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.

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.

Setting System Package Identity

When installing MTS, set the System package identity before creating any new packages as follows:

1. Create a new local Windows NT group named “MTS Administrators” and a new local user named “MTS Administrator”.
2. Add the “MTS Administrator” user to the “MTS Administrators” and "Administrators" groups.
3. Set the identity of the System package to “MTS Administrator”.
4. Shut down the System package so that it will be restarted with the new identity.

User Rights for MTS Packages

Make sure the user account for the identities of the System package and other MTS packages have the Windows NT "Log on as a service" user right. You can verify this by using the Windows NT User Manager:

1. From the **Policies** menu, choose **User Rights**.
2. Click **Show Advanced User Rights**.

Oracle Client Software Requirements for Use with MTS

Your Microsoft Transaction Server applications can access an Oracle database located on the same system as Microsoft Transaction Server. In this case, ensure that you install all of the software described in the Microsoft Transaction Server Help file in the section “Configuring Microsoft Transaction Server to Access Oracle."

Your Microsoft Transaction Server application can also access an Oracle database located on a remote Windows NT or UNIX system. To do this you must install the proper versions of Oracle’s client software on the system containing your Microsoft Transaction Server components. Your Oracle client software must be equal to or greater than the following versions:

Component	Version
Oracle SQL*Net	2.3.3
Oracle OCIW32.DLL	1, 0, 0, 5
Oracle SQLLIB18.DLL	1.8.3.0.1
Oracle XA73.LIB	7.3.3.2.0

Using Oracle Integrated Security with MTS

Oracle databases have integrated security that uses Windows NT authentication to validate database users. This allows a user to log on to Oracle without supplying a separate user name or password. Users can maintain one user name and password for both Windows NT and Oracle.

If your MTS components always supply a user name and password when connecting to Oracle databases, then you are not using integrated security. This is true whether your applications specify the user name and password directly or indirectly through an ODBC DSN. In either event, you are not using integrated security and can ignore this step.

If you use integrated security, you must configure MS DTC to run under a user name and password authorized to connect to your Oracle database. This is required because during database recovery, MS DTC opens your Oracle database to report the outcome of in-doubt transactions. See "Changing the User Account Under Which Microsoft Distributed Transaction Coordinator Runs" below. Use the

Oracle security administration tools to ensure that the user name you specify is authorized to open your Oracle database.

For more information on Oracle's integrated Windows NT security facilities, consult your Oracle documentation.

Changing the User Account Under Which Microsoft Distributed Transaction Coordinator Runs

If you configure the Microsoft Distributed Transaction Coordinator (DTC) to run under an account other than the System account, you must specify an account that is a member of the Administrators group. If not, MS DTC may fail to start, and you will see the following message in the Windows NT event log.

```
XATM log object failed to set log encryption key
```

You can configure the user ID for MS DTC as follows:

1. Stop MS DTC.
2. From the **Start** menu select **Settings** and then **Control Panel**.
3. Double-click the **Services** icon in the control panel.
4. Double click on **MS DTC** in the Service list box.
5. Select **Log On As** and specify a user name and password.
6. Restart MS DTC.

Configuring Oracle to Support a Large Number of Connections

If you want to create more than a few dozen connections to an Oracle database, you must configure the Oracle server to support additional database connections. If you do not do this, you may experience one or more of the following errors:

- Failures on **SQLConnect** calls.
- Failures to enlist on the calling object's transaction, which may result in any of the following errors in the Oracle trace file:
    - Too many sessions
    - TNS server failed to locate the server name
- Too many distributed transactions
- Timeouts while waiting for database locks. This is likely to occur if the configured number of locks is insufficient for the number of concurrently active transactions.
- Record collision due to locks held by in-doubt transactions.

If you experience any of these problems, consider increasing the following Oracle server configuration parameters:

- Sessions
- distributed_lock_timeout
- distributed_transactions
- dml_locks
- max_transaction_branches
- open_cursors
- Processes
- Queuesize

Sessions value

The Sessions configuration value should typically be three times the total number of database connections that you expect your applications to make. This is because the Oracle system typically opens more than one session (often as many as three sessions) for each transactional database connection that your application opens.

Queuesize value

The Oracle listener process sometimes falls behind and rejects database open requests if its queuesize is not adequate. Connection requests that come in too quickly for the Oracle listener to handle and that exceed the listener's queue (as determined by the QUEUESIZE parameter in LISTENER.ORA, TNSNET.ORA, and NAMES.ORA), are returned with an ECONNREFUSED error. A client that encounters this error returns the message "ORA-12541: No Listener" and the client log or trace files will show the ECONREFUSED message.

To correct this problem:

1. Stop the Oracle listener on the system containing the Oracle database server.
2. Increase the QUEUESIZE parameter in the LISTENER.ORA, TNSNET.ORA, or NAMES.ORA files on the Oracle database server system. Choose the queue size based on the number of simultaneous, or nearly simultaneous, connection requests you anticipate.

   For example, to accommodate 100 requests change the LISTENER.ORA file as follows:

   ```
   QUEUESIZE = 100
   ```
3. Restart the Oracle listener.

MTS Packages on MSCS Clusters Cannot Access XA-Compliant Databases

MTS packages on MSCS clusters cannot access an XA-compliant database (such as Oracle).

Security and Communication Between Remote MS DTC Computers

If your application performs transactions spanning two or more computers, you must ensure that the MS DTCs on these computers are configured to communicate with each other. MS DTC runs under the identity of the System account by default. The MS DTC on one computer can communicate with the MS DTC on another computer, only if the Guest accounts on both computers are enabled. If you disable the Guest account on either machine or in the Windows NT domain to which they belong, you must configure MS DTC to run under some other user name that can communicate with the remote computer. See "Changing the User Account Under Which Microsoft Distributed Transaction Coordinator Runs" above.

MS DTC with MSCS Requires QFE

Customers who use MS DTC in an MSCS environment should install Windows NT QFE Q116844. This QFE fixes a Remote Procedure Call (RPC) problem that can cause an MS DTC client to hang when attempting to connect to MS DTC on a clustered server.

Remote Server Name Must Specify a Virtual Server on MSCS Servers

Clustered servers must set the **Remote Server Name** property to a virtual server name using the MTS Explorer. This enables remote clients to access MTS components within the cluster, and, in the event of failure, ensures that clients are referencing the cluster, not a specific node within that cluster.

MTS Package Identities Must Be in Administrator Group on MSCS Clusters

For MTS packages on MSCS clusters, the user account associated with the package identity must be a system administrator or must have Full Control cluster access permissions (as set in cluadmin). If you do not run the package under the identity of a system administrator, MTS will be unable to connect to MS DTC.

MTS on MSCS Clusters Requires Same Windows NT Directory Path on All Nodes

MTS requires that both the Windows NT directory have the same path on all nodes within an MSCS cluster (for example, c:\winnt).

Shut Down MSCS Cluster Resources Before Installing MTS

You must shut down all MSCS cluster resources (such as SQL Server) that are affected by MTS setup prior to running the MTS setup program.

Installing SQL Server 6.5 After Installing MTS

You can install SQL Server 6.5 after installing MTS. It is recommended that you install SQL Server 6.5 before installing MTS. If you install SQL Server 6.5 after installing MTS, MTS Setup will attempt to ensure that MS DTC works properly.

Run the INSTCAT.SQL Script

As a convenience, MTS includes the INSTCAT.SQL script. This script updates the catalog stored procedures for the SQL Server ODBC driver. If you have not done so already, run this script using the SQL Server **isql** utility.

Documentation Errata on Security

Security authorization is not enabled by default.

You cannot set a package's identity to a group.

When setting package identity, MTS validates the password that you enter. However, if you change the password for the account without updating the password in the MTS Explorer, the package cannot run.

MS DTC Log File Can Be Compressed

The MS DTC log file can now be located in a compressed directory. However, for optimum performance, do not compress the Microsoft DTC log file or store it in a compressed directory. MS DTC must decompress the log file before it can be used.

Monitoring MTS Transactions on Windows 95

The documentation for starting and stopping MS DTC on Microsoft Windows 95 should read as follows:

By default, the Microsoft Distributed Transaction Coordinator (MS DTC) is configured to start automatically when a Windows NT or Windows 95 system starts. To prevent MS DTC from automatically starting after rebooting a Windows 95 computer, use the registry editor to find the **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices** registry key and then delete the value entry named **MSDTC**. If you want to enable automatic startup of MS DTC again, use the registry editor to create a value entry named **MSDTC** with the string value "`msdtcw -start`" under the **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices** registry key.

Note that the name of the service is "`msdtcw`", not "`msdtc`", and that there is a space before the hyphen.

No Remote Administration of MTS 2.0 Computers from MTS 1.*x* Computers

You cannot remotely administer an MTS 2.0 installation from a computer running a previous version of MTS. Attempting to do so will return an error code or cause the MTS Explorer to terminate unexpectedly.

No Individual Package Shutdown when Remotely Administering an MTS 1.*x* Computer

Individual package shutdown is not supported when remotely administering an MTS 1.*x* computer.

Visual Basic 5.0 Shutdown with MTS 2.0

After installing MTS 2.0, you may see an Access Violation when shutting down Microsoft Visual Basic 5.0. This is a known Visual Basic 5.0 bug that can be fixed by installing Visual Studio Service Pack 2 from http://www.microsoft.com/vstudio. Refer to Knowledge Base article Q167213 for more information about this problem.

Multiple Server Processes on Windows 95

There is a known problem that occurs when, if an MTS server process terminates unexpectedly (because of a failfast, an assert, or an access violation), subsequent attempts to restart the application create multiple MTS processes. (The correct behavior should be only one MTS process per package.) In case of unexpected server process termination, you must reboot your Windows 95 computer.

Deadlocks on Windows 95 When Using Named Pipes

There is a known problem that may cause deadlocks when using more than approximately 40 Named Pipe connections from a Windows 95 client to MS Distributed Transaction Coordinator (DTC). To fix this problem, change the protocol between the ODBC driver and SQL Server to TCP/IP.

Components and TreatAs and AutoTreatAs Registry Keys

COM allows one server to emulate another using the **TreatAs** or **AutoTreatAs** registry keys. Suppose a client creates a component of a given CLSID using **CoCreateInstance**. If the **TreatAs** or **AutoTreatAs** key is present under that CLSID in the registry, COM will instead create the component indicated by this key. This effectively routes client requests to the emulating server. (The same routing is possible using the ProgID if the client refers to its servers by ProgID instead of CLSID.)

Components being emulated using **TreatAs** or **AutoTreatAs** are not valid MTS components and cannot be imported. If the emulation is to be added after the component is imported, the component must be deleted from the MTS catalog, then replaced with the emulating component.

Mixed Threading Models in a Server Process

Within a server process, objects in the same activity must be registered with compatible threading models. The following combinations are valid:

- Any combination of apartment, both, or free
- Single-threaded only

Note that this means that single-threaded MTS components cannot be created by an Active Server Page running in-process with IIS or in-process with an IIS application using process isolation.

Do Not Create Non-MTS Local Servers Using CreateInstance

Creating non-MTS local servers using the context object's **CreateInstance** method is strongly discouraged. Such servers will not scale well and may result in additional server instances that are not shut down by MTS. If you need to create a non-MTS local server, use **CoCreateInstance**.

Do not use COM surrogates within MTS. Instead, load in-process servers into MTS server processes.

MTS Security Does Not Work With IIS Impersonation

MTS role-based package access control does not work properly when packages are front-ended by mutliplexers that impersonate multiple clients (for example, Internet Information Server). MTS packages used in these environments should be configured with security disabled.

This can be fixed by installing Windows NT QFE Q147222. After installing this QFE, role-based security will work properly with IIS.

When you create secure IIS applications that use process isolation, IIS creates an MTS package that is set to run as IWAM_<*computer name*>. If you wish to change the identity of these packages, you should also add the new package identity to the "MTS Trusted Impersonators" group or security will not work properly in other out-of-process MTS components called by your package. If this is not done, the caller will appear to be IWAM_<*computer name*> rather than the actual client of IIS.

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/or connections as well as explicitly release the ADO objects.

No Sample Bank ASP Client

The documentation incorrectly states that an Active Server Page (ASP) version of the Sample Bank client is provided.

Maximum Number of Methods on MTS 2.0 Components Is 1024

MTS 1.0 had a limit of 100 methods per component. In MTS 2.0 this limit is now 1024 methods.

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.

Location of Visual Basic Script Samples

The Visual Basic Scripting (VBScript) samples for automating MTS administration are located in \Program Files\Mts\Samples, not \Program Files\Mtx\Samples\WSH as stated in the documentation.

Using the Microsoft Transaction Server API to Develop Applications

Developing Components with Microsoft Visual C++

MFC Extension DLLs

MTS components should not be built as MFC Extension DLLs because such DLLs can be loaded only by MFC applications. A COM component, and therefore an MTS component, should be built so that it can be loaded into any process, regardless of the type of application that started the process.

For more information on MFC Extension DLLs, see the Microsoft Visual C++ Programmer's Guide.

Developing Components with Microsoft Visual Basic

ObjectControl

- The **ObjectControl** interface is the Automation counterpart of the C++ **IObjectControl** interface. The **ObjectControl** interface is not supported unless you have installed Microsoft Windows NT 4.0 Service Pack version 2 or later.
- The current Microsoft Transaction Server run-time implementation will call the **CanBePooled** method of a Transaction Server object, as specified in the Transaction Server online documentation. However, the return value is currently ignored. Returning TRUE does not stop Transaction Server from releasing your object.

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.

Maintaining the MTS Catalog when Developing Components with Visual Basic

Whenever you recompile an OLE DLL project in Visual Basic 4.0, Visual Basic rewrites all of the registry entries for all of the components (Visual Basic classes) that live in that DLL. Additionally, Visual Basic may generate new GUIDs (depending on your project configuration) to identify the components in that DLL. This means that your MTS components are no longer properly registered in the MTS catalog.

There are two solutions to this problem. First, there is a **Refresh** button in the Transaction Server Explorer, as well as the **Refresh All Components** command on the **Tools** menu. If you use this command Microsoft Transaction Server will repair all inconsistencies in registry entries of components currently in the right pane of the Explorer.

If you install the development version of Microsoft Transaction Server, check the **VB Addin** option to enable a feature that will automatically refresh your components after recompiling them. The next time you run Visual Basic 4.0, the add-in will be automatically installed in your Visual Basic IDE. You can also turn the feature on and off on a per-project basis by selecting or deselecting the **MTxServer RegRefresh | AutoRefresh after compile** command on the Visual Basic **Add-Ins** menu. If you decide you want to refresh all of your Microsoft Transaction Server components at any given time, you can use the **MTxServer RegRefresh | Refresh all components now** command on the **Add-Ins** menu.

Important The Visual Basic Add-in has been updated to work with both Visual Basic versions 4.0 and 5.0. Once installed, it will automatically refresh the Transaction Server catalog with the changes made during each compile. The add-in menu option to enable and disable the automatic refresh is no longer supported.

Using the add-in will properly refresh the MTS catalog, even after Visual Basic compilations that generate new component GUIDs. Note also that the **Compatible OLE Server** setting in the Visual Basic **Project Options** dialog box can be used to stop Visual Basic from generating new component GUIDs.

Refreshing the MTS catalog depends on you not changing the ProgIDs of your components. In Visual Basic 4.0, a component's ProgID is formed by the following concatenation: *project name.class name*. If you change either of these items, you will have to reinstall your component(s) in the MTS Explorer.

Building your component in Visual Basic 4.0 without selecting the **Compatible OLE Server** option replaces your old CLSID and IID on each compile. This has disadvantages when developing an MTS component, even with the Visual Basic add-in enabled.

- Roles you assigned to the interface using the MTS Explorer are lost, since the interface IID is obsolete.
- Proxies and registry configurations you distributed to remote machines no longer refer to your component and must all be updated.
- Packages you exported which contain your component require re-exporting since the package definition file GUIDs are now out of sync.

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.