Microsoft Transaction Server Readme

Welcome to Microsoft Transaction Server (MTS), a powerful environment that makes it easier to develop and deploy high-performance, scalable, and robust enterprise, Internet, and intranet applications. Microsoft Transaction Server provides an application programming model and a run-time infrastructure for developing and deploying distributed, component-based applications.

This Readme file lists late-breaking feature information not contained in the regular documentation, known limitations of this release of MTS, and information on reporting bugs or problems with the product.

NOTE: The documentation contained in the Microsoft Transaction Server 2.0 SP1 Help file supersedes the MTS documentation in Windows NT Option Pack.

Late-Breaking Information and Known Limitations

The following sections contain late-breaking information that is not covered in the regular documentation, as well as known bugs and limitations.

Accessing Property Groups in LockMethod Mode

One method call can not simultaneously access more than one Property Group in LockMethod mode. An attempt to access multiple property groups results in an application process failure (failfast) with the following message:

> An attempt was made to open an SPM Property Group in LockMethod mode, by an object without context, or by an object that has already opened another Property Group in LockMethod mode.

To allow an application to lock more than one property group, select one property group as the main property group of a set of property groups. The main property group can uses mode LockMethod, and the other property groups use mode LockSetGet. If the application obtains a lock on the main property group first, then the whole set of property groups is effectively locked.

SecurityProperty and ObjectContext Interfaces Provided for Visual C++ Developers

Visual C++ developers can now call the **SecurityProperty** and **ObjectContext** interfaces. The SecurityProperty interface is used to determine the current object's caller or creator. The ObjectContext interface allows you to do the following:

- Declare that the object's work is complete.
- Prevent a transaction from being committed, either temporarily or permanently.
- Instantiate other MTS objects and include their work within the scope of the current object's transaction.
- Find out if a caller is in a particular role.
- Find out if security is enabled.
- Find out if the object is executing within a transaction.
- Retrieve Microsoft Internet Information Server (IIS) built-in objects.

Although these interfaces are not documented in the MTS Reference, the Visual Basic objects and methods provide the same functionality. See the Visual Basic section of the MTS Reference in the Programmer's Guide for **SecurityProperty** and **ObjectContext** object and method descriptions. Additionally, Java components can access the **SecurityProperty** methods through the new Java **Context** class. For complete documentation of the **Context** class, see the Visual J++ section of the MTS Programmer's Reference.

Oracle Support Not Available on Alpha Platforms

MTS components cannot participate in transactions with an Oracle database on Alpha platforms.

New Java Context Class

If you are building applications using Visual J++, use the new Context class instead of **IContextObject**. The **Context** class allows you to do the following using Visual J++:

- Declare that the object's work is complete.
- Prevent a transaction from being committed, either temporarily or permanently.
- Instantiate other MTS objects and include their work within the scope of the current object's transaction.
- Find out if a caller is in a particular role.

- Find out if security is enabled.
- Find out if the object is executing within a transaction.
- Access the **SecurityProperty** methods.

See the Visual J++ section of the Programmer's Reference for complete documentation of the new class.

Using ErrorInfo with ADO

ADO erases any information that you might have put in the ErrorInfo object. There is a code work-around for this problem that needs to be implemented in your error handler.

For Visual Basic, the code is similar to the following:

```
ErrorHandler:
    ' cleanup
    If Not adoRS Is Nothing Then
        Set adoRS = Nothing
    End If
    If Not adoConn Is Nothing Then
        Set adoConn = Nothing
    End If

    Err.Raise Err.Number, "Bank.Accout.Post", Err.Description

Exit Function
```

For Visual C++, the code is similar to the following:

```
//
// ErrorInfo is saved here because the following
// ADO cleanup code may clear it.
//
IErrorInfo * pErrorInfo = NULL;
GetErrorInfo(NULL, &pErrorInfo);

if (adoRsBalance) adoRsBalance->Release();
if (adoCoConnection) adoCoConnection->Release();

AtlReportError( CLSID_CAccount, pErrMsg, IID_IAccount, hr);

//
// put the error back in TLS
//
SetErrorInfo(NULL, pErrorInfo);
```

For Visual J++, the code is similar to the following:

```
if (adoRsBalance != null) {
   if (adoRsBalance.getState() == ObjectStateEnum.adStateOpen)
      adoRsBalance.Close();
   ComLib.release (adoRsBalance);
}

if (adoConn != null) {
   if (adoConn.getState() == ObjectStateEnum.adStateOpen)
      adoConn.Close();
   ComLib.release (adoConn);
}
```

Note In Java you must explicitly close recordsets and connections, as well as explicitly releasing the ADO objects.

Components Using RDO 2.0 on Multiprocessor Computers May Experience Access Violations

Components that use RDO 2.0 and are accessed concurrently by multiple clients may experience access violations. This has been observed on servers that have multiple processors. This problem has not been observed for components that use ADO.

Java Sample Bank Does Not Compile on Windows 95

To compile the Java Sample Bank components using Visual J++ 1.1 and earlier, on Windows 95, run midl.exe on the Account.idl file, then run SetJavaDev.bat. If you are using Visual J++ 6.0, see the next topic.

Building the Account.VJ Sample in Visual J++ 6.0

The batch file SetJavaDev.bat is used to create the appropriate type library and wrapper classes to build the Account.VJ sample, and works with Visual J++ 1.1 and earlier. If you are using Visual J++ 6.0 to build the Account.VJ sample, you must run the following commands instead:

```
midl AccountLib.idl
jactivex /javatlb /d . AccountLib.tlb
jactivex /javatlb /d . "%SystemDrive%\Program Files\Common
    Files\System\ADO\msado15.dll"
```

The jactivex application is located in the VJ98 directory and may not be in your path.

Known Problem with Mtx.exe Shutdown When Using Visual Basic 4.0 Components

When an Mtx.exe process running Visual Basic 4.0 components is "Shutdown because idle for *x* minutes", there is a known problem that results in a Read Access Violation. Currently there is no known solution to this problem. It is thought that if more than two Visual Basic 4.0 components are used in a process it is more likely to happen. One workaround is to set your package to "Leave running when idle". This is only a problem when using components built with Visual Basic 4.0 -- no such problem exists with Visual Basic 5.0.

Avoiding Visual Basic 4.0/RDO Deadlocks

If your server component uses RDO, never let an **rdoConnection** with an active **rdoResultset** fall out of scope or a deadlock may result. You must manually close the **rdoResultset** or **rdoConnection** to avoid the deadlock. The following sequence produces a deadlock:

1. Create an instance of the server component. Call a method, which creates **rdoConnection** and then **rdoResultset**. You may fetch from, but do not exhaust or close the **rdoResultset**.
2. Release this instance. (As this connection and resultset go out of scope, the ODBC calls required are not immediately made. Instead, a message is posted to a hidden window to do the **SQLFreeStmt+SQLDisconnect+SQLFreeConnect**).
3. Create a second instance of the server component. Create an **rdoConnection** and attempt to update a row, which was included in the resultset from Step 1 above. This is now a deadlock because the locks held for the resultset created in Step 1 have not been released. And the message pump which will process the message posted in Step 2 above will now never have a chance to run.

The deadlock is avoided by manually closing the resultset (or connection) at the end of Step 1, which immediately makes the ODBC calls to close the statement (and connection). The root of the problem is the delay introduced by the message posted to the hidden window to do the ODBC closes.

Unresolved Externals on MTS-specific GUIDs

If you are getting linker errors because IID_IGetContextProperties or other MTS-specific GUID definitions are not found, it is because the compiler is using an obsolete version of the MTS include and library files, which does not include these definitions. Verify that your compiler path lists the MTS include and library directories before the include and lib directories of your compiler.

Updating MTS 2.0 on computers running Microsoft Cluster Server

If you are running MTS on computers that also run Microsoft Cluster Server, it is important that the MTS system registry entries be synchronized on both servers in the cluster. If these entries are synchronized, MTS packages in the cluster can access XA-compliant databases such as Oracle. Further, the Distributed Transaction Coordinator (DTC) will support failover, even if the path of the system directory on the two computers is not the same.

To synchronize the MTS system registry entries upon installation of the service pack, you must make sure both nodes of the cluster are running and then manually run the msdtc.exe utility on each server. To run msdtc.exe, enter the following in the Command Prompt window:

```
msdtc -mts2sp1fix systempath
```

Where *systempath* is the path of the Windows NT system files. For example, if the path of your system files is c:\winnt\system32, you would enter this command:

```
msdtc -mts2sp1fix c:\winnt\system32
```

MTS Components and the Visual Basic 6.0 Debugger

Visual Basic 6.0 supports debugging MTS components, but there are several issues to keep in mind. For details, see the Visual Basic 6.0 Readme file.

MTS Components and the Visual J++ Debugger

Visual J++ 6.0 provides full debugging support for MTS components in the MTS runtime environment. For details, see the Visual J++ documentation.

If you are using Visual J++ 1.0 or 1.1, you cannot debug MTS components in the MTS run-time environment. You can set breakpoints and debug MTS components that have not been installed in a package, but functionality provided by the MTS run-time environment is not available. The effect is that you can debug the component as a Java class, not as an MTS component. The object context will not be created in the debugger, so operations involving the object context will fail.

Specifying Path of MTS Component for exegen.exe in Visual J++ 1.0 or 1.1

In Visual J++ 1.0 or 1.1, when you create an MTS component using exegen.exe, you should not specify the path of the component using the multibyte character set (MBCS). Because of a problem in exegen.exe, components created this way cannot register themselves. To work around this problem, specify the path using the single-byte character system (SBCS) when you create the component.

- This behavior does not occur in Visual J++ 6.0, because the Visual J++ development environment subsumes the behavior of exegen.exe.

Reporting Problems and Bugs

When reporting a bug, please include the following information in your bug report:

- The error number and description.
- Any relevant Windows NT event-log messages. (Microsoft Transaction Server reports errors in the Windows NT event log. If an error occurs, please check the Windows NT event log).
- The configuration for your client computer, if applicable.
- The configuration for your server computer.
- The components that are causing the problem.
- The language that you used to develop your component (Microsoft Visual Basic, Microsoft Visual C++, and so on).
- The type of interfaces your component(s) implements. For example, custom interfaces, dual interface, or dispinterface.
- The ODBC version, if applicable.
- The name and version of the resource manager.
- Any third-party components, resource dispensers, or resource managers that are causing problems.
- If possible, the line or lines of code where the problem occurs.