynchronous or breaking the transaction into two independent phases.

Events also raise problems with DCOM security configuration (call direction is reversed so it may require making security configuration changes on clients, or modifying firewall configurations), and bring performance problems due to the extra roundtrips required to establish the Connection Point callbacks.

As your application evolves, you may find it necessary to partition functionality across several machines. Be careful when using methods that are primarily tested and supported for applications boundaries of no more than one machine. These will not allow you the flexibility to later distribute your application without a high degree of rework.

#### How To

Make sure you do not use **RaiseEvent** in your COM+ objects. If you need to notify many unknown server objects of a certain event, you may evaluate using COM+ Events. For asynchronous communications look at Queued Components, MSMQ, or some other notification scheme.

#### References

COM+ Technical Series: Loosely Coupled Events (http://msdn.microsoft.com/library/techart/compluscouple.htm)

Trading Up To Queued Components (http://msdn.microsoft.com/library/techart/qchanson.htm)

## Implementing the Business Layer Servers in Visual Basic

At this point, you know how your application will be structured. If you have chosen a stateful design, you are aware of it and the potential tradeoffs. The implementation of your design is perhaps the ultimate factor that will determine its success. All these items give suggestions on how to write your code and properly configure your project and classes.

You might want to visit the link below as well. It has some useful information regarding MTS/COM+ component debugging.

INFO: Visual Basic 6.0 Readme Part 12 – MTS Issues (Q170156) at
http://support.microsoft.com/support/kb/articles/Q170/1/56.ASP

### Specify a Naming Convention

When writing the code for your components it is always good to rely on a set of rules regarding the names of prefixes and things. These will make the code more maintainable, easier to troubleshoot, and less prone to errors while coding.

#### More Information

You may want to specify naming conventions for:

- Variables
- Classes
- Functions and Parameters
- Projects
- Database Objects (tables, stored procedures, and so on)
- Shared Properties

#### How To

There is a lot of information on this topic. However, keep these guidelines in mind:

- Don't over abbreviate (for example, **fopen** vs. **OpenFile**).
- Stick to a Noun-Verb or Verb-Noun scheme (**FileOpen** vs. **OpenFile**).
- Use good capitalization (**DeleteExpiredSubscribers** vs. **deleteexpiredSubscribers**).
- Use underscores consistently (**sp_add_user** vs. **sp_removeuser**).
- Use Plural-Singular consistently (**SELECT * FROM *Author*s INNER JOIN *Book***).

#### References

INFO: Microsoft Consulting Services Naming Conventions for VB (Q110264) at
http://support.microsoft.com/support/kb/articles/Q110/2/64.asp

### Do Not Depend On Global Variables Being Shared

Visual Basic 6.0 stores a separate instance of a global value on a per-thread basis. This means that if you have a global variable that's being written to by an MTS object, other MTS objects will not see the same value unless they happen to execute on the same thread. Chances of this happening on a reliable basis in a stateful or stateless programming model are quite low. Lesson: Don't rely on all objects seeing the same data in global variables.

### How To

Be sure to check that your application doesn't rely on a global variable being shared across multiple activities, to store, for example, a counter. Place a transient global state in the SPM if it meets SPM's qualifications. Otherwise, place state in a database.

### References

The Shared Property Manager (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_spm_8awi.htm)
A Thread to Visual Basic by Dan Appleman (http://www.desaware.com/articles/threadingL3.htm)

## Avoid Calling GetObjectContext in _Activate if You Don't Need the Context Continuously

### Details

Although the object context lifetime spans the time between the Activate and the Deactivate methods of the **ObjectControl** interface, it is usually better to acquire a reference to the context every time it is needed.

### More Information

A typical function call uses the **ObjectContext** only once or twice to set the vote on the current transaction, or to use some attributes. In that case, using **GetObjectContext** every time will not create significant overhead in machine instructions executed and will keep the code less cluttered and easier to maintain.

However, if you use the object context repeatedly (say, if you use the ASP object **Response** through the **ObjectContext.Item** collection) it is better to acquire a reference within the function and set it to nothing when you're finished doing the heavy task. You can also use a WITH statement to avoid repeated use of the same variable. This will make your code more readable/maintainable and improve execution times.

## Use ObjectControl_Activate and Deactivate Instead of Class_Initialize and Terminate

The Class_Initialize event happens before the object is put under COM+ activity control. This has puzzling consequences that make your project less predictable and harder to debug. For example, the Object Context is unavailable, database connections are not enlisted in transactions, and VB may crash while debugging these methods.

### More Information

When an object is created to run in COM+ there is an internal structure called *context* for it that encapsulates information, such as transaction enlistment, threading model, and so on. When the **Class_Initialize** event runs, the context has not been associated yet with the object, therefore these side effects will happen:

- Database connections won't enlist in current transactions.
- Child objects created won't share the context information.

  **Note** There should be very specific reasons why any of the above actions are performed, since the constructor or destructor methods should not be required in a purely stateless object design.

Avoid doing the following operations in **ObjectControl_Activate** and **ObjectControl_Deactivate**:

- Calling **SetComplete/SetAbort**
- Calling other objects
- Performing error-prone operations (accessing databases) that may fail the current transaction

### How To

Add **Implements ObjectControl** at the top of your class module. Move all your code from **Class_Initialize** to **ObjectControl_Activate**, and all code from **Class_Terminate** to **ObjectControl_Deactivate** (if any). Then make sure you are not doing anything that may prevent your object from working appropriately.

Within these guidelines, both of these events should be used instead of the Visual Basic Initialize and Terminate events for components that are running in COM+.

## Avoid Expensive or Error-prone Activities in Sub Main()

Once your component is started in COM+, its Sub Main will be invoked (this will be upon the first object request). However, if you have long-running code in there (such as data access, loading large files, parsing and string manipulation, and so on) you can usually expect an error message ID 10010 in the Event Viewer stating "The server {your CLSID here} did not register itself with DCOM within the required timeout," even if the code is fine. If this isn't enough reason to avoid this then you should be aware of another limitation: your clients won't be able to create the components and won't get any useful error information either.

### How To

If you rely on some transient or cached state being initialized for your component, it is better to initialize it on-demand. It would not be advisable to subject your component to an operation that could fail, before it has enough life in it to tell you why it died. Avoiding the Sub Main () routine will make it easier to maintain, debug, and troubleshoot.

### More Information

This error event occurs because DCOM requested a certain class to be provided as it started the server, but the server never completed its initialization phase within a reasonable timeout (not registering its class factory).

## Acquire Resources Late, Release Them Early

Avoid holding on to resources (open files, database connections, database locks, memory, and so on) for more time than absolutely required. For example, do not establish a connection to a database until you are ready to use it (which would be just before executing a command or refreshing parameters). Also, release them early, that is, close your connections and clean up shareable resources as soon as you don't need them anymore. This will increase the scalability of your application because of the more efficient resource allocation time spans.

For example, when executing a stored procedure you would set as many options on your ADO objects, such as connection options, command parameters, and so on, as possible before connecting to the database. Once you've connected, you would only do the required operations (execute a stored procedure, retrieve a result set, and disconnect the recordset), and then close the connection as soon as possible so you can continue running without holding a connection unnecessarily.

### More Information

If a server object holds onto any resource longer than needed, that resource is not available for any other client. Resources like database connections, threads, memory, locks, and so forth are finite. Use them sparingly. In the client server world, you might be able to get away with such practices for some amount of time before users felt the impact. On even medium volume *n*-tier distributed applications, the immediate effect is felt by thousands. Plan and develop for efficient use of resources from square one.

### References

MSJ: Q&A ActiveX/COM by Don Box (http://msdn.microsoft.com/library/periodic/period98/activex0398.htm)

Holding State in Objects (http://msdn.microsoft.com/library/psdk/mts20sp1/building_5xmb.htm)

## Program Using Thread-Safe Operations

While Visual Basic is a great language for developing server components, there are certain sets of functionality within the product that are not intended for high volume "GUI-less" or server-side use. Choosing to implement a server-side solution with these single purpose features will eventually bring instability to your server processes. The following scenarios are greatly discouraged from participating in server-side business objects:

- Calling **DoEvents**
- Raising an event that's handled by an object on another thread, or in another process
- Showing a form
- Invoking a library or component that has not been designed and rigorously tested for server use

### References

Designing Thread-Safe DLLs (http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbcondesigningthreadsafedlls.htm)

## Use Early Binding When Possible

Using early binding (that is, declaring object variables as being of the specific class or interface, and not as an object) greatly improves application performance.

### More Information

You are using early binding when you are using an object through a variable declared to be of the correct type, as in:

```
Dim oHouse As CHouse
```

Or

```
Dim oHouse As Iasset
```

You are using late binding when your variable is declared as in:

```
Dim oHouse As Object
```

Or

```
Dim oHouse As Variant.
```

Please note that this is not related in any way to the operator used to instantiate the object (using **CreateInstance**, **CreateObject**, or **New**). In all cases, early binding requires you to have the correct references set in your VB project.

Late-bound calls are done through a COM interface called **IDispatch** that requires two calls to execute a specific method. The first, **GetIDsOfNames**, is used to retrieve an ID for a certain method, and the second, **Invoke**, to actually send the parameters and make the call.

For production-ready implementations, it is extremely rare that you would want to distribute late bound objects. Some development environments benefit from the loose coupling of late binding in the early stages of prototyping and usability testing. However, as projects grow larger and more developers are brought online, early binding is a good frontline defense against incompatible interfaces and confusing code trees. Another huge benefit to early binding is pre-compiler checking and the ability to take advantage of the VB IntelliSense® feature.

### How To

Add references to the library that have the interfaces or objects you want to use:

```
Dim As [your classname/interface here].
```

#### References

Declaring Object Variables (http://msdn.microsoft.com/library/officedev/odeopg/decondeclaringobjectvariables.htm)

INFO: Using Early Binding and Late Binding in Automation (Q245115) at http://support.microsoft.com/support/kb/articles/Q245/1/15.asp

### Declare Parameters as ByVal Whenever Possible

Visual Basic defines parameters by reference (By Ref) by default. To pass these parameters by value (so that the parameter doesn't get transmitted back to the caller using the network unnecessarily), declare the parameters using the ByVal keyword. Any parameter not having ByVal is defaulted to ByRef by Visual Basic.

#### More Information

Passing arguments to a function can be done by value or by reference. In the case of a remote call, a by reference argument has to be sent to the server and then sent back to the client. This involves extra network traffic. Also, if the client does not require the data being sent back, having the parameter as ByVal matches more closely to the interface's semantics.

See the Queued Components and COM+ Events sections for specific caveats when using these services with functions that have ByRef parameters.

#### How To

For every parameter of every function for which a copy back to the client is not required, add the ByVal keyword.

#### References

ByVal keyword (http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbenlr98/vakeybyval.htm)

INFO: Only Write-Only Interfaces Can be Marked as Queued in COM+ (Q246625) at http://support.microsoft.com/support/kb/articles/Q246/6/25.asp

### Do Not Use the GetSetting/SaveSetting Functions in Your Projects

VB provides these functions as an easy, direct way to read the registry. However, they are designed for desktop applications, and they read a portion of the registry that is specific to the unique user identity of the person using the application **HKEY_CURRENT_USER.**

#### More Information

The VB-provided functions to access the registry are not designed for use in a server environment. You may evaluate using other ways to access the registry or default to loading files. In server environments, the user-specific portion of the registry is not loaded for the specific MTS package identity. The user's default key is used instead. This sometimes causes applications that work fine under Interactive User to break when there is nobody logged on to the server.

#### How To

There are many objects that allow reading and writing to the registry that are more stable and can access any part of it. Third parties provide some and others are provided as part of the MSDN samples.

Also, consider using files instead of the registry. Files in an NTFS drive can be secured, maintained, replicated, and programmed much more easily than registry settings.

## Set Threading Model=Apartment Threaded on Your Project

In Visual Basic 6.0, your DLLs can be either Single Threaded or Apartment Threaded—make sure you have Apartment Threaded selected.

### More Information

Visual Basic can create Apartment Threaded components or Single Threaded components. Apartment Threaded is the model required to work concurrently under MTS. If Single Threaded is selected, all method calls into any instance of any object in the DLL will be serialized. The use of any single threaded components will result in an application that lacks scalability, since all method calls will come in single-line formation, even if executed on different object instances.

### How To

Go to **Project**, then **Properties**, and in the General tab select the **Apartment Threaded** item in the Threading Model box.

### References

Components and Threading (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_synchronization_1lgn.htm)

COM+ Neutral Apartments (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_synchronization_6d83.htm)

FIX: VB 6 DLL Settings Cause Access Violation During MTS Shutdown (Q214755) at http://support.microsoft.com/support/kb/articles/q214/7/55.asp

Apartment-Model Threading in Visual Basic (http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbconapartmentmodelmultithreadinginvisualbasic.htm)

## Set Unattended Execution on Your Project

In Visual Basic 5.0 and Visual Basic 6.0, a checkbox under the Project's properties specifies Unattended Execution. Make sure this is checked before compiling.

### More Information

This attribute specifies your component will run without user interaction. This means that all message boxes raised with **MsgBox** will be logged as NT Events (see below for a better way to log events).

In Visual Basic 6.0, unlike VB 5.0, it will not change your component's threading model.

### How To

Go to **Project**, then **Properties**. In the General tab, check the **Unattended Execution** box.

## Check "Retained in Memory" on Your Project

In Visual Basic 6.0, a Project's properties check box specifies Retained in Memory. Make sure this is checked before compiling.

### More Information

This attribute specifies the way your component's lifetime impacts the loading/unloading of required DLLs. If this box in unchecked, DLLs may be released in an unexpected order that may cause your application to crash on shutdown, interrupting normal cleanup procedures and producing ambiguous errors as well as some undefined behaviors.

### How To

Go to **Project**, the **Properties** and in the **General** tab, you'll see the Unattended Execution and Retained in Memory. Check **Unattended Execution** to enable Retained in Memory, and check them both.

## Do Not Use Global Multi-Use Classes

Global Multi-Use classes are useful for client-side development. In server-side DLLs, where a stateless model prevails and good knowledge of the lifetime of the server objects is important, this setting should not be used.

### More Information

The Global Multi-Use setting allows your clients to have a certain class instance available and to have this class' methods available in the global namespace. This setting is a programming facility that makes it simpler to write client-side code at the expense of letting VB take control of the object's lifetime on the server. Avoid this in a distributed environment.

### How To

You should set the class to MultiUse instead of GlobalMultiUse, and create and destroy it from the client as required.

### References

BUG: IDE Crash with Compiled GlobalMultiUse (Q187983) at http://support.microsoft.com/support/kb/articles/q187/9/83.asp

Class Instancing Property (http://msdn.microsoft.com/library/devprods/vs6/vbasic/vb98/vbprocreatable.htm)

## Do Not Declare Variables "As New…"

When you declare a variable of an object type, you can use the As New… keyword. This saves the effort of writing the line of code that actually creates the object. However, this will cause every reference to the object to be slower, and may surprise you with an unexpected object lifetime. Furthermore, when used in combination with private classes it may bring unexpected thread behavior.

### More Information

When you declare a variable with the As New… keyword, you are telling Visual Basic to create an object instance for you whenever you need one. This will cause Visual Basic to compile your source with extra checks to see if the object is set to Nothing. If the object has not been destroyed, Visual Basic will create a new object every time the variable is used.

Some results in object lifetime may be surprising as well.

```
Dim o As New Project1.CSample
o.Foo
Set o = Nothing
MsgBox o Is Nothing 'False! o will never be visibly Nothing
```

This kind of behavior will make your code harder to maintain and may cause errors that are difficult to spot. Using the As New keyword also has repercussions on how the object is created (read the following item for more information).

### How To

Make sure that you declare and create your objects in distinct steps.

### References

INFO: Use of New Keyword In MTS Environment (Q202535) at
http://support.microsoft.com/support/kb/articles/Q202/5/35.asp

## Do Not Create Objects in the Same Project with "New"

When using the New keyword to create an object in the same project, you are bypassing COM+ object creation mechanisms, and therefore your objects will not have the expected behavior under transactions, when working with security or regarding threading.

In COM+, both the New keyword and the **CreateObject** function will flow the context (transaction, security, and so on) into the child object if the object being created is in another project. If both the creating (parent) class and the child class are in the same Visual Basic project, you will need to use **CreateObject**.

### How To

See the item below for guidelines regarding object creation.

### References

INFO: Use of New Keyword In MTS Environment (Q202535) at
http://support.microsoft.com/support/kb/articles/Q202/5/35.ASP
How Object Creation Works in Visual Basic Components
(http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbconhowobjectcreationworksinvbcomponents.htm)

## Create Your Objects with the Correct Operator

**CreateObject** and New have different behaviors depending on the scenario. Objects running in COM+ can be created using either **CreateObject** or New. Both of these ensure that the child correctly inherits the parent's context, and that the object is created on the correct apartment. You don't need to use **ObjectContext.CreateInstance** in COM+.

Using **CreateObject** allows you to specify the ProgID and location (computer) of the object, whereas using New will allow you to create an instance of the specific class that your project has references to using the local registration information.

A common trap is using the New operator to create objects in the same DLL. The Visual Basic New operator bypasses standard COM object creation mechanisms, and therefore prevents context flow to the child object.

### How To

Use New:

- To create utility objects or objects such as ADO, MSXML, CDO, MSMQ.
- To create custom objects in other project or DLL.

Use CreateObject:

- To create custom objects in the same project or DLL.
- To create objects by a runtime-specified ProgID and/or computer.

The following samples illustrate the appropriate use of the creation operators:

Scenario 1: Creating objects, registered or not, in COM+:

```
Set oXMLDom = CreateObject("Microsoft.XMLDOM")
Set oTransfer = New OtherProject.CTransfer
Set oDBConnection = New ADODB.Connection
```

Scenario 2: Creating an object in the same Visual Basic DLL (thus in COM+):

```
Set oISPSubs = CreateObject("MyProject.CSubscription")
```

### References

INFO: Use of New Keyword In MTS Environment (Q202535) at
http://support.microsoft.com/support/kb/articles/Q202/5/35.ASP
How Object Creation Works in Visual Basic Components
(http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbconhowobjectcreationworksinvbcomponents.htm)

## Compile with Symbolic Debug Info

This section will assist you in troubleshooting the application if it crashes. It will allow you to see in which function the problem happened, the contents of your variables, and other relevant information.

### How To

When compiling your DLL, make sure to check Create Symbolic Debug Info on the Options tab. This will create a PDB file just where you compiled your DLL. When deploying the DLL, make sure the PDB ends in the same directory as the DLL file. Also, make sure that your PDB and DLL match in the sense that both must be created and deployed together to work together.

Don't be concerned about run-time overhead as it's not an issue with the creation of symbolic debug information that resides outside of the executable or DLL. Your server environment will not be affected either. These files remain static until your application has an exception and they are used to inspect the context of the failure.

For example, if your server application is crashing on an apparently random basis, the first step is to determine exactly when and why it is failing. If you approach the problem with a low-level debugger and you have the PDB files for your component, you might have all the information you need to isolate the issue, such as which function calls that led to the problem, the function parameters, and the state of the DLL's variables.

### References

Native Code Compiler Switches
(http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbconnativecodecompilerswitches.htm)

## Application Lifecycle

During the lifetime of your project, many decisions have to be made with regard to operational procedures. Things such as planning ahead for versioning, stress testing, and troubleshooting are crucial at different points in time. Planning will save you a lot of headache and time in the end. Check out Microsoft Solution Framework for a lifecycle guideline at www.microsoft.com/msf.

### Know When to Set Binary Compatibility

Compiling a DLL specifying Binary Compatibility means that the newly generated DLL will have the same TypeLibraryID, Class IDs, Interface IDs, and other identifiers as the old one. This is required once subsequent versions of a DLL are deployed to working clients. However, its use during the development cycle may be detrimental to productivity. Keep Project Compatibility until you are willing to freeze all interfaces, then go with Binary Compatibility and stick with it.

#### More Information

COM rules state interfaces cannot be changed if they keep their IID. Visual Basic uses a feature of COM known as interface forwarding to shield developers from this rule. However, this ends up creating complex registry entries. This forwarding is in reality a chain of lookups that eventually points to the latest and greatest compiled COM component.

The ability to drop a component into MTS is lost once the forwarding chain is set. This can cause confusion when you are in the midst of a rapidly changing interface specification. Eliminate as much of this chain as possible.

In addition, there is an application cycle issue. Once interfaces have been set, going back and changing them may put other concurrent development activities at risk.

#### How To

During development cycle, your object interfaces are usually undergoing many changes required to correct errors not detected during the design phase. During this time of evolution, it is wise to maintain your DLL with project compatibility. In extreme cases, you might even rely on late binding. This, however, will take away the huge developer performance benefit of the Visual Basic IntelliSense AutoComplete feature. You might want to take a step back, get your interfaces right, and then resume with concurrent development.

Once you are certain you can roll out the component, store a separate copy of your DLL in a centralized agreed upon location for future reference (perhaps in Visual SourceSafe with Read Only Permissions), point your Compatibility dialog to it, and specify binary compatibility.

#### References

PRB: Adding a New Method to a VB Component in MTS Breaks Existing Client (Q241637) at http://support.microsoft.com/support/kb/articles/q241/6/37.asp

MSJ: Visual Basic Design Time Techniques to Prevent Runtime Version Conflicts (http://msdn.microsoft.com/library/periodic/period00/versioning.htm)

BUG: MTS is Unable to Delete VB6 Project Compatible Server (Q190175) at http://support.microsoft.com/support/kb/articles/q190/1/75.asp

PRB: This Interface Has Been Forwarded Error in Visual Basic with MTS (Q222634) at http://support.microsoft.com/support/kb/articles/Q222/6/34.asp

### Stress Test Your Application for Realistic Capacity Planning

The scalability of an application has more to do with resource contention, queue topology, and application architecture than physical resources available on a given box. Adding more processors to a machine or threads to a process will not always measurably increase performance, especially if the current resources are not being adequately exercised. The best way to determine if an application can scale to certain loads is through stringent stress testing (besides an actual production environment testing, which may not be realistic).

#### More Information

Even the simplest models for scalability are based on nonlinear relationships and are derived from simple systems. With complex systems, it is much harder to model projected performance, not to mention performance under extraordinary conditions. To have reasonably accurate metrics for system capabilities, test your application using different stress levels and thoroughly document actual performance numbers. You might want to graph a Performance/Load or Load/Resources chart and observe the trend for more realistic expectations. However, a projected value never replaces the actual empirical results you will obtain from stress testing.

#### How To

With MTS, you can use the Status View in the snap-in or use XTremesoft's performance monitor counters to determine object usage under a certain stress level. If you see the total number of active objects for a package approach 100, you might need to tune your MTS thread pool. On the other hand, your architecture may be forcing an unrealistic load on the MTS activity threads.

#### References

Performance Testing a Scalable Application (http://msdn.microsoft.com/library/techart/d4perftest.htm)

Consequences of Application Deployment Without MTS (http://msdn.microsoft.com/library/psdk/bdg/cmldd04_2zcj.htm)

Web Workshop: Load Test Your ASP Application (http://msdn.microsoft.com/workshop/server/asp/server092799.asp)

Windows DNA Performance Kit (http://www.microsoft.com/com/resources/WinDNAperf.asp)

XTremesoft (http://www.xtremesoft.com)

### Design Your Test Environment to Closely Match Your Production Environment

There are many problems people run into when moving an application from testing, QA, or staging, into production. They include:

- Different operating system versions
- Different services running
- Different number of processors
- Different security boundaries being crossed by the application
- Different domains, domain trusts, or local privileges or identities
- Network traffic impacts application's ability to communicate with connected resources
- Different connectivity settings (DNS, firewalls, and so on)
- Different resource availability (problems with hard-coded paths, registry, and so forth)

#### How To

Many of these problems can be avoided by involving your network administrators and security engineers during the specification, design, and development stage. This includes keeping them in the loop during all critical testing suites. Also, be sure to know your application requirements and test your system using the same kind of locked-down behavior one expects in production.

#### References

INFO: Stress Tools to Test Your Web Server (Q231282) at
http://support.microsoft.com/support/kb/articles/Q231/2/82.asp

### Test Deployment and Have Backup Plans if Something Fails

The biggest cause of grief during the life cycle of a distributed application is the day when that application goes live. The lack of rehearsal and planning for this event is only eclipsed by the behavior of all the parties involved in attempting to breath air into the lifeless application. Before your "Going Live Day," you need to rehearse the operation in a clean environment with all the participants present. Work the little problems out in a controlled environment without the added pressure of impacting production users. Avoid the turmoil and embarrassment of a failed deployment by planning and documenting the production installation process and the contingency plans.

#### How To

Create an environment to be called the "clean system." It does not know about your application and test deployment. Starting from a well-known configuration (a clean install of Windows 2000 Advanced Server, or whatever maps to your production environment), begin to clearly document every step involved in the installation of the distributed application. Note the location of the binaries and where the installation point will be.

Pretend you will not be there to troubleshoot during the entire process. Ensure that you have included small test scenarios the installers can use as waypoints to verify that the installation is going well. Lastly, document how to uninstall your application, including registered components, registry keys, temporary files, special users created for it, or services installed, started, or stopped.

#### References

Deployment and Registration of COM+ Applications
(http://msdn.microsoft.com/library/psdk/cossdk/pgdeployment_toplevel_5kdv.htm)

### Use Source Control Software, and a Source Control Strategy and Policy

Distributed applications have many components distributed throughout the network and this is usually a good thing. However, if your source code is distributed throughout the entire enterprise, you can be sure this is not a good thing. Before you begin development, define what your source code policies are going to be. Keep your documentation in source code, as well as all your test suites and results. Make sure the debug extensions are included with your compiled code. (See Compile with Symbolic Information above.)

#### How To

In your installation instructions, be sure to include a simple text file with instructions on how to get a copy of the source code in case it is needed for debugging.

#### References

Visual SourceSafe 6.0 Datasheet (http://msdn.microsoft.com/ssafe/prodinfo/datasheet/default.asp)

Setting Up Visual SourceSafe
(http://msdn.microsoft.com/library/officedev/odecore/deconsettingupvisualsourcesafe.htm)

### Learn How to Debug and Analyze Live-Running Applications or Dumps

Once you have finished your project, you will compile it and deploy it in a mock test environment, and then finally in a production environment. At anytime your application could exhibit unwanted behavior due to a configuration, hardware, logic, or runtime error.

In these scenarios, one lacks the development and debugging tools usually present during development, and has to fall back on another debugging mechanism.

In some instances, the presence of the debugger will attenuate error information. The development environment frequently suppresses timing errors and race conditions. Fortunately, there are numerous debugging tools to apply to such situations. Most of them, however, are not trivial to setup and configure.

There are two main kinds of debugging user-mode processes, such as instances of MTX.EXE. They are interactive debugging and dump debugging.

Interactive debugging means loading tools, such as Visual C++, that allow for a run-time control of threads, browsing of memory such as processor registers, stack and heap space, all during the execution of the actual program. This debugger can be specified in the property sheet of a COM+ application. For more information on interactive debugging, see Debugging Visual Basic MTS Components (http://msdn.microsoft.com/library/psdk/mts20sp1/building_1ur7.htm).

The dump debugging is useful when your application is crashing and it's hard to know why or when your server applications start running with 100 percent CPU utilization. This mainly revolves around getting a snapshot of the running process, usually a crash dump, and loading it into the debugger to analyze the state at the instant of the fault. It is harder than interactive debugging because it requires more low-level knowledge of the machine, stack, and heap architecture, but once you have the memory dump, it can be done at any time, on any machine.

### How To

Many tools are useful for the applications you develop. This will be especially true when you find yourself interfacing with legacy code, inherited black-box DLLs, or other scenarios. You will also need to generate PDB files for your components, check the Compile with Symbolic Debug Info section above.

A fine tool to choose for this type of behavior is the IIS Exception Monitor. It lets you "hook" to an MTS/IIS/COM+ process to monitor for any exceptions that occur in the running process.

The easiest tool for interactive debugging is Visual C++, since it is a GUI-oriented debugger. However, there are some limitations to its usefulness. If you are wary of polluting your production or test environments with a Visual Studio setup, then the other debuggers will better suit your needs.

Other available debuggers include the venerable WinDBG, available in the Win32 SDK. Its power comes at a price. A strong working knowledge of assembly code and its complexity is highly recommended.

### References

Low-level debugging is not a common task for a Visual Basic developer, but if you ever need to do it, the links below will help you learn and use the tools.

COM+ Configuration Settings: Applications (http://msdn.microsoft.com/library/psdk/cossdk/pgconfiguring_configuring_0777.htm#_cos_application_level_settings)

Web Workshop: Developing a Visual Basic Component for IIS/MTS (http://msdn.microsoft.com/workshop/server/components/vbmtsiis.asp)

Windows SDK Debuggers (http://msdn.microsoft.com/downloads/sdks/platform/windbg.asp)

Debugging Visual Basic MTS Components (http://msdn.microsoft.com/library/psdk/mts20sp1/building_1ur7.htm)

Web Workshop: Troubleshooting with the IIS Exception Monitor (http://msdn.microsoft.com/workshop/server/iis/ixcptmon.asp)

Web Workshop: Analyzing Logs from IIS Exception Monitor
(http://msdn.microsoft.com/workshop/server/iis/readlogs.asp)

## State Management

State management is a complex topic. State is any piece of information or data that alters the behavior of the application: a list of books, a shopping cart, user options, a list of countries, a connection string, or a hit counter are all examples. What makes this subject complex are the daunting combinations of information usage patterns, types, access methods, storage strategies available, and many other options.

To undertake this problem better, it is important to distinguish between the different attributes our information will have:

- Logical scope (organization, application, user session, activity, and so on)
- Physical scope (worldwide, corporate network, farm, server, process, and so on)
- Lifetime (persistent, per logical period, per session, per activity, transient, and so on)
- Usage patterns (write once, read many; write often, but read seldom; and so on)
- Purpose (is the info used in the application [a list of customers] or is it used by the application to work [a database connection string, metadata, and so on])

Other characteristics that will greatly affect your application's scalability and reliability include:

- Size of the information being managed and its impact on storage and network usage.
- Storage strategy (SQL Server as the recommended RDBMS, Active Directory™ services, files, in-memory such as Session, Application, or Shared Property Manager; Constructor Strings, Registry, MSMQ, and so on).

### Store Your Application State in SQL Server

Your application state contains all the information your application needs to perform its function, such as lists of customers, transactional information, and so on. This is best stored in SQL Server, even if you decide to cache it somewhere else later.

Something you should avoid is storing part of your application's information in SQL Server and part in external storages, such as a file, Session/Application variables, or SPM. For example, having transactions in SQL Server tables, but having tax percentages in an INI file or a constructor string is not a good idea.

What happens if your INI file is modified on the fly while transactions using the tax info are taking place? How can you guarantee consistency if there is a physical disk problem? Using mixed storage mechanisms for pieces of data that are closely related may compromise data integrity and make your application hard to deploy and troubleshoot. Storing other information, such as a tax percentage, in constructors also complicates things by mixing the logical application space and the administrator's physical application space.

### Build to Scale Out

The best scalability numbers obtained for any platform were done by both partitioning the functionality across different layers (a functional decomposition), and by partitioning the load for each layer across a number of devices.

Without going into all the advantages the *scale out* model brings to your application, it is important to note that this approach gives the best results when it is planned for from the design stage. This means that if you intend to make your application scalable, you must not rely on any activity being locked down to a certain machine. Common traps that violate this concept are:

- Storing critical counters or other application state in ASP Application variables or the Shared Property Manager, which have a machine or process scope.

- Storing user session information on a certain server, effectively locking down the user to that computer's resources and lifetime. This makes load balancing harder and may put at risk the availability and user experience of your application.

## Allow for Multivalued Constructors

COM+ allows you to administratively specify a string that the object will have access to before activation. It is wise to allow this one string to contain different values that your application may use for different purposes. That is unless you are sure that your class will always need only a single datum. This setting is useful to control so many things (specifying tracing levels, branching security behavior, and so on) that you will probably want to add more values in the future.

As a general guideline, the COM+ Constructor event should only be used for initial state management and to handle constructor information. It should not do any long-running or expensive tasks. For activity-specific, initialization tasks use **ObjectControl_Activate**. Remember that the Activate event happens after the Construct event, so you can use the constructor data in your activation. See the Use ObjectControl_Activate and Deactivate Instead of Class_Initialize and Terminate paragraph for specific guidelines on how to use Activate and Deactivate.

### How To

You can use the Split function to partition a separated string into an array of elements. This can be used to make an efficient constructor string parser, as in the following example:

```
Option Explicit
Implements IObjectConstruct  'Found in "COM+ Services Type Library"


'Sample variables we want to get off the constructor
'(In a stateless class these would be the only class members)
Private miTraceLevel As Integer
Private msLogPath As String

'The constructor ordinal indexes, starting at one:
Private Enum eConstructVars
   eConstructTraceLevel = 1
   eConstructLogPath = 2
End Enum
...
...
...
Private Sub IObjectConstruct_Construct(ByVal pCtorObj As Object)
   Dim vValues As Variant
   Dim sConstruct As String

   sConstruct = pCtorObj.ConstructString

    'This assumes that the first character is the separator itself.
   'This allows for more flexible configuration
   vValues = Split(sConstruct, Left(sConstruct,1))

   'For numbers we append a 0 to account for empty strings.
   'Optionally, you could do some 'default' logic
    miTraceLevel = CInt("0" & Trim(vValues(eConstructTraceLevel)) )

   'We can read strings directly. Here we implement a small default
   'logic - if nothing was specified then we return App.Path
   'Using VB's inline IF.
   msLogPath = IIF(vValues (eConstructLogPath)="",App.Path, _
         Trim(vValues(eConstructLogPath)) )

End Sub
```

There are a couple things to note, mainly the management of empty values for numbers (the array element will have an empty string—""—which cannot be cast to an Integer), the trimming of the strings, and the inclusion of fallback default values.

Keep in mind that the administrator will have to append trailing separators so that the Split function returns an array of the expected size. Simply adding an extra load of separators at the end or resizing the array (as follows) can overcome this potential nuisance:

Add this element to the Enum:

```
eConstructMAX = 2   'The amount of values I expect
```

Then check and adjust the array right after the Split call:

```
If Ubound(vValues) < eConstructMAX
Then Redim Preserve vValues(1 To eConstructMAX)
```

So in our example, the constructor string `;3 ; C:\Logs\Foo` would be loaded, parsed with the semicolon separator, and each element would be trimmed of leading and trailing white spaces, and assigned to its appropriate variable. Again, keep in mind that the first character is the separator.

### References

HOWTO: Using COM+ 1.0 Constructor Strings from Visual Basic (Q246138) at http://support.microsoft.com/support/kb/articles/q246/1/38.asp

## Don't Store Passwords in Constructor Strings

The constructor string is a tempting place to store your resource connection strings, all with database usernames, passwords, and what not. However, keep in mind that this information:

- Can be seen by anyone browsing the administrative UI.
- Can be obtained through the COM Administration objects.
- Is stored in the RegDB in cleartext.

### How To

Probably the best way to deal with sensitive information is to rely on NTFS security. That is, store it in a file that has read permissions for your COM+ application identity and read/write permissions for the appropriate administrators only. You can then load this file and parse it out as if it were the construction string itself, or do a combination of both methods.

Keep in mind that ADO connections can be opened by specifying a UDL file path instead of the full-fledged connection string. There is more on this in the Data Access section. Other information can be kept in INI or XML files. This allows you to set appropriate security on these files.

If you must store sensitive information in the constructor string, verify the following:

1. Only the appropriate users/groups belong to the Reader role of the System Package. This has to be carefully managed to prevent COM+ from being unable to read its own configuration.
2. You have controlled and audited access to the \WINNT\Registration folder, where the COM+ configuration database (RegDB) stores its files.

Constructor strings are not hashed or shuffled in any way when stored to disk. Keeping the right tabs on security is enough to prevent unauthorized access. If you have a loose security environment, you may want to encrypt or hash these sensitive strings on storage. Another alternative is to use the LSA API. These sets of functions allow you to store your sensitive information in a secure portion of the registry. They are a more complex solution but you may find it suits your needs better than INI or XML files with NTFS security. See the link below for the LSA API reference.

### References

Platform SDK: LSA Authentication (http://msdn.microsoft.com/library/psdk/logauth/lsaauthpp_1rxh.htm)

## Using Queued Components

Queued Components (QC) are one of the main services provided by COM+ 1.0. It eliminates the need to develop your own infrastructure to use MSMQ (or some other asynchronous mechanism), letting you keep a COM programming model on the client and server, while giving you the benefits that MSMQ provides as a transport. It also provides great support for security, transactions, and error handling.

When using MSMQ you have to:

- Design, construct, validate, and parse your own message formats.
- Use a special API or object model to compose, send, and receive messages.
- Develop your own message listener as a Windows NT service, managing threads, and security manually.
- Design your own scheme to handle bad message processing.

Queued Components takes care of it all!

Visit the link below for a simple overview of QC's architecture. You may find it useful to identify all the pieces of the QC infrastructure.

Queued Components Architecture
(http://msdn.microsoft.com/library/psdk/cossdk/pgservices_queuedcomponents_9p45.htm)

Check the following articles for in-depth information on Queued Components.

MSJ: Building a Store-and-Forward Mechanism with Windows 2000 Queued Components
(http://msdn.microsoft.com/library/periodic/period99/queued.htm)

Writing Queued Components
(http://msdn.microsoft.com/library/psdk/cossdk/pgwritingcomponents_queuedcomponents_6ir7.htm)

### Install MSMQ in Active Directory Mode

MSMQ 2.0 provides two main setup options:

1. Active Directory mode, in which a domain controller (DC) keeps the catalog of all the machines, networks, queues, and so forth.
2. Workgroup mode, which is intended for standalone operation of MSMQ or when a DC is unavailable.

If you have an option, install MSMQ in Active Directory mode. Read the Knowledge Base articles below for more information.

#### References

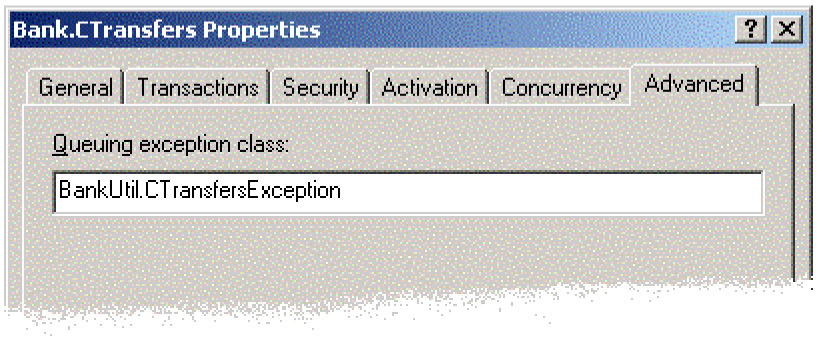
HOWTO: Determine if MSMQ 2.0 Is Installed in Workgroup or Domain Mode (Q248500) at
http://support.microsoft.com/support/kb/articles/q248/5/00.asp

INFO: Changes to MSMQ COM Components in MSMQ 2.0 (Q247780) at
http://support.microsoft.com/support/kb/articles/q247/7/80.asp

### Build an Exception Class

Queued Components have a mechanism to handle exceptions both on the client (recorder) and server (playback) sides. This mechanism relies on an interface called **IPlaybackControl** and the configuration of an Exception Class for your queued object.

An Exception Class is a COM object that is invoked to handle the errors that arise on the client or server. You can specify its ProgID or CLSID in the **Properties** (see Figure 2 below).



Queuing exception class:

BankUtil.CTransfersException

**Figure 2. Specifying an Exception Class for Bank.CTransfers**

To get the most out of it, an exception class has to support two (or more) interfaces—**IPlaybackControl** and the interfaces for your queued object. For example, if your object is called **CTransfers**, a good exception class for it will implement **IPlaybackControl** and **_ Ctransfers**. The underscore is appended by VB to name the default interface of a class.

```
Implements IPlaybackControl
Implements _CTransfers
```

**IPlaybackControl** is a simple interface that has two methods, **FinalClientRetry** and **FinalServerRetry**. QC uses this method to notify your exception class that it is handling a client-side or server-side error. When handling an error, QC will first create an instance of your exception class, invoke **IPlaybackControl_FinalClientRetry** or **IPlaybackControl_FinalServerRetry** as appropriate, and then call the methods on your custom interface as recorded in the MSMQ message.

The main goals of your exception class are to:

- Provide useful information on what operation is being attempted, such as the method calls, parameters, and so on. The most efficient way to do this on the server is to log method calls with parameters to a known log. On a client, this log should be checked and the end user should be notified about the failure, giving the user an option to cancel the operation altogether or to retry it.
- Provide information on the context in which the error happened. This is the usual logging, such as time, computer, Thread ID, Activity ID, and so on.

On the client, a QC error occurs when MSMQ doesn't allow the message to be delivered to the server queue (for security reasons is a good example), or after a long time has passed and there was no connection to the server. MSMQ will keep a message around in an outbound queue until the message's time-to-live expires, or if the server rejects the message (security problems or if the target queue has been deleted). The default time-to-live is years, unless you have overridden the time-to-reach-queue message property in the queue moniker.

On the server, where the recorded calls are executed, QC has a retry mechanism that starts working whenever the called methods return an error, the caller is not authenticated, or the transaction is aborted.

### References

Handling Server-Side Errors (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_queuedcomponents_8q9f.htm)

Persistent Client-Side Failures (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_queuedcomponents_4h0z.htm)

IPlaybackControl Reference (http://msdn.microsoft.com/library/psdk/cossdk/iplaybackcontrol_23vw.htm)

Poison Messages (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_queuedcomponents_1b5f.htm)

## How Queued Components Affect Transactions

You may experience significant changes when your component's transactional behaviors are being invoked by clients through COM (or DCOM) versus Queued Components.

When invoked by Queued Components, the COM+ Player (the infrastructure code that reads the MSMQ message and executes the recorded calls on your component) initiates a new transaction when creating the queued components proper.

This means that if your component is marked to Support Transactions, then the component will not run under a transaction when invoked by non-transactional COM clients. However, the component will run under a transaction when invoked as a Queued Component.

If this is unexpected and changes the way you think your objects should behave, reevaluate the object's transactional setting. The transactional attribute of an object should not depend on its caller.

There is another issue to keep in mind. When a transactional server-side component calls another server component through QC, there are two distinct transactions: the caller transaction and the queued component's (callee) transaction. Remember, these are two separate transactions. You can also consider the message delivery a transaction in itself. The failure of the first will cause the MSMQ message to not be delivered, therefore the server object will not execute.

If the first transaction commits successfully, then the MSMQ message will be delivered. On the receiving side, the player will start the second transaction. Any failure in the second transaction will not affect the first transaction in any way.

## Understanding the Basics of Queued Component Security

Security works differently when you are calling an object through DCOM versus invoking it through Queued Components. On the client, when you are making (or *recording*) calls there are no security checks, so all method calls will appear to succeed. The QC recorder registers two identities in the message: the identity that made the QC call on the client, and the identity that sent the MSMQ message.

These are usually the same, but they can be different when using out-of-process servers. For example, if the client you are interacting with has an out-of-process server with identity X that creates the QC instance, you are the caller, but X is the sender.

If both are the same, the QC Player will check this identity versus the roles set on the method, allowing access as appropriate. If they are different, QC will allow the call only if the message sender identity (X in the previous example) is in the QC Trusted User role of the System Application on the server.

When and if the QC allows the call to execute on the server, your component can use all the security properties of the **ObjectContext**, such as **IsCallerInRole**, to perform its task.

If the identities registered in the calls do not match the roles, or the sender is not trusted, you will get access denied errors. When designing your application, keep in mind that this will happen once the message is delivered and processed on the server, totally independently of the client.

In all QC cases, since the actual caller is not available (the calls are parsed out off an MSMQ message), Impersonation is not available.

### How To

When designing a disconnected application, take into account that security will be checked at the server beyond the client application's control.

### References

Using Queued Components Security
(http://msdn.microsoft.com/library/psdk/cossdk/pgservices_queuedcomponents_1khl.htm)

## Watch Out When Debugging QC

When debugging Queued Components, watch out for timeouts. In short, the Player starts a transaction. If you take too long debugging your code, you will exceed the Distributed Transaction Coordinator (DTC) timeout for a transaction. This will cause the transaction to abort, and QC will retry playing back the message. The DTC timeout is configured to be 60 seconds by default, which is probably not enough time to do any relevant debugging.

### How To

When debugging a QC server, set the DTC timeout to a large value. You can do this in the property sheet for My Computer in the Component Services MMC Snap-in. The Transaction Timeout in the Options tab allows you to enter a timeout value in seconds. Enter '0' to disable the timeouts.

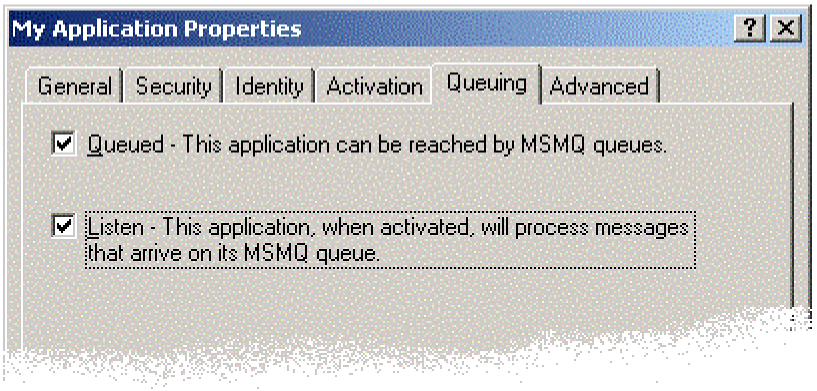
Typical symptoms of this problem are:

- Your Transactions Aborted counter on the DTC Statistics will increase for no apparent reason.
- The exception class for your component is called.

## Mark Applications to Listen

When configuring Queued Components, one of the things you probably want to do is to check your COM+ Application to Listen. This will let your application monitor the MSMQ queues and automatically pick up and play back a QC MSMQ message upon arrival. You can set your COM+ Application to Listen through the Queuing tab of the COM+ Application property sheet, as shown in Figure 3.

If you don't check this, the messages containing the recorded calls will pile up in your application's MSMQ queue and won't be processed.



[✓] Queued - This application can be reached by MSMQ queues.

[✓] Listen - This application, when activated, will process messages that arrive on its MSMQ queue.

**Figure 3. Marking a COM+ application to listen for QC messages**

## Using COM+ Events

COM+ Events or *Loosely Coupled Events* (LCE) is another new COM+ service oriented towards making applications extensible in a configurable way. The main concept of LCE is that you can configure a class as a dispatcher of events, called an Event Class. Other classes with similar interfaces and methods can be configured as subscribers. Whenever a client creates an instance of our Event Class and calls a method, COM+ steps in and makes all the subscribers receive that method call. Any association with normal "events," such as **RaiseEvent**, **WithEvents**, and so on, is purely a terminology mishap. COM+ events are not associated with traditional **ConnectionPoints** events and are used in completely different scenarios.

There are many ways to tune the behavior of these events. For example, you can:

- Make these events fire in succession or in parallel.
- Set filters so that a subscriber will only receive method calls when the parameters contain data it's interested in.
- Make the event notification be dispatched through Queued Components.
- Add/Remove existing object instances as subscribers during run time (this is called a *Transient Subscription).*
- Implement special interfaces that let you control how events are dispatched in a detailed fashion.

As you can see, all this flexibility ends up giving you many architecture and implementation options to use.

To understand the basics of COM+ Events and see an example, read the three articles below.

COM+ Events Architecture (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_events_20rp.htm)

COM+ Events Model (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_events_1oz0.htm)

COM+ Technical Series: Loosely Coupled Events
(http://msdn.microsoft.com/library/techart/compluscouple.htm)

This MSJ article explains most of the COM+ Events workings with more detail.

The COM+ Event Service Eases the Pain of Publishing and Subscribing to Data
(http://msdn.microsoft.com/library/periodic/period99/com+event.htm)

### Do Not Use COM+ Events for Multicast Scenarios

A common first reaction to developers approaching COM+ Events is to envision a multicast scenario in which a server-side event (stock value passed $150) triggers countless client applications to display a notification.

COM+ Events, however, were not designed and built to tackle this scenario. They call subscribers synchronously, so a large number of clients will hold the publisher for a long time, and they need to know about every subscriber, making the registration and unregistration of these necessary to your application. In short, LCE was not designed for it.

When you need to notify a large number of subscribers (as in a multicast scenario) that live on a remote location, you usually don't know and don't care whether they received the notification or not, and don't want to be kept waiting for all the subscribers to receive the call. In these cases, a good alternative is to use straight UDP (datagram-based IP protocol) to deliver a message over the network. The Winsock Control or API makes this very easy. Check out the links below for more information. Using the API is recommended, but if it turns out to be too hard, you can use the Winsock control instead.

#### References

Two Examples Using the Winsock ActiveX Control (Q163999) at
http://support.microsoft.com/support/kb/articles/Q163/9/99.asp

## Understand the Implications of Queuing Subscribers

When you mark a subscription as queued, COM+ will deliver the event call treating the subscriber as a Queued Component. This usually brings in all the changes in behavior you would expect, but some implications can occur.

The major implications are:

- The publisher continues execution sooner (the subscriber is called asynchronously).
- Access checks aren't done immediately, but deferred.
- The subscriber doesn't take the impersonation value of the publisher.
- The publisher and subscriber run in separate transactions.
- The subscriber won't get called if a subsequent error prevents the publisher transaction from completing.
- The subscriber will get called after the publisher transaction finishes and commits.
- The publisher transaction will be oblivious to any errors happening in the subscribers.
- There will be more disk activity (caused by MSMQ activity on transactional queues).

The lesson is to treat that "Queued" checkbox on the Subscriber property sheet with respect, and use a design decision rather than a performance tuning tweak.

### References

Composing Events with Queued Components (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_events_5sab.htm)

## Select an Appropriate LCE Extension Strategy

Your application event scenario will probably fall under one of these two categories:

1. You want your implementation to execute on a certain method invocation no matter what, and you want to have other subscribers execute only in certain cases (if there are any). In this case, you probably want to code your class to do its work and then call the Event Class, as shown in Figure 4. You can even have a flag (set in the constructor, or the SPM) to tell your code whether to invoke the event at all.

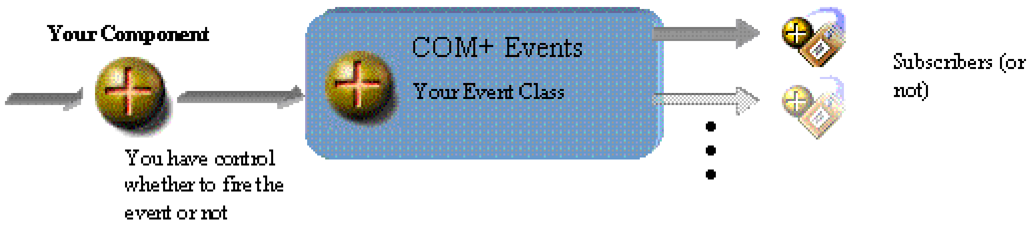


**Figure 4. Your component executes as normal and invokes the Event Class before, during, or after its processing. The interfaces do not need to be identical, and your code executes.**

In this configuration:

- Your clients invoke your component, not the Event Class.
- You have more control over when the event is invoked.
- You can have an Event Class with a different, perhaps more specific, interface than the one required by your code.
- You can decide whether it's necessary to create the Event Class in run time, gaining performance.

- Your code executes every time.

2. Another configuration is required when you want your implementation to be executed with the same characteristics as any future subscriber. In some cases, you might even envision you or your customer removing your particular subscriber implementation from the application in the future. In the scenario depicted in Figure 5, creating an Event Class up front and configuring your component as a subscriber may be the best choice even though you will have a decrease in performance.

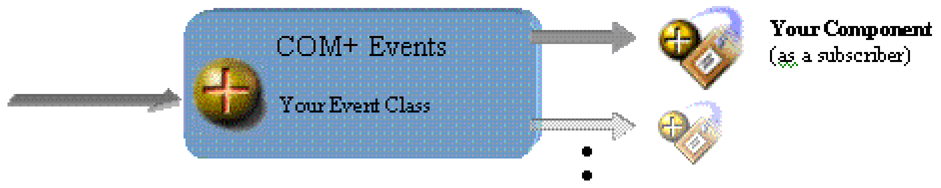


**Figure 5. Your component is configured as a subscriber to the Event Class, and conforms to the interfaces of the event. Your code should not be dependant on subscriber execution sequence.**

In this case:

- Your clients create and invoke the Event Class directly.
- Your component has to implement the right interfaces/functions just like any other subscriber.
- Your component should not depend on being executed first, last, or in between.
- The Event Class is always created, even if your component is the only subscriber.

There are countless other scenarios which you may need to implement. If that is the case, LCE provides mechanisms by which your publisher, Event Class, and subscribers can interact in a more complex way. Read the Publisher Filtering article (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_events_7xpj.htm) for additional details.

## Do Not Make Events Out Of Everything

With all the added flexibility, it's tempting to just configure all your classes as Event Classes and add one subscriber from day one. You can just leave it this way, and if you ever need other subscribers, you'll add them, right? Don't give in.

When you create a class and you configure it as an Event Class, COM+ has to step in, synthesize the right interface from the type library information, and set up the internals to get the subscribers called. As expected, this takes significantly longer than just creating the real class and calling the method.

Plan ahead and make COM+ Events only out of the required operations. If you expect some actions to require LCE-style extension, create the appropriate Event Classes and add your default subscriber, or call the Event Classes only when needed from your component.

## Watch Out for Argument Changes

When you invoke an event, the call is propagated to all subscribers. These subscriber objects usually receive the same call with the same parameters. There is a generic scenario to watch out for—whenever you are passing strings, arrays, UDTs, or objects (that is data types that are pointed to, rather than passed in the call stack), and your subscriber components change the values, subsequent subscribers will see the new values. This breaks the subscriber independence idea.

It may also bring random and unexpected problems to your application, especially when firing the subscribers in parallel. For example, a subscriber could change the data while another subscriber is using it. This will translate into crashes and hard to reproduce problems. The issue can be overcome with a bit of discipline when coding subscribers.

The most generic workaround is to copy the incoming arguments to other variables in the subscribers code. All subsequent operations in the function will act on the copy. Watch out for object arguments, since you cannot copy the whole instance. You shouldn't set them to Nothing either. This will result in subsequent subscribers trying to operate or execute methods on thin air (invalid memory).

## Code to Amortize Event Class Creation

As mentioned before, whenever you create an Event Class, the event interface is synthesized and many things happen behind the scenes. All these operations have an associated cost you want to leverage.

### How To

Whenever you have to fire multiple Events, try to code so that rather than creating and destroying Event Class after Event Class, you create one and call many times into it before destroying it. Keep in mind the other guidelines, such as don't store objects in the SPM, keep your objects stateless, and so on. These guidelines take precedence over this optimization.

## Data Access Technologies

Most of the time, you are not calculating Pi to the *n*th decimal or training a neural network. You are hitting a database with queries and updates and you are hitting this database hard. Correct usage of data access technologies, plus knowledge of your backend database is crucial to maintaining high performance and sustained scalability requirements. Here are some items you might want to take into account from your business layers.

#### General References

INFO: Tips for Using Database Components with MTS (Q169210) at http://support.microsoft.com/support/kb/articles/Q169/2/10.asp

ADO Examples and Best Practices by William R. Vaughn—published by Apress! (http://www.apress.com/).

### Use the Latest Version of the Microsoft Data Access Components

Windows 2000 ships with Microsoft Data Access Components (MDAC) 2.5. This MDAC version includes improved functionality and other features, such as XML integration without requiring you to create a file, URL resource support, and hierarchical namespace navigation.

MDAC 2.5 is also available for Windows NT 4.0 and Windows 9*x* computers. See the link below for more information and downloads.

MDAC includes both the MS ODBC and OLE DB components. Use the latest version of these components as shipped in the MDAC package. Do not mix and match ODBC and OLE DB components to create the "flavor of the week." The MDAC solution is tested and deployed in a discrete package of DLLs. The key word here is tested. Other configurations have a high probability of working sporadically.

#### References

Get the latest MDAC version and information from www.microsoft.com/data/.

### Close and Set Your ADO Objects to Nothing As Soon As Possible

When you use ADO objects, be sure to explicitly call the Close method and set the recordsets and connections you used to Nothing as soon as you are done with them. One of the most common coding errors in ADO is forgetting to close once you're done working with the object. While explicitly closing the object is not mandatory, doing so can be the difference between a working and failing application.

Setting ADO objects to Nothing clears up your error information stored in **Err**. This may bring problems if you cleanup your ADO objects in an error handler. Therefore, if you are expecting this information to be preserved, store the Number, Source, and Description in variables during the ADO object cleanup process before closing the object.

> **Note** If you are going to return a disconnected recordset, you should not close the recordset. By closing the recordset, you will be destroying the internal cache for it, so all you need to do is put its ActiveConnection to Nothing.

#### How To

To confirm that your **Connection** or **Recordset** objects are closed correctly, you can implement the following code:

```
If Not oMyRecordset Is Nothing Then
   If oMyRecordset.State <> 0 Then oMyRecordset.Close
   Set oMyRecordset = Nothing
End If
```

As mentioned before, you shouldn't call the **Close** method in a recordset that you will return to the client.

## Avoid Reusing ADO Connection Objects to Execute Many Commands

Avoid reusing open connection objects over multiple database commands. That is, open, use, and close a connection whenever needed.

If you have wrapped ADO code in a data-access object, this will not be a problem, since these wrappers perform in a stateless fashion. However, if you have ADO code in your objects, and especially if you have ADO connections stored at class level, please review the Knowledge Base article below. The main problem lies in keeping open database cursors while other things are being executed. If you are retrieving recordsets to send to the client, disconnect them from the RDBMS.

Implementing a way to reuse ADO Connection objects greatly increases the maintenance effort and sometimes leads to design mistakes that could be avoided by good encapsulation. The ADO-OLE DB layer implements its own internal connection pooling. This connection pooling makes the following flow the preferred way to access data on the server:

- Create ADO connection
- Open connection
- Use it
- Close connection
- Set ADO connection object to Nothing

This will be easy to develop, easy to maintain, as well as fast and scalable.

### References

HOWTO: Reusing ADO Connections Within MTS Transactions (Q234218) at http://support.microsoft.com/support/kb/articles/Q234/2/18.asp

## Use OLE DB Providers Instead of ODBC Drivers

When using OLE DB through ADO, you can choose to use a native OLE DB provider, or choose a special OLE DB provider (MSDASQL—the "OLE DB Provider for ODBC Drivers") that will transform and then forward all calls to an ODBC driver.

Use OLE DB providers that speak natively with the database if there is one available for better performance characteristics, more functionality, forward compatibility, and the participation of a vibrant OLE DB third party solutions market.

### How To

You can tell if you are using ODBC if any of the below is true:

- You are configuring a System, User, or File DSN in the ODBC snap-in to use an application.
- You have "MSDASQL" or "DSN" in your connection string.
- You do not indicate a provider in your connection string.

If you are using ODBC, see if a native provider accesses the same database with the same functionality support. Key areas to look for are Connection Pooling, Distributed Transaction support, and so on.

### References

MDAC Partners Area: OLE DB Products on the Market Today (http://www.microsoft.com/Data/partners/products.htm)

OLE DB for the ODBC Programmer (http://msdn.microsoft.com/library/techart/msdn_ole4odbc.htm)

Merant Corporation (http://www.merant.com/)

## Use Universal Data Link (UDL) Files to Persist Connection Strings

You can use a UDL file to store database connection very much in the same way you use ODBC Data Source Names (DSNs). A UDL file stores OLE DB connection information, such as Provider, username, password, and other options. You can open your ADO connections with the information stored in this UDL file, thus letting your administrator change it as required and avoiding the need to open the registry or to use ODBC.

Make sure your package identity has read access to the file and that the right Administrators have read-write access to it (to actually edit it). In the case of a Web site, make sure your DLL and your UDL files are not in a virtual directory, which would make them directly accessible and downloadable through the Web.

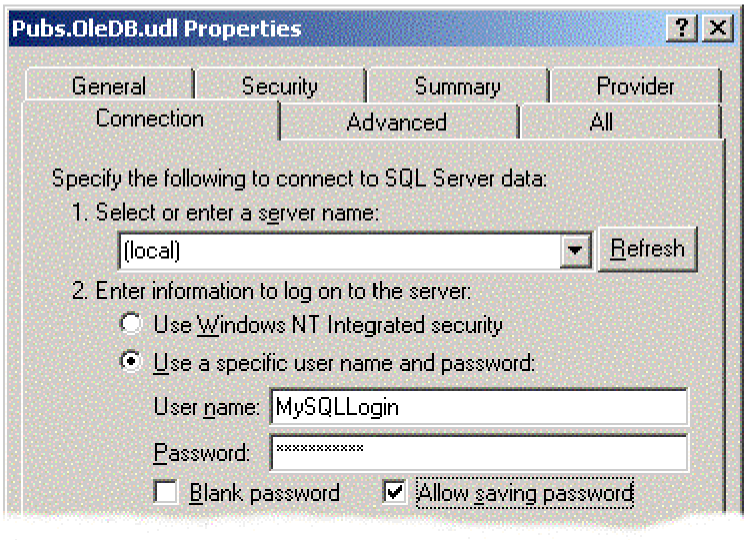
To create a UDL file, create an empty file and name it with a UDL extension. You enter all the information by double-clicking the file and entering the information in the property sheets for it, as shown below.

To open an ADO connection with the UDL, use the following syntax:

```
oOConnection.Open "FILE Name=C:\SecureStuff\MyDataLink.UDL"
```

You may instruct an administrator to place the DLL and UDL on the same secure directory, in which case you do need to hardcode the full UDL file path. As seen in Figure 6, you will need to set Allow saving password to store the full connection string. You will get a warning indicating that you are storing your password in cleartext in a file. If you set the right NTFS permissions on the file, this will not be a problem.

Do not place a UDL file in a file share so that other servers can open it. If you have many servers, keep a copy of the UDL file on each one.


Pubs.OleDB.udl Properties

Specify the following to connect to SQL Server data:

1. Select or enter a server name: (local)
2. Enter information to log on to the server:
   - Use Windows NT Integrated security
   - [✓ Use a specific user name and password:]

User name: MySQLLogin

Password: ************

Blank password [✓ Allow saving password]

**Figure 6. Setting Allow saving password**

You might be concerned about performance; after all, this is hitting the disk, right? Actually, when using this technique the file system will cache the file in memory and stress tests show the performance degradation is negligible, if measurable at all, when comparing it to accessing System or File DSNs.

## Use Stored Procedures Instead of Dynamic SQL Strings

Using stored procedures has many benefits including:

- Adds an abstraction level from the database schema, by isolating the conceptual data command from the command's implementation. Stored procedures also make the application code less coupled, allowing for easier testing.
- Provides for better code distribution. There is less logic in packing up parameters and returning result sets than in building, sending, and parsing a SQL string, plus doing all the above.
- Is more efficient in runtime since their execution plans are precompiled. Dynamic SQL strings have to include parameters to have their query plans cached optimally.
- Allows for more runtime flexibility. SQL strings compiled in your DLLs are harder to change once deployed than a stored procedure.
- Allows for better-engineered code since their parameters have declared directions and types.
- Allow for better security configuration. Having stored procedures restricted to different application/package identities or to different connection strings in different server applications is usually easier to maintain.
- If you have used stored procedures, then you will have an easier growth path if that ever needs to change.

### How To

Adding a stored procedure is easy. Just look up the CREATE PROCEDURE statement in T-SQL and follow the guidelines. If you choose to use stored procedures in the implementation of your data access objects, then be consistent for maintenance reasons. Use them either entirely or not at all. This will aid troubleshooting or feature building for later development efforts without having to second-guess the mix of dynamic SQL and Stored Procedures.

## Use OUTPUT Parameters on Stored Procedures for Single-Row Return Values

When you return a single item of data or a single row from a database lookup, do so using OUTPUT parameters rather than a result. OUTPUT parameters bypasses any cursor/result set building on the server and allows for a more efficient data transmission between the components and the database.

However, depending on your architecture, you may not wish to have different code paths for single-row and multi-row queries. You must also take into consideration how dynamic your database schema is. With more strongly typed stored procedures you are coupling them more to your database. Flexibility, as usual, challenges performance.

## Do Not Pass ADODB.Connection Objects Across Apartments

There are several considerations regarding cross-apartment/process marshalling of database connections. It is recommended that you do not do this. You'll see unexpected behaviors that are imposed by the limitations of marshalling and database drivers.

If you are doing this, it is probable that you are attempting to manage a pool of connections and have implemented your own connection dispenser. In most cases, you are working too hard to provide pooling functionality yourself. Usually, even the most stringent resource managers allow for different connections on different activities that happen on the same apartment. If your ODBC driver or OLE DB provider allows for pooling (which it usually does), close connections and open them as required, and let the OLE DB/ODBC layer provide the infrastructure work.

If the reason you are using cross-apartment/process marshalling is to provide for the proper connection string, then store the string in SPM as discussed earlier and recreate the connection in each method call as we have suggested.

### References

Pooling in the Microsoft Data Access Components (http://msdn.microsoft.com/library/techart/pooling2.htm)

## Remember the ACID Rules

Remember the rules of transactional operations, usually described as the ACID properties: Atomicity, Consistency, Isolated, and Durable. In application design, one might forget that the Atomicity and Consistency properties of transactions require all operations to be "serializable." The action of serialization imposes locking mechanisms on all the appropriate resources. If you would like a transaction to have a larger span than the locks for it, then you probably want more than one transaction.

Entire books have been written on this subject. Just keep in mind that these concepts are tightly coupled with each other: Transactions, ACID properties, and Locks.

### References

ACID Culture (http://msdn.microsoft.com/library/partbook/dcom/acidculture.htm)

Principles of Transaction Processing by Philip A. Bernstein and Eric Newcomer (ISBN: 1558604154)

These books are a solid foundation for database and transaction understanding, but they are definitively oriented to a more academic community and are harder to follow.

*Transaction Processing: Concepts and Techniques* by Jim Gray and Andreas Reuter (ISBN: 1558601902)
*Introduction to Relational Database Systems* by C. J. Date (ISBN: 020154329X)

## Use SetComplete and SetAbort to Vote on Transactions

When you call either of these methods, you are placing your vote on the current transaction, and telling COM+/MTS that you are ready to be deactivated (destroyed or set to Nothing in VB talk). All methods should place their vote on the current transaction. This allows for a better encapsulation of logic and therefore increases any opportunities for reusability or integration.

Voting is a good thing to do while developing your code. It forces you to consider the error cases and makes you think about resource management. Once any participating method call has voted to abort the transaction, the transaction is doomed. When the method exits, the object becomes available for reuse. There is no reason to continue any more processing of the transaction at the point of calling **SetAbort**.

Keep in mind this will deactivate your objects after the method call is completed and therefore will require a stateless implementation. See above for more details on the differences between stateful/statelessness.

### How To

Just before returning from your method, call **GetObjectContext.SetComplete**. In your error handler, call **GetObjectContext.SetAbort** before calling **Err.Raise**. This will insure that you are placing the appropriate vote regardless of the caller. Of course, more complex voting schemes may be implemented but the one just described is a good start.

### References

MSJ: How Microsoft Transaction Server Changes the COM Programming Model (http://msdn.microsoft.com/library/periodic/period98/mtscom.htm)

MSJ: Writing MTS-style Transactions with Visual Basic and SQL Server 7.0 (http://msdn.microsoft.com/library/periodic/period99/mtsvb.htm)

## Understand the Implications of Using Command.Parameters.Refresh

The ADO Command object can be used to execute parameterized stored procedures. There are two ways to let the Command object know about the parameters and their data types, direction, and other information.

The first method, **.Parameters.Refresh**, requires a call to the database engine to retrieve this information from the stored procedure's declaration. Note that this method requires the Command's ActiveConnection to have an open connection object or a valid connection string.

The second method is specifying the parameter information yourself. In this case, you have to know the exact names, types, and directions of the parameters being added and call **Parameters.Append** repeatedly to populate the collection.

### Issues

When invoking the **Parameters.Refresh** method, you are actually executing a query against the database server for the parameter information. Take into account network traffic, run-time adaptability of your component, and proximity of the database (measured in communication cost with the DB), along with the benefits of having accurate parameter information before deciding to use the **.Refresh** method or not.

This decision is a compromise between run-time flexibility versus performance. Adding hard-coded values yourself will always be faster, but any other method (retrieving these values from the SPM for example) will probably be slower than issuing the **.Refresh** method. However, if you wrap your MDAC code in a data-access helper class, you will find that the **.Refresh** method allows for added runtime flexibility with the results being a less frequent recompilation and redeployment of components.

Just knowing that this facility is available to you should be an important factor in your implementation. The **.Refresh** method can help you achieve a good degree of encapsulation without being too tightly coupled to the backend.

### References

Microsoft Internet Developer (MIND): Using Stored Procedures from ADO 2.1: Enhancing the Refresh Method (http://msdn.microsoft.com/library/periodic/period99/storedpr8oc.htm)

## Don't Use Data Environments on Server-side Components

Visual Basic 6.0 Data Environments greatly assist using ADO to perform operations on databases. However, for components intended to run under MTS or COM+, use straight ADO code, which is more stable under concurrent environments. If you have wrapped ADO in a helper class or function, development is even faster and easier than with Data Environments.

### More Information

The Data Environment wraps ADO with its own policies regarding connection and object reuse. Some of the implementations the Data Environment uses were driven by client-side technology decisions. Some of these design decisions were implemented in such a way as to not be fully reliable when used as a server-side facility. If you have used the Data Environment in your distributed solutions on the server, you can expect some instability at some point in your application life cycle.

### How To

Rather than relying on the abstraction level provided by Data Environments, build a wrapper for your data access code that allows you direct manipulation of the data access components without any intermediate handlers. Pure ADO code is MTS friendly and meets the base requirements for functioning well in a multi-threaded environment.

## Understand the Impact of Changing a Transaction's Isolation Level

All SQL Server transactions initiated from transactional COM+ objects are promoted to have serializable isolation specified. The default is Read Committed. This is the highest isolation level and it guarantees consistency in all scenarios.

However, this means that you will see more locks being held when your database operations are executing from transactional COM+ objects. These locks are good and important. They are in place to keep your data operations consistent. However, if consistency falls behind performance in your goals, you may evaluate tuning the isolation level. To put it another way, only resort to changing isolation levels when the cost of inconsistency is lower than the cost of low performance.

There are ways to change isolation levels as a performance optimization, but you must understand the implications of doing this, which may include incorrect data being sent to the client, unexpected calculation results, and more.

### Issues

The easiest way to determine if a piece of data is eligible for a less restrictive isolation level requires that you address the full impact of exposing the application logic to inconsistent results. For some applications and use scenarios, a certain level of error or miscalculation is acceptable at the expense of accurate, but late, data.

Different isolation levels protect you against different hazards that may occur when successive read or write operations take place on the same piece of data from two transactions. For example:

- Write-After-Read (protected when using Serializable level)
- Read-After-Write (protected under Cursor Stability and Serializable levels)
- Write-After-Write (protected under Browse, Cursor Stability, and Serializable levels)

As a rule of thumb, try assessing the precision required by each command, as opposed to deciding it application-wide. If precision is not an issue, then your database access methods could be eligible for a lower isolation level and better scalability. However, you should document your decision well and explain this in your metrics.

### How To

You can change the isolation level on a command in two ways.

1. Set the isolation level for the transaction. All commands executed against the connection will share it. On SQL Server, run the following line after opening a database connection:

```
oConn.Execute "SET TRANSACTION ISOLATION LEVEL x"
Where "x" stands for the isolation level desired.
```

Another way to do this is to specify a hint in the actual statement or stored procedure.

2. See the online SQL Server books for the WITH keyword and valid isolation level values. Other RDBMSs and resource managers have different implementations of isolation across transactions, connections, and commands.

### References

SQL Server: SET TRANSACTION ISOLATION LEVEL (T-SQL) at http://msdn.microsoft.com/library/psdk/sql/set-set_39.htm

SQL Server: Locking Hints (http://msdn.microsoft.com/library/psdk/sql/8_con_7a_17.htm)

SQL Server: sp_lock (T-SQL) at http://msdn.microsoft.com/library/psdk/sql/sp_la-lz_2.htm

The links under the Remember the ACID Rules section will be useful as well.

## Security, Communication, and Configuration

Hackers are out there to get your server. They are smart, numerous, and have only one thing in mind: getting in. They could be anybody—internal and external individuals or groups. They have many smart tools to choose from, however persistence is their greatest weapon. Paranoia is your best defense.

This may sound extreme, but correct planning for internal and external security is needed in environments where sensitive data and applications are manipulated. Keeping your systems correctly tuned and configured will help you avoid trouble.

Your security policy has to make the appropriate compromises between an environment with tight access checks and detailed auditing where almost no resource can be shared, and a resource-oriented or task-oriented environment, where a coarser granularity on identity may be acceptable in exchange for more opportunities for resource sharing and pooling. Remember, if you don't have a security policy, you don't have security, period. Any implementation effort has to have security thought into it from the first day.

#### References

Security in COM+ (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_security_5jbz.htm)

Deciding Where To Enforce Security
(http://msdn.microsoft.com/library/psdk/cossdk/pgservices_security_9zax.htm)

Using Distributed COM with Firewalls
(http://msdn.microsoft.com/library/backgrnd/html/msdn_dcomfirewall.htm)

### Change the Default Passwords

An incredible amount of critical information—financial, marketing, personal—has been stolen off the Internet not only because of deficient security architectures, but because databases and systems were irresponsibly left with their default installation passwords. If you don't want to be part of this group, be sure to change the default login passwords for well-known users in the RDBMS, Windows NT computer, and other resource.

This is not a securing-your-system checklist, but rather a don't-get-caught-with-your-pants-down reminder:

- Change the SQL Server "sa" user password to something other than blank. Don't use a dummy value such as password, sa, secret, and so on.
- Do not hardcode user names or passwords in ASP pages. These may be handed around artists, localization engineers, translators, and so forth.
- Keep secure information in secure data stores, such as the registry and file system. Make your application load passwords at run-time from the registry or the file system.

If an outside attacker has a means of attempting to log in to internal resources, you will need to reconfigure your firewall and network architecture. Changing the password is a first step, but attacks are going to be persistent and the hacker may eventually get in. Auditing failed attempts is always a good source to consider as well.

#### References

SQL Server 7.0 Security (http://msdn.microsoft.com/library/techart/sql7security.htm)

Connecting to SQL Server Over the Internet (http://msdn.microsoft.com/library/psdk/sql/1_server_9.htm)

PRB: Cannot Change SA Password in Enterprise Manager (Q218172) at
http://support.microsoft.com/support/kb/articles/Q218/1/72.asp

See the Use UDL Files to Persist Connection Strings section for more helpful links.

### Check Security at the Door

There are many ways and places to perform security checks. However, one good practice always prevails—check security on the first entry point possible. This simplifies administration, decouples your application from the resources it uses, and improves its scalability by not forcing it to waste time doing things that will fail later.

### More Information

Generally, you can check security and put barriers either at the entry point—setting restrictions on who may call components—or at the resource levels—setting restrictions on who may update a table. While checking security at resource levels seems enough at a first glance, this decision decreases the opportunities for growth and flexibility.

You must ensure that you're not making a business logic security check at the data layer. For example, you don't want to setup your application so that the permission to add a customer is enforced by allowing or denying INSERTS on a Customer's table. This model may eventually fail if your application grows. What happens if you need to send the customer e-mail?

This results in a layer mix-up, tying together the security of the components to both the administration and implementation of the back-end resources they use.

### How To

Set security using roles on your objects. Group methods on your front-line objects so that classes are more task-oriented, rather than entity-oriented—a Transfers class versus Checking and Saving classes for example. This will make administration easier, allowing you to set permission at object/interface level rather than method level.

### References

Assigning Roles to Components, Interfaces, or Methods (http://msdn.microsoft.com/library/psdk/cossdk/pgconfiguring_security_9ncj.htm)

Security Boundaries (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_security_4zsj.htm)

Deciding Where To Enforce Security (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_security_9zax.htm)

## Use Roles to Set Security for Your Application

The idea of Roles is to abstract the application's security namespace from the network or domain namespace, and to allow your code to react to being used by different application actors instead of specific users. These actors may represent groups of people (managers) or applications (banking components).

You can use roles declaratively (just like setting permissions on a file) or programmatically from within your business logic code.

### More Information

Hard-coding user identities, group names, or other domain or network information in your code couples your application to your particular security mechanisms and network namespace, not to mention to your network administrator. Your application will probably break when your administration procedures change or when a security reorganization takes place. Using roles as your primary security methodology will ensure future interoperability and easier administration.

Role definition is part of the initial application design, not final deployment. If you are writing business logic code and have not thought about roles, you are taking a huge risk in having to make major redesigns later. Take the time upfront to ambitiously study your business solution in terms of roles and how they will be handled by your implementation.

By appropriately mapping the security accounts to your roles, you will be able to control security in your application as it evolves. This can be done declaratively by setting permissions on objects, interfaces, or methods or programmatically by changing your logic in-code depending on the user.

#### How To

You create roles for a given COM+ application by adding them to the Roles folder in the Component Services MMC Snap-in. You can then add network groups and users to the roles as easily as you add permissions on a file.

Once you have determined the roles, you assign Windows NT groups (and users) that belong in them. You then add and remove roles to/from the appropriate objects, interfaces, and methods. The presence of a role for a certain object, interface, or method indicates access to it is granted for all Windows NT groups and users belonging to that role. Keep in mind that when an object has certain roles to it, its interfaces and methods automatically inherit those roles. If a Manager's role has access to the **MoneyTransfer** object, then the Manager's role will automatically have access to the interfaces and roles in the object.

You can also use the **ObjectContext** function **IsCallerInRole** to programmatically react to the user's identity—for example, to stop money transfers of more than $10,000 to anyone not a member of a 'Senior Account Managers' role.

#### References

Designing Roles Effectively (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_security_12t5.htm)

Using Role-Based Security (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_security_9lo9.htm)

Authorizing Clients Using Roles (http://msdn.microsoft.com/library/psdk/cossdk/pgservices_security_0b3n.htm)

Configuring Role-based Security (http://msdn.microsoft.com/library/psdk/cossdk/pgconfiguring_security_5w55.htm)

### Select the Right Security Granularity

A typical problem that hurts scalability is when identity granularity is too small. For example, do you need to have each specific end-user registered in your database (as an example of a secure resource) or is it enough to know a specific role or functional area to which this operation belongs?

In many cases, from the back end's point of view, an application rather than a specific user uses a resource. Having coarser security granularity will bring the following benefits:

- Simpler administration—the resource only manages a handful of user applications.
- More scalability—components running activities for different end users may share connections or take advantage of connection pooling.

However, this comes at a cost of less fine-grained auditing capabilities. While this may sound like a big cost, be realistic regarding your auditing plans and processes.

Try not to use separate SQL Server logins for every application user. Use different logins for different roles. Thus, rather than restricting or auditing users A and B and C, you will be setting database permissions on your roles, Managers, Clerks, Account Managers, and so on. This can be accomplished by using a different connection string depending on the roles to which the current user belongs.

You can use the following function to accomplish this task. This code makes a number of simple assumptions:

- Some roles will have specific SQL Server logins that want to be enforced.
- Some roles precede others. For example, if a user is both an Employee and a Manager, you probably want it to connect to the database with Manager access.

- The component has an array that may have been obtained from the object's constructor with a list of the roles to use in appropriate precedence. The last element in the list is the default connection.
- There is a set of UDLs with the same name as the roles on the same path as the DLL.

```
Public Function GetRoleConnectionString(RoleList As Variant) As String
    Dim i As Integer
    Dim oCtx As ObjectContext

    Set oCtx = GetObjectContext

    For i = LBound(RoleList) To UBound(RoleList) - 1
        If oCtx.IsCallerInRole(RoleList(i)) Then Exit For
    Next i

    'We build a connection string out of the UDL name by
    'adding FILE, the path and the extension
    GetRoleConnectionString = "FILE Name=" & App.path  & "\" &_
        RoleList(i) & ".UDL"


    Set oCtx = Nothing

End Function
```

This function may be used the following way:

```
' Assuming we have a comma-delimited list of special roles
' e.g.: "Managers,Employees,HelpDesk,DefaultUser"
' that may have been obtained from the constructor string for the object…

'Declare a place to put our array of roles
Dim vRoles As Variant

'As an example, create the array from a comma-delimited string…
'you may have obtained this string from the object constructor or the SPM.
'Note the precedence of the role list – a user in both the Managers
'and HelpDesk roles
'will get the Managers login.

vRoles = Split("Managers,Employees,HelpDesk,DefaultUser" , ",")

'Open the connection!
oADODBConnection.Open GetRoleConnectionString(vRoles)
```

## Do Not Use Delegate for Integrated Security Database Logins

As mentioned in the Select the Right Security Granularity section, specifying resource logins on a per-user basis is usually overkill. However, if you need this fine control, COM+ has new functionality that allows COM+ components running as a certain user to impersonate that user when accessing certain resources. This simplifies the component logic since there is no need to do an artificial caller id-login association.

All COM+ applications have a setting for Impersonation Level. This setting will determine to what level the component will act as being the calling user. The highest setting, Delegate, allows your component to access remote resources appearing as its current caller. This allows you to leverage SQL Server integrated security, specifying SQL Server NT logins for your application users.

The main benefit Delegate impersonation brings is that the resource may manage its own security at a fine granularity. For example, the DBA can manage SQL Server security and auditing at a per-user level.

The drawbacks of Delegate impersonation to login to databases lies in that connections cannot be pooled, since they are opened for a specific user. This will drastically increase the connections required to work and will cause delays when connecting/disconnecting to the database.

If you are using SQL Server Integrated Security, then you may evaluate using lower impersonation levels that allow you to have a coarser granularity for the logins. You can also use the code sample in Select the Right Security Granularity to have a flexible compromise between the "one user-one login" and the "all users-one login" approach.

## Distribute Active Directory and Application Server Functionality

You should avoid making application servers out of machines hosting Active Directory. These machines are usually Domain Controllers (DCs) and there will be heavy competition for network resources among the services you host on these machines. In addition, memory resources consumed by a domain controller can be quite high, leaving little resources for your application.

### More Information

Application servers must be able to respond consistently to varying user loads and server state scenarios. Users expect the same response to their request throughout the workday. Any concurrent DC duties will impact this consistent performance and lead to end user dissatisfaction with your solution. The role of a Domain Controller machine is different enough from that of an application server that they each merit their own dedicated machine.

### How To

As a rule, try to isolate the resource usage and reduce the shared failure point between your services. Putting two stressful and important tasks on a single machine does not meet your ultimate objective of high availability, reliability, manageability, or failure isolation.

## Make Sure that Authentication Is Possible if Required

Authentication is the act of positively identifying the user based upon the credentials presented to the authenticating authority. The server machine does this by contacting a Domain Controller to request acknowledgement of the calling identity. There are many levels of DCOM authentication ranging from None, in which the server does not attempt to validate the calling user, to Privacy, in which the network packets are encrypted and data integrity and privacy are guaranteed.

The level of authentication for a communication is determined by a negotiation between the client and the server machines. The higher of both settings is used. In other words, the server attempts to enforce the stricter of the client and server settings should they differ.

### More Information

When a call or object creation request comes in to a server for a DCOM connection with an authentication level higher than None, the server must verify the identity of the caller with a DC. However, if no DCs are available, or if the call comes from a domain with which there is no trust, this cannot be accomplished, and the call will fail with the message Access Denied.

In this case, you must either change the network or trust configuration, or decrease the authentication level. At some point, you can decrease the authentication level to the point where any credentials are accepted by the Service Control Manger (SCM).

### How To

To set the authentication level on the server, open the **COM+ Application Property Sheet** and select the **Authentication Level** from the **Security** tab. Shutdown your package after changing this setting to get it enforced.

On the client, the authentication level is set for both the application and the machine level. In a typical Windows NT 4.0, or Windows 9*x* machine, you must open DCOMCNFG and change the default authentication level and the application's authentication level. Remember that the effective level enforced by the server will be the highest of both.

If you want to use your application from a Windows 9*x* client that does not log on to the domain, you will need to set the authentication level to None for the client computer and the server COM+ application. You can do so in the Security tab of the application's properties.

### References

Setting COM+ Application Authentication (http://msdn.microsoft.com/library/psdk/cossdk/pgconfiguring_security_87ji.htm)

## Tune Your Application Timeout

COM+ server applications have a timeout setting that you set to schedule an automatic shut down after the application has been sitting idle for a specified time. Every COM+ server application runs as a distinct **DLLHOST.EXE** process on a machine. With every process there are certain resources allocated. In many cases, application processes fragment their private heap of memory or even leak memory if they use libraries or components not designed to run in servers.

When the process is shut down, this pool of memory is returned to the operating system to be reallocated for another process. Your application can get a fresh start on life and receive the merits of a fresh pool of memory whenever it restarts. For most sites, having an automatic shutdown is not possible due to the constant and frequent hits. However, when you do have a choice, perform this proactive maintenance action.

You may have a decrease in usage frequency on weekends, late at night, or some other moment. Tune your timeout to happen in these usage valleys.

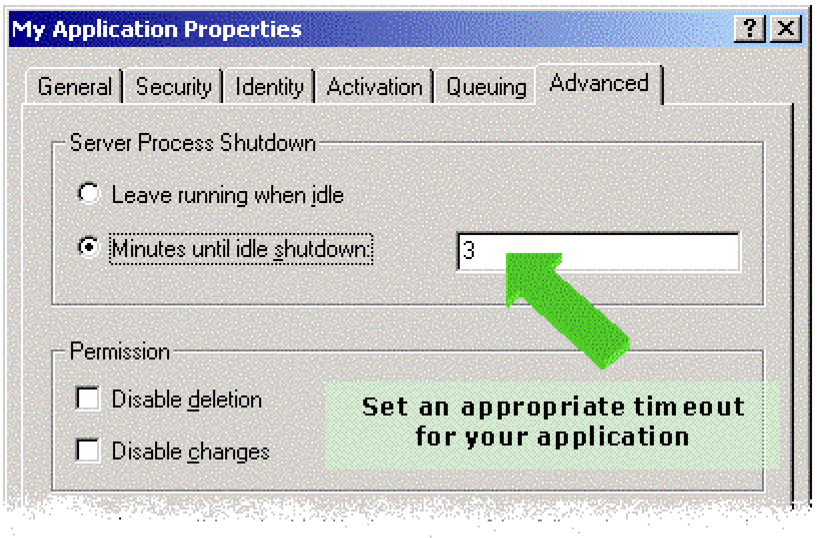
### More Information

Some problems regarding server applications, such as memory leaks, handle leaks, state expiration, and so on, are wiped out when their process closes. Although this won't cause your problems to go away and they should be addressed immediately, you might sidestep the sometimes too harsh symptoms of resource exhaustion by specifying a proactive shutdown at a low-impact moment.

The only impact that you will see is that your database connection pool will be reset and that the first user to require the application since the shutdown will experience a slight delay due to the time required to start the process again.

### How To

To set the timeout value, open the **Component Services** snap-in in Administrative Tools, then open the COM+ Application Property Sheet and click on the **Advanced** tab. Then set the value as shown in Figure 7.


My Application Properties
General
Security
Identity
Activation
Queuing
Advanced
Set an appropriate timeout for your application

Server Process Shutdown
- ( ) Leave running when idle
- (●) Minutes until idle shutdown: 3

Permission
- [ ] Disable deletion
- [ ] Disable changes

**Figure 7. Setting an application timeout**

Remember that if a client holds a reference to a server object without using it, it won't prevent the shutdown from happening and the client will need to re-acquire the connection.

The application must be activated on the server. Library applications don't have a timeout since they don't have their own process.

### References

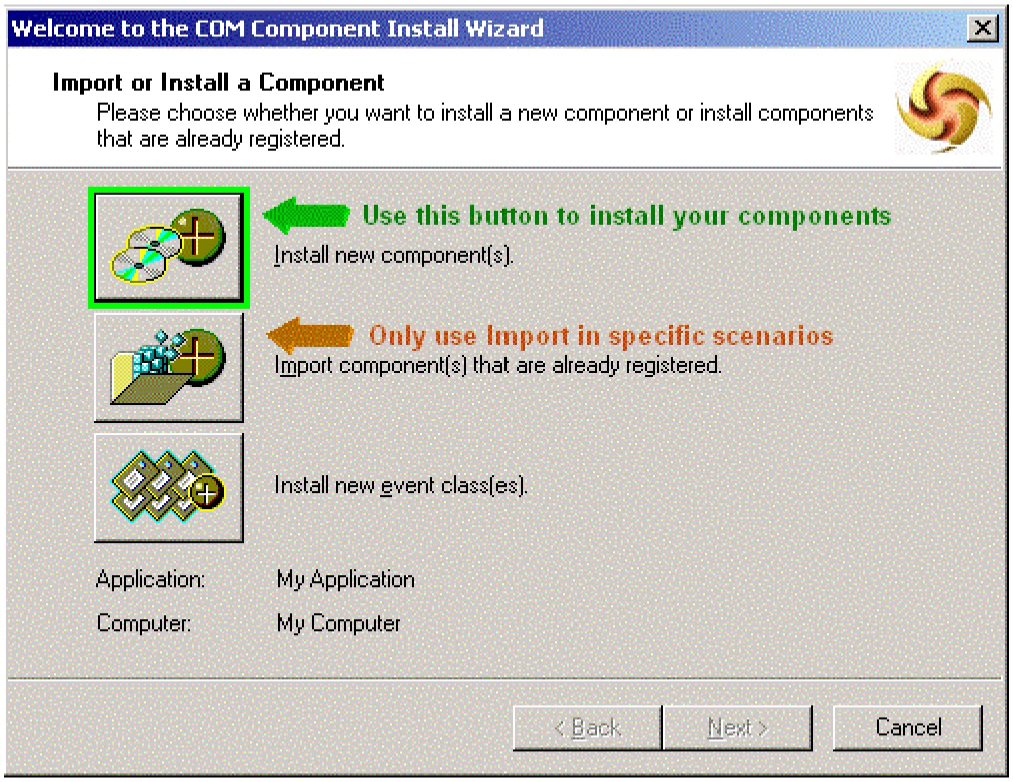
Types of COM+ Applications (http://msdn.microsoft.com/library/psdk/cossdk/pgintro_applicationoverview_8ub7.htm)

## Install (not Import) Your Components into COM+

When adding components to a COM+ application through the Component Services MMC Snap-in you have one of two choices:

1. Dragging and dropping the DLL file into the components list.
2. Right clicking under the Components folder for the package, selecting New, and choosing Import objects already registered or Install new components.

If you use this second method, use Install as depicted in Figure 8.


Welcome to the COM Component Install Wizard

Import or Install a Component

Please choose whether you want to install a new component or install components that are already registered.

Use this button to install your components
Install new component(s).
Only use Import in specific scenarios
Import component(s) that are already registered.
Install new event class(es).

Application: My Application

Computer: My Computer

Figure 8. The Component Installation dialog box

### More Information

When installing a component, MTS will query the DLL for specific information regarding the objects, interfaces, and their attributes. It will also configure the component taking into account all this information.

When importing, however, MTS will only change the local registration information to point to MTS, so the objects get instantiated within it. However, it does not validate that the component can actually be installed in MTS (not applicable in Visual Basic) and perhaps more critical, it does not query the DLL for all the classes' attributes. Therefore, if a class module was marked in Visual Basic as requiring transactions, it will appear as not supporting transactions if imported into MTS.

In addition, upgrading your MTS server to Windows 2000 and COM+ will make migrating the MTS packages to COM+ applications a lot easier if the components are installed and not imported.

### How To

In the Add Component Wizard, always choose Install and then select the DLL, or drag and drop the DLL from Windows Explorer to the Components folder of the package.

### References

Installing New Components (http://msdn.microsoft.com/library/psdk/cossdk/pgcreatingapplications_creatingapplications_06ib.htm)

Importing Components (http://msdn.microsoft.com/library/psdk/cossdk/pgcreatingapplications_creatingapplications_8yr7.htm)

## Make Sure There Is Good RPC Connectivity Between COM+ and SQL Server Machines

When there are separate application and database server machines connected in a network, all transactions are coordinated by the Distributed Transaction Coordinator (DTC) services on both machines. These DTC services require Remote Procedure Call (RPC) communication with each other, in a connection-oriented protocol. This might be ncacn_ip_tcp, ncacn_spx, or ncacn_nb_nb. They also refer to each other using host names instead of IP addresses, so you have to make sure that name resolution is working correctly in both directions.

#### More Information

The DTCs in different computers coordinate transactions amongst each other and their respective local resource managers. To communicate, they need RPC connectivity. This implies:

- Correct ports are available for RPC (if there is a firewall in between).
- Correct name resolution is working in both directions.

Since the DTC requires reliable communications, it needs a transport that is connection-oriented. For example, UDP/IP will not suffice, but TCP/IP will. These are identified by *ncacn_* prefixed RPC transports.

#### How To

You must make sure both computers can ping each other by name. If you have a firewall, make sure to read the RPC and firewall documentation. You need to open ports 135 plus a certain range (e.g. 1100 to 1130) in both directions, and configure your computers to use this range.

To diagnose communication problems you might use RPCPing, a utility that lets you test RPC connectivity. When using RPCPing, make sure to look for the transports listed above by selecting Endpoint Search on the client-side of the pinged application. Note however, that the utility uses a specific port to test connectivity, which is not very faithful to DTC behavior. It is, however, a significant test.

More tools will be posted in the Microsoft Knowledge Base as they become available and have been adequately tested.

#### References

XCLN: How to use RPCPing to Test RPC Communication (Q167260) at http://support.microsoft.com/support/kb/articles/Q167/2/60.ASP

### Use Terminal Services as a Remote Administration Tool

You can configure Microsoft Terminal Server on a Windows 2000 server so that remote administration is made easier. Terminal Server can also be configured specifically for this task so that the terminal management does not affect the quality of service of the server. Terminal Services can also be used to control another user's session, enabling joint-administration and helpdesk scenarios.

Using Terminal Services for Remote Administration of the Windows 2000 Server Family (http://www.microsoft.com/WINDOWS2000/library/operations/terminal/tsremote.asp)

How to Use the Terminal Services Remote Control Feature (Q232792) at http://support.microsoft.com/support/kb/articles/Q232/7/92.ASP

### Use TCP/IP Instead of Named Pipes to Connect to SQL Server

There are several ways to connect to SQL Server, the common ones being TCP/IP, Named Pipes, and multiprotocol RPC.

#### More Information

Using TCP/IP to connect to SQL Server will give you the following benefits:

- Simpler connectivity configuration, even over a firewall.

- You will not have the security side effects of using Named Pipes across the network (security issues other than SQL Server access control).
- You will have better performance at higher volumes and more scalability.

Looking ahead, you may want to evaluate the use of XML on HTTP as your database protocol. This will become available with SQL Server 2000, which is still in beta. As more information becomes available and we see successful implementations, this document will be updated to convey the best practices.

#### How To

The protocol choice may be configured from the SQL Servers Client Network Utility, or by specifying the desired protocol in the connection string. The SQL Server 7.0 default is usually named Pipes, so you might want to verify this configuration after installation.

#### References

How to Configure a Client to Use TCP/IP (Client Network Utility) (http://msdn.microsoft.com/library/psdk/sql/cliepcc_10.htm)

Using the SQL Server Client Network Utility (http://msdn.microsoft.com/library/psdk/sql/1_client_3.htm)

HOWTO: Change SQL Server Default Network Library Without Using Client Network Utility (Q250550) at http://support.microsoft.com/support/kb/articles/Q250/5/50.asp

INF: ODBC SQL Server Connection Parameters (Q137635) at http://support.microsoft.com/support/kb/articles/Q137/6/35.asp

## Best Practices for Using Oracle and COM+

Here are some guidelines to help you put together a COM+/Oracle application. Before beginning such an application, the developer and the architect should completely read the following Knowledge Base article:

INFO: Using Oracle Databases with Microsoft Transaction Server (Q193893) at http://support.microsoft.com/support/kb/articles/q193/8/93.asp

### Oracle Server Configuration

#### Put the Oracle Server on a Dedicated Windows 2000/Windows NT Server

Oracle server was designed to be the only application running on a machine. If you are running Oracle on Windows 2000 or Windows NT, it is assumed you have set up Oracle to run on its very own dedicated machine. This will reduce the contention for resources on the server, resulting in achieving higher performance. Customers have reported problems when trying to run other software, such as COM+, on the same Windows 2000 or Windows NT server as Oracle.

#### Configure the Oracle Server to Support Large Number of Connections

The article INFO: Using Oracle Databases with Microsoft Transaction Server (Q193893) at http://support.microsoft.com/support/kb/articles/q193/8/93.asp discusses the general rules for setting the Oracle Server Configuration parameters to support a Large Number of Connections. Adjusting these parameters has resolved many COM+/Oracle issues. Please refer to the Administering Oracle and Microsoft Distributed Transaction Coordinator section in this article for more information on this topic.

### COM+/Oracle Client Configuration

#### Build Your COM+ Server the Right Way

Setting up and maintaining a COM+ machine that works with Oracle can be complicated. For the best performance, keep your COM+ server as clean as possible by installing these applications on a freshly formatted hard drive in the following order:

1. Windows 2000
2. Oracle Client 8.0.5 with patch 80524 or later
3. Oracle Net8 with patch 8.0.5.0.3 or later
4. Update the client software registry keys

   **Note** MDAC 2.5 comes preinstalled with Windows 2000, so installing MDAC is not required.

#### Install Oracle Client 8.0.5 with the Latest Patches

Install the Oracle 8.0.5 client with the latest Oracle client patch (80524 or later) for the transactional components to work with Oracle. Oracle actively patches their software. Install these patches when they become available. Many times, the patches include fixes that relate directly to COM+/Oracle applications. See the following article for information on where to obtain these patches.

INFO: Location of Oracle's Public FTP Site (Q193941) at http://support.microsoft.com/support/kb/articles/q193/9/41.asp

#### Install Oracle Net8 with Patch 8.0.5.0.3 or Later

Install Oracle Net8 patch 8.0.5.0.0.3 or later after you install the Oracle Net 8. See the Knowledge Base article above for information on where to obtain these patches.

### Update the Client Software Registry Keys

If you are using Oracle8 Client software, you must modify the values of the following registry keys. Make sure you have the registry key values specified in the table below.

| Oracle Client | Windows NT or Windows 9*x* | Windows 2000 |
|---|---|---|
| 7.*x* | [HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\Transaction Server\Local Computer\My Computer]<br><br>"OracleXaLib"="xa73.dll"<br>"OracleSqlLib"="SQLLib18.dll"<br>"OracleOciLib"="ociw32.dll" | [HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\MSDTC\MTxOCI]<br><br>"OracleXaLib"="xa73.dll"<br>"OracleSqlLib"="SQLLib18.dll"<br>"OracleOciLib"="ociw32.dll" |
| 8.0 | [HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\Transaction Server\Local Computer\My Computer]<br><br>"OracleXaLib"="xa80.dll"<br>"OracleSqlLib"="sqllib80.dll"<br>"OracleOciLib"="oci.dll" | [HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\MSDTC\MTxOCI]<br><br>"OracleXaLib"="xa80.dll"<br>"OracleSqlLib"="sqllib80.dll"<br>"OracleOciLib"="oci.dll" |
| 8.1 | [HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\Transaction Server\Local Computer\My Computer]<br><br>"OracleXaLib"="oraclient8.dll"<br>"OracleSqlLib"="orasql8.dll"<br>"OracleOciLib"="oci.dll" | [HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\MSDTC\MTxOCI]<br><br>"OracleXaLib"="oraclient8.dll"<br>"OracleSqlLib"="orasql8.dll"<br>"OracleOciLib"="oci.dll" |

## Testing Installation and Configuration of COM+ Support for Oracle

### Tools to Test the Configuration of COM+ Support for Oracle

After following the steps for configuring COM+ support for Oracle, test to make sure the configuration is working properly. For information on tools and applications refer to the Test Installation and Configuration of COM+/MTS Support for Oracle section in the Knowledge Base article Q193893 at http://support.microsoft.com/support/kb/articles/q193/8/93.asp. You may also test the configuration by writing your own sample application.

### Use Component Checker to Test MDAC Deployment

The Component Checker Tool is designed to help you determine installed version information and diagnose installation issues with the MDAC. Component Checker runs Windows 95, Windows 98, Windows NT 4.0, and Windows 2000. Component Checker has been updated to include MDAC 2.5 data. You can obtain this tool at www.microsoft.com/data/download.htm.

## Known Limitations Using Oracle with COM+

### Oracle Client 8i Limitation with Transactional Components

There is a bug in the Oracle8i XA client library that causes all DTC transactions to fail. This bug shows up when DTC's XA support is used to perform transactions against an Oracle database using Oracle8i client libraries. This is due to a bug in the xa_close implementation. Until the bug is fixed, is it recommended that you use Oracle client 8.0.5 with patch 80524 or later with transactional components.

### Watch Out for Transaction Control in Stored Procedures

Keep in mind that you are working from a global transaction when calling a stored procedure from COM+. Calling **COMMIT**, **ROLLBACK**, or **SAVEPOINT** could have unintended results or errors. Microsoft has not tested the use of Oracle stored procedures from COM+ so caution should be exercised when using them.

### Test Oracle Software Thoroughly Before Using it in Production with COM+

Customers have problems implementing the following Oracle software in a COM+ environment:

- Oracle Names Server
- Application Server
- Connection Pooling/Multiplexing
- Fault Tolerant Software

The problem usually arises when deploying into a production environment where these services are running, so make sure that you are testing your application appropriately.

### Resources

If you have problems implementing a COM+/Oracle application, review the following articles:

INFO: Using Oracle Databases with Microsoft Transaction Server (Q193893) at http://support.microsoft.com/support/kb/articles/q193/8/93.asp

INFO: Error: "-2147168246 (8004d00a)" Failed to Enlist on Calling Object's Transaction (Q191168) at http://support.microsoft.com/support/kb/articles/q191/1/68.asp

INFO: Oracle TRC Files and MTS (Q246006) at http://support.microsoft.com/support/kb/articles/q246/0/06.asp

INFO: Location of Oracle's Public FTP Site (Q193941) at http://support.microsoft.com/support/kb/articles/q193/9/41.asp

All Microsoft Knowledge Base Articles on Oracle (http://search.support.microsoft.com/kb/psssearch.asp?)

## Client-Server Interaction

Your server is not alone. Distributed applications imply managing issues including, but not limited to, bandwidth, databases, client licenses, transactions, storage partitioning, caching, and more. How you move data between the layers is one of the most challenging assignments of building *n*-tier architectures using Windows DNA.

### Use XML Strings or Recordsets to Send Data Between Layers

There are many techniques in existence for sending and receiving data between client and server, such as parsing XML strings or Recordset objects. Here is a list of the common techniques and their pros and cons.

The big winners in terms of programmer preference are disconnected ADO Recordsets, with XML strings on the rise. The rapid growth of XML use, together with the industry-wide shift to HTTP and Web services, and the efficiency and openness of the method, earn it the choice of best practice for Windows DNA 2000 applications.

#### XML Strings

XML Strings can be obtained by using the **ADODB.Stream** object to save a recordset, by using the XML DOM, or by concatenating a string manually through VB.

Transmit XML back and forth as a simple String, not as **XML DOM** objects or **ADODB.Stream** references. Remember to use ByVal whenever possible to avoid unnecessary back and forth marshalling, especially when dealing with large documents.

The main key to building fast middle-tier components or applications that rely on MSXML is to keep documents as small and simple as possible. There is a worse-than-linear relationship between performance and document size, and the size of the XML documents being used seems to be a major factor separating fast and sluggish applications.

**Pros:**

- Disconnected from the database and application server
- Can be persisted to and loaded from files, MSMQ messages, and so on
- Powerful but new navigation semantics
- Easily transformed into different structures with XSL
- Metadata
- When correctly structured, they can be opened as recordsets
- Easily extensible
- Easily integrated with ASP and HTML applications
- Negligible impact on performance when extracted from ADO Recordsets
- Easily incorporated into schema/data validation frameworks
- Easily integrated with XML-based services such as BizTalk

**Cons:**

- Small list of UI controls that can be bound to/initialized directly from XML (but it's growing)
- Unwieldy to manipulate—it takes a considerable amount of steps to get a certain piece of data out of an XML document in comparison to a variant array

- Can stress Visual Basic's less than optimal string management with big documents if you do lots of hand concatenation

### Disconnected ADO Recordsets

ADO Recordsets are a popular data transmission strategy. Their advantages typically outweigh their caveats and their ease of development makes them a fast and productive strategy.

**Pros:**

- Disconnected from the database and application server
- Can be persisted to and loaded from files, MSMQ messages, and so on
- Have popular navigation semantics
- Have column metadata (by name and by position referencing)
- Allow data-bound controls to be used on Windows clients
- Easily manipulated within the business rules layers
- Can be manipulated to output their XML representation directly to an ASP page

**Cons:**

- Depend on ADO being deployed on client and server, but not necessarily the same version
- Hard to extend with extra information

### Variant Arrays

Although Variant Arrays have been popular in the past, disconnected Recordsets or XML strings quickly displaced them.

**Pros:**

- Easily understood and manipulated without any ADO knowledge
- Flexible scheme, not tight-bound to a schema with fixed attributes and data types
- Easily passed by value

**Cons:**

- Inefficient across the wire, sometimes taking 50 percent more packets than XML strings, and many more packets than a marshalled-by-value Recordset object.
- Costly to obtain from a recordset—a **GetRows** call can take considerably more time than obtaining an XML string via an **ADODB.Stream** object, depending on the recordset characteristics.
- The identification of columns is done by index—a reshuffling of columns in a SELECT statement could render an array with an unsuitable layout. This feature may be a plus or a minus on maintainability depending on how flexible your components and UI are to changing conditions.
- No intrinsic metadata (no field names, no types, and so on), making the run-time flexibility mentioned above hard to come by.
- Low-level manipulation functionality, such as filtering, sorting, and so on.

Other less frequently used options are:

- Property Bags Contents/Byte Arrays: These are an interesting data transmission option, but not recommended as an application-wide data transfer mechanism. VB6 objects can save and load themselves in and out of property bags, and its flexibility (the ability to add properties in a collection) seems attractive.

However, use XML strings for these types of scenarios where flexibility is paramount for better performance and maintainability.

- Comma/Separator-Delimited Strings: This strategy could be useful in some scenarios where you need interoperability with something else that speaks in separator-delimited strings. Leaving this exceptional case aside, this mechanism has little appeal, bringing most of the disadvantages of all other techniques depicted above in term of maintainability, ease of development, flexibility, and performance.

An option that you should avoid implementing unless under special conditions is using the XML DOM objects directly instead of a string.

### XML Document Object References

When making a function that returns XML, should you declare it As String or As [some member of the XML DOM family]? Using the XML DOM object reference may seem beneficial, however it will not work as expected across processes, apartments, and may limit your future growth path as the XML DOM reference evolves and morphs. There are other options that should be avoided as well.

- **ADODB.Stream** objects: The **ADODB Stream** object is a helper object that assists the conversion of a recordset into a data stream. It was not designed or tested as a way to pass information between components. All the previous options are better suited for this task.
- **Dictionary**, **Collection**, or **PropertyBag** objects: These data structures are not designed to send information between layers. They have a place in client-side programming and maybe within ASP pages. Most attempts to use a **Collection**, **Dictionary**, or **PropertyBag** will fail in real testing due to performance, security, and stability problems.

### More Information

Disconnected recordsets transmit data internally through ADTG (Advanced Data Table Gram) format, a binary representation that is efficiently packed. XML strings appear to be segmented in a more inefficient manner and may seem longer, but benchmarks have determined there may not be much difference in regard to network impact.

Using disconnected recordsets couples you to the evolution of MDAC, its deployment on client and servers, and the use of data structures that MDAC supports, such as tabular, multidimensional, hierarchical, and so on. This is not necessarily a bad thing. This plumbing is well understood and the semantics have been popular since DAO became a big hit years ago. ADO has many current advantages as XML tools catch up (and they are doing so very quickly). Even more ADO can now transform itself into XML, and back. Hence, in a loose manner, by choosing ADO you are not locking into a data access strategy that doesn't allow a path for evolution.

On the other hand, the use of XML implies the standardization of a structure in which to send data between clients and servers. Usually the recordset-generated XML is good enough, however if your needs are more demanding in terms of flexibility and manipulation, you can transform the XML.

XML is more flexible from the point of view of interface definitions, but it transfers part of the burden to the layers manipulating it. It takes several lines of code to open an XML document and get to a given piece of data.

While modern browsers have built-in support for XML, XSL, Schemas and other XML-related technologies for data manipulation and presentation, Windows NT 4.0 and Windows 9*x* clients don't come with this support natively unless they have Internet Explorer 5.0 or higher. Windows 2000 computers have XML support out of the box.

There is no blanket rule as to what can be the best choice for your particular application. Each option frees you from some technologies but binds you to others, such as DCOM, ADO, XML, and so on. Each option has performance characteristics that vary with the way they are used. To make sure you are choosing the best for your application, you should test with data sets that you expect to use, taking into account number of rows, data types, number of columns, and so forth. If you are testing, keep in mind that ADO Recordsets and ADO Recordset-generated XML strings are easy to interchange since one can be obtained from the other, back and forth. That is, if your component returns an XML string, your client can still open an ADO Recordset from it and use it without knowing the source.

Variant arrays pose a slightly bigger challenge to test since the code on the client side is drastically different from code used to consume ADO Recordsets.

As mentioned before, choosing the best option for your particular application demands an effort, and you must balance developer, time-to-market, and run-time performance as appropriate for each system you build.

### References

Optimizing Interactions Between the COM+ Business Logic Tier and the Presentation Tier (http://msdn.microsoft.com/library/psdk/cossdk/pgdistributed_design_6ofm.htm)

HOWTO: Get XML Representation Of an ADO Recordset in VB (Q252767) at http://support.microsoft.com/support/kb/articles/q252/7/67.asp

MIND: Beyond the Browser: An XML Parser in Visual Basic (http://msdn.microsoft.com/library/periodic/period99/beyond0999.htm)

INFO: Using Disconnected Hierarchical Recordsets (Q213856) at http://support.microsoft.com/support/kb/articles/Q213/8/56.ASP

www.microsoft.com/data/ for MDAC updates for Windows platforms

## Do Not Pass Client Object References to the Server

When calling a method on the server, you may be tempted to pass as a parameter an instance of an object or control that is living on the client to the server, so that the server can query it for more information. Don't do this. The effects and problems it creates are too abundant for a short description but here are two:

- It entails coupling between the business layer and the presentation layer.
- It is an inefficient usage of the network and it may mess up your security picture since the calls would go from the server to the client, reversing the usual flow of communication.

### How To

Make sure your server-side components take all the data required in every method call, thus making these components stateless. If you feel a server has to call the client, make sure that:

1. You are not ruining client interoperability for your business layer. That is, what happens if the client has to be migrated to ASP? To a touch-tone program? To a user-interfaceless XML service?
2. You are not trying to emulate an asynchronous scenario. That is, the server can tell the client when it has finished and when it will know implicitly.
3. You do not drastically change your environment security-wise because of calls coming out of the server to the client.

If you feel your server must notify a client about something, do it through Queued Components or some asynchronous notification scheme. Above all, do not taint your business logic with user interface.

## Declare XML Functions and Parameters as Strings, not XML Objects

If you decide to use XML strings as your data transmission strategy, you are faced with the simple question of how to declare functions and parameters. You should declare parameters as strings and not as members of the XML object model. That is, declare a function following this style:

```
Public Function SaveAuthor(AuthorXML As String) As String
```

And not like this:

```
Public Function SaveAuthor(AuthorXML as MSXMLDOMDocument) _
As MSXML.DOMDocument
```

This will make your objects less dependent on the current XML version on the machines involved. Performance numbers suggest that for most cases (documents of reasonable size without too much complexity), the cost of reparsing the string is minimal.

#### Do Not Pass Collections or Dictionaries as Method Arguments

The Visual Basic collection or **VBScript Dictionary** objects were not designed for this use and the application might fail randomly. Try using recordsets, arrays, or XML strings instead of collection or dictionary objects for cross-component communication.

#### More Information

There are many ways to have named property collections. Depending on the usage you give to it, you might find that passing recordsets, arrays, XML strings, or PropertyBag content is more appropriate. The **Dictionary**, **Property Bag**, and **Collection** objects were never intended or designed for remoting, therefore they were never tested in this fashion.

#### How To

Make sure your components don't take in parameters or have return values that are declared as, or set to be, **Collections**, **Property Bags**, or **Dictionaries**.

#### References

INFO: VB 6.0 Readme Part 13: Dictionary Object (Q191792) at http://support.microsoft.com/support/kb/articles/q191/7/92.asp

### Be Sure to Pass Property Values and not Objects as Parameters

#### Details

When programming a Visual Basic client application, it's common to use the default properties of controls and other objects as parameters for function calls. Using default properties can cause unexpected failures when using remote components or ADO. If you are calling a remote object directly from your User Interface, and the call accepts variant parameters, be sure that you are passing the actual value and not the object itself (such as a **TextBox**).

#### More Information

If the parameter of a function is a Variant, Visual Basic cannot determine whether you really want to pass the object or the object's default value, so it assumes you want to pass what you have explicitly written. Many Visual Basic programmers depend on Visual Basic to address the ambiguity in the usage of objects by implementing the default property. This implementation is not the best choice. When you want to pass something, make sure you are very explicit.

ADO Parameters and method arguments seem to have a special dislike for default properties. In many cases, strange error messages are returned just because a text box's contents are specified as `txtFoo` instead of `txtFoo.Text`.

#### How To

Be sure to correctly reference the property, such as in `Text1.Text`.

#### Take advantage of Windows 2000 On-Demand Client Registration

Windows 2000 has new functionality that allows your clients to automatically locate, install, and register your components whenever required.

Configuring the Active Directory the right way can make your network aware of the existence of certain classes. Whenever a client with the right policies tries to create the object and the object registration is missing, Windows 2000 will automatically locate this central Class Store, download and install the component, and transparently continue with the application.

Leveraging this great functionality takes two steps:

1. Configuring the component in the Class Store.
2. Enabling the **Download missing COM components** option in the **Group Policy Snap-in** to modify the client-side COM runtime behavior so that it checks the Class Store when it fails to locate the component in the local registry.

### References

Deploying COM Applications Using the Class Store (http://msdn.microsoft.com/training/offers/winvco_bld/topics/winvc00073.htm)

Publishing COM and COM+ Components (http://msdn.microsoft.com/training/offers/WINVCO_BLD/Topics/winvc00120.htm)

## Guidelines for ASP Development

Your application server is, or eventually will be used by a Web server, usually an IIS computer running ASP pages. ASP is a unique client to your objects. It brings in special threading and security considerations.

While many Web sites using ASP do not use components at all, in this paper ASP is the glue between an Internet client and your components. The article ASP Component Guidelines (http://msdn.microsoft.com/workshop/server/asp/server01242000.asp) provides a short checklist to a healthy Web application.

### Partition Services Between ASP and Components

ASP is mostly used to create HTML or XML files on the server to be consumed by clients, so we'll focus on that usage scenario. This raises the common question, if ASP pages are on the server are they part of the business layer? In a component world, the answer is usually no. The fact that ASP runs on a server, probably in the same box or room as your application server, does not make it part of your business logic.

Making this clear distinction will have huge payoffs as your user interface tools evolve or as you enable more business-to-business scenarios.

Having said that, let's review some of the most important business vs. presentation partitioning guidelines:

- Isolate UI code from business logic. This includes writing UI-coupled code, such as MTS objects using ASP intrinsics, separate from business logic code, as in different DLLs.
- Isolate your transactions from your ASP pages. Transactional ASP is great in some scenarios, but having components and a multi-tiered application changes this. Components should not rely on having their transactions, and therefore business logic semantics, managed by the client layer.
- Place presentation components (those using Request and Response) in the same machine and/or process as the Web server. If you put objects using the ASP intrinsic objects on remote machines, all invocations on the intrinsics will happen in a callback fashion. That is the COM+ server calling the IIS client, which noticeably decreases performance and complicates security configuration. You can place these rendering objects in a COM+ application marked for Library activation.

ASP lives on the server, so your ASP pages must conform to resource-sharing rules and keep scalability in mind. The following should help clarify:

- Manage as little user-specific state as possible in Session.
- Keep your ASP pages stateless and allow for resource pooling when possible.

#### How To

When evaluating if a piece of code belongs in the business logic or presentation layer, ask yourself, "If I had to swap my ASP pages for a touch-tone telephony application, would this code be useful?" If the answer is yes, try to categorize it as business logic code or user interface helper code.

If the code isn't useful if changing clients, or if it is a helper in building the user interface, that code belongs in your presentation services layer. That is, in ASP pages or components using the ASP intrinsics. It would not belong in your business object components.

### Understanding the Differences Between Desktop and ASP Clients

ASP is a particular client to your components, different from traditional single-threaded Win32 applications on desktops. The main differences are outlined below.

- Thread Management: ASP is a multithreaded client. This means that there will be many concurrent activities running, probably processing different ASP pages at the same time. This means that you can't make an object monopolize the system by pretending it's the only one around. This may have unexpected repercussions, such as making it a bad practice to store objects in ASP Session or Application variables.

- Security Context: ASP is executed by Internet Information Services 5.0 in Web sites with low, medium, or high isolation. Even more, these Web sites can have different security settings, allowing or denying anonymous access, authenticating clients, and so on. All these settings yield a large number of scenarios in which different user accounts end up calling your objects.
- Ease of Growth: This is not a technical issue but rather a side effect of the facilities Web applications provide. Traditionally, increasing the user base for a desktop application demanded carefully planned rollouts to a known amount of clients. ASP has changed this procedure. Once up and running, your ASP-Visual Basic application may be easily opened up to regional or worldwide use, by all employees, all business partners, and all customers. Picture it this way—a single e-mail with a hyperlink may increase your user base tenfold. Is your application ready for this? The only way to know is to stress test your Web site to obtain realistic performance expectations. See the Application Lifecycle section for more information on stress testing.

How should you use your Visual Basic objects within ASP? Create your objects and destroy them within page scope. That is, make your ASP pages stateless if possible, relying on Session or Application variables for transient state only. Do not store objects in Session or Application variables. This will lock an ASP thread to your session, destroying all scalability expectations. That is, your Web server will handle no more than a couple of dozen users. If you need to store something in Session or Application, make it data and not objects.

There are many other guidelines to follow. It is recommend that you read J. D. Meier's "Servin' it Up" column on MSDN Voices. The column includes a wealth of techniques, practices, and tips that will help you develop scalable, reliable ASP and component applications.

### References

MSDN Voices: Servin' it Up Column (http://msdn.microsoft.com/voices/archive.asp#server)

SeminarOnline: Using Custom COM Components Under ASP (http://www.microsoft.com/Seminar/1033/Webcast0909ASPCOM/Seminar.htm)

MSDN Magazine (http://msdn.microsoft.com/msdnmag/default.asp)

## Do Not Store References to VB Objects in Session or Application

All Visual Basic 6.0 components are Apartment Threaded, meaning they run in single-threaded apartments (STA). This means that when you create an object on a thread, all calls to that object must be serviced using that same thread. Putting one instance of an STA object to be used by many threads (from concurrent Web site users) will result in serialized activities and will probably bottleneck in your application.

Furthermore, storing an **STA** object created with **Server.CreateObject** in Session scope will effectively tie the executing thread to the current user, therefore limiting the maximum number of concurrent users of your application to a default of 20x*N* (*N*=Number of processors).

### How To

If you have followed the recommendations of making your objects stateless, there is no need to store references for reuse by clients and storing them in application scope. Clients will be able to create, use, and destroy their own objects independently. This will reduce the need to keep session-specific objects, since they will be holding no session-specific state.

The recommended way is to make objects stateless that will access the database or other stores, such as cookies and LDAP, as required.

If you need to use Session or Application scoped data, store the data and not the objects manipulating it in there. You can create a class that will encapsulate the manipulation of the desired values.

#### References

INFO: Do Not Store STA Objects in Session or Application (Q243543) at http://support.microsoft.com/support/kb/articles/q243/5/43.asp

PRB: Storing STA COM Component in Session Locks Session Down to Single Thread (Q243815) at http://support.microsoft.com/support/kb/articles/Q243/8/15.ASP

INFO: Component Threading Model Summary Under ASP (Q243544) at http://support.microsoft.com/support/kb/articles/Q243/5/44.ASP

### Learn What's New in IIS 5.0

Internet Information Server 5.0 brings many new enhancements. These improvements are well documented in J.D. Meier's MSDN article, Leveraging ASP in IIS 5.0 (http://msdn.microsoft.com/workshop/server/asp/server02282000.asp). Here is an overview of the most important improvements described in the article.

- Improved, out-of-the-box performance
- Server.Transfer and Server.Execute methods
- Centralized error handling
- Improved Browser Capabilities
- Improved Script engines
- A Regular Expression Parser
- Integration with ADO Recordsets XML capabilities
- New security, caching, isolation, and administration features

#### References

Important Changes in ASP (http://msdn.microsoft.com/library/psdk/iisref/iiwachng.htm)

INFO: What's New in ASP with IIS 5.0 (Q222487) at http://support.microsoft.com/support/kb/articles/Q222/4/87.ASP

## Error Handling

Distributed systems will have distributed errors. The information you decide to collect, store, discard, and display is crucial in determining whether something is going wrong technically or logically in your application. Is your application ready for the challenge of real-world use? Do your users know what to do in the case of a problem? Do you know what to do?

Building a good error-handling framework can prevent problems from escalating undetected. Your reputation is going to be enhanced or damaged by the way you handle your error and exceptional conditions. Begin attacking this issue before you cut your first line of code, and save yourself from needless stress later.

Microsoft has released two important fixes regarding error handling in COM+ applications. These fixes prevent your error information from being erased in specific scenarios and prevent unexpected "Method '~' of Object '~' failed" error descriptions. These articles explain the exact symptoms and describe how to obtain the hotfixes.

FIX: Calls Between Configured Components Built with Visual Basic Causes Loss of Extended Error Information (Q255735) at http://support.microsoft.com/support/kb/articles/Q255/7/35.asp

FIX: Calling IUnknown Across Computer Boundaries Causes Loss of Extended Error Information (Q255733) at http://support.microsoft.com/support/kb/articles/Q255/7/33.asp

## Give the Right Error Information to the Right Audiences

Distributed application should avoid sending too much information to the end users regarding errors. End users should not be subject to cryptic messages. Tell them what they need to know to continue with their work, and convey the nature of the error in relevant terms.

If it wasn't something they did, then tell them so. Then tell them what to do. Do not show useless messages like "Contact your network administrator with this message." Consider creating an e-mail alias for the application manager and giving the user the option of clicking on a button and mailing a reference to the error on this alias. Do all the work for your user when it comes to reporting errors. They shouldn't have to feel like they need to give a deposition to someone when they bump into an error. They have already spent enough time on the matter.

Imagine your user is adding a new customer to the user base, and after entering the information, the user tries to save it. A message telling the clerk to notify the administrator is useless. A reasonable example of an error message could be:

> "There were problems while attempting to perform the following action: *Save a Customer*. System administrators have all the information they need to diagnose this and have been notified. However, you may choose to retry the operation, or cancel it. If you decide to cancel, follow these procedures:
>
> *Write the customer information on the XYZ123 form, and submit it to Sales at the end of the day.*"

On the other hand, system administrators and developers need as much information as possible in the case of an error. There is no such thing as too much data in regards to an error. This means that error information must be preserved in such a way that the right people are notified. Using a database to preserve this information is a good idea. Log all your distributed application information to one relational database device and use the proven power of SQL Server to analyze this information.

### More Information

People who interface with the system must get the exception information they need to proceed with their role in the organization. A bank clerk, for instance, should not be burdened with capturing information for your needs. Their role has them servicing the customer. Therefore, your error messaging strategy should be tuned to help them accomplish their jobs.

Put yourself in your user's shoes. Was this error caused by data entered? What data caused the error? Will operation be able to complete? Was the operation actually performed or not? Can the customer process the reservation later or should he go to the pen-and-paper procedure? Where are the pen-and-paper procedures?

### How To

The easiest way to do this is to encapsulate all your exception-state logic in a class or project. This code need not be super-optimized or sacrifice functionality for the sake of speed. Remember, an error in an application may be a symptom of a bug in implementation and design, and may not be easily reproduced or may depend on transient conditions. Saving as much information as possible will reduce isolation and troubleshooting time and effort.

Finally, an error is better reported to the end user as an information message. Sending fatal errors or scalding messages to your end users will make users less inclined to use your application. While somewhat advanced by current standards, a user agent is another good way to interact with the user.

### References

Visual Basic Concepts: Debugging Your Code And Handling Errors (http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbcondebuggingyourcodehandlingerrors.htm)

Get a Handle on Error Handling (http://msdn.microsoft.com/library/periodic/period97/gethandleonerrorhandling.htm)

When Things Go Wrong: Interacting with Users (http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbconwhenthingsgowronginteractingwithusers.htm)

Visual Basic Programmer's Journal (VBPJ): Create an Animated Desktop Assistant (http://msdn.microsoft.com/library/periodic/period98/vbpj0898.htm)

*The Inmates Are Running the Asylum* by Alan Cooper (ISBN:0672316498)

*How the Mind Works* by Steven Pinker (ISBN: 0393045358)

### Log All Required Information

Errors may not happen all the time and may be caused by hard-to-identify and/or duplicate conditions. Whenever you have an error condition, your code must log all the information available for you to analyze later. Thread IDs, security identities, and even the time and location of the error can be crucial depending on the problem at hand.

### How To

An adequate, though certainly not complete error log must include the following information:

- Date and time of the error (use one time zone if applicable)
- Physical machine name
- Error number
- Error source
- Error description
- Current object and method call
- DLL version number
- Call arguments that might be useful in reproducing the problem
- Information from the **ObjectContext** (Transaction/Security info)
- Application name, Process ID, and Thread ID

### How To

Encapsulating all the error handling in a class is probably the best bet. Place the following code in a class, build a DLL (with all the recommendations outlined in this document), and place it in a COM+ library application on the server.

The following function will take an **ObjectContext** object, and create an XML string that represents it. XML was chosen to store this information since the context has hierarchical collections that may or may not be there depending on different configuration options:

```
Private Function GetContextXML(ByVal Context As ObjectContext,_
        ByVal SecurityCallContext As SecurityCallContext) As String
    Dim sXML As String

    sXML = "<ObjectContext>" & vbCrLf

    sXML = sXML & T("Time", Now)
    sXML = sXML & T("Application", App.Title)
    sXML = sXML & T("ApplicationVersion", App.Major & "." & App.Minor)
    sXML = sXML & T("UnattendedAndRetained", -(App.UnattendedApp And App.RetainedProject))
    sXML = sXML & T("Thread", App.ThreadID)
```

```
If Context Is Nothing Then
    sXML = sXML & T("Valid", "0")
Else
    sXML = sXML & T("Valid", "1")
    With Context

        With .ContextInfo
            sXML = sXML & T("ActivityId", .GetActivityId)
            sXML = sXML & T("ContextId", .GetContextId)
            sXML = sXML & T("IsInTransaction", .IsInTransaction)

            If .IsInTransaction Then
                sXML = sXML & T("TransactionId", .GetTransactionId)
            End If

        End With


        sXML = sXML & T("IsSecurityEnabled", .IsSecurityEnabled)
        sXML = sXML & T("ItemCount", .Count)

        If .IsSecurityEnabled Then
            With .Security
                sXML = sXML & T("DirectCallerName", .GetDirectCallerName)
                sXML = sXML & T("DirectCreatorName", .GetDirectCreatorName)
                sXML = sXML & T("OriginalCallerName", .GetOriginalCallerName)
                sXML = sXML & T("OriginalCreatorName", .GetOriginalCreatorName)
            End With
        End If

    End With


    If SecurityCallContext.IsSecurityEnabled Then
        With SecurityCallContext

            Dim i As Integer
            Dim oCallers As SecurityCallers

            Set oCallers = .Item("Callers")


            sXML = sXML & T("CallerCount", oCallers.Count)
            For i = 1 To oCallers.Count
                With oCallers.Item(i)
                    sXML = sXML & "<Caller>" & vbCrLf
                    sXML = sXML & T("AccountName", .Item("AccountName"))
                    sXML = sXML & T("AuthenticationService",
                        .Item("AuthenticationService"))
                    sXML = sXML & T("ImpersonationLevel",_
                        .Item(" ImpersonationLevel"))
                    sXML = sXML & T("AuthenticationLevel",_
                        .Item("AuthenticationLevel"))
                    sXML = sXML & "</Caller>" & vbCrLf
                End With
            Next i
            Set oCallers = Nothing
        End With
        End If

    End If

    sXML = sXML & "</ObjectContext>"

    GetContextXML = sXML
End Function

Private Function T(Name As String, Value As String) As String
    T = "<" & Name & ">" & Value & "</" & Name & ">" & vbCrLf
End Function
```

**Note** This is practically the only instance in which you would pass a reference to an **ObjectContext** as a function argument. A context belongs with its object. Barring this scenario, passing **ObjectContexts** is usually not a good thing to do.

This code will loop through your ADO Errors collection and generate another piece of XML with that information.

```
Private Function GetADOErrorXML(ByVal ADOErrors As ADODB.Errors) As String

    Dim sXML As String
    Dim oADODBError As ADODB.Error

    For Each oADODBError In ADOErrors
        sXML = sXML & "<ADODBError>"
        With oADODBError
            sXML = sXML & T("Description", .Description)
            sXML = sXML & T("HelpContext", .HelpContext)
            sXML = sXML & T("HelpFile", .HelpFile)
            sXML = sXML & T("NativeError", .NativeError)
            sXML = sXML & T("Number", .Number)
            sXML = sXML & T("Source", .Source)
            sXML = sXML & T("SQLState", .SQLState)
        End With
        sXML = sXML & "</ADODBError>"
    Next

    GetADOErrorXML = sXML
End Function
```

And last, this code will store the **Err** object to an XML string.

```
Private Function GetVBErrorXML(ByVal Error As ErrObject) As String
    Dim sXML As String

    sXML = sXML & "<ErrorInfo>"
    With Error
        sXML = sXML & T("Description", .Description)
        sXML = sXML & T("HelpContext", .HelpContext)
        sXML = sXML & T("HelpFile", .HelpFile)
        sXML = sXML & T("LastDllError", .LastDllError)
        sXML = sXML & T("Number", .Number)
        sXML = sXML & T("Source", .Source)
    End With
    sXML = sXML & "</ErrorInfo>"

    GetVBErrorXML = sXML
```

```
End Function
```

When all of this code is combined, the result is a function that writes out to a specific database if available. If the database is unavailable, it writes the information to a file, or notifies it in the event log. The database schema should have enough explicit fields for all the data by which you care to filter/sort.

## Rely on Logging and Not Direct User Interaction

Components that run on the server typically have no graphical interface and often run in a hidden Windows station, so any dialog box your application raises will not be seen. If your server experiences an error that displays a dialog box, there is no way to detect it. If it is a modal dialogue box, then your application goes into a permanent hang condition since there is no way to clear it.

A simple way to write to the Windows NT event log is using the **App.LogEvent** method. For example:

```
App.LogEvent Err.Number & ";" & "[Object.Method]" & Err.Description.
```

Of course, you would log all the information you need as previously discussed.

But then again, this is a basic error logging implementation. If your application has moderate error management requirements, it is suggested that you avoid this and aim for a more comprehensive logging framework from the start.

When it comes time to isolate problems, it is necessary to merge multiple machine logs and begin the huge task of making some sense out of these distributed logs. To be realistic, in all but the simplest cases, you should implement a centralized error handling and logging facility. Using SQL Server for this is not overkill. See the next section for more information.

### References

HOWTO: Write to the Windows NT Event Log from Visual Basic (Q154576) at http://support.microsoft.com/support/kb/articles/q154/5/76.asp

INFO: App.LogEvent Only Logs in Compiled Applications (Q161306) at http://support.microsoft.com/support/kb/articles/q161/3/06.asp

INFO: Default Filename for Text File Error Log: VBEVENTS.LOG (Q161283) at http://support.microsoft.com/support/kb/articles/q161/2/83.asp

## Use a Central Database for Error Logging

A centralized error logging facility will be of great help when diagnosing, troubleshooting, and testing your applications.

The Windows 2000 event log is designed for event logging, but for many complex scenarios, it is not enough. Imagine a distributed application in which two machines are facing errors. Having them log to different stores makes reconciliation a tough job when trying to determine what went wrong. Tracking, monitoring, and analyzing errors are much simpler when a single database holds the information.

However, it is a good idea to send high severity operational exceptions to the Windows 2000 event log, such as "could not connect to the main database" rather than application specific errors, such as "could not delete user with outstanding debts." System administrators usually monitor the Windows 2000 event logs, so try to log events and information there that will help them perform their role.

Do not implement the registry as a store for your error messages. This causes problems that are difficult to track down later. The registry was not designed with logging in mind. A verbose logging process can easily fill the registry to its maximum capacity (as configured by the administrator), causing other applications to fail.

### How To

Different logging mechanisms can be used in a hierarchical approach, in which a primary method exists, but if it fails, a secondary fallback mechanism is used. If the primary mechanism fails, the problem may be relevant to the administrator's turf so you should log this to the Windows 2000 event log.

See the following examples for ideas on what the error handling logic should do.

Primary Logging:

- Relational Database: failed to log to a relational database? Log the failure to the Windows NT log and write the application error as a string to a flat file.
- Primary Relational Database: failed to log to a relational database? Write to a message queue on the local machine.
- Primary: failed to write to a message queue? Write to a message queue that can later be processed and placed in a database. Write the error information as a delimited string to a flat file.

In all cases, your primary goal should be focused on getting this information into a relational database for further analysis. Rummaging through 7,000 entries in a flat file is tedious and unproductive, as well as expensive, for your application to do. Once you have this error information in a relational database, you can use ASP pages or your own VB applications to publish, monitor, and analyze it. This also opens the door for novel ways of automating part of your helpdesk or user feedback cycle.

## Make an Error Handling Administration/Log-Viewing GUI

Provide an administrative user interface for error handling and logging that is relevant to your troubleshooting and error tracking tasks. Make the selection of logging characteristics easy for field deployment and maintenance teams by providing them with information regarding the complexity of logging of the errors, the performance trade-offs, an how to proceed with analyzing the information.

Doing this correctly one time can prove very rewarding for it is an effort that can be shared across many projects of different nature. Good logging architecture and frameworks may be the most reusable code an enterprise can leverage across one or many platforms.

What kinds of things can the error administration GUI do for you?

- Centralize activation/deactivation of logging
- Show real-time activity through PerfMon, System Events, Queued Components, MSMQ, and other mechanisms
- Provide roll-up queries on different parameters and dimensions
- Provide a way to identify top issues to follow up with more user education, improved error handling, more thorough testing, and so forth
- Provide field information to an automated helpdesk system or helpdesk staff

The many possibilities are limited only by your imagination and resources. However, building your own user application for error management, no matter how simple or complex, is hardly ever a waste of time.

## Test Your Error Handling Design and Implementation

Specify and test your error handling and logging with the same diligence as you test other features. Test scenarios in which connections fail, resources are disabled, there are insufficient security credentials for a DCOM call, and so forth. Review the error logs and determine if you have provided an accurate description of the error to enable rapid field troubleshooting or, at the least, a good start at isolation.

You must test if the error information provided to the end users is adequate, complete, and helpful in allowing them to make a decision with regard to their work. This should be practiced with all applications and users whether internal or external. This includes Internet or extranet users, company employees, and IT staff. Confidence in an application, product, or partner is greatly undermined by bad exception handling.

## Add a Run-Time Monitoring System

Logging is appropriate but sometimes you need to record and access other dynamics in your runtime environment. Windows Management Instrumentation (WMI), Windows Performance Monitor (PerfMon), and other tools allow you to do this.

Be sure to check the links below if your application has this monitoring requirement. These are worthwhile features in complex applications and they don't require much development effort.

WMI is the preferred mechanism with system events and notification. It is a powerful and extensible mechanism, but may seem a bit daunting at first. Check the links for more information on how to take advantage of WMI from your Visual Basic application. A common homegrown way to do this is to write this tracing to a SQL database or file. This strategy has a higher toll on performance but since tracing should only be used when needed, the performance toll is minimal.

It is useful to implement a scaled approach to tracing. That is, your components can implement different levels of tracing with increased details, from nothing to a full-blown log of every internal activity. This can be extremely useful when troubleshooting.

Implementing tracing for the following information is recommended:

- Object Call: executed whenever a call is made to an object.
- Object Parameters: trace all method parameters, including arrays, recordsets, and so on. Remember that recordsets can be easily transformed to an XML string for storage in your tracing database.
- Code Flow: traces code execution step-by-step within a component, specifying the before and after of function calls, method level variables, and so forth.

Things most traces should have as appropriate are:

- Source: Machine, Time, Project, Class, and Method.
- Environment Information: security context, Process ID, Thread ID, and so on.

Good tracing is dictated by the object's role. Special objects, such as data access wrappers, Active Directory wrappers, Error Handling helpers, and so on, will have implementations of tracing that makes sense in their particular functionality and the external resources they use.

A good place to place the tracing level flag is in the constructor string for your object. This way you can take a phased approach at isolating a problem, turning on and off different types of tracing without impacting users in other areas of the application.

See the Allow for Multivalued Constructors section for more information on how to extract information from the constructor string.

### References

WMI Scripting Sample Programs (http://msdn.microsoft.com/library/psdk/wmisdk/sdkintro_9zg3.htm)

WMI Architecture (http://msdn.microsoft.com/library/psdk/wmisdk/archover_4gv9.htm)

Alerts Are Cheap Insurance (http://msdn.microsoft.com/library/techart/smartalerts.htm)

MSJ: Using PerfMon with COM (http://msdn.microsoft.com/library/periodic/period00/comperf.htm)

Finding Leaks and Bottlenecks with a Windows NT PerfMon COM Object (http://msdn.microsoft.com/library/techart/perfmon.htm)

## Learn ASP Error Handling Specifics

ASP error handling can be very different from the traditional Visual Basic error handling procedures. A basic difference is the lack of the On Error Goto [MyLabel] statement, reducing error handling to either:

- An inline approach that uses On Error Resume Next and checking for an error condition after each operation.
- A centralized scheme in which the 500-100.asp page is customized for your application's needs.

The following links will help you make better error handling decisions with regard to your ASP pages by helping you develop proven techniques that have been field tested.

#### References

Bulletproofing Your ASP Components by Charles Alexander (http://msdn.microsoft.com/library/periodic/period99/aspcomp.htm)

Fitch & Mather Stocks: Web Application Design (http://msdn.microsoft.com/library/techart/fmstocks_web.htm)

Handling and Avoiding Web Page Errors Part 1: The Basics (http://msdn.microsoft.com/workshop/Author/script/WebErrors.asp)

Handling and Avoiding Web Page Errors Part 2: Run-Time Errors (http://msdn.microsoft.com/workshop/Author/script/WebErrors2.asp)

Handling and Avoiding Web Page Errors Part 3: An Ounce of Prevention (http://msdn.microsoft.com/workshop/Author/script/WebErrors3.asp)

## Other Considerations

COM+ is constantly evolving and the incoming Microsoft Application Center Server brings some exciting issues into the arena of application deployment and scalability. MSDN will be adding code and links, refining the concepts, and expanding on the new areas as crucial points are identified. You're encouraged to send ideas, wishes, or complaints.

### Know Your Information Sources

Microsoft works hard to provide you with Knowledge Base articles, backgrounder white papers, regular columns, samples, WebCasts, and a wealth and variety of information resources. If you are evaluating the technology, designing or implementing an application, or troubleshooting some problems you've run into, Microsoft probably has the information you need. Visit these links when you're looking for information regarding MTS.

MSDN (http://msdn.microsoft.com/default.asp)

Microsoft Knowledge Base – COM+ Articles (http://search.support.microsoft.com/kb/)

COM+, DTC, and MTS White Papers on ftp.microsoft.com (ftp://ftp.microsoft.com/bussys/viper/docs)

Visual Basic Programmer's Journal (http://www.vbpj.com/)

Enterprise Development (http://www.enterprisedev.com/)

The Microsoft Knowledge Base is a powerful tool. To learn how to maximize the benefits it can provide, read this article on keywords.

HOWTO: Search for COM+ Knowledge Base Articles (Q252318) at http://support.microsoft.com/support/kb/articles/q252/3/18.ASP

### Give Us Feedback!

It is important to us to know how you make use of our technologies. Your feedback is invaluable for us at design, development, test, and maintenance phases to make sure we address your needs.

#### How To

For feedback on the content of this article, please e-mail the author at edjez@microsoft.com.

> **Note** This address is not a channel to submit technical support questions, bug reports, requests for information, or any other questions about Microsoft products or services beyond this specific document.

Do you have an idea, comment, or suggestion you'd like to pass on about a Microsoft product or service? Microsoft Wish is the place to go. You can go to www.microsoft.com/mswish or e-mail one of our many Wish aliases. Our product engineers and managers monitor these aliases.

IIS Public Wish: iiswish@microsoft.com

Internet Explorer Wish: iewish@microsoft.com

Visual Basic Wish: vbwish@microsoft.com

SQL Server Wish: sqlwish@microsoft.com

Visual Basic Windows CE Wish: vbcewish@microsoft.com

SOAP Wish: soapwish@microsoft.com

IIS Exception Monitor Wish: emwish@microsoft.com

> **Note** Microsoft Wish aliases are not a place to submit technical support questions, bug reports, requests for information, or any other kinds of questions about Microsoft products or services.

Here is a short list of products to keep an eye on in the coming months:

- Application Center Server
- SQL Server 2000
- BizTalk Server 2000
- Visual Studio 7.0

We encourage you to enlist in the beta programs, learn from them, proactively plan your development strategy, and help shape the products you'll be using in the future.

## About the Author

Edward A. Jezierski is a COM+ DS Engineer currently located on planet Earth who wants the world to enjoy the existence of COM+. Outside of Microsoft, he contemplates genetics and cognitive systems while skydiving.

This white paper has contributions from the Microsoft development, MSDN, and field teams—MTS/COM+ Developer Support, Microsoft Consulting Services, and Application Development Consulting—as well as from numerous partners and customers.

Special thanks to: Terrence Nevins, J. D. Meier, Joe Long, Dick Dievendorff, Mary Kirtland, Jim Carley, Bill Vaughn, Erik Gunvaldson, Subash Bhamidipati, Srinath Vasireddy, Bernard Chen, and all the people at Microsoft and partners who helped get this document published.