COM+

Jim Fawcett

CSE775 – Distributed Objects

Spring 2006

References

- COM and COM+ Primer, Alan Gordon, Prentice Hall, 2000
- COM Programming with Microsoft .Net, Templeman, Mueller, Microsoft Press, 2003
- COM and .NET Component Services, Juval Lowy, O'Reilly, 2001
- Understanding COM+, David Platt, Microsoft Press, 1999
- Windows 2000 Brings Significant Refinements to the COM(+) Programming Model, Don Box, MSJ, May 99
- House of COM, Don Box, MSJ, May 99

COM+ Objectives

- Provide the infrastructure for Enterprise Computing:
  - Application is used by many types of people in an organization
  - Application is intranet and/or internet capable
  - Provides support for security
  - Provides reliable data access and communication over unreliable network connections
- Fundamental three tiered computational model
  - Presentation layer on client's desktop
  - Business logic layer on application server
  - Data access layer on remote database servers

What is COM+ ?

- COM+ provides the following services:
  - Transaction services
  - Security services
  - Synchronization services
  - Queued components
  - Event Service
  - JIT Activation and Object Pooling
  - In Memory Database
  - Load Balancing

- Many of these services are available administratively as well as programmatically.

COM+ Services

- ***Transaction services:***
  Coordinates the use of transactions across several objects.
- ***Security services:***
  Provides both programmatic and administrative security services at the interface and method levels
- ***Synchronization services:***
  Provides both programmatic and administrative synchronization of components using the Thread Neutral Apartment (TNA), sometimes also called the Rental Apartment model.
- ***Queued components:***
  Implements store and forward messaging using MSMQ.
- ***Event Service:***
  Provides event objects and subscription lists stored in COM+ catalog.
- ***In-Memory Database:***
  Automatic caching of back-end tables on middle-tier machines (ADO.Net)
- ***Load Balancing:***
  Distributes object creation requests among a number of servers in a cluster.

COM+ Vs. .Net

- "COM+ component services are .Net component services."

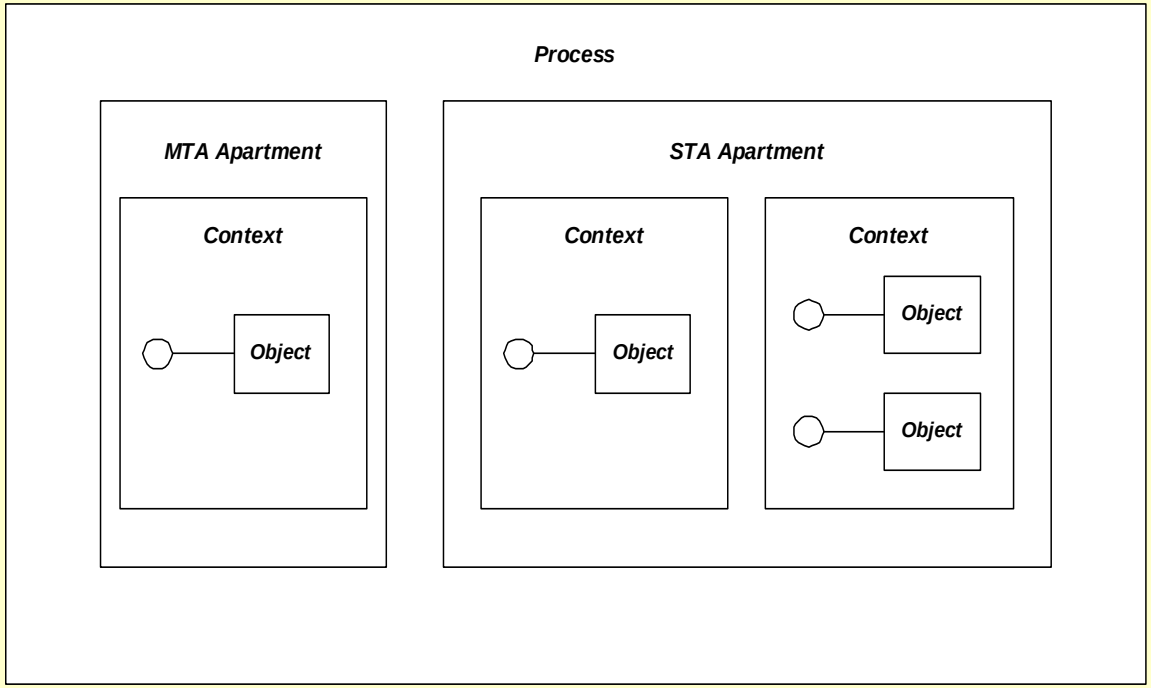
  ".Net does not replace COM+ because .Net does not provide component services! This implementation has always been the intention, and the reason that we had COM+ when Windows 2000 was released is that Microsoft decided that that part of the component framework was ready for release and would be useful to COM but the managed runtime part (what we call .Net) wasn't."

COM+ Architecture

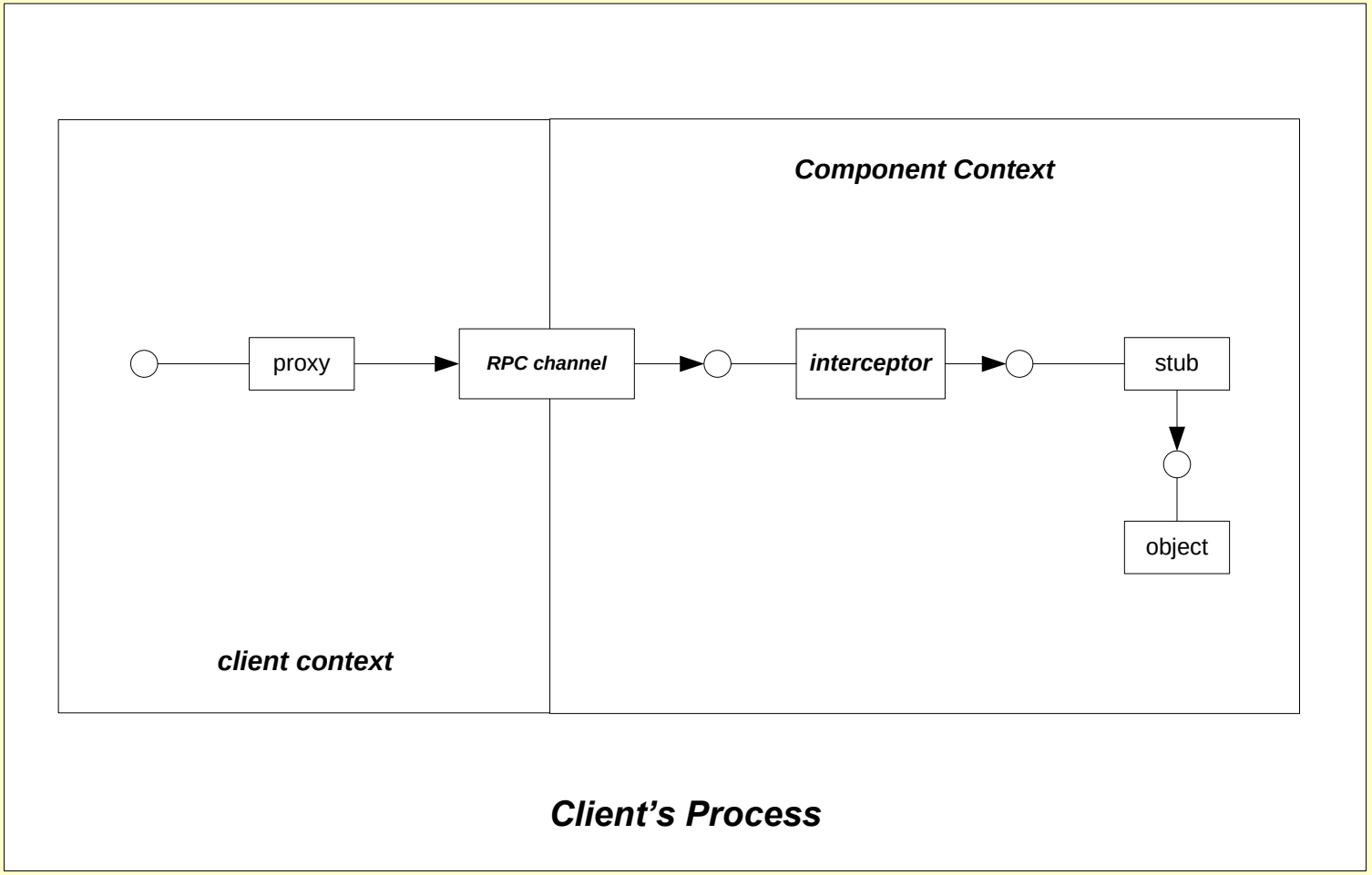
- Attributes
  - Attribute based computation allows the designer or a user to specify attributes that determine how a component behaves.
  - COM+ recognizes attributes for:
    - Transactions, synchroniation, object pooling, JIT activation, events, security, queuing (Gordon, pg 428)
- Applications
  - A COM+ application is a group of one or more components that are administered as a unit and run in the same process.
  - A COM+ application can include components from multiple COM servers.
- Catalog
  - COM+ stores its attribute values in a new database called the Catalog
  - The Component Services Explorer is a visual interface for the COM+ catalog.
- Configured Component
  - To configure a component to use COM+ services you must first add the component's server to a COM+ application.

Context

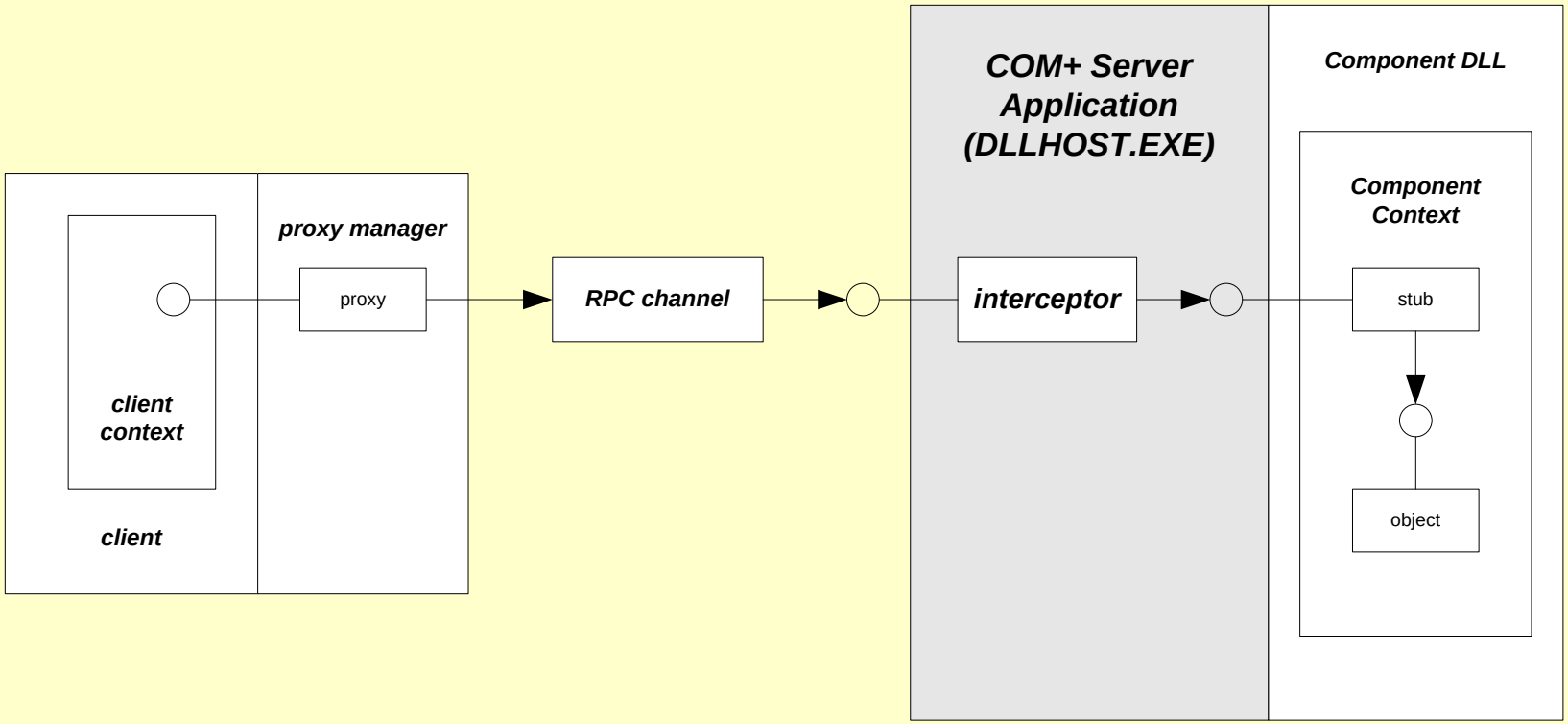
- A COM+ context is the run-time environment in which one or more compatible COM+ objects in a particular process execute.
  - A compatible object is one that shares the runtime requirements specified for the context.
  - All of the objects that reside in a context share the same attribute settings



COM+ Library Application



COM+ Server Application



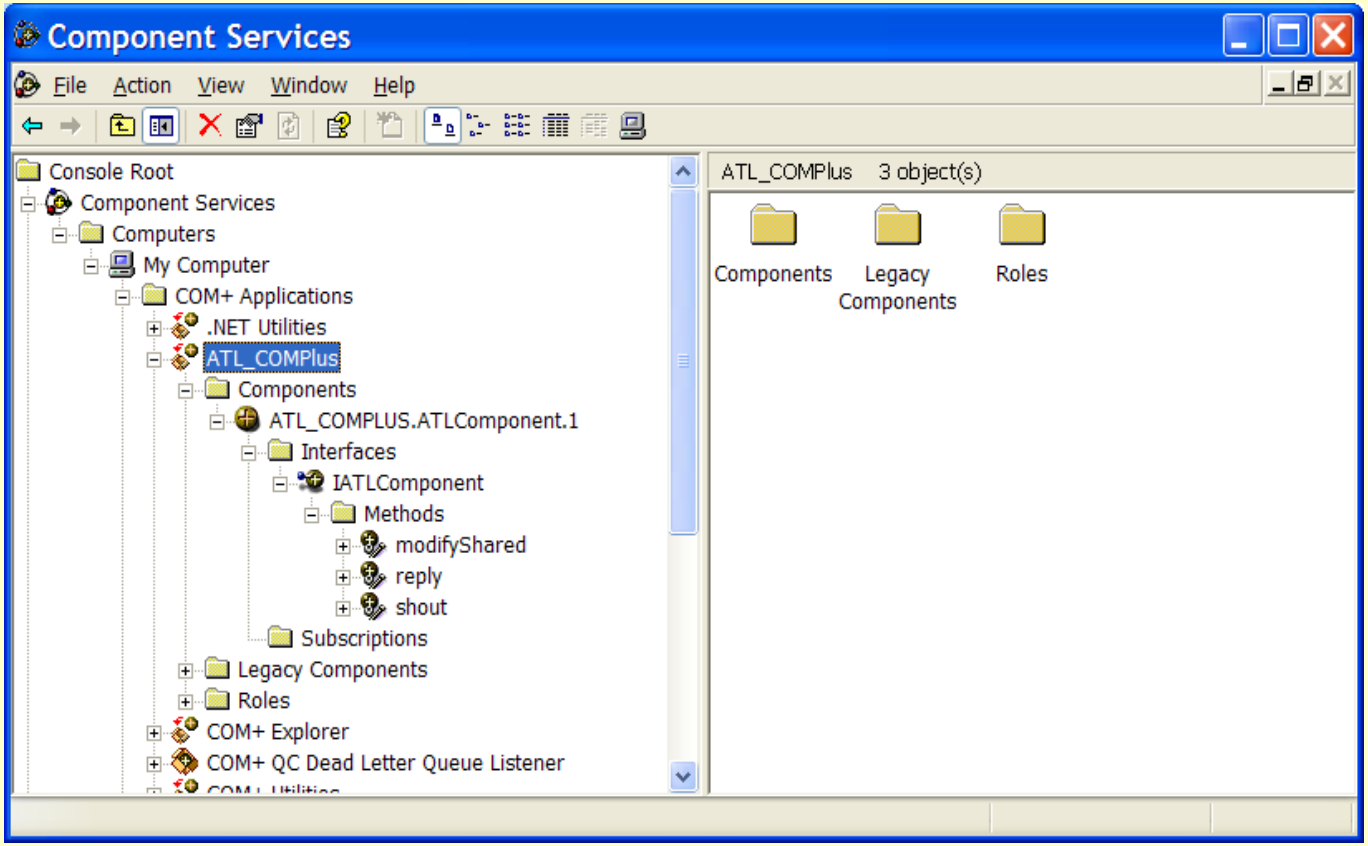
Interception

- If the client of an object configured for COM+ resides in a different context than the object, then a light weight proxy, called an interceptor, is set up between the client and object.
- It is the interceptor that handles transactions, security, and TNA synchronization.
- The nature of these services is determined by the COM+ object's context attributes.
  - The context attributes can be set programmatically or administratively

Context

- Under Windows COM+ partitions a process into contexts.
  - Each context is a collection of objects that share runtime requirements.
  - A process may contain more than one context to separate incompatible objects from one another.
  - Each context in a process has a COM object called the object context (OC).
  - Objects in the context access OC by calling CoGetObjectContext and using the OC interact with the services provided by their context.
  - Proxies are used to allow objects to make calls across context boundaries.
- Each COM+ Application defines a Context.

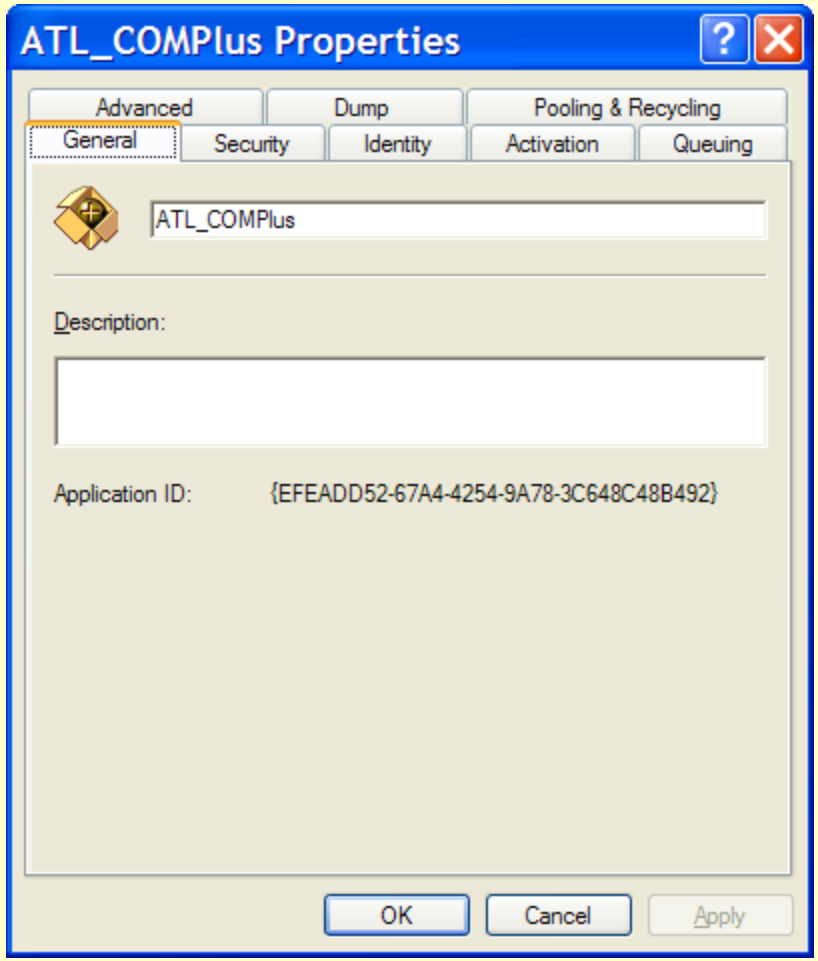
Applications



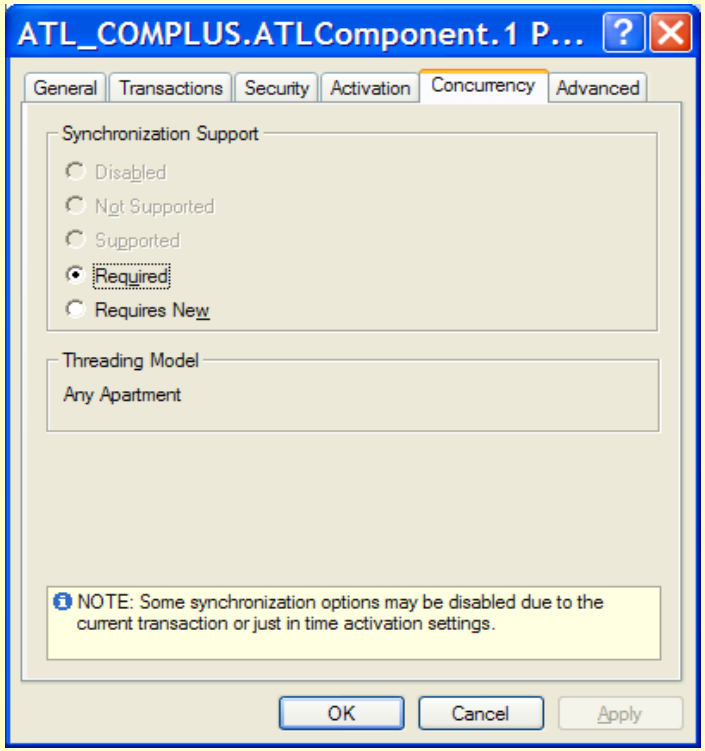
- Applications are Containers for Components

Application Properties

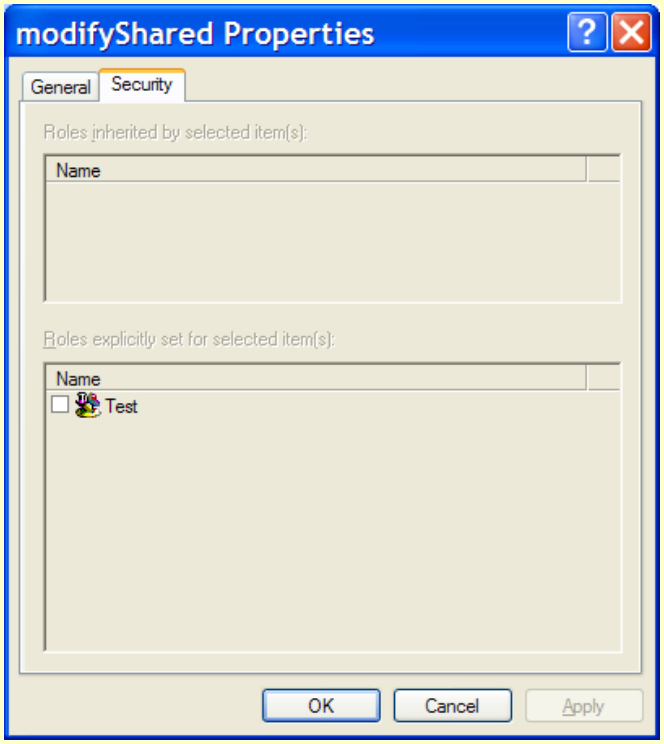
- A COM+ Application is a group of one or more COM+ components that are administered as a group and run in the same process.
- A COM+ Application can include components from multiple COM servers (dlls).
- An application has a set of properties that define the context of the components it contains.



Component Properties



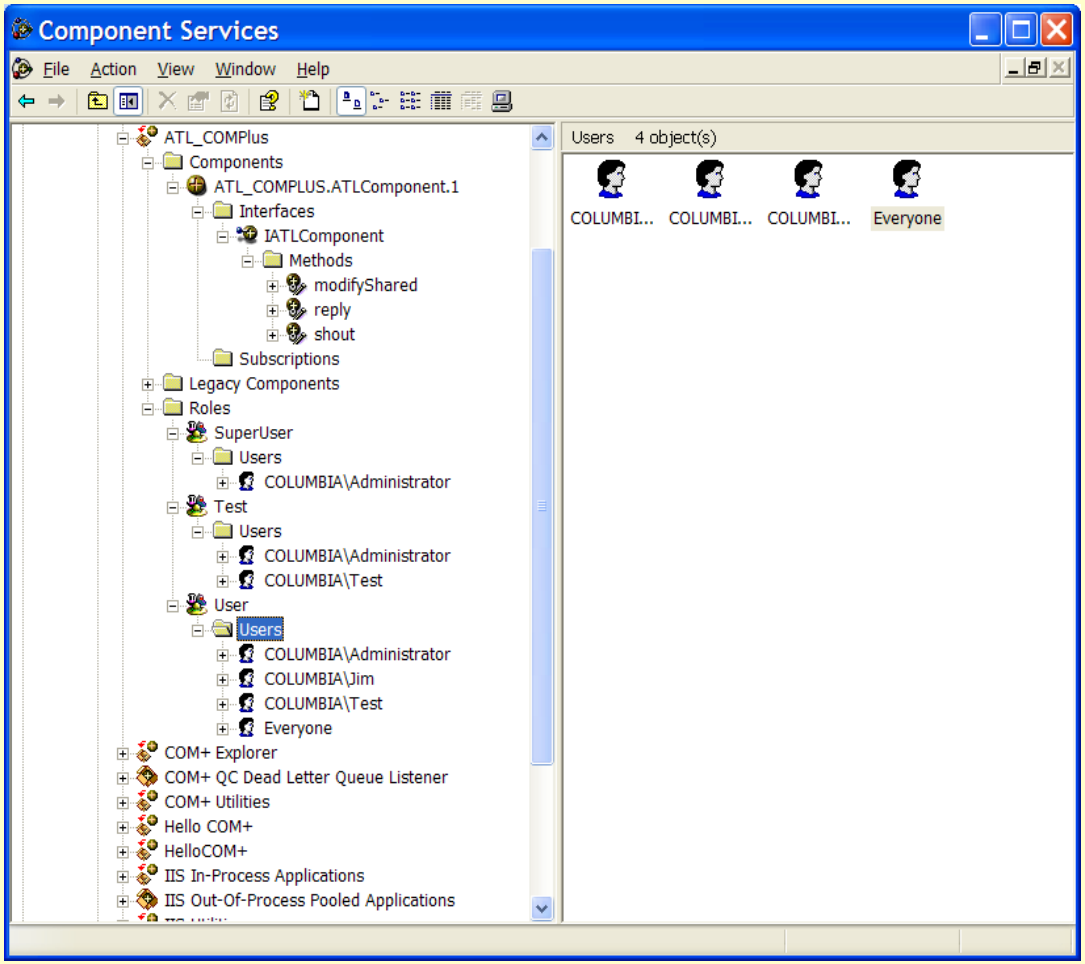
- The properties that can be set for individual components are affected by the Application's property configuration.



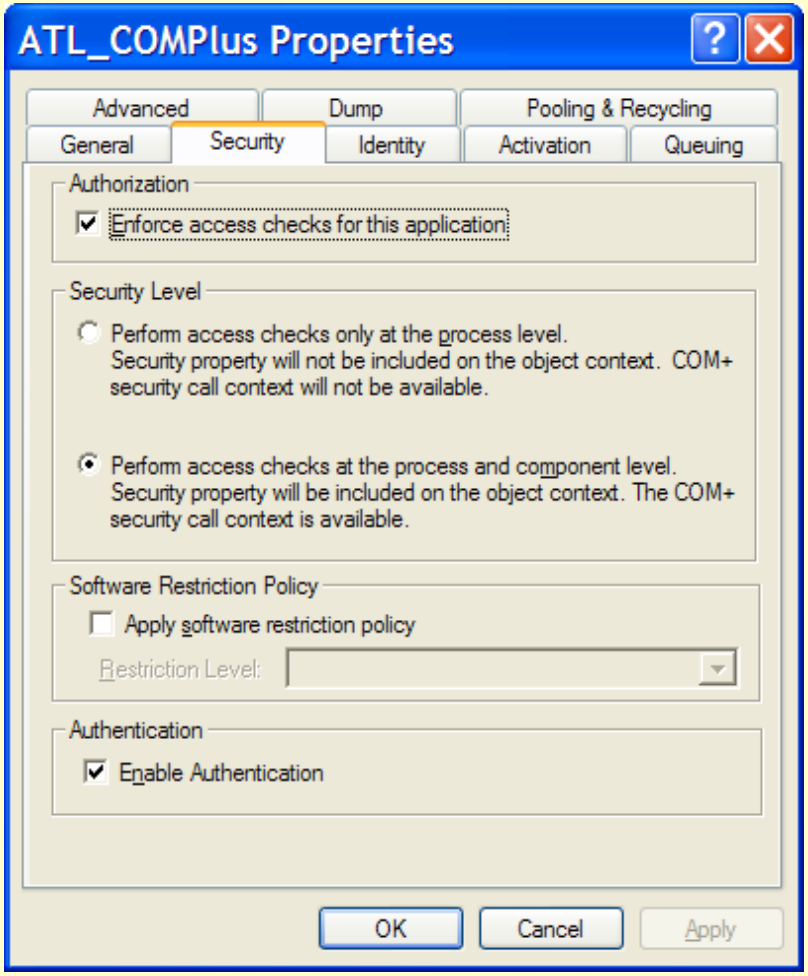
- Individual methods can have security roles applied to them.

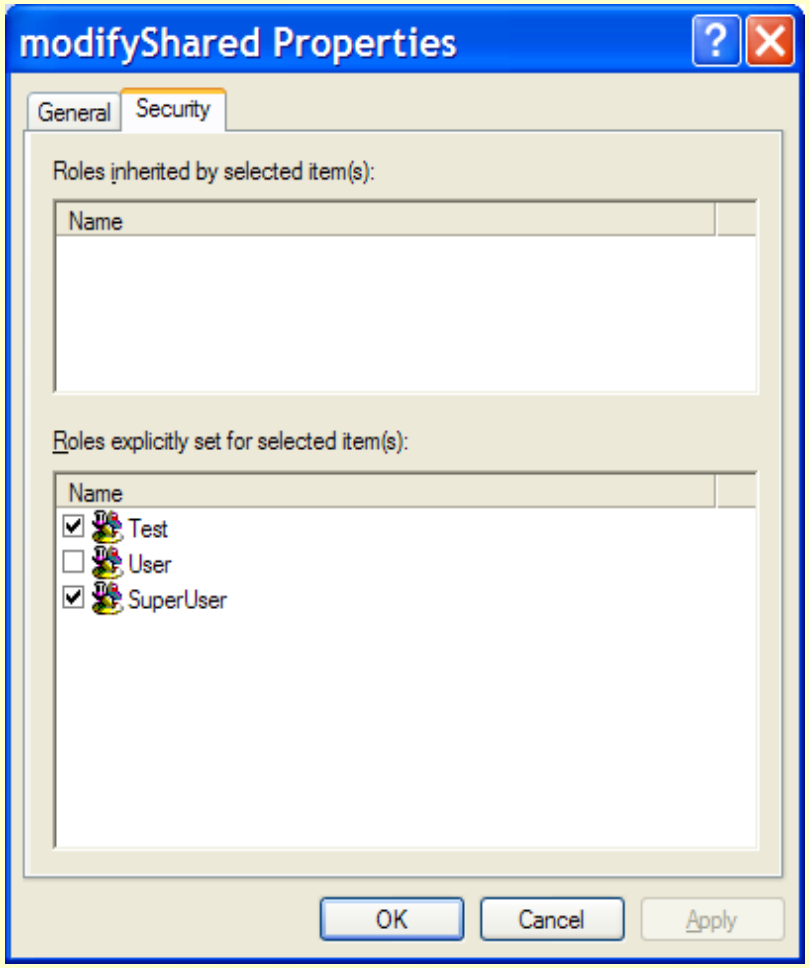
Security

- Define Roles
- Add users to roles
- Select allowed roles for each method

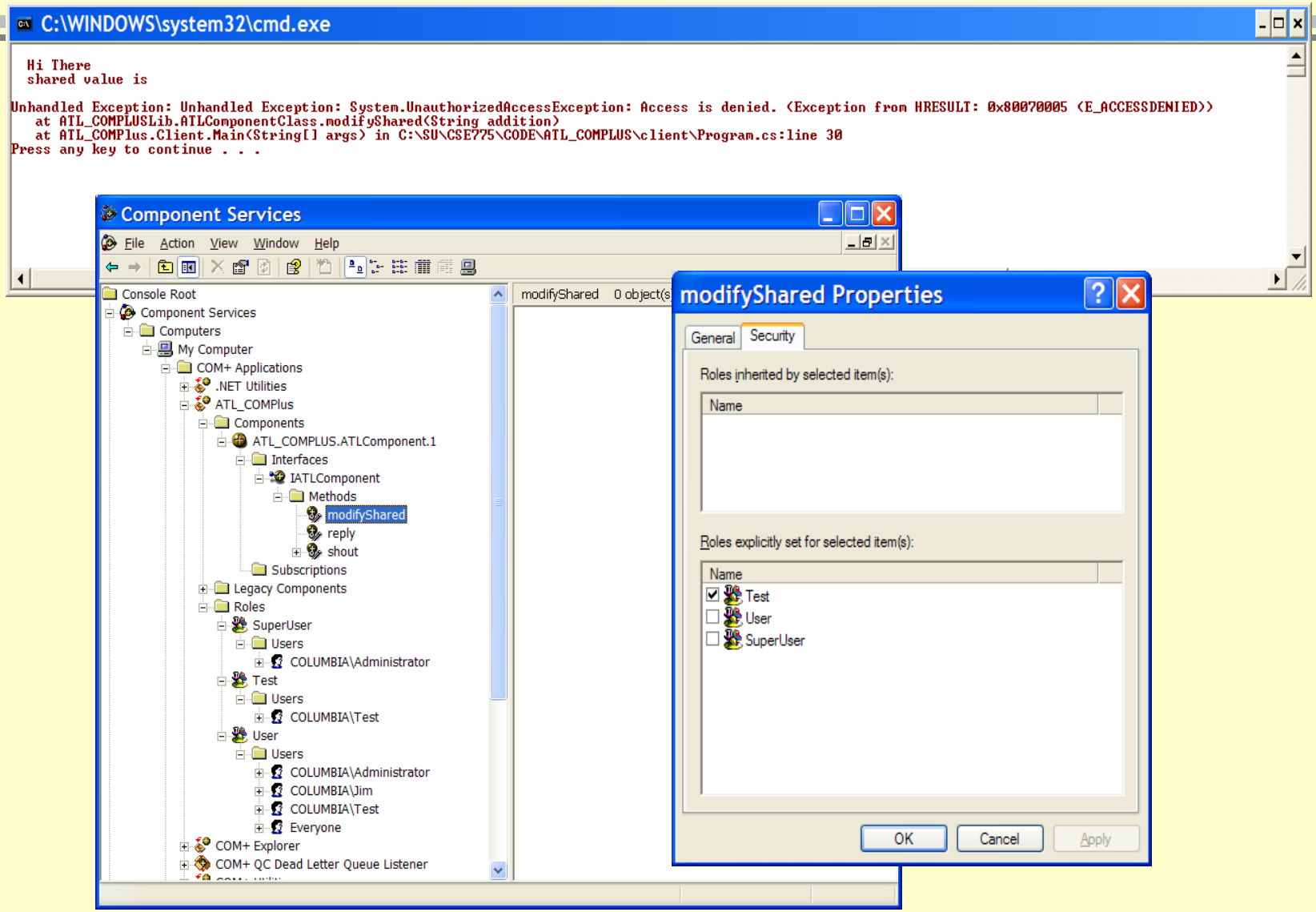


Security

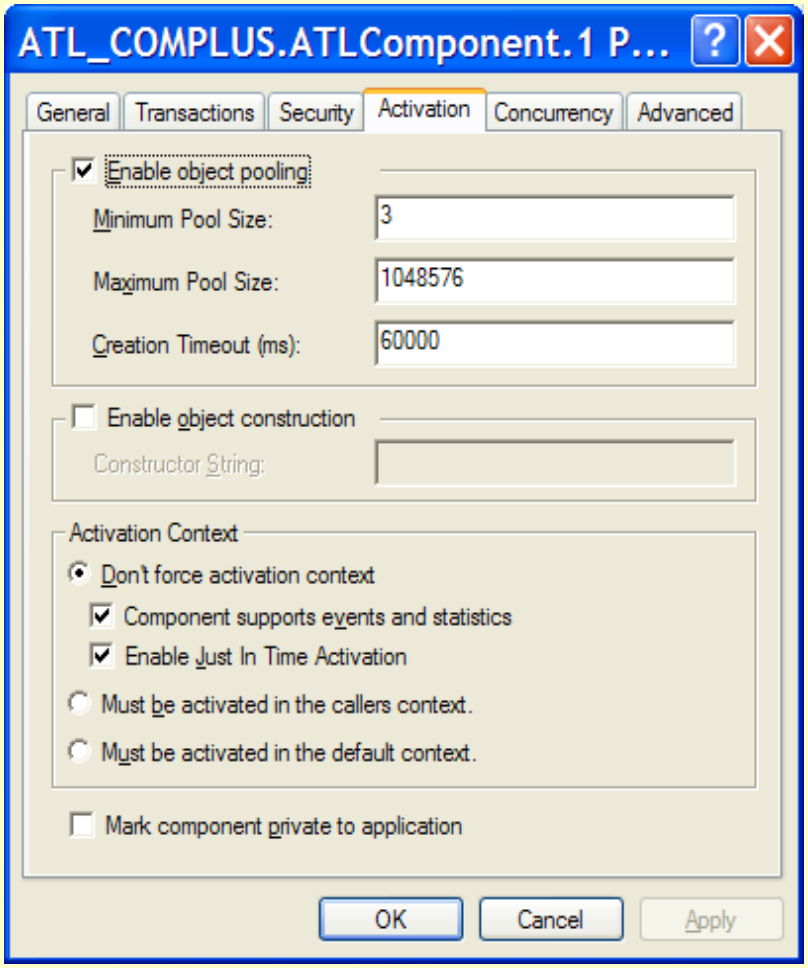

ATL_COMPlus Properties
Advanced
Dump
Pooling & Recycling
General
Security
Identity
Activation
Queuing
Authorization
Enforce access checks for this application
Security Level
Perform access checks only at the process level.
Security property will not be included on the object context. COM+ security call context will not be available.
Perform access checks at the process and component level.
Security property will be included on the object context. The COM+ security call context is available.
Software Restriction Policy
Apply software restriction policy
Restriction Level:
Authentication
Enable Authentication
OK
Cancel
Apply

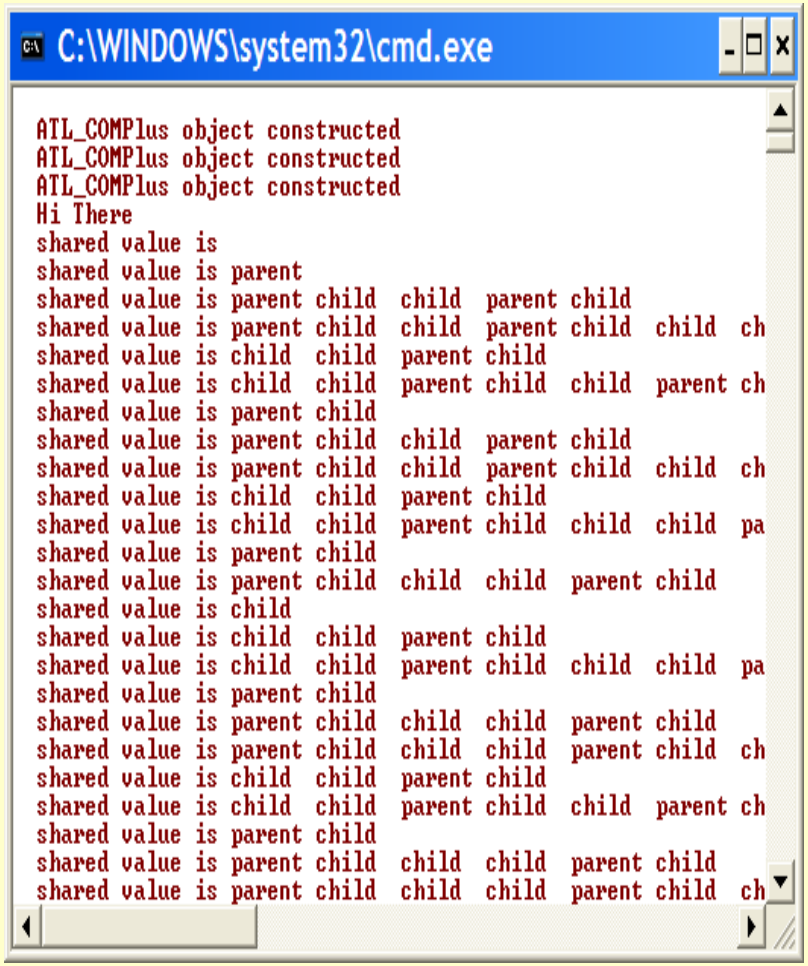

modifyShared Properties
General
Security
Roles inherited by selected item(s):
Name
Roles explicitly set for selected item(s):
Name
Test
User
SuperUser
OK
Cancel
Apply

Unauthorized User

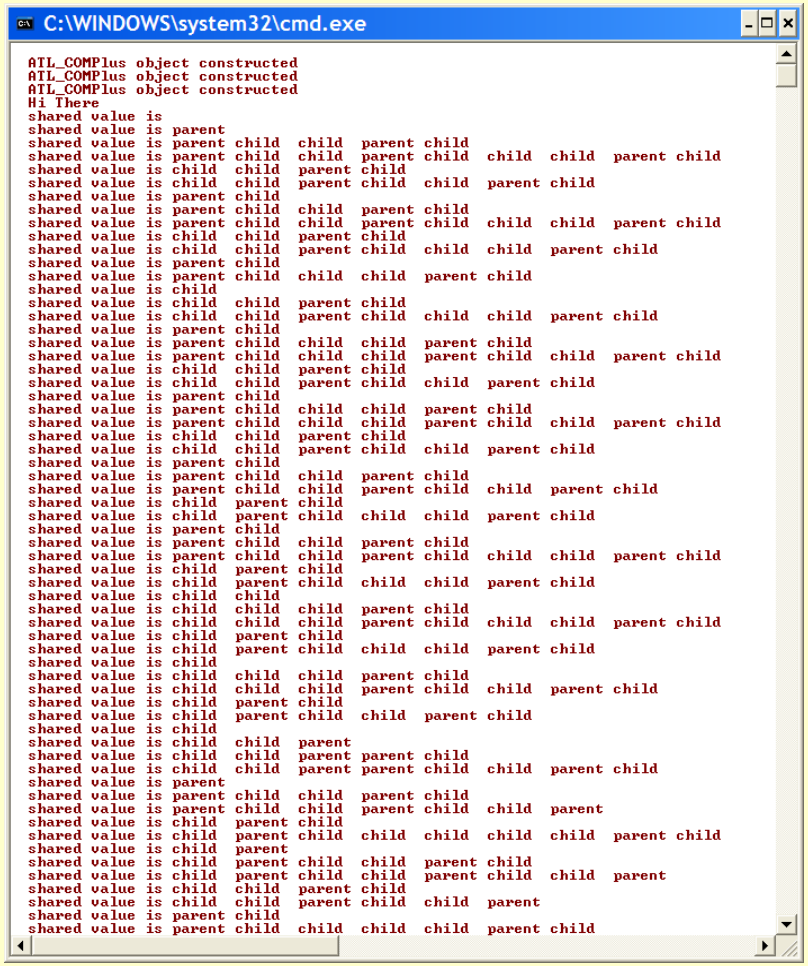


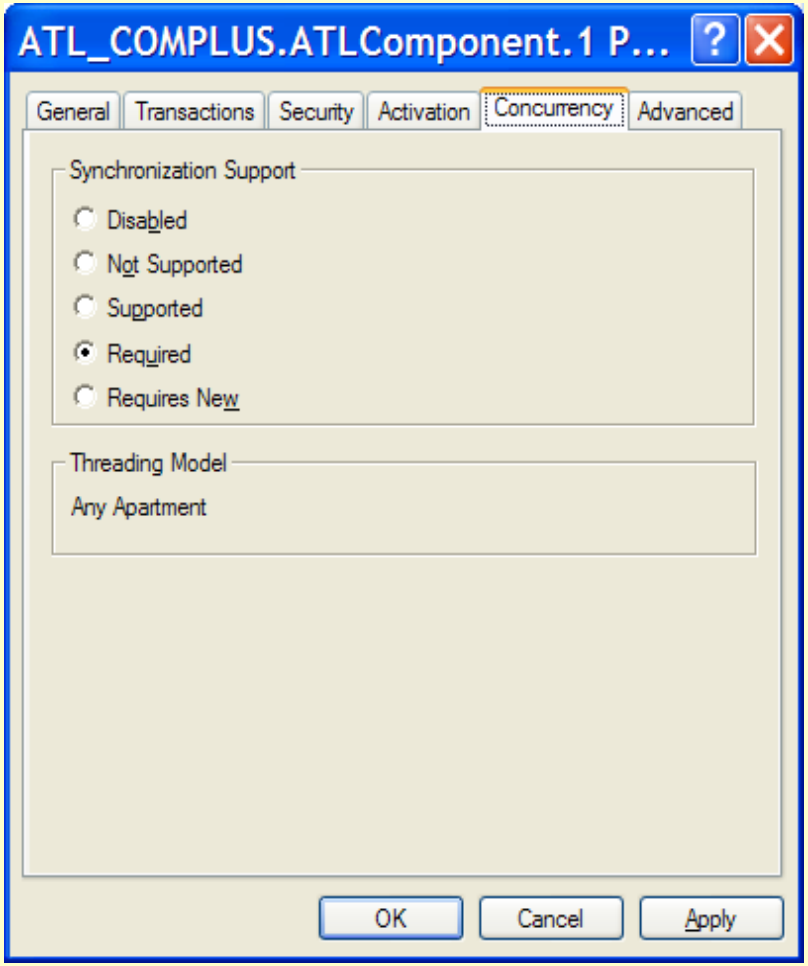
Object Pooling





Concurrency Control

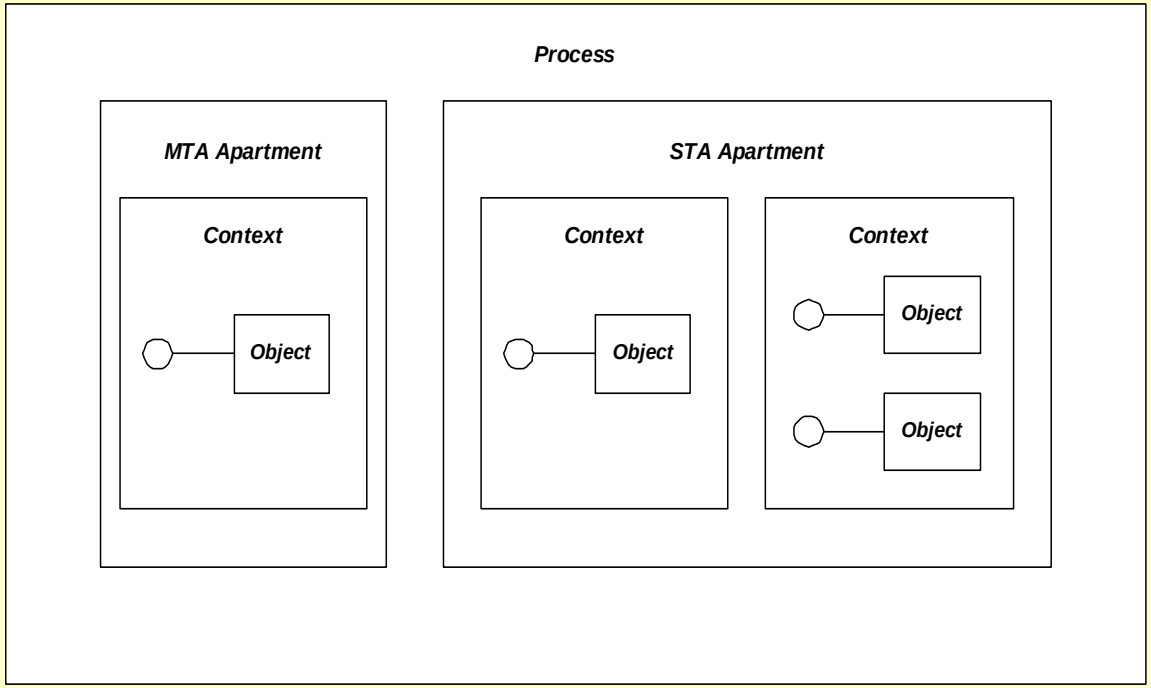




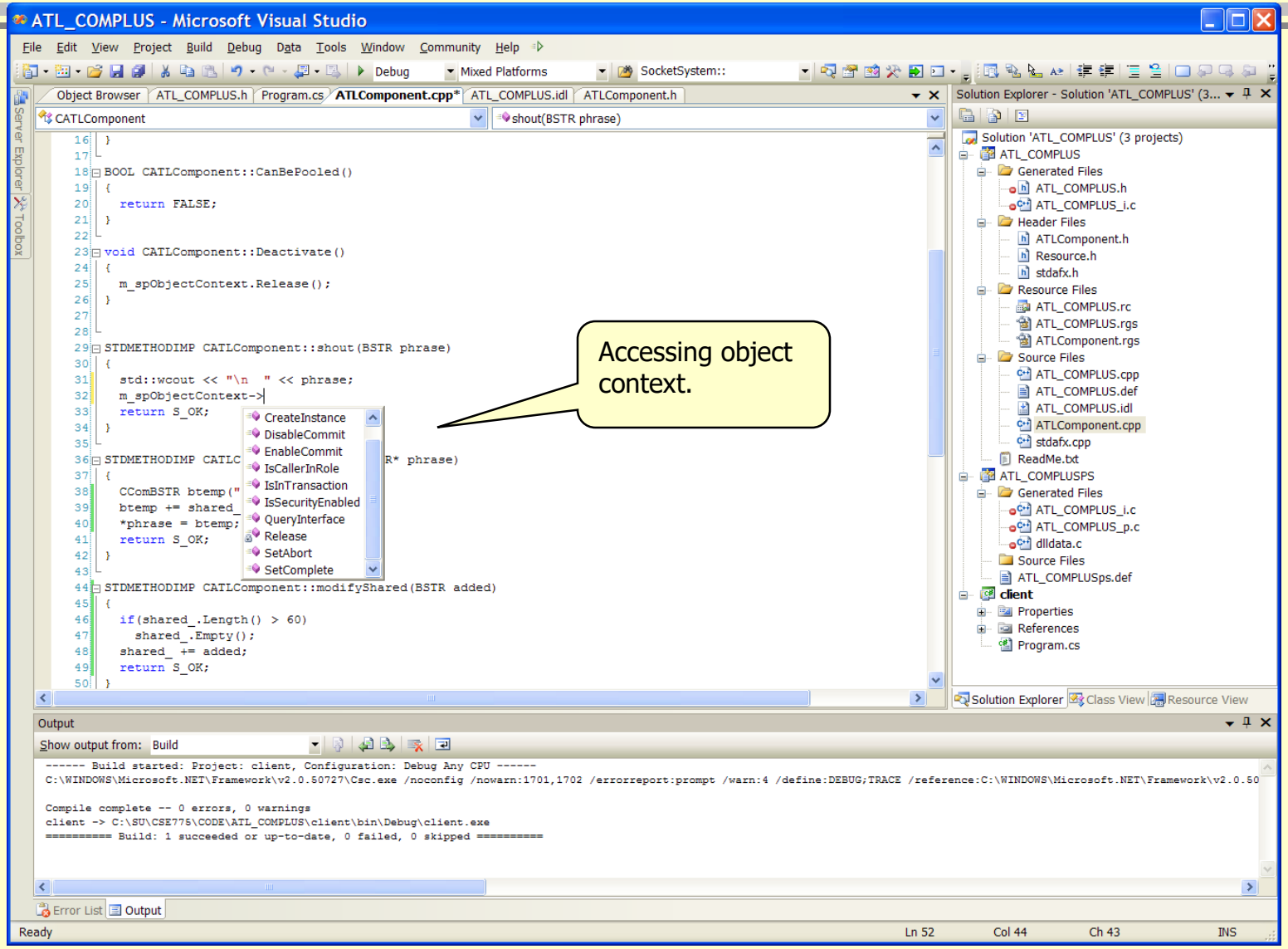
Synchronization via TNA

Context

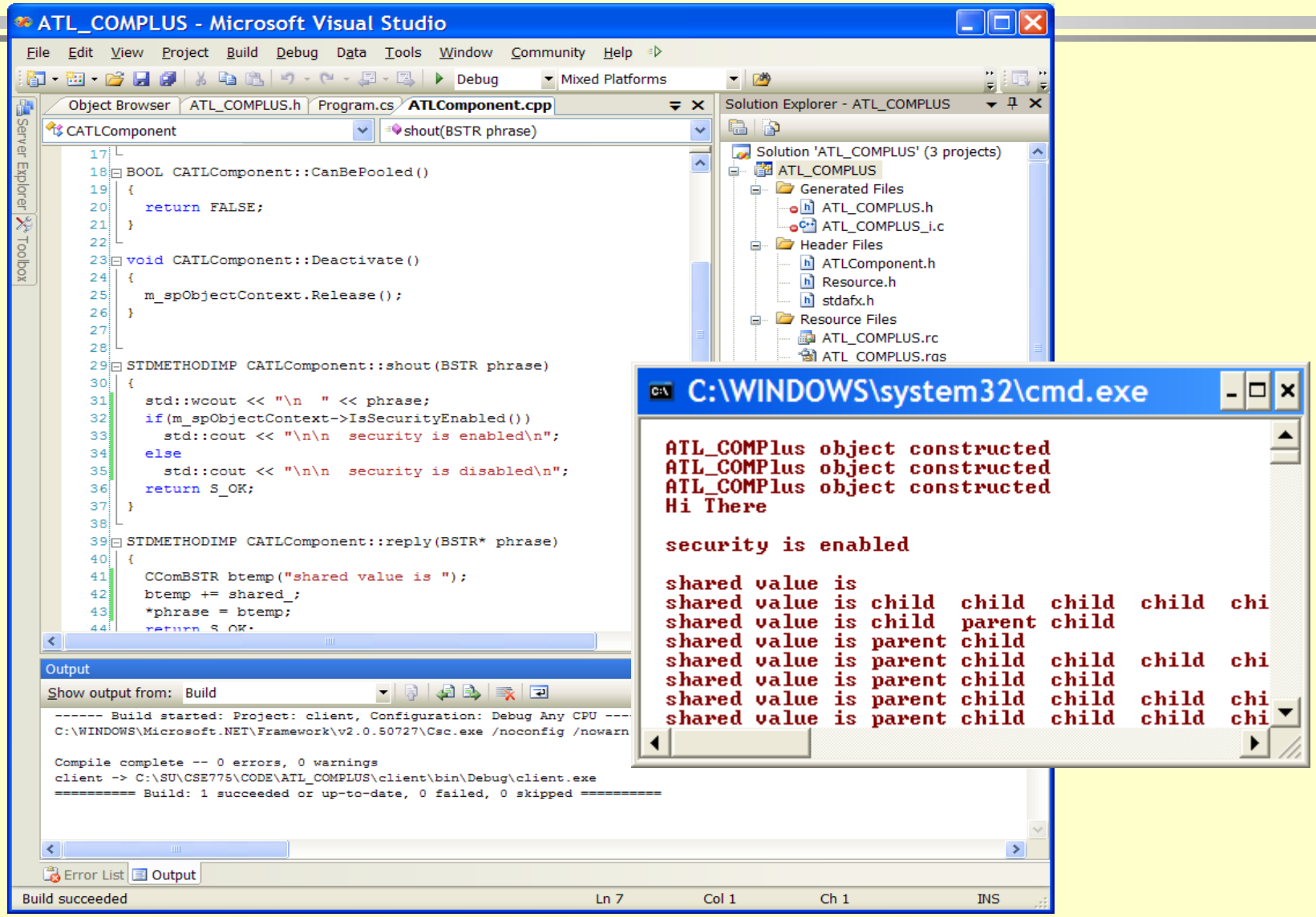
- A COM+ context is the run-time environment in which one or more compatible COM+ objects in a particular process execute.
  - A compatible object is one that shares the runtime requirements specified for the context.
  - All of the objects that reside in a context share the same attribute settings



Programmatic Access to Context



Accessing Context Properties



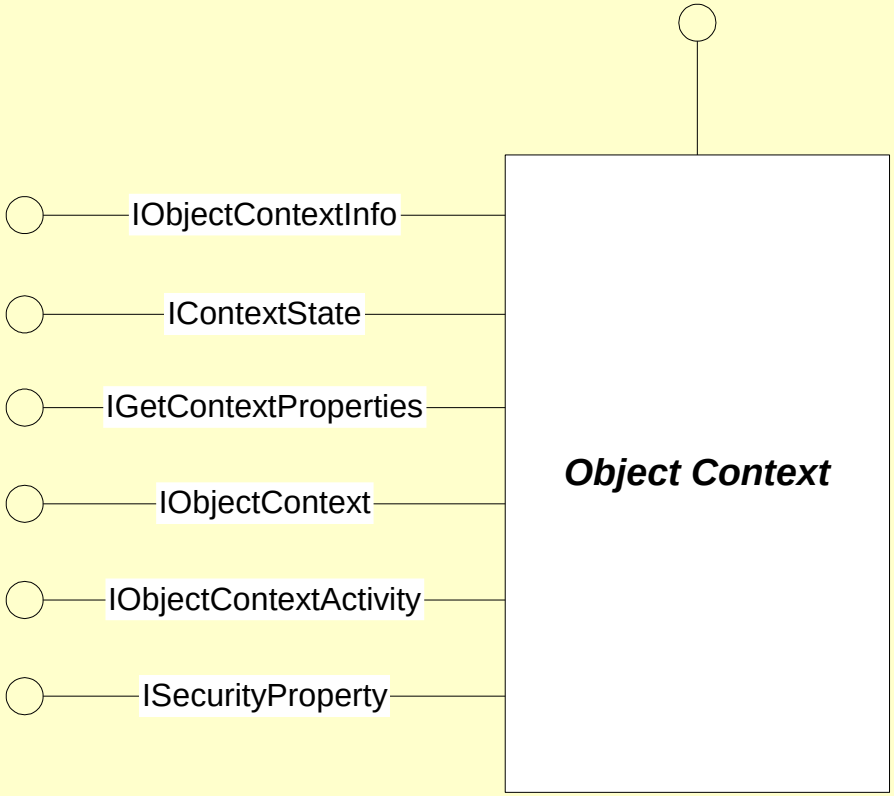
Context and Apartments

- Context determines what service activities are performed for a COM+ configured object.
  - A context belongs to one and only one apartment
  - Communication between contexts results in light-weight marshaling via interceptor
- Apartment determines what marshaling activites are performed for any COM object.
  - An apartment can contain many contexts

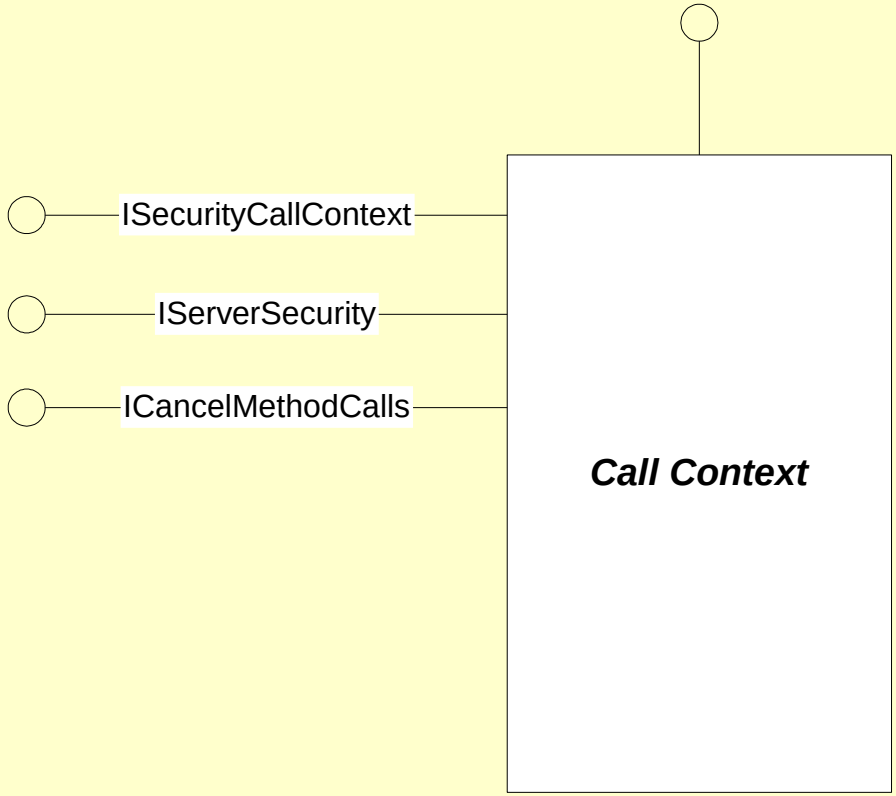
Context

- Under Windows COM+ partitions a process into contexts.
  - Each context is a collection of objects that share runtime requirements.
  - A process may contain more than one context to separate incompatible objects from one another.
  - Each context in a process has a COM object called the object context (OC).
  - Objects in the context access OC by calling CoGetObjectContext and using the OC interact with the services provided by their context.
  - Proxies are used to allow objects to make calls across context boundaries.

HRESULT CoGetObjectContext(REFIID riid, LPVOID **ppv)



HRESULT CoGetCallContext(REFIID riid, LPVOID **ppv)



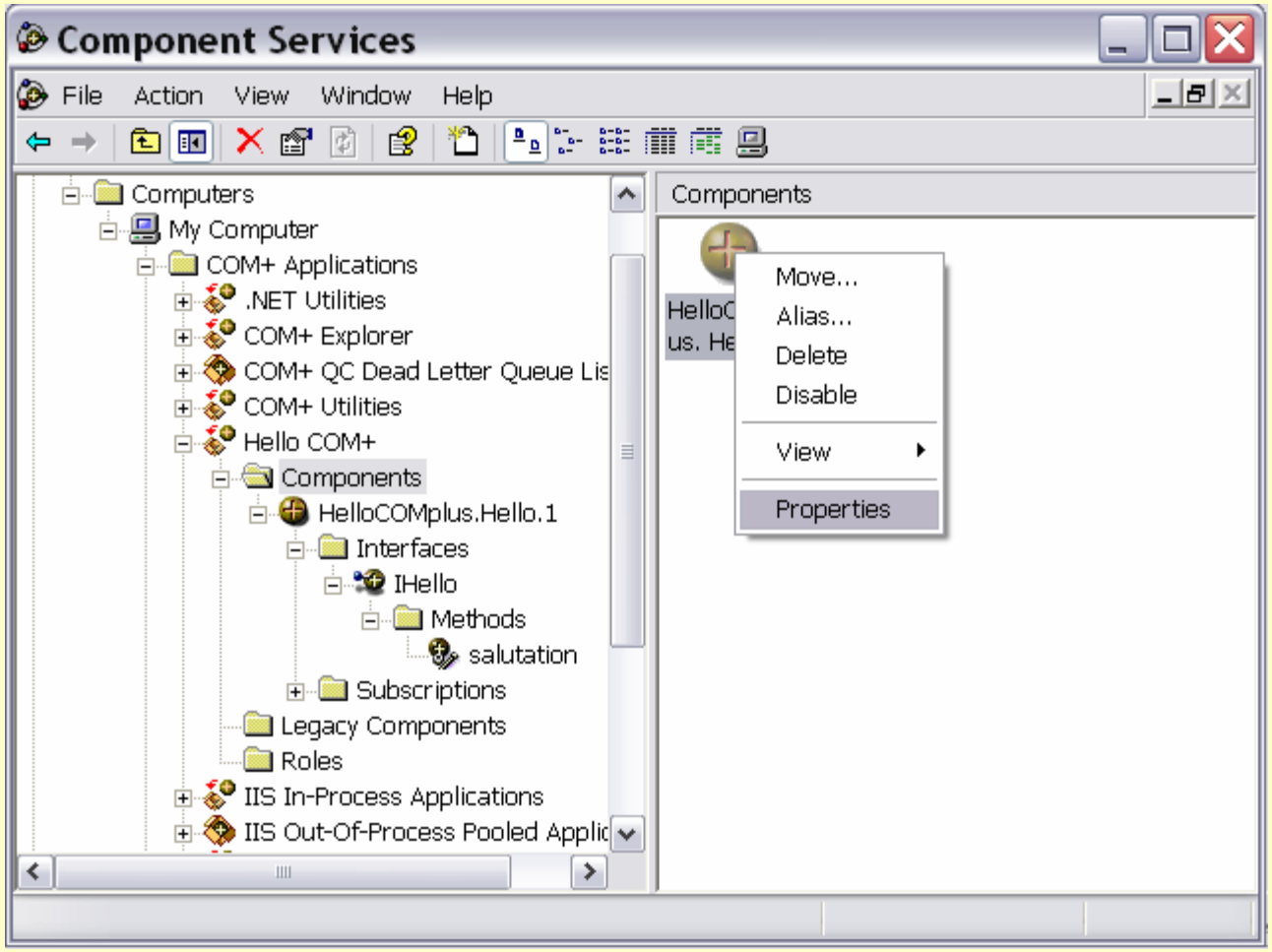
Configuration and Interception

- COM+ has a catalog manager that uses a configuration catalog database (RegDB). It describes services that are carried out by COM+ executive before and after forwarding calls to component.
- Configured services include:
  - transactions
  - syncronization
  - object pooling
  - declarative authorization
  - queueing
- These services are rendered by interception:
  - If needed to provide services COM will provide a proxy that intercepts component method calls to inject configured services.

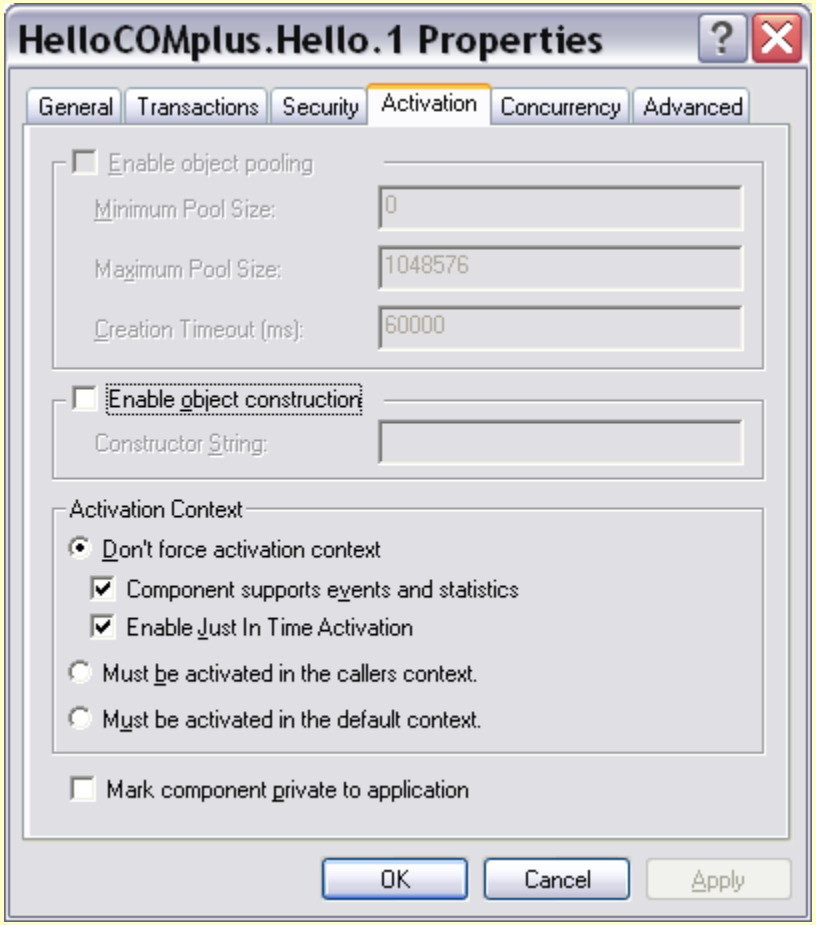
Configured Components

- To build and use a COM+ configured component you must:
  - Create a COM+ application using the Component Services Explorer (Gordon, pg 446).
  - Set context-wide attributes through its properties dialog (Gordon, chap 12).
  - Add the component to this application (Gordon, pg 451).
  - Set context attributes for this class and its methods using Component Services Explorer (Gordon, chap 11).

Administrative View of Configured Component



Configuring Component



Requirements to Use COM+

- Must be part of an NT Domain or Windows Active Directory to use all of COM+ facilities.
- COM+ components must be “configured”.

Requirements for COM+ Components

- Must be In-Proc (dll-based) components.
  - These may run in a client's address space, or hosted by a surrogate process, called dllHost.exe.
- Must be self registering.
  - Provide DllRegisterServer and DllUnregisterServer functions.
- All interfaces and methods must be described in a type library.
  - You can bind the type library into the dll that contains the component.
- Must provide support for marshalling their interfaces.
  - Use typelibrary marshaling via the oleautomation attribute in IDL or provide proxy/stub dll.

Architecture Summary

- COM+ is fully integrated with Windows and COM.
  - Windows 2000, Windows XP, Windows Server 2003
- It is a run-time environment designed to host in-proc COM components.
- COM+ is designed to provide scalable distributed enterprise applications.
- It uses declarative statements to take advantage of transactions, serialization, and security. It does this by using classes with attributes and a context manager.

COM+ Services Summary

- COM+ supports
  - Just In Time (JIT) activation ◆ object not instantiated until client makes a call to an interface method
  - early deactivation ◆ COM+ may deactivate component before client calls final release and transparently reactive when needed
  - pooling of objects ◆ COM+ maintains a pool of activated objects for distribution to incoming clients. This avoids many activations and deactivations.
  - shared properties and database access through ODBC
  - transactions
  - activities
  - security identities

COM+ Transactons

- Satisfy the ACID properties:
  - Atomic: all operations that make up a transaction will succeed or fail as a unit.
  - Consistent: data being operated on during a transaction will continue to be internally self consistent
  - Isolated: concurrent transactions can not see each others partial and uncommitted results.
  - Durable: once committed all updates will persist even in the event of a system failure.

COM+ Security

- COM+ supports a role based security model
  - roles are used to provide declarative authorization that grants or denies specific permissions
  - the system administrator can bind roles defined in a component to specific users and user groups

- COM+ also supports programmatic security
  - programmatic security is defined by interfaces and code used to determine proper access to a component through the
    IObjectContext::IsSecurityEnabled and
    IObjectContext::IsCallerInRole
    methods and the ISecurityProperty interface.

Synchronization

- Starting with Windows 2000 COM has a new Thread Neutral Apartment (TNA).
  - Each process has at most one TNA.
  - All incoming calls to a TNA are serviced by the caller's thread (so there will be no windows message loop bottleneck).
  - ThreadingModel=Neutral is the preferred model for components with no user interface.
  - User interfaces should still be housed in STAs.
  - The combination of ThreadingModel=Neutral and Synchronization=Required is equivalent to the rental model where any thread can call into the object, but only one thread at a time.

Synchronization (continued)

- The Single Threaded Apartment (STA) is no longer the primary means of serializing calls to an object (it is still necessary for user interface code).

- All COM components can now use activities to control concurrent access. Using this approach:
  - a thread switch is no longer required
  - the windows message queue is no longer used for serialization

Activity

- An activity is a collection of one or more contexts that share a concurrency model.
- Activities are used to enforce call synchronization.
- Contexts that do not belong to an activity get no call serialization. Any thread in the context's apartment can enter the context at any time.

Contexts vs. Activities

- A context is a set of objects that share common run-time attributes.
- A context is the unit of interception. Calls between incompatible contexts are marshalled through proxies.
- An activity is a collection of one or more contexts that share common synchronization requirements. COM allocates a process-wide lock to each activity.
- Under COM+ clients don't share objects. They are based on private objects that may share common state protected by transactions.

Transaction Management

- Transactions are managed by the Distributed Transaction Coordinator (DTC).
- The DTC is responsible for coordination of transaction outcome.
  - it is responsible for maintaining the ACID properties
  - it uses a lock management strategy based on transaction-affinity locks.
  - When a lock is held by a transaction resource like a database manager it can be reentered from anywhere in the transaction since all objects in the transaction stream share a single DTC transaction.
- Transaction aware resources like ODBC and Microsoft Message Queue can access the context's transaction when an object requests service.

Transaction Streams

- A transaction stream is a collection of contexts in space (across process and machine boundaries) that share a transaction.
- A transaction stream is completely contained inside an activity.
- All objects inside a transaction stream share access to a single transaction in time.
- COM will automatically start a new transaction if the stream's previous transaction has ended.

Transaction Failures

- Each context in a transaction stream keeps track of whether the its objects are satisfied with the current state of the transaction. Each object can set or clear a "happy" state by calling IContextState::SetMyTransactionVote.
  - A transaction can commit only if all contexts in the transaction stream are satisfied.
  - If one or more contexts are "unhappy" when the transaction root deactivates the transaction will be aborted and any operations that are protected by the transaction will be rolled back.
  - When an object returns control with its unhappy state set and its done state set this tells COM it has detected an unrecoverable error and the transaction is doomed to failure.

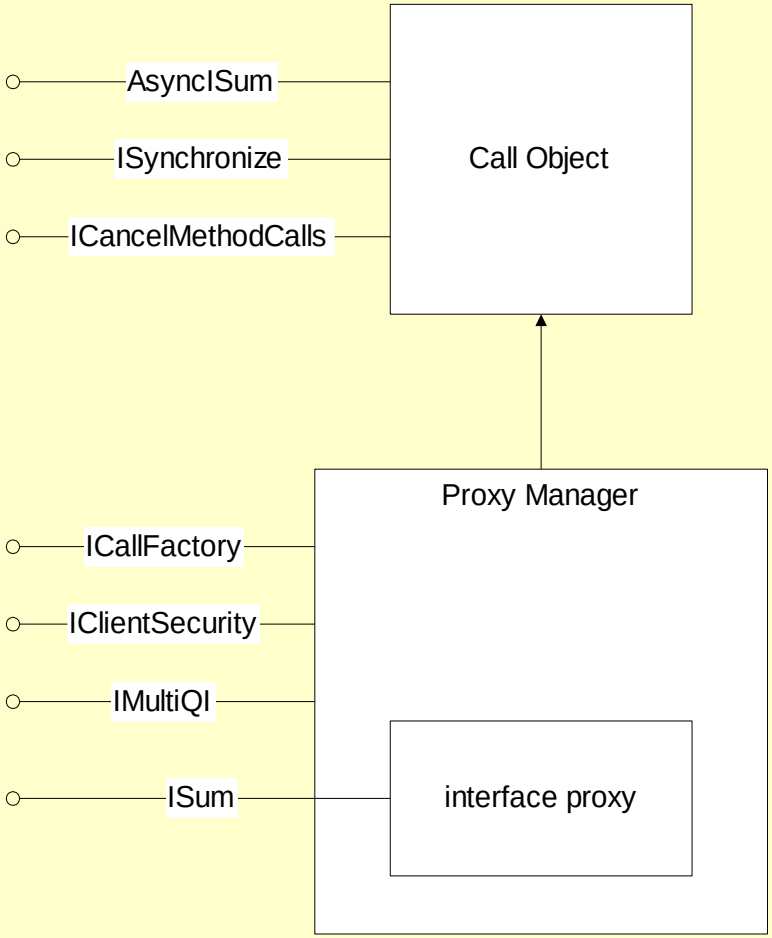
Non-Blocking Method Invocation

- An invocation of a method now has a COM object associated with it to give the client more control
- Client-side threads can now issue asynchronous calls and regain control immediately - they do not block waiting for the method to return.
- On the server side objects can free up the RPC thread used to invoke the method to allow more concurrent calls to be serviced.
- The client and server can elect to use non-blocking invocation independently.
- For this to work the interface is annotated in IDL as supporting non-blocking invocation using the [async_uuid] attribute.

Asynchronous Calls

- Interfaces supporting asynchronous calls provide the usual synchronous method, Sum(…) for example.
- They also provide two additional methods:
  - Begin_Sum(…) uses the [in] parameters
  - End_Sum(…) uses the [out] parameters
- The standard proxy manager implements the ICallFactory interface to allow the caller to create a call object that implements the asynchronous version of the interface.
- Clients can either:
  - poll the object for completion
  - implement the ISynchronize interface to allow the call object to signal completion.

Call Objects



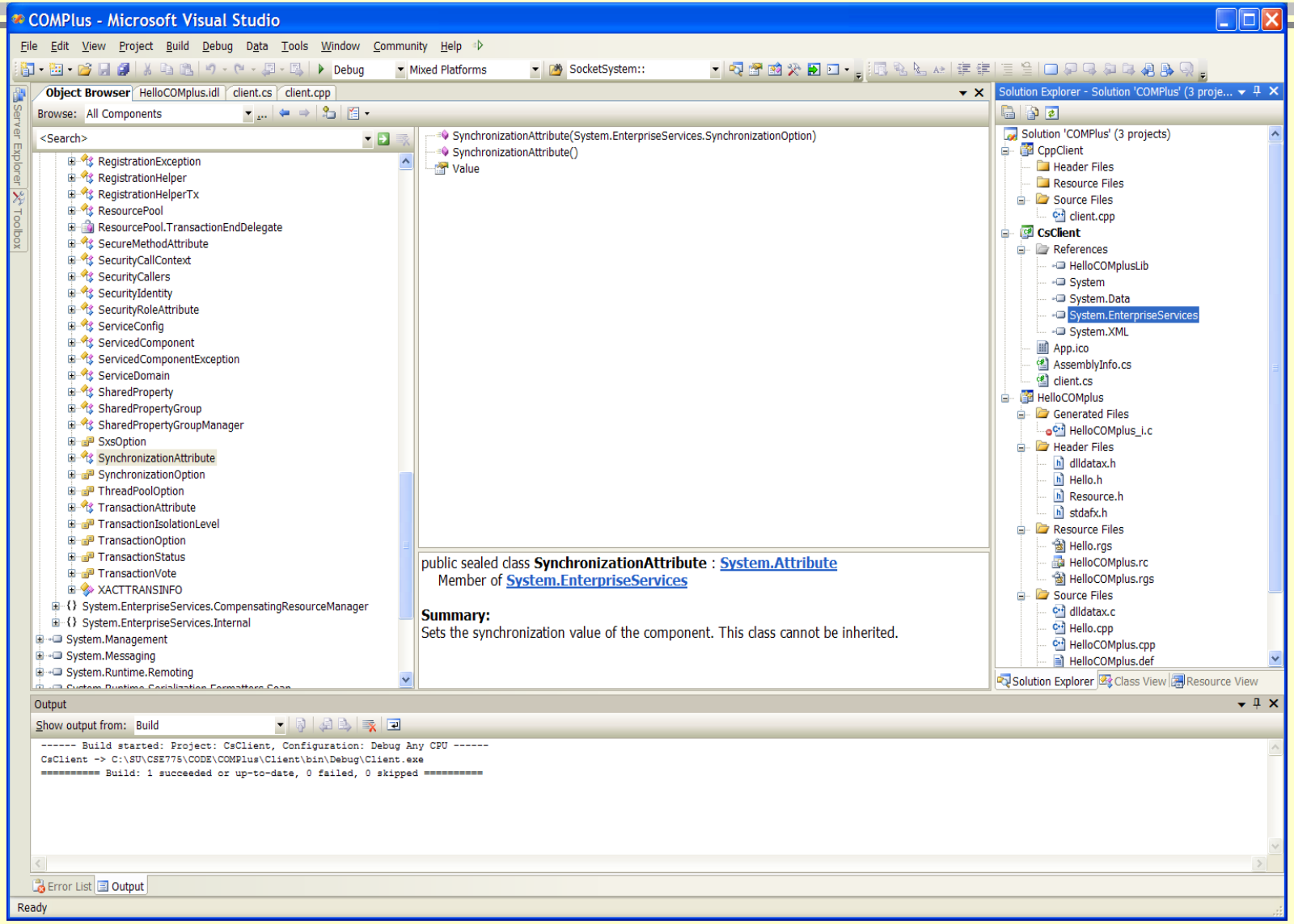
Integration with MSMQ

- Starting with Windows 2000 a class can be configured to support transparent queuing using Microsoft Message Queue (MSMQ).
- This allows the client to create a queue aware proxy that buffers all calls until the object is released.
- Once released the proxy uses MSMQ to send a message to the server which then replays the method calls issued by the client.
- MSMQ supports guaranteed delivery over unreliable network links.

Other Features

- COM+, under Windows 2000, supports:
  - a new variant of marshalling between standard and custom marshalling to allow components to create client side handlers that do some work before method invocation
  - COM supports pipes to facilitate bulk transfer of data within a method call.
  - improvements to the security model
    - better control over when caller's tokens are inspected
    - supports delegated trusts
  - activation-time load balancing by picking lightly loaded remote machines for activation
  - object pooling

Relationship with .Net



End of Presentation