ftf@□[fXfg fXfefbfv fKfCfh

Microsoft® Transaction Server (MTS) ,Í□AfRf“f|□[flf“fgfx□[fX,ÌfgfŠ“fUfNfVf‡f“□ˆ—□fVfXfef€ ,Æ,µ,Ä□A□,□«”\,ÅfXfP□[fŠuf‹□A,©,ÂŒ˜˜S,ÈŠé‹ÆŒü,¯fCf“fˆ□[flfbfg□A,¨,æ,ÑfCf“fgfŠflfbfg fT□[fo□[fAfvfŠfP□[fVf‡f“,ÌŠJ”□A”z’u□A,¨,æ,ÑŠÇ—□‹@”\,ð’ñ‹Ÿ,µ,Ü,·□BMTS ,Å,Í□AfRf“f| □[flf“fgfx□[fX,Ì•ªŽUfAfvfŠfP□[fVf‡f“,ðŠJ”,·,é,½,ß,ÌfvfŠfOf‰f~f“fO f,ffƒ‹,ª’è‹`,³,ê,Ä,¢,Ü,·□B,Ü,½□A,± ,ê,ç,ÌfAfvfŠfP□[fVf‡f“,Ì”z’u,ÆŠÇ—□,Ì,½,ß,Ìf‰f“fˆfCf€ŠÂ‹«,ÆfOf‰ftfBfJfƒ fc□[fƒ‹,ª—pˆÓ,³,ê,Ä,¢,Ü,·□B

,±,±,Å,Í□AMTS ,Ì□V,µ,¢‹@”\,ÆfhfLf…f□f“fg,ÌŠT—v,ð□à–¾,µ□A—pŒê,É,Â,¢,Ä,à‰ð□à,µ,Ü,·□B,±,± ,Å,Íˆè‰ºˆ,ÌfgfsfbfN,É,Â,¢,Ä□à–¾,µ,Ü,·□B

- □V,µ,¢‹@”\
- fhfLf…f□f“fg,Ì□\□¬
- —pŒê□W
- ft□[fefBfŠfefB
- ,æ,Šñ,¹,ç,ê,éŽ¿–â (FAQ)

□V,µ,¢‹@”\

Microsoft Transaction Server (MTS) Version 2.0 ,É,Í□AŒ˜˜S,ÅƒXƒP□[ƒ‰ƒuƒ‹,ÈƒCƒ“ƒ^□[ƒIƒbƒg,¨,æ,ÑƒCƒ“ƒgƒ‰ƒIƒbƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì”z’u,ð—eˆÕ,É,·,é‘½,,Ì□V,µ,¢‹@”\,ª—pˆÓ,³,ê,Ä,¢,Ü,·□B,±,±,Å,Í MTS ,Ì□V,µ,¢‹@”\,ÌŠT—v,ð□à–¾,µ,Ü,·□B

Internet Information Server (IIS) Version 4.0 ,Æ,ÌŠ®‘S,È“□□‡

MTS 2.0 ,Í IIS 4.0 ,ÆŠ®‘S,É“□□‡,³,ê,Ä,¨,è□AWeb □ã,ÌƒrƒWƒIƒX ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,É,Æ,Á,Ä□Å“K,Èƒvƒ‰ƒbƒgƒtƒH□[ƒ€,ð’ñ‹Ÿ,µ,Ü,·□BMTS ,Æ IIS ,Ì“□□‡,É,æ,è□AŽŸ,Ì,æ,¤,È□V,µ,¢‹@”\,ª’ñ‹Ÿ,³,ê,Ü,·□B

- ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÅŽg—p,Å,«,é Active Server Pages

  MTS ,ªŠÇ—□,·,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ““à,Å Active Server Pages ,ÌƒXƒNƒŠƒvƒg,ðŽÀ□s,Å,«,Ü,·□B,± ,Ì,½,ß□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,É‘Î,·,é•ÛŒì,ª Web ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‘S‘Ì,É“K—p,³,ê,Ü,·□B

- IIS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,É‘Î,·,éƒNƒ‰ƒbƒVƒ…•ÛŒì

  IIS Web ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð Web ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“Ž©□g,Ì MTS ƒpƒbƒP□[ƒW“à,ÅŽÀ□s,Å,«,é,Ì,Å□AWeb ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìƒvƒ□ƒZƒX,ð•ª—£,µ□AƒNƒ‰ƒbƒVƒ…,©,ç•ÛŒì,·,é,±,Æ,ª,Å,«,Ü,·□B

- ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÉŠî,Ã,ƒCƒxƒ“ƒg

  ŠJ”ŽÒ,Í Active Server Pages ,ÌƒXƒNƒŠƒvƒg,ÉƒRƒ}ƒ“ƒh,ð–„,ß□ž,Þ,±,Æ,ª,Å,«,é,Ì,Å□AWeb ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì‰ž“š,ðƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŒ‹‰Ê,ÉŠî,Ã,¢,ÄƒJƒXƒ^ƒ}ƒCƒY,Å,«,Ü,·□B

- IIS ‘g,Ý□ž,ÝƒIƒuƒWƒFƒNƒg,ÌƒIƒuƒWƒFƒNƒg ƒRƒ“ƒeƒLƒXƒg

  MTS ƒIƒuƒWƒFƒNƒg ƒRƒ“ƒeƒLƒXƒg,É,æ,Á,Ä□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŠJ”-ŽÒ,ªƒt□[ƒU□[,Ì□ó‘Ô□î•ñ,ð’Ç□Õ,·,é,Æ,«,Ì•¡ŽG,³,ÍŒyŒ¸,³,ê□AIIS ‘g,Ý□ž,ÝƒIƒuƒWƒFƒNƒg,ÉŠÇ—□,³,ê,½□ó‘Ô□î•ñ,ð’Ç□Õ,·,é,±,Æ,ª,Å,«,Ü,·□B,±,Ì,½,ß□AWeb ŠJ”ŽÒ,Í MTS ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒ‚ƒƒ‹,ÌŠÈˆÕ□«,ðŠˆ—p,Å,«,Ü,·□B

- ‹¤’Ê,ÌƒCƒ“ƒXƒg□[ƒ‹,ÆŠÇ—□

  MTS ,Æ IIS ,ª‹¤’Ê,ÌƒCƒ“ƒXƒg□[ƒ‹,Æ‹¤’Ê,ÌŠÇ—□ƒRƒ“ƒ\□[ƒ‹,ð‹¤—L,Å,«,é,Ì,Å□AWeb □ã,ÌƒrƒWƒIƒX ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì”z’u,ÆŠÇ—□,Ì•¡ŽG,³,ªŒyŒ¸,³,ê,Ü,·□B

Oracle ,ÌƒIƒCƒeƒBƒu ƒTƒ|□[ƒg,ðŠÜ,Þ XA ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒvƒ□ƒgƒRƒ‹,ÌƒTƒ|□[ƒg

- MTS 2.0 ,Í XA ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒvƒ□ƒgƒRƒ‹,ðƒTƒ|□[ƒg,µ,Ä,¢,Ü,·□B,±,Ì,½,ß□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í ODBC ,ð‰î,µ,Ä Windows NT Server □ã,¨,æ,Ñ (UNIX ,ÌŠeƒo□[ƒWƒ‡ƒ“,ðŠÜ,Þ) Microsoft ˆÈŠO,ÌƒIƒyƒŒ□[ƒeƒBƒ“ƒO ƒVƒXƒeƒ€□ã,Å“®□ì,·,é IBM DB2□AInformix ,È,Ç,Ì XA □€‹’ƒf□[ƒ^ƒx□[ƒX,Æ‘g,Ý□‡,í,¹,ÄŽg—p,Å,«,Ü,·□B

- MTS 2.0 ,É,Í□A Microsoft Oracle Driver for Oracle ,Ì‰ü’ù”Å,ªŠÜ,Ü,ê,Ä,¢,é,Ì,Å□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í ODBC ,ð‰î,µ,Ä Oracle Version 7.3 ˆÈ□~,Æ˜AŒg,µ□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð’¼□ÚŠÇ—□,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒfƒXƒNƒgƒbƒv ƒIƒyƒŒ□[ƒeƒBƒ“ƒO ƒVƒXƒeƒ€,ÌƒTƒ|□[ƒg

MTS 2.0 ,Í Windows NT Version 4.0 ˆÈ□~,Æ Windows 95/98 ,Ì—¼•û,ÌƒIƒyƒŒ□[ƒeƒBƒ“ƒO ƒVƒXƒeƒ€,ÅŽg—p,Å,«,Ü,·□BMTS ,Í□A Microsoft ,ÌƒfƒXƒNƒgƒbƒv ƒIƒyƒŒ□[ƒeƒBƒ“ƒO ƒVƒXƒeƒ€,É‘Î‰ž,µ,Ä,¢,é,½,ß□AŠé‹Æ,Í“ÆŽ©,Ì MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒXƒ^ƒhƒAƒ□ƒ“ ƒo□[ƒWƒ‡ƒ“,ð”z’u,·,é,±,Æ,ª,Å,«,Ü,·□B

- Windows NT Server ,É MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð”z’u,·,é‘O,É□AMTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŠJ”,·,é,½,ß,ÌŠJ”ƒvƒ‰ƒbƒgƒtƒH□[ƒ€,Æ,µ,Ä Windows 95/98 ,ð—˜—p,Å,«,Ü,·□B,Ü,½□AWindows NT □ã,ÅŽÀ□s,µ,Ä,¢,é MTS ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŠÇ—□ƒNƒ‰ƒCƒAƒ“ƒg,Æ,µ,Ä Windows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,ðŽg,¤,± ,Æ,à,Å,«,Ü,·□B,³,ç,É□AMTS ,Í Windows 95/98 □ã,ÅŽÀ□s,·,é Personal Web Server (PWS) ,Ìƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,à’ñ‹Ÿ,µ,Ü,·□B

- Windows NT ,Æ Windows 95/98 ,Æ,Ìˆá,¢,É,æ,è□AWindows 95/98 □ã,Ì MTS ,Í Microsoft Cluster Server ,â MTS ,Ìƒf□□[ƒ‹,ÉŠî,Ã,ƒZƒLƒ…ƒŠƒefB,ðƒTƒ|□[ƒg,µ,Ä,¹,ñ□BWindows NT ,Ü,½,Í Windows

95/98 ,ðŽÀ□s,µ,Ä,¢,éƒRƒ“ƒsƒ…□[ƒ^,©,ç MTS ,ðŽÀ□s,µ,Ä,¢,é Windows 95/98 ƒRƒ“ƒsƒ… □[ƒ^,ðƒŠƒ,□[ƒg,ÅŠÇ—□,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B,Ü,½□AMTS ŠÇ—□,ÌŽ©“®‰»,ðŽ¦,· Microsoft Visual Basic® Scripting Edition (VBScript) ,ÌƒTƒ“ƒvƒ‹,Í□AWindows 95/98 ƒRƒ“ƒsƒ… □[ƒ^,É,ÍƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ü,¹,ñ□B

Microsoft Cluster Server ,ÌƒTƒ|□[ƒg

MTS 2.0 ,Í Microsoft Cluster Server (MSCS) ,ðƒTƒ|□[ƒg,µ,Ä,¢,Ü,·□B,±,Ì,½,ß□AƒNƒ‰ƒXƒ^“à,Ì MTS ƒpƒbƒP□[ƒW,ÌŽ©“®ƒtƒF□[ƒ‹ƒI□[ƒo□[,ª‰Â”\,Å,·□BŽ©“®ƒtƒF□[ƒ‹ƒI□[ƒo□[,É,æ,è MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì□,‰Â —p□«,ªŽÀŒ»,³,ê,Ü,·□B

LU 6.2 Sync Level 2 ,ð‰î,µ,½ CICS ,¨,æ,Ñ IMS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌƒTƒ|□[ƒg

MTS 2.0 ,ðŽg,¤,Æ□AŠé‹Æ,Í MVS □ã,Ì CICS ,¨,æ,Ñ IMS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒTƒ|□[ƒg,·,é MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð”z’u,·,é,±,Æ,ª,Å,«,Ü,·□BMTS ,Í□AMTS ,Æ CICS/MVS ,¨,æ,Ñ IMS/MVS ,Æ,ÌŠÔ,Ì‘ŠŒÝ ‰^—p□«,ð’ñ‹Ÿ,·,é SNA ƒT□[ƒo□[ƒRƒ“ƒ|□[ƒlƒ“ƒg,Å, ,é COM ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒCƒ“ƒefƒOƒŒ□[ƒ^ (COMTI) ,ðŠ®‘S,ÉƒTƒ|□[ƒg,µ,Ü,·□B

ŠÇ—□‹@”\,ÌŠg’£

MTS 2.0 ,Å,ÍŽŸ,Ì,æ,¤,ÈŠÇ—□‹@”\,ÌŠg’£,É,æ,è□AMTS ƒpƒbƒP□[ƒW,Ì”z’u,ÆŠÇ—□,ª,³,ç,É—eˆÕ,É,È,Á,Ä,¢ ,Ü,·□B

- MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒXƒiƒbƒvƒCƒ“

  MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Í□AMicrosoft Management Console (MMC) ,ÌƒXƒiƒbƒvƒCƒ“,É,È,Á,Ä,¢,Ü,·□B,± ,Ì,½,ß□AIIS ,È,Ç,Ì,Ù,©,Ì□»•i,Æ“¯,¶ŠÇ—□ƒRƒ“ƒ\□[ƒ‹,ðŽg,Á,Ä MTS ƒpƒbƒP□[ƒW,ðŠÇ—□,Å,«,Ü,·□BMMC ,¨,æ,ÑƒXƒiƒbƒvƒCƒ“,ÌŽg,¢•û,Ì□Ú□×,É,Â,¢,Ä,Í□AMMC ,ÌƒhƒLƒ…ƒ□ƒ“ƒg,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

- ŒÂ□X,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,ÌƒVƒƒƒbƒgƒ_ƒEƒ“

  MTS 2.0 ,Å,Í□AMTS ƒT□[ƒo□[ƒvƒ□ƒZƒX,ðƒVƒƒƒbƒgƒ_ƒEƒ“,¹,¸,É□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ÅŒÂ□X,ÌƒpƒbƒP□[ƒW,ðƒVƒƒƒbƒgƒ_ƒEƒ“,Å,«,Ü,·□BƒpƒbƒP□[ƒW,ðƒVƒƒƒbƒgƒ_ƒEƒ“,·,é,Æ□A,»,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“, ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,Í□I—¹,µ,Ü,·□BMTS 1.x ƒRƒ“ƒsƒ…□[ƒ^□ã,ÌƒŠƒ,□[ƒg,ÅŠÇ—□,³,ê,Ä,¢ ,éƒpƒbƒP□[ƒW,Í□AŒÂ•Ê,ÉƒVƒƒƒbƒgƒ_ƒEƒ“,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

- ‰ü—Ç,³,ê,½ƒpƒbƒP□[ƒW,ÌƒAƒNƒefBƒu‰»,Ì□Ý’è

  MTS 2.0 ,Å,Í□AƒAƒAƒNƒefBƒu‰»,ÍƒpƒbƒP□[ƒW ƒŒƒxƒ‹,¾,¯,É□Ý’è,³,ê,Ü,·□B“¯,¶ƒpƒbƒP□[ƒW“à,Ì•¡□”,ÌƒRƒ“ƒ| □[ƒlƒ“ƒg,ÉˆÙ,È,éƒAƒNƒefBƒu‰»,ð□Ý’è,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B,Ü,½□AƒŠƒ,□[ƒg,Å,ÌƒAƒNƒefBƒu ‰»,ð□Ý’è,µ,ÄƒpƒbƒP□[ƒW,ðŽÀ□s,·,é,±,Æ,à,Å,«,Ü,¹,ñ□BƒpƒbƒP□[ƒW,Í□A**ƒT□[ƒo□[** (ƒ□□[ƒJƒ‹,Å,ÌƒRƒ“ƒ| □[ƒlƒ“ƒg,ÌƒAƒNƒefBƒu‰») ,Ü,½,Í**ƒ‰ƒCƒuƒ‰ƒŠ** (ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŒÄ,Ñ□o,µ,½ƒNƒ ‰ƒCƒAƒ“ƒg,Æ“¯,¶ƒvƒ□ƒZƒX,Å,ÌƒAƒNƒefBƒu‰») ,Ì,¢,¸,ê,©,É□Ý’è,³,ê,Ü,·□B

  ŠÜ,Ü,ê,Ä,¢,é,·,×,Ä,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ªƒCƒ“ƒvƒ□ƒZƒX,ÉŽw’è,³,ê,Ä,¢,é MTS 1.0 ,ÌƒpƒbƒP□[ƒW,Í□AMTS 2.0 ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,Æ□AŽ©“®“I,Éƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,ÉƒAƒbƒvƒOƒŒ□[ƒh,³,ê,Ü,·□BŠÜ,Ü,ê,Ä,¢ ,é,·,×,Ä,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ªƒ□□[ƒJƒ‹,Å,ÌƒAƒNƒefBƒu‰»,ÉŽw’è,³,ê,Ä,¢,é MTS 1.0 ,ÌƒpƒbƒP□[ƒW,Í□AŽ©“®“I,ÉƒT□[ƒo□[ƒpƒbƒP□[ƒW,ÉƒAƒbƒvƒOƒŒ□[ƒh,³,ê,Ü,·□BˆÙ,È,é•ûŽ®,ÌƒRƒ“ƒ| □[ƒlƒ“ƒg,ÌƒAƒNƒefBƒu‰»,ª□¬□Ý,µ,Ä,¢,é MTS 1.0 ,ÌƒpƒbƒP□[ƒW,Í□AŽè“®,Åƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,Ü,½,ÍƒT□[ƒo□[ƒpƒbƒP□[ƒW,Æ,µ,Ä□\□¬,·,é•K—v,ª, ,è,Ü,·□B

- ƒvƒ□ƒpƒefB,Ì□X□VŽž,É•¡□”ƒAƒCƒef€,Ì‘I‘ð,ª‰Â”\

  MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å•¡□”,ÌƒAƒCƒef€,Ìƒvƒ□ƒpƒefB,ð“¯Žž,É‘I‘ð,µ□A•Ï□X,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO‹@”\,ÌŠg’£

MTS 2.0 ,Å,ÍŽŸ,Ì,æ,¤,Èƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO‹@”\,ÌŠg’£,É,æ,è□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì□ì□¬,ª,³,ç,É— eˆÕ,É,È,Á,Ä,¢,Ü,·□B

- MTS ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì’Ç‰Á,Æ□X□V

  Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª□X□V,³,ê□A□V,½,É 2 ,Â,ÌƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª’ñ‹Ÿ,³,ê,Ü,·□BTic-

Tac-Toe ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í□A‹¤—L□ó‘Ô,ðŠÇ—□,·,é”ñƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì‹@”\,ðŽ¦,·ŠÈ’P,Èƒ}ƒ‹ƒ`ƒ†□[ƒU□[ƒQ□[ƒ€,Å,·□BAdministrative Sample Scripts ,Í□AŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŽg—p,µ□AWindows® Scripting Host ƒXƒNƒŠƒvƒg,É,æ,Á,ÄƒGƒNƒXƒvƒ□□[ƒ‰,Ì□ˆ—□,ðŽ©“®‰»,·,é•û–@,ðŽ¦,µ,Ü,·□B

- ŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg

  ŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,É,æ,éƒpƒbƒP□[ƒW,Ì”z’u,Æ•ÛŽç,ðŽ©“®‰»,·,é,± ,Æ,ª,Å,«,Ü,·□BƒI□[ƒgƒ□□[ƒVƒ‡ƒ“ŒÝŠ·,Ì”CˆÓ,ÌŒ¾Œê,ðŽg,¤,± ,Æ,É,æ,è□AƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,È,Ç,ÌŠÇ—□Žè□‡,ðŠÈ’P,ÈƒXƒNƒŠƒvƒg,ðŽg,Á,ÄŽ©“®‰»,·,é,± ,Æ,ª,Å,«,Ü,·□B

- MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì□ÝŒv,ÆŽÀ‘•,ÉŠÖ,·,éƒhƒLƒ…ƒ□ƒ“ƒg

  □wProgrammer's Guide□x,Í□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□ì□¬,·,é,Æ,«,Ì□ÝŒv,¨,æ,ÑŽÀ‘•,ÉŠÖ,·,éŽèˆø,«□‘,Å,·□BƒfRƒ“ƒ|□[ƒlƒ“ƒg,Ö,ÌƒrƒWƒlƒXƒ□ƒWƒbƒN,Ì‘g,Ý□ž,Ý,©,ç–â‘è,Ì□f’f,ÆƒfƒoƒbƒO,Ü,Å,ÌƒgƒsƒbƒN,É,Â,¢,Ä□à–¾,µ,Ü,·□B

ŠÖ˜A□€–Ú

□wŠÇ—□ŽÒƒKƒCƒh□x□AMTS Overview and Concepts (ŠT—v,Æƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO,ÌŠT”O)

ƒhƒLƒ…ƒ□ƒ“ƒg‚Ì□\□¬

Microsoft Transaction Server (MTS) ‚É‚Í□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚Ì□ÝŒv□A□\'z□A"z'u□A‚¨‚æ‚ÑŠÇ—□‚ÌŠw□K‚É–ð—§‚Â‚ŽŸ‚Ìƒhƒ‚ƒLƒ…ƒ□ƒ“ƒg‚ª—pˆÓ‚³‚ê‚Ä‚¢‚Ü‚·□B

ƒhƒLƒ…ƒ□ƒ“ƒg	"à—e
□wƒZƒbƒgƒAƒbƒv□x	MTS ‚Æ MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ÌƒZƒbƒgƒAƒbƒv•û–@‚ð□à–¾‚µ‚Ü‚·□BMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚©‚ç Oracle ƒf□[ƒ^ƒx□[ƒX‚ÉƒfƒAƒNƒZƒX‚·‚éŽè□‡‚Æ□AMTS ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ðƒCƒ“ƒXƒg□[ƒ‹‚·‚éŽè□‡‚É‚Â‚¢‚Ä‚à□à–¾‚µ‚Ü‚·□B
□wƒtƒ@□[ƒXƒg ƒXƒefbƒv ƒKƒCƒh□x	MTS ‚Ì□V‚µ‚¢‹@"\‚ÆƒhƒLƒ…ƒ□ƒ“ƒg‚ÌŠT—v‚ð□à–¾‚µ□A—pŒê‚É‚Â‚¢‚Ä‚à‰ð□à‚µ‚Ü‚·□B
□wƒNƒCƒbƒN ƒcƒA□[□x	MTS ‚ÌŠT—v‚ð□à–¾‚µ‚Ü‚·□B
□wŠÇ—□ŽÒƒKƒCƒh□x ŠÇ—□ŽÒƒKƒCƒh	MTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ŽğŽg‚Á‚ÄƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚Ì"z'u‚ÆŠÇ—□‚ð□s‚¤‚Ü‚´‚Ü‚È•û–@‚Æ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ÌƒOƒ‰ƒtƒBƒJƒ‹ƒCƒ"ƒ^□[ƒtƒFƒCƒX‚ÌŠT—v‚ð□à–¾‚µ‚Ü‚·□B
ƒpƒbƒP□[ƒW‚Ì□ì□¬	MTS ƒpƒbƒP□[ƒW‚Ì□ì□¬‚Æ‘g‚Ý□ž‚Ý‚É‚Â‚¢‚Ä□A□ì‹Æ‚ÆŽžŠÔ‚É‰ˆ‚Á‚Ä□à–¾‚µ‚Ü‚·□B
ƒpƒbƒP□[ƒW‚Ì"z•z	MTS ƒpƒbƒP□[ƒW‚Ì"z•z‚É‚Â‚¢‚Ä□A□ì‹Æ‚ÆŽžŠÔ‚É‰ˆ‚Á‚Ä□à–¾‚µ‚Ü‚·□B
ƒpƒbƒP□[ƒW‚ÌƒfƒCƒ"ƒXƒg□[ƒ‹	MTS ƒpƒbƒP□[ƒW‚ÌƒfƒCƒ"ƒXƒg□[ƒ‹‚Æ□\□¬‚É‚Â‚¢‚Ä□A□ì‹Æ‚ÆŽžŠÔ‚É‰ˆ‚Á‚Ä□à–¾‚µ‚Ü‚·□B
ƒpƒbƒP□[ƒW‚Ì•ÛŽç	MTS ƒpƒbƒP□[ƒW‚Ì•ÛŽç‚ÆŠÄŽ‹‚É‚Â‚¢‚Ä□A□ì‹Æ‚ÆŽžŠÔ‚É‰ˆ‚Á‚Ä□à–¾‚µ‚Ü‚·□B
ƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ“‚ÌŠÇ—□	•ªŽUƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ“‚Æ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ðŽg‚Á‚½ƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ“‚ÌŠÇ—□‚É‚Â‚¢‚Ä□à–¾‚µ‚Ü‚·□B
ŠÇ—□‚ÌŽ©"®‰»	ŠT"O"I‚ÈŠT—v□A'€□ìŽè□‡□AƒTƒ"ƒvƒ‹ƒR□[ƒh‚ðŽ¦‚µ□AMTS ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg‚ÌƒXƒNƒŠƒvƒg‚ðŽg‚Á‚Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰‚Ì□ˆ—□‚ðŽ©"®‰»‚·‚é•û–@‚É‚Â‚¢‚Ä□à–¾‚µ‚Ü‚·□B
□wProgrammer's Guide□x	

MTS Overview and Concepts (ŠT—v,Æfvƒ□fOf ‰ƒ~ƒ“ƒO,ÌŠT”O)	MTS ,ÌŠT—v,¨,æ,Ñ MTS ,ÌƒRƒ“ƒ| □[ƒlƒ“ƒg,Ç,¤,µ,ª‹¦’²,µ,Ä“®□ì,·,é,µ,- ,Ý,ð□à–¾,µ,Ü,·□B,Ü,½□AƒNƒ ‰ƒCƒAƒ“ƒg/ƒT□[ƒo□[ŠJ”- ŽÒ,ÆƒVƒXƒeƒ€ŠÇ—□ŽÒ,Ìƒj□[ƒY,É MTS ,ª,Ç,Ì,æ,¤,É‰ž,¦,é,©,ð□à–¾,µ□AMTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒOƒ ‰ƒ~ƒ“ƒO,ÌŠT”O,É,Â,¢,Ä□Ú,µ,- ‰ð□à,µ,Ü,·□B
Building MTS Applications (ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì□ì□¬)	MTS —p,Ì ActiveX® ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŠJ”- ,É,Â,¢,Ä□A□ì‹Æ,ÆŽè□‡,É‰ˆ,Á,Ä□à– ¾,µ,Ü,·□B
MTS Administrative Reference (ŠÇ— □ƒŠƒtƒ@ƒŒƒ“ƒX)	MTS ƒXƒNƒŠƒvƒg ƒlƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì□ˆ—□,ðŽ©“® ‰»,·,é,½,ß,ÌŽQ□Æ□î•ñ,ª‹L□Ú,³,ê,Ä,¢ ,Ü,·□B
MTS Reference (ƒŠƒtƒ@ƒŒƒ“ƒX)	MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒCƒ“ƒ^□[ƒtƒFƒCƒX (API) ,ÉŠÖ,·,éŽQ□Æ□î•ñ,ª‹L□Ú,³,ê,Ä,¢ ,Ü,·□B

ƒt□[ƒeƒBƒŠƒeƒB

MTS ‚É‚Í□Aƒofbƒ` ƒtƒ@ƒCƒ‹‚Å“Á’è‚Ì□ì‹Æ‚ðŽ©“®‰»‚·‚é‚½‚ß‚ÌƒRƒ}ƒ“ƒhƒ‰ƒCƒ“ ƒt□[ƒeƒBƒŠƒeƒB‚ª—p^Ó‚³‚ê‚Ä‚¢‚Ü‚· (‚±‚ê‚ç‚Ìƒt□[ƒeƒBƒŠƒeƒB‚ÍƒRƒ}ƒ“ƒh ƒvƒ□ƒ“ƒvƒg‚©‚ç¼□ÚŽg—p‚Å‚«‚Ü‚·)□B

ƒRƒ}ƒ“ƒhƒ‰ƒCƒ“ ƒt□[ƒeƒBƒŠƒeƒB

ŽŸ‚Ì•\‚Í□AMTS ‚Æˆê□□‚ÉƒCƒ“ƒXƒg□[ƒ‹‚³‚ê‚éƒRƒ}ƒ“ƒhƒ‰ƒCƒ“ ƒt□[ƒeƒBƒŠƒeƒB‚ÌƒNƒCƒbƒN ƒŠƒtƒ@ƒŒƒ“ƒX‚Å‚·□B

ƒt□[ƒeƒBƒŠƒeƒB	**‹@”**
MTXSTOP.exe	MTS ‚Ì‚·‚×‚Ä‚Ìƒvƒ□ƒZƒX‚ðVƒƒbƒgƒ_ƒEƒ“‚µ‚Ü‚·□B[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ‚ðƒ}ƒEƒX‚Ì‰Eƒ{ƒ^ƒ“‚ÅƒNƒŠƒbƒN‚·‚é‚Æ•\Ž¦‚³‚ê‚éƒ□ƒj…□[‚Ì [ƒT□[ƒo□[ƒvƒ□ƒZƒX‚ÌVƒƒbƒgƒ_ƒEƒ“] ƒRƒ}ƒ“ƒh‚É‘Î‰ž‚·‚éƒRƒ}ƒ“ƒhƒ‰ƒCƒ“ ƒt□[ƒeƒBƒŠƒeƒB‚Å‚·□B
MTXREREG.exe	ƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^‚É“o˜^‚³‚ê‚Ä‚¢‚é‚·‚×‚ÄƒRƒ“ƒ|□[ƒlƒ“ƒg‚ð□X□V‚µ‚Ü‚·□B‘I‘ð‚µ‚½ƒRƒ“ƒsƒ…□[ƒ^‚ðƒ}ƒEƒX‚Ì‰Eƒ{ƒ^ƒ“‚ÅƒNƒŠƒbƒN‚·‚é‚Æ•\Ž¦‚³‚ê‚éƒ□ƒj…□[‚Ì [‚·‚×‚Ä‚ÌƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ì□X□V] ƒRƒ}ƒ“ƒh‚É‘Î‰ž‚·‚éƒRƒ}ƒ“ƒhƒ‰ƒCƒ“ ƒt□[ƒeƒBƒŠƒeƒB‚Å‚·□B
MTXREPL.exe	MTS ƒT□[ƒo□[‚ðƒŒƒvƒŠƒP□[ƒg‚µ‚Ü‚·□BƒŒƒvƒŠƒP□[ƒgŒ³ƒRƒ“ƒsƒ…□[ƒ^‚ÆƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^‚Ì—¼•û‚ðŽQ□s‚µ‚Ä‚¢‚é•K—v‚ª, ‚è‚Ü‚·□B
TestOracleXAConfig .exe	MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ðŠÜ‚Þ•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠm”F‚·‚é‚½‚ß‚É Oracle ‚Ì□\□¬‚ðƒefƒXƒg‚µ‚Ü‚·□B ‚±‚ƒt□[ƒeƒBƒŠƒeƒB‚ªŽ¸”s‚µ‚½□ê□‡□AOracle ƒf□[ƒ^ƒx□[ƒX‚ðŽg—p‚·‚é•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Í“®□ì‚µ‚Ü‚¹‚ñ□B

Windows NT ‚ÌŠÇ—□ƒc□[ƒ‹

Windows NT ‚É‚à MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ÌŠÇ—□‚ÉŽg—p‚Å‚«‚é□[ƒ‹‚ª‚¢‚‚Â‚©‚ ‚è‚Ü‚·‚ª‚¨Žg‚¢‚ÌŠÂ‹«‚É‚æ‚Á‚Ä‚Í‚»‚ê‚ç‚ðŽg—p‚Å‚«‚È‚¢‚±‚Æ‚à‚ ‚è‚Ü‚·□B ƒ{ƒ^ƒ“[ƒXƒ^□[ƒg] ƒ{ƒ^ƒ“‚ðNƒŠƒbƒN‚µ□A[ƒvƒ□ƒOƒ‰ƒ€] ‚ðƒ|ƒCƒ“ƒg‚µ‚Ü‚·□BŽŸ‚É□A[ŠÇ—□ƒc□[ƒ‹ (‹¤’Ê)] ‚ðƒ|ƒCƒ“ƒg‚µ‚Ü‚·□B

ƒc□[ƒ‹		**‹@”**
	ƒCƒxƒ“ƒg ƒrƒ…□[ƒA	Windows NT ‚Å□AƒCƒxƒ“ƒg‚Æ‚Í□AƒVƒXƒefŠ€‚Ü‚½‚Íƒvƒ□ƒOƒ‰ƒ€‚Å”□¶‚µ□Aƒt□[ƒU□[‚É‘Ê’m‚·‚é•K—v‚Ì‚È‚¢ƒ□ƒj…□[ƒA‚Í□AƒCƒxƒ“ƒg‚ðƒt□[ƒU□[‚É‘Ê’m‚µ‚½‚è□Aƒ□ƒO‚É‹L˜^‚µ‚½‚è‚µ‚Ü‚·□BMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚Å”□¶‚µ‚½‚è–â‘è‚è□ƒ‘f‚·‚é‚Æ‚«‚Í□A‚Ü‚¸ƒCƒxƒ“ƒg ƒrƒ…□[ƒA‚ðŽQ□Æ‚µ‚Ä‚-‚¾‚³‚¢‚¢□B
	ƒpƒtƒH□[ƒ}ƒ“ƒX ƒ‚ƒjƒ^	ƒpƒtƒH□[ƒ}ƒ“ƒX ƒ‚ƒjƒ^‚Í□AŽ©•ª‚ÌƒRƒ“ƒsƒ…□[ƒ^□A‚Ü‚½‚Íƒlƒbƒgƒ□□[ƒN□ã‚Ì‚Ù‚©‚Ì□[ƒ^□A‚Ü‚½‚Íƒ□ƒj…□[ƒ^‚ÌƒpƒtƒH□[ƒ}ƒ“ƒX‚ðŠÄŽ‹‚·‚é‚½‚ß‚Ìƒ□ƒc□[ƒ‹‚Å‚·□B
	ƒT□[ƒo□[ƒ}ƒl□[ƒWƒƒ	ƒT□[ƒo□[ƒ}ƒl□[ƒWƒƒ‚Í□A“¯ˆê‚ƒhƒ□ƒCƒ“‚ÌƒNƒ□□[ƒNƒXƒe□[ƒVƒ‡ƒ“‚ÆƒT□[ƒo□[‚ðˆê——‚ð•\Ž¦‚µ‚Ü‚·□B
	ƒhƒ□ƒCƒ“ ƒt□[ƒU□[ƒ}ƒl□[ƒWƒƒ	ƒhƒ□ƒCƒ“ ƒt□[ƒU□[ƒ}ƒl□[ƒWƒƒ‚ðŽg‚¤ Æ□Aƒhƒ□ƒCƒ“‚Ìƒt□[ƒU□[ƒAƒJƒEƒ“ƒg‚ð□ì□¬‚Ü‚½‚Íœ□‚µ‚Ü‚·□B

		A–³Œø,É,µ,½,è,·,é,±,Æ,ª,Å,«,Ü,·□B,Ü,½□AƒZƒLƒ…ƒŠƒeƒB,ÌŒ´‘¥,ð□Ý’è,µ□Aƒ†□[ƒU□[ƒAƒJƒEƒ“ƒg,ðƒOƒ‹□[ƒv,É’Ç‰Á,·,é,±,Æ,à,Å,«,Ü,·□B
	Windows NT □f’ƒfƒvƒ□ƒOƒ‰ƒ€	Windows NT □f’ƒfƒvƒ□ƒOƒ‰ƒ€,Í□AƒRƒ“ƒsƒ…□[ƒ^,ÌƒŠƒ\□[ƒX,ÉŠÖ,·,é□î•ñ,ð•\Ž¦,µ,Ü,·□B

’□ˆÓ ,±,ê,ç,Ì Windows NT ,ÌŠÇ—□ƒc□[ƒ‹,ðŽg,¤,É,Í□AŠÇ—□ŽÒŒ ŒÀ,ðŽ□,ÂƒAƒJƒEƒ“ƒg,ÅƒRƒ“ƒsƒ…□[ƒ^,Éƒ□ƒOƒIƒ“,·,é•K—v,ª, ,è,Ü,·□B

ŠÖ˜A□î•ñ

□wŠÇ—□ŽÒƒKƒCƒh□x

,æ,Šñ,¹,ç,ê,éŽ¿–â (FAQ)

Microsoft Transaction Server ,É,Â,¢,Ä,æ,Šñ,¹,ç,ê,éŽ¿–â,Í□Ahttp://www.microsoft.com/japan/support/ ,É‹L□Ú,³,ê,Ä,¢,Ü,·□B

Microsoft Transaction Server ,ÌƒNƒCƒbƒN ƒcƒA□[

Microsoft® Transaction Server (MTS) ,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒgƒx□[ƒX,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□ˆ—□ƒVƒXƒeƒ€ ,Æ,µ,Ä□A□,□«”\,ÅƒXƒP□[ƒ‰ƒuƒ‹□A,©,ÂŒˇ˜S,ÈŠé‹ÆŒü,¯ƒCƒ“ƒ^□[ƒlƒbƒg□A,¨,æ,ÑƒCƒ“ƒgƒ‰ƒlƒbƒg ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŠJ”□A”z’u□A,¨,æ,ÑŠÇ—□‹@”\,ð’ñ‹Ÿ,µ,Ü,·□B

ˆÈ‰º,Ìƒgƒsƒbƒ‰N,Å,Í□AMicrosoft Transaction Server ,ÌŠe‹@”\,É,Â,¢,Ä□à–¾,µ,Ü,·□B

- Microsoft Transaction Server ,Æ,Í
- Microsoft Transaction Server ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«
- Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰
- Microsoft Transaction Server API
- Microsoft Transaction Server ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“

Microsoft Transaction Server ,Æ,Í

MTS ,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒgƒx□[ƒX,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□^—□ƒVƒXƒeƒ€
,Æ,µ,Ä□AŒ˜˜S,ÈƒCƒ“ƒ^□[ƒlƒbƒg,¨,æ,ÑƒCƒ“ƒgƒ‰ƒlƒbƒg ƒT□[ƒo□[ƒAƒvƒSƒP□[ƒVƒ‡ƒ“,Ì□\'z□A”z'u□A,¨,æ,ÑŠÇ—
□‹@”\,ð'ñ‹Ÿ,µ,Ü,·□B,Ü,½□AMTS ,Å,Í□A–L•x,È‹@”\,ð”õ,¦,½ƒOƒ‰ƒtƒBƒJƒ‹ ƒc□[ƒ‹ (MTS ƒGƒNƒXƒvƒ□□[ƒ‰)
,ðŽg,Á,Ä MTS ƒT□[ƒo□[ƒAƒvƒSƒP□[ƒVƒ‡ƒ“,ð”z'u,µ□AŠÇ—□,·,é,±,Æ,ª,Å,«,Ü,·□B

MTS ,ÍŽŸ,Ì,æ,¤,È‹@”\,ð'ñ‹Ÿ,µ,Ü,·□B

- MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«□B
- ƒAƒvƒSƒP□[ƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì”z'u,ÆŠÇ—□,Ì,½,ß,ÌƒOƒ‰ƒtƒBƒJƒ‹
  ƒt□[ƒU□[ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Å, ,é MTS ƒGƒNƒXƒvƒ□□[ƒ‰□B
- ƒXƒP□[ƒ‰ƒuƒ‹,ÅŒ˜˜S,ÈƒAƒvƒSƒP□[ƒVƒ‡ƒ“,Ì□\'z,Ì,½,ß,ÌƒAƒvƒSƒP□[ƒVƒ‡ƒ“
  ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Æ "ƒŠƒ\□[ƒX ƒƒBƒXƒyƒ“ƒT"□BƒŠƒ\
  □[ƒX ƒƒBƒXƒyƒ“ƒT,Í□AƒvƒZƒX“à•”,ÌƒAƒvƒSƒP□[ƒVƒ‡ƒ“ ƒRƒ“ƒ|
  □[ƒlƒ“ƒg,É'ã,í,Á,Ä□AŽ□'±,µ,È,¢‹¤—L□ó'Ô,ðŠÇ—□,·,éƒT□[ƒrƒX,Å,·□B
- ƒAƒvƒSƒP□[ƒVƒ‡ƒ“ ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒCƒ“ƒ^□[ƒtƒFƒCƒX (API) ,ðŽg,Á,Ä
  MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□ì□¬,·,é•û–@,Æ□AŠÇ—□ƒXƒNƒŠƒvƒg
  ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å,Ì”z'u□^—□,ðŽ©“®
  ‰»,·,é•û–@,ðŽ¦,· 3 ,Â,ÌƒTƒ“ƒvƒ‹ ƒAƒvƒSƒP□[ƒVƒ‡ƒ“□B

MTS ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒ,ƒfƒ‹,Í□AƒrƒWƒIƒX ƒ□ƒWƒbƒN,ðƒJƒvƒZƒ‹‰»,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŠJ”-
,·,é,½,ß,ÌƒtƒŒ□[ƒ€ƒ□□[ƒN,ð'ñ‹Ÿ,µ,Ü,·□BMTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,Í□A,±,ê,ç,ÌƒRƒ“ƒ|
□[ƒlƒ“ƒg,ðŽÀ□s,·,é,½,ß,Ì'†ŠÔ'w,Ìƒvƒ‰ƒbƒgƒtƒH□[ƒ€,Å,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AMTS ƒ
‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÅŽÀ□s,³,ê,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ð‘o˜^,µ□AŠÇ—□,·,é,±,Æ,ª,Å,«,Ü,·□B

,±,Ì,æ,¤,È 3 ŠK'w,Ìƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒ,ƒfƒ‹,É,æ,Á,Ä□AŠJ”ŽÒ,ÆŠÇ—□ŽÒ,Í 2
ŠK'w,ÌƒNƒ‰ƒCƒAƒ“ƒg,ƒT□[ƒo□[ƒAƒvƒSƒP□[ƒVƒ‡ƒ“,Ì□§–ñ,ð□æ,è‰z,¦,é‰Â”\□«,ð“¾,Ü,·□B3
ŠK'w,ÌƒAƒvƒSƒP□[ƒVƒ‡ƒ“,Å,Í□AŽŸ,ÉŽ¦,·—□—R,©,ç□A”z'u,ÆŠÇ—□,Ì□_“î□«,ªŒü□ã,µ,Ü,·□B

- 3 ŠK'w,Ìƒ,ƒfƒ‹,Å,Í□AƒAƒvƒSƒP□[ƒVƒ‡ƒ“,Ì•¨—□ƒA□[ƒLƒeƒNƒ`ƒƒ,æ,è˜_—□ƒA□[ƒLƒeƒNƒ`ƒƒ,Ì,Ù,¤
  ,ª□dŽ‹,³,ê,Ü,·□B,Ç,ÌƒT□[ƒrƒX,©,ç,Å,à,Ù,©,ÌƒT□[ƒrƒX,ðŒÄ,Ñ□o,·,±,Æ,ª,Å,«,Ü,·□B,Ü,½□AƒT□[ƒrƒX,ª,Ç,±
  ,É'¶□Ý,µ,Ä,¢,Ä,à,©,Ü,¢,Ü,¹,ñ□B
- ,±,ê,ç,ÌƒAƒvƒSƒP□[ƒVƒ‡ƒ“,Í”z•z,³,ê,Ü,·□B,±,ê,Í□A“K□Ø,ÈƒRƒ“ƒ|□[ƒlƒ“ƒg,ð“K□Ø,È□ê□Š,ÅŽÀ□s,Å,«,é,±
  ,Æ,ðˆÓ–¡,µ,Ä,¢,Ü,·□B,»,ÌŒ‹‰Ê□Aƒ†□[ƒU□[,É—˜“_,ª, ,é,Æ“¯Žž,É□Aƒlƒbƒgƒ□□[ƒN ƒŠƒ\□[ƒX,ÆƒRƒ“ƒsƒ…
  □[ƒ^ ƒŠƒ\□[ƒX,ÌŽg—pŒø—¦,ª□Å“K‰»,³,ê,Ü,·□B

ŠÖ˜A□€–Ú

ƒhƒLƒ…ƒ□ƒ“ƒg,Ì□\□¬

Microsoft Transaction Server ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«

MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,É,æ,Á,Ä□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŠJ”ŽÒ,âƒVƒXƒeƒ€ŠÇ—□ŽÒ,Í□A•ïŠ‡“I,ÅŽg,¢,â,·,¢ˆê˜A,ÌƒVƒXƒeƒ€ ƒT□[ƒrƒX,ð—˜—p,µ□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð—eˆÕ,ÉŠJ”□A”z’u□A,¨,æ,ÑŠÇ—□,Å,«,é,æ,¤,É,È,è,Ü,·□B,±,ê,ç,ÌƒT□[ƒrƒX,É,Í□AŽŸ,Ì,æ,¤,È,à,Ì,ª, ,è,Ü,·□B

- •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□B"ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“" ,Æ,Í□AƒAƒgƒ~ƒbƒN,È□ˆ—□,Æ,µ,ÄŽÀ□s,³,ê,é“®□ì,Ì’Pˆ Ê,Ì,±,Æ,Å,·□B,Â,Ü,è□Aˆ—□,Í‘S‘Ì,Æ,µ,Ä□¬Œ÷,·,é,©□AŽ¸”s,·,é,©,Ì,¢,¸,ê,©,ÌŒ‹‰Ê,É,È,è,Ü,·□B
- ƒvƒ□ƒZƒX,ÆƒXƒŒƒbƒh,ÌŽ©“®ŠÇ—□□B
- ƒIƒuƒWƒFƒNƒg ƒCƒ“ƒXƒ^ƒ“ƒXŠÇ—□□B
- ƒIƒuƒWƒFƒNƒg,Ì□ì□¬,ÆŽg—p,ð□§Œä,·,é•ªŽUƒZƒLƒ…ƒŠƒeƒB ƒT□[ƒrƒX□B
- ƒVƒXƒeƒ€ŠÇ—□,ÆƒRƒ“ƒ|□[ƒIƒ“ƒgŠÇ—□,Ì,½,ß,ÌƒOƒ‰ƒtƒBƒJƒ‹ ƒCƒ“ƒ^□[ƒtƒFƒCƒX□B

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŠJ”ŽÒ,Í□A,±,ê,ç,ÌƒVƒXƒeƒ€ ƒT□[ƒrƒX,ð—˜—p,·,é,±,Æ,É,æ,Á,Ä□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉƒXƒP□[ƒ‰ƒrƒŠƒeƒB,ÆŒ˜˜S□«,ð,à,½,ç,·,±,Æ,ª,Å,«,Ü,·□B,±,Ì,½,ß□AƒVƒXƒeƒ€ŠÂ‹«,ÌŠJ”,©,ç‰ð•ú,³,ê□A‹Æ–±□ã,Ì–â‘è‰ðŒˆ,É□W’†,Å,«,é,æ,¤,É,È,è,Ü,·□B

MTS ,Í□AActiveX™ ƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL) ,ð□¶□¬,Å,«,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŠJ”-ƒc□[ƒ‹,Å, ,ê,Î□A,Ç,Ì,æ,¤,Èƒc□[ƒ‹,Æ,Å,à‘g,Ý□‡,í,¹,ÄŽg—p,Å,«,Ü,·□B,½,Æ,¦,Î□AMicrosoft Visual Basic®□AMicrosoft Visual C++®□AMicrosoft Visual J++®□A,Ü,½,Í,»,Ì,Ù,©,Ì ActiveX ƒc□[ƒ‹,ðŽg,Á,Ä□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŠJ”,Å,«,Ü,·□B

MTS ,Í□AƒŠƒŒ□[ƒVƒ‡ƒiƒ‹ ƒf□[ƒ^ƒx□[ƒX ƒVƒXƒeƒ€□Aƒtƒ@ƒCƒ‹ ƒVƒXƒeƒ€□AƒhƒLƒ…ƒ□ƒ“ƒgŠi”[ƒVƒXƒeƒ€,È,Ç□A,³,Ü,´,Ü,ÈƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,Æ‘g,Ý□‡,í,¹,Ä—˜—p,Å,«,é,æ,¤,É□ÝŒv,³,ê,Ä,¢,Ü,·□B,±,Ì,½,ß□AŠJ”-ŽÒ,âƒ\ƒtƒgƒEƒFƒA ƒ□□[ƒJ□[,Í□A,³,Ü,´,ÜƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,Ì’†,©,ç•K—v,È,à,Ì,ð‘I‘ð,µ□Aƒ□□[ƒJƒ‹ ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ü,½,Í•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì—˜“_,ðŠˆ—p,µ,È,ª,ç□A’Pˆê,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Å•¡□”,ÌƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,ðŠÈ’P,É—˜—p,·,é,±,Æ,ª,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

MTS Overview and Concepts (ŠT—v,Æƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO,ÌŠT”O)

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰

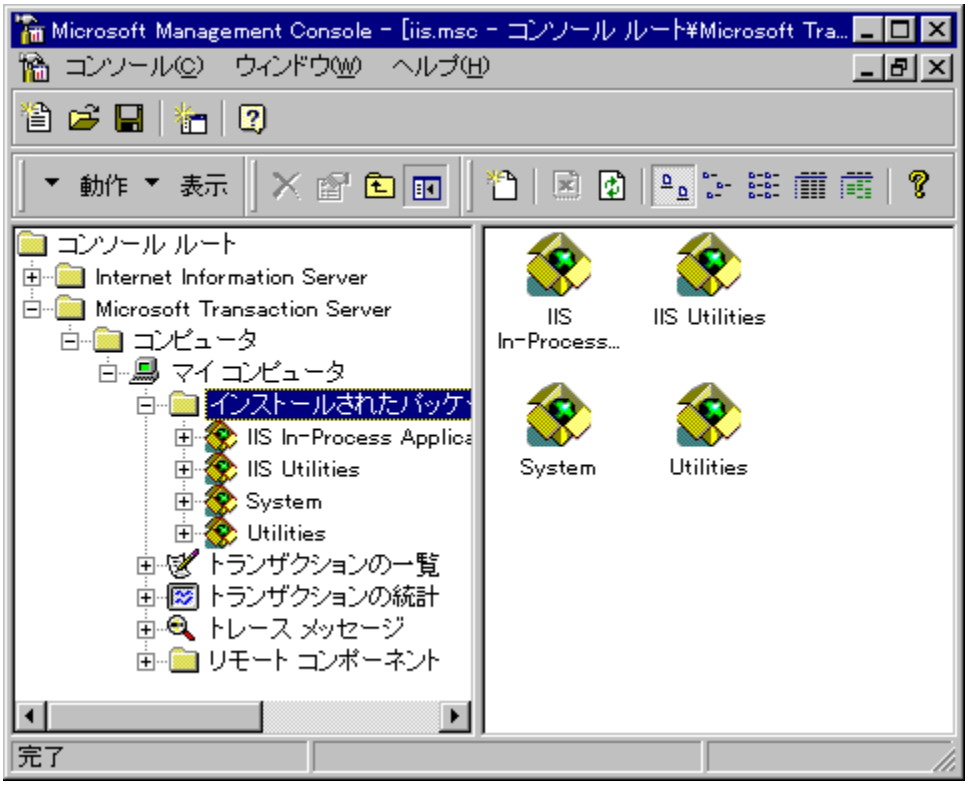
MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Í□AMTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŠÇ—□,µ□A"z'u,·,é,½,ß,ÌƒOƒ‰ƒtƒBƒJƒ‹ ƒt□[ƒU□[ƒCƒ"ƒ^□[ƒtƒFƒCƒX,Å,·□BŠJ"ŽÒ,Æ"¯—l,ÉƒVƒXƒeƒ€ŠÇ—□ŽÒ,â Web ŠÇ—□ŽÒ,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒpƒbƒP□[ƒW,ÌŠÇ—□□A"z•z□AƒCƒ"ƒXƒg□[ƒ‹□A"z'u□A,¨,æ,ÑƒeƒXƒg,ð□s,¤,±,Æ,ª,Å,«,Ü,·□BŠJ"-ŽÒ,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AƒRƒ"ƒ|□[ƒlƒ"ƒg,ð'g,Ý□‡,í,¹,ÄŠù□¬,ÌƒpƒbƒP□[ƒW,ð□ì□¬,µ,½,è□AMTS ŠÂ‹«,ÅƒRƒ"ƒ|□[ƒlƒ"ƒg,ð"z•z,µ,½,è□AƒeƒXƒg,µ,½,è,·,é,±,Æ,ª,Å,«,Ü,·□B,Ü,½□AŠÇ—□ŽÒ,âŠJ"ŽÒ,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒRƒ"ƒ|□[ƒlƒ"ƒg,ÆƒpƒbƒP□[ƒW,ÌƒCƒ"ƒXƒg□[ƒ‹□A"z'u□A,¨,æ,Ñ•ÛŽç,ð□s,¤,± ,Æ,à,Å,«,Ü,·□B,³,ç,É□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,¤,Æ□Aƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ" ƒRƒ"ƒ|□[ƒlƒ"ƒg,ÌƒgƒŠ‰ƒ"ƒUƒNƒVƒ‡ƒ",ðŠÄŽ‹,µ□AŠÇ—□,·,é,±,Æ,ª,Å,«,Ü,·□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK'w,Í□AŽŸ,Ì,æ,¤,Èƒ‰ƒ"ƒ^ƒCƒ€ŠÂ‹«,ÌƒAƒCƒeƒ€,ª,Ç,Ì,æ,¤,É□\□¬,³,ê,Ä,¢,é,©,ð•\ ,µ,Ü,·□B

- ƒRƒ"ƒsƒ…□[ƒ^
- ƒpƒbƒP□[ƒW
- ƒRƒ"ƒ|□[ƒlƒ"ƒg
- ƒ□□[ƒ‹
- ƒCƒ"ƒ^□[ƒtƒFƒCƒX
- ƒ□ƒ\ƒbƒh

MTS ƒpƒbƒP□[ƒW,ÍƒRƒ"ƒsƒ…□[ƒ^,ÉƒCƒ"ƒXƒg□[ƒ‹,³,ê□AƒRƒ"ƒ| □[ƒlƒ"ƒg,ðŠÜ,Ý□Aƒ□□[ƒ‹,ð'è‹`,µ,Ü,·□Bƒpƒbƒp□[ƒW"à,ÌƒRƒ"ƒ|□[ƒlƒ"ƒg,ÍƒCƒ"ƒ^□[ƒtƒFƒCƒX,Æƒ□ƒ\ ƒbƒh,ð'è‹`,µ,Ü,·□B□ê—p,ÌƒEƒBƒ"ƒhƒE,ðŽg,Á,Äƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ",ÆƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW,ÉŠÖ,·,é□î•ñ,ð•\ Ž¦,·,é,±,Æ,ª,Å,«,Ü,·□B

ŽŸ,Ì□},Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ"ƒhƒE,Ì□¶'¤,ÉŠK'w,ª,Ç,Ì,æ,¤,É•\Ž¦,³,ê,é,©,ðŽ¦,µ,Ä,¢,Ü,·□B

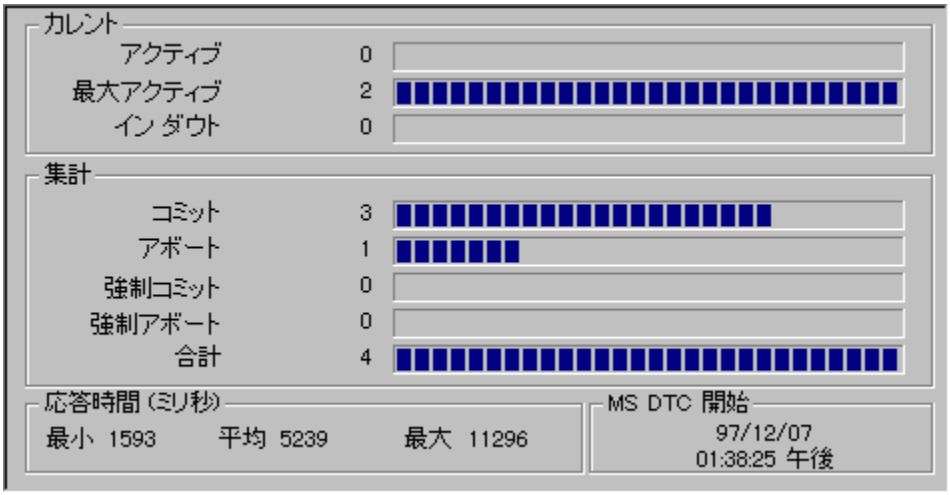


ƒvƒ□ƒpƒeƒB•\Ž¦,ÌƒEƒBƒ"ƒhƒE,ðŽg,Á,Ä□AƒpƒbƒP□[ƒW"à,ÌƒRƒ"ƒ|□[ƒlƒ"ƒg,Ìƒvƒ□ƒpƒeƒB,âƒRƒ"ƒsƒ…

□[ƒ^,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,ð•\Ž¦,·,é,±,Æ,ª,Å,«,Ü,·□B

Prog ID	トランザクション	DLL	CLSID	スレッド	セキュリティ
Bank. Account	必要	C:¥Progra...	{731A63A...	Apartment	Y
Bank. Account. VC	必要	C:¥Progra...	{04CF0B7...	Both	Y
Bank. Account. VJ	必要	C:¥Progra...	{9FAF8612...	Both	Y
Bank. CreateTable	新しく必要	C:¥Progra...	{731A63B5...	Apartment	Y
Bank. GetReceipt	サポート	C:¥Progra...	{731A63B3...	Apartment	Y
Bank. GetReceipt. VC	サポート	C:¥Progra...	{A81260B2...	Both	Y
Bank. GetReceipt. VJ	サポート	C:¥Progra...	{AB077646...	Both	Y
Bank. MoveMoney	必要	C:¥Progra...	{731A63B1...	Apartment	Y
Bank. MoveMoney. VC	必要	C:¥Progra...	{04CF0B7...	Both	Y
Bank. MoveMoney. VJ	必要	C:¥Progra...	{B790937A...	Both	Y
Bank. UpdateReceipt	新しく必要	C:¥Progra...	{731A63B7...	Apartment	Y
Bank. UpdateReceipt. VC	新しく必要	C:¥Progra...	{A81260B8...	Both	Y
Bank. UpdateReceipt. VJ	新しく必要	C:¥Progra...	{C25E3B4...	Both	Y

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv] ƒEƒBƒ“ƒhƒE,ðŽg,Á,Ä□A□Å‹ß,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì□ó‘Ô,ðŠT—ª,Æ,µ,Ä•\
,·“□Œv□î•ñ,ð•\Ž¦,·,é,±,Æ,à,Å,«,Ü,·□B



MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì‘€□ì

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,É,ÍŠK‘w,ª•\Ž¦,³,ê□A‰E‘¤,É,Í□¶‘¤,ÅƒNƒŠƒbƒN,µ,½ƒAƒCƒef€,Ì“à—e,ª•\Ž¦,³,ê,Ü,·□BŠK‘w,Í□AƒtƒHƒ‹ƒ_,Æ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□\□¬,Å,«,é,·,×,Ä,ÌƒAƒCƒef€,ðŠÜ,ñ,¾ƒcƒŠ□[□\‘¢,É,È,Á,Ä,¢,Ü,·□B

ƒAƒCƒef€,Ì“à—e,ð•\Ž¦,·,é,É,Í□AƒEƒBƒ“ƒhƒE,Ì‰E‘¤,ÅƒtƒHƒ‹ƒ_,âƒAƒCƒef€,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ,Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK‘w,ðˆÚ“®,µ,Ü,·□BƒEƒBƒ“ƒhƒE,Ì□¶‘¤,ÅƒtƒHƒ‹ƒ_,âƒAƒCƒef€,ðƒNƒŠƒbƒN,µ,Ä□A‰E‘¤,É,»,Ì“à—e,ð•\Ž¦,·,é,±,Æ,à,Å,«,Ü,·□BŠK‘w,Ì’†,Ì”CˆÓ,ÌƒAƒCƒef€,ð“WŠJ,·,é,É,Í□AƒAƒCƒef€,Ì‰¡,ÌƒvƒoƒX‹L□† (+) ,ðƒNƒŠƒbƒN,µ,Ü,·□B,»,ÌƒAƒCƒef€,ÌŠK‘w,ªƒEƒBƒ“ƒhƒE,Ì□¶‘¤,É•\Ž¦,³,ê,Ü,·□BƒEƒBƒ“ƒhƒE,Ì□¶‘¤,ÅƒtƒHƒ‹ƒ_,âƒAƒCƒef€,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,·,é,Æ□A,»,Ì“à—e,ª‰E‘¤,É•\Ž¦,³,ê□AŠK‘w,Ì•\Ž¦,ð“WŠJ,µ,½,è□A□Ü,è,½,½,ñ,¾,è,Å,«,Ü,·□B

ƒEƒBƒ“ƒhƒE,Ì□¶‘¤□A‰E‘¤,Ì‚Ç,¿,ç,Å,à□A•ûŒüƒL□[,ðŽg,Á,ÄƒAƒCƒef€,ð‘I‘ð,·,é,±,Æ,ª,Å,«,Ü,·□BƒEƒBƒ“ƒhƒE,Ì‰E‘¤,Å **Enter** ƒL□[,ð‰Ÿ,·,Æ□A,»,ÌƒAƒCƒef€,Ì“à—e,ª•\Ž¦,³,ê,Ü,·□B**Tab** ƒL□[,ð‰Ÿ,·,Æ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‰E,ðŒðŒÝ,ÉˆÚ“®,Å,«,Ü,·□B

Windows® 95/98 ,Å,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,ÉƒcƒŠ□[□\‘¢,ª•\Ž¦,³,ê,È,¢,± ,Æ,É’□ˆÓ,µ,Ä,,¾,³,¢□BŠK‘w,ð
‰º,É,ÍˆÚ“®,·,é,É,ÍƒAƒCƒRƒ“,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ□AŠK‘w,ð□ã,ÉˆÚ“®,·,é,É,Íƒc□[ƒ‹ ƒo□[,Ì [1 ,Â□ã,ÌƒfŒƒxƒ‹]

ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□BWindows 95/98 ,Å,Ì MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŽg,¢•û,Ì□Ú□×,É,Â,¢,Ä,Í□A□wŠÇ—□ŽÒƒKƒCƒh□x,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒAƒCƒeƒ€,Ìƒvƒ□ƒpƒeƒB,Ì□Ý’è,Æ•\Ž¦

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK‘w,É’Ç‰Á,³,ê,½ƒAƒCƒeƒ€,ÉŠÖ,·,éŠî–{“I,È□î•ñ,Í□A,»,ÌƒAƒCƒeƒ€,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,É•\Ž¦,³,ê,Ü,·□B•\Ž¦,³,ê,é□î•ñ,Í□AƒAƒCƒeƒ€,ÌŽí—Þ,É,æ,Á,Äˆ Ù,È,è,Ü,·□B,½,Æ,¦,Î□AƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,É,Í□AƒRƒ“ƒsƒ…□[ƒ^–¼□Aƒ□ƒO ƒtƒ@ƒCƒ‹,Ì□ê□Š□A□X□V,Ì□Ý’è,È,Ç,ª•\Ž¦,³,ê,Ü,·□BƒpƒbƒP□[ƒW,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,É,Í□AƒZƒLƒ…ƒŠƒeƒB,â,»,Ì,Ù,©,Ìƒvƒ□ƒZƒXŒÅ—L,Ì□Ý’è,ÉŠÖ,·,é□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

[ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ð•\Ž¦,·,é,É,Í□AƒAƒCƒeƒ€,ð‘I‘ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ì [ƒvƒ□ƒpƒeƒB] ,ðƒNƒŠƒbƒN,·,é,©□A,Ü,½,ÍƒAƒCƒeƒ€,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒpƒeƒB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B[ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,É,Í•¡□”,Ìƒ^ƒu,ª, ,è□A,»,ê,¼,ê,Ìƒ^ƒu,ðƒNƒŠƒbƒN,µ,Äƒ^ƒu,ð□Ø,è‘Ö,¦,Ü,·□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹,Æ‰ðŒˆ

MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,É,Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□ˆ—□,ð□L”Íˆ Í,ÉƒTƒ|□[ƒg,·,é‹@”\,ª—pˆÓ,³,ê,Ä,¢,Ü,·□BMicrosoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒl□[ƒ^ (MS DTC) ,Í□AMTS ,Å,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□ˆ—□,ðƒTƒ|□[ƒg,µ,Ü,·□BMS DTC ,Í Windows NT® ,ÌƒT□[ƒrƒX,Å,·□BMTS ,Í MS DTC ,ðŽg,Á,Ä□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠ®—¹‘O,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì,·,×,Ä,ÌŠÖ˜A•”•ª,ª¯ˆÓ□ó‘Ô,É, ,é,±,Æ,ðŠm”F,µ,Ü,·□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌƒTƒ|□[ƒg,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK‘w,Ì’†,Å□AŽŸ,Ì 3 ,Â,ÌƒEƒBƒ“ƒhƒE,ð’Ê,¶,Ä’ñ‹Ÿ,³,ê,Ü,·□B

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——] ƒEƒBƒ“ƒhƒE,Í□AƒAƒNƒeƒBƒu ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì□ó‘Ô,ðŠÄŽ‹,·,é,Æ,«,ÉŽg,¢,Ü,·□B

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv] ƒEƒBƒ“ƒhƒE,Í□A□Å‹ß,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv□î•ñ,ð•\Ž¦,·,é,Æ,«,ÉŽg,¢,Ü,·□B

[ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ƒEƒBƒ“ƒhƒE,Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□ˆ—□,ÉŠÖ˜A,·,éƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW,ð•\Ž¦,·,é,Æ,«,ÉŽg,¢,Ü,·□B

ŠÖ˜A□€–Ú

□wŠÇ—□ŽÒƒKƒCƒh□x

Microsoft Transaction Server API

MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒCƒ“ƒ^□[ƒtƒFƒCƒX (API) ,ðŽg—p,µ,Ä MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,Ì‹@”\ ,ðŠˆ—p,·,éƒXƒP□[ƒ‰ƒuƒ‹,ÅŒ˜˜S,ÈƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŠJ”,µ,½,è□AƒpƒbƒP□[ƒW,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŠÇ— □,ðŽ©“®‰»,µ,½,è,Å,«,Ü,·□B

ƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŠJ”

MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÌŠO•”,ÅŽÀ□s,³,ê,éƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í□A•W□€,Ì COM ƒ‰ƒCƒuƒ ‰ƒŠŠÖ□” (C++ ,Å,Í **CoCreateInstance**□AVisual Basic ,Å,Í **CreateObject** ƒ□ƒ\ƒbƒh) ,ðŽg,Á,Ä MTS ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŠJ”

MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg (MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,É“o˜^,³,ê,éƒT□[ƒo□[ƒRƒ“ƒ|□[ƒlƒ“ƒg) ,ðŠJ”,·,é□ê□‡,Í□AMTS ,Ì **IObjectContext**□A**ISharedPropertyGroupManager**□A**ISharedPropertyGroup**□A,¨,æ,Ñ **ISharedProperty** ,ÌŠeƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŽg—p,µ,ÄŽŸ,Ì□ì‹Æ,ð□s,¤,±,Æ,ª,Å,«,Ü,·□B

- ƒIƒuƒWƒFƒNƒg,Ì“®□ì,ÌŠ®—¹,ð□éŒ¾,·,é□B
- ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒRƒ~ƒbƒg,³,ê,È,¢,æ,¤,É,·,é□B
- ,Ù,©,Ì MTS ƒIƒuƒWƒFƒNƒg,ð□ì□¬,·,é□B
- ,Ù,©,ÌƒIƒuƒWƒFƒNƒg,Ì“®□ì,ðƒJƒŒƒ“ƒg ƒIƒuƒWƒFƒNƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌƒXƒR□[ƒv,ÉŠÜ,ß,é□B
- ŒÄ,Ñ□o,µŒ³,ª“Á’è,Ìƒ□□[ƒ‹,ðŽ□,Á,Ä,¢,é,©,Ç,¤,©,ð”»’f,·,é□B
- ƒZƒLƒ…ƒŠƒefB,ª—LŒø,É,È,Á,Ä,¢,é,©,Ç,¤,©,ð”»’f,·,é□B

MTS ŠÇ—□,ÌŽ©“®‰»

MTS ŠÇ—□ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä□AƒpƒbƒP□[ƒW,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŠÇ—□,ðŽ©“®‰»,Å,«,Ü,·□BVisual Basic□AVisual Basic Scripting Edition (VBScript)□A,Ü,½,Í,»,Ì,Ù,©,Ì”CˆÓ,ÌƒI□[ƒgƒ□□[ƒVƒ‡ƒ“ŒÝŠ·Œ¾Œê,ðŽg,Á,Ä□AŠù□¬,ÌƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,©,çŠÖ˜AƒRƒŒƒNƒVƒ‡ƒ“,Ì—ñ‹“,Ü,Å,Ì MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì□ˆ—□,ðŽ©“®‰»,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

□wŠÇ—□ŽÒƒKƒCƒh□x□AMTS Overview and Concepts (ŠT—v,Æƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO,ÌŠT”O)□AMTS Reference (ƒŠƒtƒ@ƒŒƒ“ƒX)□AMTS Administrative Reference (ŠÇ—□ƒŠƒtƒ@ƒŒƒ“ƒX)

Microsoft Transaction Server ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“

MTS ,É,Í□AƒhƒLƒ…ƒ□ƒ“ƒg,Ì,Ù,©,É□AŠw□Kƒc□[ƒ‹,Æ,µ,Ä–ð—§,ÂƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª—pˆÓ,³,ê,Ä,¢,Ü,·□B,±,ê,ç,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìˆê•”,ðŽ©•ª,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉƒRƒs□[,µ□A•K—v,É‰ž,¶,Ä□C□³,µ,ÄŽg,¤,±,Æ,à,Å,«,Ü,·□B

□wProgrammer's Guide□x,Ì‘S‘Ì,ð’Ê,¶,Ä□AMTS ,ÌƒvƒOƒ‰ƒ~ƒ“ƒO‹Z–@,ðŽ¦,·ƒTƒ“ƒvƒ‹ ƒR□[ƒh,âƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª‹L□q,³,ê,Ä,¢,Ü,·□B,±,ê,ç,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìƒtƒ@ƒCƒ‹,Ì‘½,-,Í□AƒZƒbƒgƒAƒbƒvŽž,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ü,·□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìƒ\□[ƒX ƒtƒ@ƒCƒ‹,Í□AMTS ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒfƒBƒŒƒNƒgƒŠ,Ì \Samples ƒtƒHƒ‹ƒ_,É, ,è,Ü,·□B

MTS ,É,Í□AŽŸ,ÌƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª—pˆÓ,³,ê,Ä,¢,Ü,·□B

ƒTƒ“ƒvƒ‹	**□à–¾**
Sample Bank	MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒCƒ“ƒ^□[ƒtƒFƒCƒX ,ÌŽg,¢•û,ðŽ¦,·ŠÈ’P,Èƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒf□[ƒ^ƒx□[ƒX ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Å,·□B
Tic-Tac-Toe	‹¤—L□ó‘Ô,ðŠÇ—□,·,é”ñƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì‹@”\,ðŽ¦,·ŠÈ’P,Èƒ}ƒ‹ƒ`ƒ†□[ƒU□[ƒQ□[ƒ€,Å,·□B
Administrative Sample Scripts	VBScript ,ðŽg,Á,Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì□ˆ—□,ðŽ©“®‰»,·,é•û–@,ðŽ¦,·ŠÇ—□ƒIƒuƒWƒFƒNƒg ƒXƒNƒŠƒvƒg,Å,·□B

ŠÖ˜A□€–Ú

ŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒ□ƒ“ƒg,ÌƒCƒ“ƒXƒg□[ƒ‹□ASample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒZƒbƒgƒAƒbƒv□ATic-Tac-Toe ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒZƒbƒgƒAƒbƒv□AŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ÌƒZƒbƒgƒAƒbƒv□AMTS Overview and Concepts (ŠT—v,Æƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO,ÌŠT”O)□AMTS ŠÇ—□Ž©“®‰» Visual Basic Script ƒTƒ“ƒvƒ‹

ƒZƒbƒgƒAƒbƒv

Microsoft® Transaction Server (MTS) ,Ö,æ,¤,±,»□BMTS ,Í□A□,□«”\,ÅƒXƒP□[ƒ‰ƒuƒ‹□A,©,ÂŒ˜˜S,ÈŠé‹ÆŒü,¯ƒCƒ“ƒ^□[ƒlƒbƒg□A,¨,æ,ÑƒCƒ“ƒgƒ‰ƒlƒbƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŠJ”,Æ”z’u,ð—eˆÕ,É,·,é‹—Í,ÈŠÂ‹«,ð’ñ‹Ÿ,µ,Ü,·□BMTS ,Å,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒgƒx□[ƒX,Ì•ªŽUƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŠJ”-,·,é,½,ß,Ìƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒ,ƒƒ‹,ª’è‹`,³,ê,Ä,¢,Ü,·□B,Ü,½□A,±,ê,ç,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì”z’u,ÆŠÇ—□,Ì,½,ß,Ìƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,à’ñ‹Ÿ,µ,Ü,·□B

MTS ,ÌƒCƒ“ƒXƒg□[ƒ‹,ÆƒZƒbƒgƒAƒbƒv,Ì•û–@,É,Â,¢,Ä,Í□AŽŸ,Ìƒgƒsƒbƒ‹,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

- ƒVƒXƒefŠ€,Ì•K—v□ðŒ□
- ŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒ□ƒ“ƒg,ÌƒCƒ“ƒXƒg□[ƒ‹
- MTS ƒT□[ƒo□[,Ì□\□¬
- MTS ,ð Microsoft Cluster Server ,Æ‘g,Ý□‡,í,¹,ÄŽg,¤,æ,¤,É□\□¬,·,é
- MTS ,ð Oracle ,ÉƒAƒNƒZƒX,Å,«,é,æ,¤,É□Ý’è,·,é
- Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒZƒbƒgƒAƒbƒv
- Tic-Tac-Toe ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒZƒbƒgƒAƒbƒv
- ŠÇ—□ƒIƒufƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ÌƒZƒbƒgƒAƒbƒv
- MTS ,ÉŠÖ,·,é□î•ñ,Ì“üŽè

ƒVƒXƒefƒ€‚Ì•K—v□ðŒ□

‚±‚±‚Å‚Í□AMTS ‚Ìƒ‰Cƒ“ƒXƒg□[ƒ‹‚É•K—v‚Èƒn□[ƒhƒEƒFƒA‚Æƒ\ftƒgƒEƒFƒA‚Ì□ðŒ□□AƒZƒbƒgƒAƒbƒv‚ÉŠÖ‚·‚é'□ˆÓŽ–□€□A‚¨‚æ‚Ñ Alpha ƒvƒ‰ƒbƒgƒtƒH□[ƒ€‚É‘Î‚·‚é MTS ‚ÌƒTƒ|□[ƒg‚É‚Â‚¢‚Ä□à–¾‚µ‚Ü‚·□B

MTS ‚ÍŽŸ‚Ì‚¢‚¸‚ê‚©‚Ì•û–@‚ÅRƒ“ƒsƒ…□[ƒ^‚ÉƒCƒ“ƒXƒg□[ƒ‹‚Å‚«‚Ü‚·□B

- Windows NT® Version 4.0 Option Pack ‚ðŽg‚Á‚Ä□AInternet Information Server (IIS) ‚È‚Ç‚Ì Option Pack ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Æ‚ê□□‚É MTS ‚ðƒCƒ“ƒXƒg□[ƒ‹‚·‚é•û–@□B
- Windows NT Version 4.0 Option Pack ‚ðŽg‚Á‚Ä□A‚Ù‚©‚Ì Option Pack ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Æ‚Í•Ê‚É MTS ‚¾‚¯‚ðƒCƒ“ƒXƒg□[ƒ‹‚·‚é•û–@□B

MTS ‚Í□AWindows NT□AWindows® 98□A‚¨‚æ‚Ñ DCOM ‚ªƒCƒ“ƒXƒg□[ƒ‹‚³‚ê‚Ä‚¢‚é Windows® 95 □ã‚Å“®□ì‚µ‚Ü‚·□B

□**d—v:** IIS 2.0 ‚Ü‚½‚Í IIS 3.0 ‚ðŠù‚ÉŽg—p‚µ‚Ä‚¢‚é□ê□‡‚Í□AWindows NT Option Pack ‚Ì IIS 4.0 ‚Æ‹¤‚É MTS ‚ðƒCƒ“ƒXƒg□[ƒ‹‚µ‚Ä‚¾‚³‚¢□B

‚Ù‚©‚Ì Option Pack ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Æ‚Í•Ê‚É MTS ‚¾‚¯‚ðƒCƒ“ƒXƒg□[ƒ‹‚·‚é‚Í□AŽŸ‚Ì'€□ì‚ð□s‚¢□B

1. Option Pack ƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€‚ðŽÀ□s‚µ□A[ƒJƒXƒ^ƒ€] ‚ðƒNƒŠƒbƒN‚µ□B
2. ‚·‚×‚Ä‚Ì Option Pack ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒ`ƒFƒbƒN ƒ{ƒbƒNƒX‚ðƒIƒt‚É‚µ□B
3. ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX‚ÍƒIƒ“‚É‚¹‚¸‚É [Transaction Server] ‚ð'I'ð‚µ□B
4. [ƒTƒuƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ì•\Ž¦] ‚ðƒNƒŠƒbƒN‚µ□B
5. [Microsoft Transaction Server ƒRƒA ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX‚ðƒIƒ“‚É‚µ□B[Microsoft Management Console] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX‚àƒIƒ“‚É‚È‚Ü‚·□B

   [Transaction Server Development] ƒIƒvƒVƒ‡ƒ“‚ð'I'ð‚·‚é‚Æ□Aƒf□[ƒ^ ƒAƒNƒZƒX ƒRƒ“ƒ|□[ƒlƒ“ƒg‚àƒCƒ“ƒXƒg□[ƒ‹‚³‚ê‚é‚±‚Æ‚É'□ˆÓ‚µ‚Ä‚¾‚³‚¢□B
6. [OK] ‚ðƒNƒŠƒbƒN‚µ□AƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€‚ðŒp'±‚µ‚Ü‚·□B

ƒn□[ƒhƒEƒFƒA‚Ì•K—v□ðŒ□

Microsoft Transaction Server ‚ðƒCƒ“ƒXƒg□[ƒ‹‚·‚é'O‚É□AƒRƒ“ƒsƒ…□[ƒ^‚ªŽŸ‚Ì•K—v□ðŒ□‚ð–ž‚½‚µ‚Ä‚¢‚é‚±‚Æ‚ðŠm”F‚µ‚Ä‚¾‚³‚¢□B

- Windows NT□AWindows 98□A‚Ü‚½‚Í DCOM ‚ªƒCƒ“ƒXƒg□[ƒ‹‚³‚ê‚Ä‚¢‚é Windows 95 ‚ðŽÀ□s‚µ‚Ä‚¢‚é i386 ŒÝŠ·ƒRƒ“ƒsƒ…□[ƒ^‚Ü‚½‚Í Alpha AXP™ ƒRƒ“ƒsƒ…□[ƒ^□Bƒn□[ƒhƒEƒFƒA‚Ì•K—v□ðŒ□‚Ì□Å□V□î•ñ‚É‚Â‚¢‚Ä□AMTS Readme ‚ðŽQ□Æ‚µ‚Ä‚¾‚³‚¢□B
- ‚·‚×‚Ä‚Ìƒ|ƒRƒ“ƒg‚ðƒCƒ“ƒXƒg□[ƒ‹‚·‚é□ê□‡‚Í□A30 MB ˆÈ□ã‚Ì‹ó‚«—e—Ê‚ª ‚éfn□[ƒh ƒfƒBƒXƒN□B
- CD-ROM ƒhƒ‰ƒCƒu□B
- Windows NT Version 4.0 ˆÈ□~‚Ü‚½‚Í Windows 95/98 'Î‰ž‚Ìƒffƒ‰ƒXƒvƒŒƒC□B
- 32MB ˆÈ□ã‚Ìƒf□ƒƒŠ□B
- ƒ}ƒEƒX‚Ü‚½‚Í‚»‚Ì‚Ù‚©‚Ìƒ|ƒCƒ“ƒeƒBƒ“ƒO ƒffƒofCƒX□B

Windows NT Version 4.0 Option Pack ƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€‚ðŽg‚Á‚ÄAMTS ‚Ìƒ‰ƒZƒbƒgƒAƒbƒv‚ð–³□lƒ‚□[ƒh‚ÅŽÀ□s‚Å‚«‚Ü‚·□B–³□lƒZƒbƒgƒAƒbƒv‚ðŽÀ□s‚·‚é'O‚É□AƒZƒbƒgƒAƒbƒv ƒtƒ@ƒCƒ‹‚ðƒZƒbƒgƒAƒbƒv‚Ì'I'ð□€–Ú‚É‰ž‚¶‚Ä□X‚µ‚Ä‚¾‚³‚¢□B–³□lƒZƒbƒgƒAƒbƒv‚Ì□Ú□×‚É‚Â‚¢‚Ä□AWindows NT 4.0 Option Pack ƒlƒ“ƒ‰ƒCƒ“ ƒhƒLƒ…ƒ□ƒ“ƒg‚Ì□wƒtƒ@□[ƒXƒg ƒXƒefƒbƒv ƒKƒCƒh□x‚Ì□uWindows NT 4.0 Option Pack ‚ÌƒZƒbƒgƒAƒbƒv□v‚Ì□uOption Pack ‚ðƒCƒ“ƒXƒg□[ƒ‹□v‚Ì□uOption Pack ‚Ì–³□lƒCƒ“ƒXƒg□[ƒ‹□v‚ðŽQ□Æ‚µ‚Ä‚¾‚³‚¢□B

–³□lƒZƒbƒgƒAƒbƒv‚ðŽÀ□s‚·‚é‚Í□A[ƒXƒ^□[ƒg ƒ{ƒ^ƒ“‚ðƒNƒŠƒbƒN‚µ□A[ƒtƒ@ƒCƒ‹‚ð–¼'è‚µ‚ÄŽÀ□s] ‚ðƒNƒŠƒbƒN‚µ□BŽŸ‚É□AŽŸ‚Ì‚æ‚¤‚É“ü—Í‚µ‚Ü‚·□B

```
setup /u:unattend.txt
```

ƒ\ƒtƒgƒEƒFƒA‚Ì•K—v‚ðŒ‾

ƒ\ƒtƒgƒEƒFƒA‚Ì•K—v‚ðŒ‾‚ÍŽŸ‚Ì‚Æ‚¨‚è‚Å‚·‚B

- MTS ‚ðƒCƒ“ƒXƒg[ƒ‹‚·‚é‘O‚É‚AWindows NT Version 4.0 ^È‚~‚AWindows 98‚A‚Ü‚½‚Í Windows 95 ‚Æ DCOM for Windows 95 ‚ðƒRƒ“ƒsƒ…[ƒ^‚ÉƒCƒ“ƒXƒg[ƒ‹‚µ‚È‚¯‚ê‚Î‚È‚è‚Ü‚¹‚ñ‚BMTS ‚ðƒZƒbƒgƒAƒbƒv‚·‚é‘O‚É DCOM for Windows 95 ‚ðƒCƒ“ƒXƒg[ƒ‹‚µ‚È‚¢‚Æ‚AŽŸƒ‚ƒGƒ‰‚[ƒ‚ƒbƒZ‚[ƒW‚ª•\Ž¦‚³‚ê‚Ü‚·‚B

  "ƒZƒbƒgƒAƒbƒv ƒ‰ƒCƒuƒ‰ƒS mtssetup.dll ‚ð“Ç‚Ý‚ž‚ß‚È‚©‚Á‚½‚©‚ÅŠÖ‚” MTSSetupProc ‚ªŒ©‚Â‚©‚è‚Ü‚¹‚ñ‚Å‚µ‚½‚B"

  DCOM for Windows 95 ‚Íhttp://www.microsoft.com/japan/win95/ ‚©‚çƒCƒ“ƒXƒg[ƒ‹‚Å‚«‚Ü‚·‚BDCOM for Windows 95 ‚Í Internet Explorer 4.0 ‚ªŽ©“®“I‚ÉƒCƒ“ƒXƒg[ƒ‹‚·‚é‚±‚Æ‚É‘‚ˆÓ‚µ‚Ä‚¾‚³‚¾‚³‚¢‚BWindows 98 ‚ðƒCƒ“ƒXƒg[ƒ‹‚·‚é‚Æ‚ADCOM ‚àƒCƒ“ƒXƒg[ƒ‹‚³‚ê‚Ü‚·‚B

  Windows 95 ƒRƒ“ƒsƒ…[ƒ^‚©‚ç MTS ‚ðí‚œ‚·‚é‚ê‡‚ÍADCOM for Windows 95 ‚ðí‚œ‚µ‚È‚¢‚Å‚¾‚³‚¾‚³‚¢‚BDCOM for Windows 95 ‚ª•K—v‚È Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR[ƒfƒBƒl[ƒ^ (MS DTC) ‚ÍAMTS ƒZƒbƒgƒAƒbƒv ƒvƒ‚ƒOƒ‰ƒ€‚É‚æ‚Á‚ÄƒCƒ“ƒXƒg[ƒ‹‚³‚ê‚Ü‚·‚B

- Windows 95/98 ƒRƒ“ƒsƒ…[ƒ^‚©‚ç Windows NT ƒRƒ“ƒsƒ…[ƒ^‚ðŠ‚[ƒg‚ÅŠÇ—‚·‚é‚ê‡‚ÍAŠŠƒ‚[ƒg ƒŒƒWƒXƒgƒŠ ƒT[ƒrƒX‚ðƒCƒ“ƒXƒg[ƒ‹‚µ‚È‚¯‚ê‚Î‚È‚è‚Ü‚¹‚ñ‚BƒŠƒ‚[ƒg ƒŒƒWƒXƒgƒŠ ƒT[ƒrƒX‚ðŽg‚¤ ‚Æ‚A(“K‚Ø‚ÈƒAƒNƒZƒXŒ ‚ª‚ ‚ê‚Î) ƒŠƒ‚[ƒg‚Ì Windows NT ƒRƒ“ƒsƒ…[ƒ^‚ìƒŒƒWƒXƒgƒŠ ƒGƒ“ƒgƒŠ‚ð•Ï‚X‚·‚é‚±‚Æ‚ª‚Å‚«‚Ü‚·‚BƒŠƒ‚[ƒg ƒŒƒWƒXƒgƒŠ ƒT[ƒrƒX‚ÍAWindows 95/98 ‚Ì CD-ROM ‚Ì \Admin\Nettols\RemotReg ƒffBƒŒƒNƒgƒŠ‚É‚ ‚è‚Ü‚·‚BƒŠƒ‚[ƒg ƒŒƒWƒXƒgƒŠ ƒT[ƒrƒX‚ðƒCƒ“ƒXƒg[ƒ‹‚·‚é•û–@‚É‚Â‚¢‚Ä‚ÍARegserv.txt ƒtƒ@ƒCƒ‹‚ðŽQ‚Æ‚µ‚Ä‚¾‚³‚¢‚B
- Microsoft Windows 95 ƒNƒ‰ƒCƒAƒ“ƒg‚ð MTS ‚Æ‹¤‚ÉŽg—p‚·‚é‚ê‡‚ÍADCOM for Windows 95 ‚ðƒCƒ“ƒXƒg[ƒ‹‚µ‚Ü‚·‚BDCOM for Windows 95 ‚ì‚Å‚V‚î•ñ‚É‚Â‚¢‚Ä‚ÍAhttp://www.microsoft.com/japan/win95/‚ðŽQ‚Æ‚µ‚Ä‚¾‚³‚¢‚BWindows 98 ‚ðƒCƒ“ƒXƒg[ƒ‹‚·‚é‚Æ‚ADCOM ‚àƒCƒ“ƒXƒg[ƒ‹‚³‚ê‚Ü‚·‚B
- Windows NT Server Version 4.0‚ðŽg—p‚µ‚Ä‚¢‚ê‡‚ÍAWindows NT 4.0 Service Pack 3 ‚ðƒCƒ“ƒXƒg[ƒ‹‚µ‚È‚¯‚ê‚Î‚È‚è‚Ü‚¹‚ñ‚BWindows NT 4.0 Service Pack 3 ‚ÍAhttp://www.microsoft.com/products/ntupdate/nt4sp3/ ‚©‚çƒ_ƒEƒ“ƒ□□[ƒh‚Å‚«‚Ü‚·‚B
- ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ðŽg‚Á‚Äƒf□[ƒX□[ƒX‚ÉƒCƒAƒNƒZƒX‚·‚é‚ê‡‚ÍAMicrosoft SQL Server Version 6.5 ^È‚~‚A‚Ü‚½‚Í‚»‚Ì‚Ù‚©‚Ì Microsoft Transaction Server ŒÝŠ·‚Ìƒf□[ƒ^ƒx□[ƒX‚ðŽg—p‚µ‚Ä‚¾‚³‚¾‚³‚¢‚BSQL Server ‚ðŽg—p‚·‚é‚É‚Í Windows NT ƒlƒyƒŒ□[ƒeƒBƒ“ƒO ƒVƒXƒeƒ€‚ª•K—v‚Å‚·‚B
- MTS ƒ\ƒtƒgƒEƒFƒA‚ðŽÀs‚·‚éƒRƒ“ƒsƒ…[ƒ^‚É‚Í SQL Server 6.5 Service Pack 4 ‚ðƒCƒ“ƒXƒg[ƒ‹‚·‚é‚±‚Æ‚ð‹-‚¨Š©‚ß‚µ‚Ü‚·‚BŠù‘m‚Ì‚áŠQ‚Ì‚¿‚Â‚Â‚©‚Í‚A±‚Ì Service Pack ‚É‚æ‚Á‚Ä‰‰ðŒˆ‚Å‚«‚Ü‚·‚BSQL Server 6.5 Service Pack 4 ‚Íhttp://www.microsoft.com/japan/bkoffice/ntsql/ ‚©‚çƒCƒ“ƒXƒg[ƒ‹‚µ‚Ü‚·‚B
- Microsoft Visual Basic® ‚ðŽg‚Á‚ÄƒRƒ|□[ƒlƒ“ƒg‚ð□ì□¬‚·‚é‚ê‡‚ÍAMicrosoft Visual Basic Version 4.0 ^È‚~‚Ì Enterprise Edition ‚ðŽg—p‚µ‚Ä‚¾‚³‚¢‚BMTS ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ì□ì□¬‚É‚Íƒ‰ƒCƒZƒ“ƒX Version 5.0 ^È‚~‚ðŽg—p‚·‚é‚±‚Æ‚ð‹-‚¨Š©‚ß‚µ‚Ü‚·

  Visual Basic ‚ðŽg‚Á‚Ä DLL ‚ð□ì□¬‚·‚é‚ê‡‚ÍAMTS ‚ÌƒCƒ“ƒXƒg[ƒ‹‚Å [ƒJƒXƒ^ƒ€] ƒlƒvƒVƒ‡ƒ“‚ð‘I‘ð‚µ‚Ä VB ƒAƒhƒCƒ“‚ðƒCƒ“ƒXƒg[ƒ‹‚·‚é‚ÆŽ©“®“I‚ÉŽg—p‚·‚é‚±‚Æ‚ª‚Å‚«‚Ü‚·‚BVisual Basic ‚ªƒRƒ“ƒpƒCƒ‹’†‚É□V‚µ‚¢ƒRƒ“ƒ|□[ƒlƒ“ƒg GUID ‚ð□¶□¬‚µ‚½Œã‚Å‚à MTS ƒJƒ^ƒ□ƒO‚ð□³‚µ‚□X□V‚·‚é‚±‚Æ‚ª‚Å‚«‚Ü‚·‚B□Ú□×‚É‚Â‚¢‚Ä‚Í□A□uMTS Component Requirements□v ‚ðŽQ‚Æ‚µ‚Ä‚¾‚³‚¢‚B
- Microsoft Visual C++® ‚ðŽg‚Á‚ÄƒRƒ|□[ƒlƒ“ƒg‚ð□ì□¬‚·‚é‚ê‡‚ÍAVisual C++ Verison 4.1 ^È‚~‚ð Active Template Libraries (ATL) Version 1.1 ^È‚~‚Æ‘g‚Ý‡‚í‚¹‚ÄŽg—p‚µ‚Ä‚¾‚³‚¢‚BVisual C++ Version 4.1 ‚ðŽg—p‚·‚é‚ê‡‚ÍAWin32® SDK ‚ª•K—v‚Å‚·‚B
- Java ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ð□ì□¬‚µ‚□AMTS ‚Æ‹¤‚ÉŽÀs‚·‚é‚ê‡‚ÍAInternet Explorer Version 4.0 ^È‚~‚Æ^ê□□‚ÉƒCƒ“ƒXƒg[ƒ‹‚³‚ê‚é Microsoft Virtual Machine for Java ‚ðŽg—p‚µ‚Ä‚¾‚³‚¢‚BInternet

Explorer ,Í□Ahttp://www.microsoft.com/japan/ie/ ,©,çƒ_ƒEƒ“ƒ□□[ƒh,Å,«,Ü,·□B

- ƒCƒ“ƒ^□[ƒlƒbƒg ƒAƒvƒSƒP□[ƒVƒ‡ƒ“,ð□ì□¬,·,é□ê□‡,Í□AMicrosoft Internet Information Server Version 4.0 ˆÈ□~□A,¨,æ,Ñ Microsoft Internet Explorer Version 4.0 ˆÈ□~,ðŽg—p,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B
- Oracle ƒf□[ƒ^ƒx□[ƒX,ð MTS ,Æ‘g,Ý□‡,í,¹,ÄŽg—p,·,é,±,Æ,ª,Å,«,Ü,·□BOracle ,ð M,s,r ,Æ‘g,Ý□‡,í,¹,ÄŽg—p,·,é□ê□‡,Í□A□uMTS ,ð Oracle ,ÉƒAƒNƒZƒX,Å,«,é,æ,¤,É□Ý’è,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

MTS 1.x ƒRƒ“ƒsƒ…□[ƒ^,©,ç MTS 2.0 ƒRƒ“ƒsƒ…□[ƒ^,ðƒŠƒ,□[ƒg,ÅŠÇ—□,Å,«,È,¢

ˆÈ‘O,Ìƒo□[ƒWƒ‡ƒ“,Ì MTS ,ðŽÀ□s’†,ÌƒRƒ“ƒsƒ…□[ƒ^,©,ç□AƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ MTS 2.0 ,ðƒŠƒ,□[ƒg,ÅŠÇ—□,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B,±,ê,ðŽŽ,Ý,½□ê□‡□AƒGƒ‰□[ƒR□[ƒh,ª•Ô,³,ê,é,©□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ª—\Šú,¹,¸□I—¹,µ,Ü,·□B

MTS 2.0 ƒCƒ“ƒXƒg□[ƒ‹Œã,Ì Visual Basic 5.0 ,ÌƒVƒƒƒbƒgƒ_ƒEƒ“

MTS 2.0 ,ÌƒCƒ“ƒXƒg□[ƒ‹Œã,É Microsoft Visual Basic 5.0 ,ðƒVƒƒƒbƒgƒ_ƒEƒ“,·,é,Æ□AƒAƒNƒZƒXˆá”½,ª”-□¶,·,é□ê□‡,ª, ,è,Ü,·□B,±,ê,Í Visual Basic 5.0 ,ÌŠù’m,Ì–â‘è,Å□AVisual Studio Service Pack 2 ˆÈ□~,ð http://www.microsoft.com/japan/developer/vstudio ,©,çƒCƒ“ƒXƒg□[ƒ‹,·,é,Æ‰ðŒˆ,Å,«,Ü,·□B,±,Ì–â‘è,Ì□Ú□×,É,Â,¢,Ä,Í□AKnowledge Base ,Ì J041291 ,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

–¼‘O•t,«ƒpƒCƒv,ðŽg—p,µ,½,Æ,«,Ì Windows 95/98 ,Ìƒƒfƒbƒhƒ□ƒbƒN

Windows 95/98 ƒNƒ‰ƒCƒAƒ“ƒg,©,ç MS SQL Server ,Ö–ñ 40 ˆÈ□ã,Ì–¼‘O•t,«ƒpƒCƒv□Ú‘±,ðŽg—p,·,é,Æ□Aƒfƒbƒhƒ□ƒbƒN,ð‹N,±,·‰Â”\□«,ª, ,è,Ü,·□B,±,Ì–â‘è,ð‰ðŒˆ,·,é,É,Í□AODBC ƒhƒ‰ƒCƒo,Æ SQL Server ŠÔ,Ìƒvƒ□ƒgƒRƒ‹,ð TCP/IP ,É•Ï□X,µ,Ü,·□B

INSTCAT.SQL ƒXƒNƒŠƒvƒg,ÌŽÀ□s

MTS ,Í INSTCAT.SQL ƒXƒNƒŠƒvƒg,ð’ñ‹Ÿ,µ,Ü,·□B,±,ÌƒXƒNƒŠƒvƒg,Í□ASQL Server ODBC Driver ,É‘Î,·,éƒJƒ^ƒ□ƒO ƒXƒgƒAƒh ƒvƒ□ƒV□[ƒWƒƒ,ð□X□V,µ,Ü,·□B,±,ÌƒXƒNƒŠƒvƒg,ðŽÀ□s,µ,Ä,¢,È,¢□ê□‡,Í□ASQL Server ,Ì ISQL ƒt□[ƒeƒBƒŠƒefB,ðŽg,Á,ÄƒXƒNƒŠƒvƒg,ðŽÀ□s,µ,Ä,,¾,³,¢□B,±,ÌƒXƒNƒŠƒvƒg,Í□ASQL Server ƒJƒ^ƒ□ƒO,Å‹N,±,è,¤,é□Aƒo□[ƒWƒ‡ƒ“”Ô□†,Ì□ö□Ý“I,È•sˆê’v,ð‰ðŒˆ,µ,Ü,·□B

MTS ,Æ‹¤,É SQL Server ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é

SQL Server 6.5 ,ð MTS ,ÌƒCƒ“ƒXƒg□[ƒ‹Œã,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é,±,Æ,ª,Å,«,Ü,·,ª□ASQL Server 6.5 ,Í MTS ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é‘O,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é,±,Æ,ð,¨Š©,ß,µ,Ü,·□BMTS ,ÌƒCƒ“ƒXƒg□[ƒ‹Œã,É SQL Server 6.5 ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,Æ□AMTS ƒZƒbƒgƒAƒbƒv,Í MS DTC ,ª□³,µ,“®□ì,·,é,©,Ç,¤,©,ðŠm”F,µ,æ,¤,Æ,µ,Ü,·□B

MTS 2.0 ,É MTS 1.0 ,ð□ã□‘,«ƒCƒ“ƒXƒg□[ƒ‹,·,é

MTS 2.0 ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒRƒ“ƒsƒ…□[ƒ^,É MTS 1.0 ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,Æ,«,Í□AMTS 1.0 ,ÌƒZƒbƒgƒAƒbƒv,ðŽÀ□s,·,é‘O,É□Asystem ƒfƒBƒŒƒNƒgƒŠ,©,çŽŸ,Ìƒtƒ@ƒCƒ‹,ð□í□œ,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

%WINDIR%\system32\adme.dll

%WINDIR%\system32\dac.exe

%WINDIR%\system32\dacdll.dll

%WINDIR%\system32\dtccm.dll

%WINDIR%\system32\dtctrace.dll

%WINDIR%\system32\dtctrace.exe

%WINDIR%\system32\dtcuic.dll

%WINDIR%\system32\dtcuis.dll

%WINDIR%\system32\dtcutil.dll

%WINDIR%\system32\dtcxatm.dll

%WINDIR%\system32\enudtc.dll

%WINDIR%\system32\logmgr.dll

%WINDIR%\system32\msdtc.exe

%WINDIR%\system32\msdtc.dll

%WINDIR%\system32\msdtcprx.dll

%WINDIR%\system32\msdtctm.dll

%WINDIR%\system32\dtccfg.cpl

%WINDIR%\system32\svcsrvl.dll

%WINDIR%\system32\xolehlp.dll

%WINDIR%\system32\mmc.exe

%WINDIR%\system32\mmc.ini

%WINDIR%\system32\mmclv.dll

%WINDIR%\system32\mmcndmgr.dll

%WINDIR%\system32\mtxinfr1.dll

%WINDIR%\system32\mtxinfr2.dll

%WINDIR%\system32\mtxclu.dll

%WINDIR%\system32\mtxrn.dll

%WINDIR%\system32\mtxdm.dll

[,·,×,Ä□í□œ] ƒIƒvƒVƒ‡ƒ“,ðŽÀ□s,·,é,Æƒt□[ƒU□['è‹`ƒpƒbƒP□[ƒW,ª□í□œ,³,ê,é

'O,Ìƒo□[ƒWƒ‡ƒ“,Ì MTS ,ðŽg—p,µ,Ä,¢,Ä□Aƒt□[ƒU□['è‹`ƒpƒbƒP□[ƒW,ðŽc,·□ê□‡,Í□AƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€,Ì [,·,×,Ä□í□œ] ƒIƒvƒVƒ‡ƒ“,Ü,½,ÍƒRƒ“ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Ì [ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì'Ç‰Á,Æ□í□œ] ƒAƒCƒRƒ“,ðŽg,í,È,¢,Å,-,¾,³,¢□BMTS
,ð□í□œ,·,é,Æ□A,·,×,Ä,Ìƒt□[ƒU□['è‹`ƒpƒbƒP□[ƒW,ª□í□œ,³,ê,Ü,·□BƒAƒbƒvƒOƒŒ□[ƒh,·,é,Æ,«,Éƒt□[ƒU□['è‹`ƒpƒbƒP□[ƒW,ðŽc,·,É,Í□AŠù'¶,Ìƒo□[ƒWƒ‡ƒ“,Ì MTS ,É□V,µ,¢ MTS ,ð□ã□',«ƒCƒ"ƒXƒg□[ƒ‹,µ,Ä,,¾,³,¢□B

ƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW ID ,ÌƒpƒXƒ□□[ƒh,ðŠÔˆá,¦,½□ê□‡

ƒZƒbƒgƒAƒbƒvŽž,ÉƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW ID ,ÌƒpƒXƒ□□[ƒh,ÌŽw'è,ðŠÔˆá,¦,é,Æ□AMicrosoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,Í"®□ì,µ,Ü,¹,ñ□B,±,Ì□ê□‡,Í□AƒJƒ^ƒ□ƒO ƒT□[ƒo□[(ƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW) ,©,ç 80008005 ƒGƒ‰ƒ□[(ƒT□[ƒo□[ŽÀ□s,ÌŽ¸"s) ,ª•Ô,³,ê,Ü,·□B•s□³,ÈƒpƒXƒ□□[ƒh,ð□C□³,·,é,É,Í□AMicrosoft Transaction Server ,ð□ÄƒCƒ"ƒXƒg□[ƒ‹,µ,Ü,·□B

,Ü,½□AƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW ID ,ÌƒpƒXƒ□□[ƒh,ÌŽw'è,ðŠÔˆá,¦,é,Æ□AƒpƒbƒP□[ƒW,ªŽÀ□s,³,ê,¸□A"¯,¶ 80008005 ƒGƒ‰ƒ□[,ª"□¶,µ,Ü,·□B,½,¾,µ□A,±,Ì□ê□‡□AƒNƒ‰ƒCƒAƒ"ƒg,Í CoCreateInstance ,ÌŽ¸"s HRESULT ,Ü,½,Í"¯"™,ÌŽ¸"s,ÆŒ©,È,µ,Ü,·□B,±,ÌƒGƒ‰ƒ□[,ð□C□³,·,é,É,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,É–ß,è□A□iƒpƒbƒP□[ƒW,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ID] ƒ^ƒu,Å) ƒpƒbƒP□[ƒW,Ì ID ,ð□C□³,µ,Ü,·□BWindows NT ,ÌƒZƒLƒ…ƒŠƒeƒB□ã,Ì□§–ñ,É,æ,è□Aƒt□[ƒU□[ID ,ÍŠm"F,Å,«,Ü,·,ª□AƒpƒXƒ□□[ƒh,ðŠm"F,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

Alpha ƒvƒ‰ƒbƒgƒtƒH□[ƒ€

Alpha AXP™ ƒRƒ"ƒsƒ…□[ƒ^,Í MTS ,ðƒTƒ|□[ƒg,µ,Ä,¢,Ü,·□B,½,¾,µ□AMicrosoft Transaction Server for Alpha platforms ,Ì□¡‰ñ,ÍƒŠƒŠ□[ƒX,É,ÍŽŸ,Ì,à,Ì,ªŠÜ,Ü,ê,Ä,¢,Ü,¹,ñ□B

- Microsoft Visual J++ ,ÌƒRƒ"ƒ|□[ƒlƒ"ƒg,ÆƒTƒ"ƒvƒ‹

Microsoft Visual C++ ,¨,æ,Ñ Visual Basic ,ÌƒTƒ"ƒvƒ‹ ƒRƒ"ƒ|□[ƒlƒ"ƒg,ÆŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ"ƒvƒ‹ ƒXƒNƒŠƒvƒg,Í□A\Program Files\Mts\Samples ƒtƒHƒ‹ƒ_,É, ,è,Ü,·□B

Alpha ƒvƒ‰ƒbƒgƒtƒH□[ƒ€,ÌƒTƒ|□[ƒg,ÉŠÖ,·,é□Å□V□î•ñ,É,Â,¢,Ä,Í□Ahttp://www.microsoft.com/japan/

products/ntserver/ ,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

Alpha ƒvƒ‰ƒbƒgƒtƒH□[ƒ€,Ö,Ì MTS ,ÌƒCƒ“ƒXƒg□[ƒ‹,ÌŒ‹‰Ê,ðŠm”F,·,é,É,Í□AAlpha ƒRƒ“ƒsƒ…□[ƒ^ (ƒT□[ƒo□[□j ,Æ x86 ƒRƒ“ƒsƒ…□[ƒ^ (ƒNƒ‰ƒCƒAƒ“ƒg) ,Ì 2 ‘ä,ÌƒRƒ“ƒsƒ…□[ƒ^,ðŽg,Á,Ä Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽÀ□s,µ,Ü,·□B□Ú□×,É,Â,¢,Ä,Í□A□uSample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒZƒbƒgƒAƒbƒv□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒƒ“ƒg,ÌƒCƒ“ƒXƒg[ƒ‹

ƒZƒbƒgƒAƒbƒv,Ì [ƒJƒXƒ^ƒ€] ƒIƒvƒVƒ‡ƒ“,ðŽg,Á,Ä Microsoft Transaction Server (MTS) ,ÌŠJ”ŽÒ—p,ÌƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒƒ“ƒg,ðƒCƒ“ƒXƒg[ƒ‹,Å,«,Ü,·BŠeƒZƒbƒgƒAƒbƒv ƒIƒvƒVƒ‡ƒ“,ÅƒCƒ“ƒXƒg[ƒ‹,³,ê,é MTS ,ÌƒRƒ“ƒ|[ƒlƒ“ƒg,ÍAŽŸ,Ì,Æ,¨,è,Å,·B

- **Å¬ƒZƒbƒgƒAƒbƒv**

  MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,Æ MTS ƒGƒNƒXƒvƒ[ƒ‰,ðƒCƒ“ƒXƒg[ƒ‹,µ,Ü,·B

- **•W€ƒZƒbƒgƒAƒbƒv**

  MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«AMTS ƒGƒNƒXƒvƒ[ƒ‰A,¨,æ,Ñ MTS ,ÌŽå,ÈƒhƒLƒ…ƒƒ“ƒg,ðƒCƒ“ƒXƒg[ƒ‹,µ,Ü,·B

- **ƒJƒXƒ^ƒ€ ƒZƒbƒgƒAƒbƒv**

  MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«AMTS ƒGƒNƒXƒvƒ[ƒ‰AMTS ,ÌŽå,ÈƒhƒLƒ…ƒƒ“ƒgAMTS ŠJ”ŽÒ—pƒTƒ“ƒvƒ‹A,¨,æ,Ñ MTS ŠJ”ŽÒ—pƒhƒLƒ…ƒƒ“ƒg,ðƒCƒ“ƒXƒg[ƒ‹,µ,Ü,·B

ƒZƒbƒgƒAƒbƒv,Ì [ƒJƒXƒ^ƒ€] ƒIƒvƒVƒ‡ƒ“,ðŽg,Á,Ä MTS ŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒƒ“ƒg,ðƒCƒ“ƒXƒg[ƒ‹,·,é,±,Æ,ð‹,,¨Š©,ß,µ,Ü,·BMTS ŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒƒ“ƒg,ð—˜—p,·,é,±,Æ,É,æ,Á,ÄAŠÇ—,Ì'Î Û,Æ,È,éƒpƒbƒP[ƒW,ÆƒRƒ“ƒ|[ƒlƒ“ƒg,ð,æ,è[,—‰ð,·,é,±,Æ,ª,Å,«,Ü,·B,½,Æ,¦,ÎASample Bank ,Æ Tic-Tac-Toe ŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ðŽg,Á,ÄAMTS ƒGƒNƒXƒvƒ[ƒ‰,É,æ,éƒpƒbƒP[ƒW,Ì”z'u,ÆŠÇ—,ð—û K,Å,«,Ü,·B,Ü,½AƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP[ƒVƒ‡ƒ“,ðŽg,Á,ÄAMTS ,ÌƒCƒ“ƒXƒg[ƒ‹,ª³,µ,Š®—¹,µ,½,©,Ç,¤,©,ðŠm”F,·,é,±,Æ,à,Å,«,Ü,·B

MTS ŠJ”ŽÒ—pƒhƒLƒ…ƒƒ“ƒg,Å,ÍAƒAƒNƒeƒBƒu‰»,ÌÝ'è,âƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌƒvƒƒpƒeƒB,È,ÇAƒRƒ“ƒ|[ƒlƒ“ƒg,ðƒpƒbƒP[ƒW,É'g,Ýž,Þ•û–@,ð‹K'è,·,éÛ,ÉÝŒv ã,Ì'ˆÓ"_,É,Â,¢,Äà–¾,µ,Ä,¢,Ü,·BwProgrammer's GuidexAMTS ƒvƒƒOƒ‰ƒ~ƒ"ƒO,ÌŠT"O (‹¤—Ló'Ô,È,Ç) ,Ì ‰ð'à,Ì,Ù,©,ÉASample Bank ƒAƒvƒŠƒP[ƒVƒ‡ƒ",ðƒCƒ"ƒXƒg[ƒ‹,µA\¬,·,é•û–@,ðŽ¦,·ƒ`ƒ…[ƒgƒŠƒAƒ‹,ª‹LÚ,³,ê,Ä,¢,Ü,·B

ƒZƒbƒgƒAƒbƒv,Ì [Å¬] ƒIƒvƒVƒ‡ƒ“,Ü,½,Í [•W€] ƒIƒvƒVƒ‡ƒ“,ðŽg,Á,ÄƒCƒ“ƒXƒg[ƒ‹,µ,½Œã,ÉAMTS ŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒƒ“ƒg,ð“üŽè,·,é,É,Í

1. [ƒXƒ^[ƒg] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µA[Ý'è] ,ðƒ|ƒCƒ“ƒg,µ,Ü,·BŽŸ,ÉA[ƒRƒ“ƒgƒ[ƒ‹ ƒpƒlƒ‹] ,ðƒNƒŠƒbƒN,µ,Ü,·B
2. [ƒAƒvƒŠƒP[ƒVƒ‡ƒ“,Ì'Ç‰Á,Æí œ] ƒAƒCƒRƒ“,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µA[Windows NT 4.0 Option Pack] ,ðƒNƒŠƒbƒN,µ,Ü,·BŽŸ,ÉA['Ç‰Á,Æí œ] ,ðƒNƒŠƒbƒN,µ,Ü,·B

   MTS ,ðíœ,·,é,ÆAƒ†[ƒU['è‹`ƒpƒbƒP[ƒW,ªíœ,³,ê,é,±,Æ,É'ˆÓ,µ,Ä,-,¾,³,¢Bƒ†[ƒU['è‹`ƒpƒbƒP[ƒW,ðŽc,µ,Ä,¨,,É,ÍA'O‰ñƒCƒ"ƒXƒg[ƒ‹,µ,½ MTS ,ðíœ,¹,¸,ÉV,µ,¢ MTS ,ðƒCƒ"ƒXƒg[ƒ‹,µ,Ü,·B
3. ƒZƒbƒgƒAƒbƒv ƒvƒƒOƒ‰ƒ€,Å ['Ç‰Á/íœ] ,ðƒNƒŠƒbƒN,µ,Ü,·B
4. [Microsoft Transaction Server] ,ðƒNƒŠƒbƒN,µA[ƒTƒuƒRƒ"ƒ|[ƒlƒ"ƒg,Ì•\Ž¦] ,ðƒNƒŠƒbƒN,µ,Ü,·B
5. ,·,×,Ä,Ì MTS ƒTƒuƒRƒ"ƒ|[ƒlƒ"ƒg,Ìƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ªƒIƒ",É,È,Á,Ä,¢,é,±,Æ,ðŠm"F,µ,Ü,·B
6. [OK] ,ðƒNƒŠƒbƒN,µ,Ü,·B

MTS ƒT□[ƒo□[,ì□\□¬

MTS ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,½,ç□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä MTS ƒpƒbƒP□[ƒW,ð”z’u,µ□AŠÇ—□,·,é,± ,Æ,ª,Å,«,é,æ,¤,É□AMTS ƒT□[ƒo□[,ð□\□¬,µ,Ü,·□BƒpƒbƒP□[ƒW,Ì”z’u,ÆŠÇ— □,ðŠJŽn,·,é‘O,É□AŽŸ,Ì‘€□ì,ðŽÀ□s,µ,Ä□AMTS ƒT□[ƒo□[,Å”z’u,ð□s,¤,±,Æ,ª,Å,«,é,æ,¤,É□Ý’è,µ,Ü,·□B

- ƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,ÆƒpƒbƒP□[ƒW ID ,ð□\□¬,·,é□B
- ŠÇ—□,Ì‘Î□Û,Æ,È,éƒRƒ“ƒsƒ…□[ƒ^,ð□Ý’è,·,é□B

,³,ç,É□AWindows NT ƒt□[ƒU□[ƒ}ƒl□[ƒWƒƒ,ðŽg,Á,Ä□ASystem ƒpƒbƒP□[ƒW,¨,æ,Ñ,Ù,©,Ì MTS ƒpƒbƒP□[ƒW,Ì ID ,ÉŠ„,è“–,Ä,éƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,ª□AWindows NT ,Ìƒt□[ƒU□[,ÌŒ —˜ "ƒT□[ƒrƒX,Æ,µ,Äƒ□ƒOƒlƒ“" ,ðŽ□,Â,± ,Æ,ðŠm”F,µ,Ü,·□B

ƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,Ì Windows NT ƒt□[ƒU□[,ÌŒ —˜,ðŠm”F,·,é,É,Í

1. ƒt□[ƒU□[ƒ}ƒl□[ƒWƒƒ,ð‹N“®,µ□A[Œ´¥] ƒ□ƒjƒ…□[,Ì [ƒt□[ƒU□[,ÌŒ —˜] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
2. [□,“x,Èƒt□[ƒU□[Œ —˜,Ì•\Ž¦] ,ðƒlƒ“,É,µ,Ü,·□B

ƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,ð□\□¬,·,é

MTS ƒpƒbƒP□[ƒW,ðˆÀ‘S,É”z’u,µ□AŠÇ—□,·,é,É,Í□AƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ì Administrator ƒ□□[ƒ‹,É“K□Ø,Èƒt□[ƒU□[,ðŠ„,è“–,Ä,È,¯,ê,Î,È,è,Ü,¹,ñ□BMTS ,ªƒZƒbƒgƒAƒbƒv,³,ê,½,Æ,«,Í□AƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ì Administrator ƒ□□[ƒ‹,Éƒt□[ƒU□[,ªŠ„,è“–,Ä,ç,ê,Ä,¢,Ü,¹,ñ□B,µ,½,ª,Á,Ä□AƒVƒXƒef€ ƒpƒbƒP□[ƒW,ÌƒZƒLƒ…ƒŠƒefB,Í–³Œø,É,È,Á,Ä,¨,è□A,Çƒt□[ƒU□[,Å,à MTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ðŽg,Á,Ä,»,ÌƒRƒ“ƒsƒ…□[ƒ^,ÌƒpƒbƒP□[ƒW,Ì□\□¬,ð•Ï□X,·,é,±,Æ,ª,Å,«,Ü,·□Bƒt□[ƒU□[,ðƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,ÉŠ„,è“–,Ä,é,Æ□AMTS ,Íƒt□[ƒU□[,ª MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÅƒpƒbƒP□[ƒW,ð•Ï□X,µ,æ,¤ ,Æ,µ,½,Æ,«,Éƒ□□[ƒ‹,ðƒ`ƒFƒbƒN,µ,Ü,·□B

“Á,ÉŽw’è,µ,È,¯,ê,Î□AƒVƒXƒef€ ƒpƒbƒP□[ƒW,É,Í Administrator ƒ□□[ƒ‹,Æ Reader ƒ□□[ƒ‹,ª, ,è,Ü,·□BƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ì Administrator ƒ□□[ƒ‹,ÉŠ„,è“–,Ä,ç,ê,½ƒt□[ƒU□[,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì,·,×,Ä,Ì‹@”\,ðŽg,¤ ,±,Æ,ª,Å,«,Ü,·□BReader ƒ□□[ƒ‹,ÉŠ„,è“–,Ä,ç,ê,½ƒt□[ƒU□[,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ÌŠK‘w“à,Ì,·,×,Ä,ÌƒIƒuƒWƒFƒNƒg,ð•\Ž¦,·,é,± ,Æ,ª,Å,«,Ü,·,ª□AƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒg□[ƒ‹□A□ì□¬□A•Ï□X□A□í□œ,âƒT□[ƒo□[ƒvƒ□ƒZƒX,ÌƒVƒƒƒbƒgƒ_ƒEƒ“□Aƒpƒ bƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg,ð□s,¤,±,Æ,Í,Å,«,Ü,¹,ñ□B,½,Æ,¦,Î□AŽ©•ª,Ì Windows NT ƒhƒ□ƒCƒ“ ƒt□[ƒU□[–¼,ðƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ì Administrator ƒ□□[ƒ‹,ÉŠ„,è“–,Ä,é,Æ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ðŽg,Á,Ä,·,×,Ä,ÌƒpƒbƒP□[ƒW,ð’Ç‰Á□A•Ï□X□A,Ü,½,Í□í□œ,ð□s,¤,±,Æ,ª,Å,«,Ü,·□Bƒvƒ‰ƒCƒ}ƒŠ ƒhƒ□ƒCƒ“ ƒRƒ“ƒgƒ□□[ƒ‰,Ü,½,ÍƒofbƒNƒAƒbƒv ƒhƒ□ƒCƒ“ ƒRƒ“ƒgƒ□□[ƒ‰,É,È,Á,Ä,¢,éƒT□[ƒo□[,É MTS ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢ ,é□ê□‡□Aƒt□[ƒU□[,ª MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÅƒpƒbƒP□[ƒW,ðŠÇ—□,·,é,É,Í□A,»,Ìƒt□[ƒU□[,Íƒhƒ□ƒCƒ“ŠÇ— □ŽÒ,Å,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ƒt□[ƒU□[,ðƒ□□[ƒ‹,ÉŠ„,è“–,Ä,é•û–@,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒVƒXƒef€ ƒpƒbƒP□[ƒW,É□V,µ,¢ƒ□□[ƒ‹,ðƒÝ’è,·,é,± ,Æ,à,Å,«,Ü,·□B,½,Æ,¦,Î□Aƒt□[ƒU□[,ÉƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ÆŽÀ□s,ð‹–‰Â,µ□A□í□œ,ÆƒGƒNƒXƒ|□[ƒg,Í‹–‰Â,µ,È,¢ Developer ƒ□□[ƒ‹,ð□\□¬,·,é,±,Æ,ª,Å,«,Ü,·□B,±,Ìƒ□□[ƒ‹,ÉŠ„,è“–,Ä,é Windows NT ƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,Ü,½,ÍƒOƒ‹□[ƒv,Í□AƒRƒ“ƒsƒ…□[ƒ^,É‘Î,·,éŠ®‘S,ÈŠ—□ŽÒ“Œ ,ðŽ□,Á,Ä,¢,È,- ,Ä,à□A,»,ÌƒRƒ“ƒsƒ…□[ƒ^,ÅƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ðƒefƒXƒg,·,é,± ,Æ,ª,Å,«,Ü,·□B□V,µ,¢ƒ□□[ƒ‹,Ì□Ý’è,Ì□Ú□×,É,Â,¢,Ä,Í□A□u□V,µ,¢ MTS ƒ□□[ƒ‹,Ì’Ç‰Á□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒRƒ“ƒsƒ…□[ƒ^,ÌƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,ð□\□¬,µ,½,ç□AƒpƒbƒP□[ƒW,Ì [ƒvƒ□ƒpƒefB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ƒZƒLƒ…ƒŠƒefB] ƒ^ƒu,Ìƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒIƒ“,É,µ,Ä”F□Ø,ÌŠm”F,ð—LŒø,É,µ,Ü,·□B”F□Ø,ÌŠm”F,ð— LŒø,É,·,é•û–@,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é□v,ðŽQ□Æ,µ,Ä,- ,¾,³,¢□B

ŽŸ,Ì MS DTC ŠÇ—□‹@”\,Å,Í□AƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ü,½,ÍƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,ªŽg—p,³,ê,È,¢,±

,Æ,É’□ˆÓ,µ,Ä,,¾,³,¢□B

- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv] ƒEƒBƒ“ƒhƒE
- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——] ƒEƒBƒ“ƒhƒE
- [ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ƒEƒBƒ“ƒhƒE
- [MS DTC ŠJŽn] ƒRƒ}ƒ“ƒh,¨,æ,Ñ [MS DTC ’âŽ~] ƒRƒ}ƒ“ƒh

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄŠÇ—□,Å,«,é,æ,¤,ÉƒRƒ“ƒsƒ…□[ƒ^,ð□Ý’è,·,é

MTS ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,½ƒRƒ“ƒsƒ…□[ƒ^,Í MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å "ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^" ,Æ,µ,ÄŠÇ—□,³,ê,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä,Ù,©,ÌƒRƒ“ƒsƒ…□[ƒ^,ðŠÇ—□,·,é,±,Æ,à,Å,«,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ð’I’ð,µ□AŽŸ,Ì,¢,¸,ê,©,Ì‘€□ì,ðŽÀ□s,·,é,Æ□AŠÇ—□,·,é•K—v,ª, ,é□V,µ,¢ƒRƒ“ƒsƒ…□[ƒ^,ð [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,É’Ç‰Á,Å,«,Ü,·□B

- [“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ□A[ƒRƒ“ƒsƒ…□[ƒ^] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
- MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ìƒc□[ƒ‹ ƒo□[,Ì [□V,µ,¢ƒIƒuƒWƒFƒNƒg,Ì□ì□¬] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B
- ƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,Å [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ðƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒRƒ“ƒsƒ…□[ƒ^] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ŽŸ,É□A•\Ž¦,³,ê,éƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å Windows NT ƒhƒ□ƒCƒ“,Å,ÌƒRƒ“ƒsƒ…□[ƒ^–¼,ð“ü—Í,µ,ÄƒŠƒ,□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,ð□Å□ãˆÊ,ÌƒtƒHƒ‹ƒ_,Æ,µ,Ä’Ç‰Á,µ,Ü,·□BƒŠƒ,□[ƒg ƒT□[ƒo□[,ÉƒAƒNƒZƒX,µ,Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ð•\Ž¦,·,é,É,Í□Aƒ□ƒOƒIƒ“ ƒAƒJƒEƒ“ƒg,ª Reader ƒ□□[ƒ‹,ÉŠ„,è“–,Ä,ç,ê,Ä,¢,é•K—v,ª, ,è,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ªƒŠƒ,□[ƒg ƒT□[ƒo□[,É‘Î,µ,ÄŠ®‘S,È“Ç,ÝŽæ,è/□‘,«□ž,ÝŒ ŒÀ,ðŽ□,Â,É,Í□Aƒ□ƒOƒIƒ“ ƒAƒJƒEƒ“ƒg,ª Administrator ƒ□□[ƒ‹,Éƒ}ƒbƒv,³,ê,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK‘w“à,ÌƒIƒuƒWƒFƒNƒg,ðŠÇ—□,·,é•û–@,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ÌŠK‘w□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

□d—v Windows 95/98 ƒRƒ“ƒsƒ…□[ƒ^□ã,Ì MTS ,ð Windows NT ƒT□[ƒo□[□ã,Ì MTS ,©,çƒŠƒ,□[ƒg,ÅŠÇ—□,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

MTS ,ð Microsoft Cluster Server ,Æ‘g,Ý□‡,í,¹,ÄŽg,¤,æ,¤,É□\□¬,·,é

Microsoft Cluster Server (MSCS) ,Í Windows NT Server ,ðŽÀ□s,µ,Ä,¢,éƒRƒ“ƒsƒ…□[ƒ^,ÌƒNƒ‰ƒXƒ^ ‰»,ð□s,¢,Ü,·□BMSCS Version 1.0 ,Í□AWindows NT Server ,ðŽÀ□s,µ,Ä,¢,é“Á•Ê,ÉƒŠƒ“ƒN,³,ê,½ 2 ‘ä,Ì ƒT□[ƒo□[,ðƒNƒ‰ƒXƒ^‰»,µ,Ü,·□BƒNƒ‰ƒXƒ^,Ìˆê•û,ÌƒT□[ƒo□[,ªŒÌ□á,µ,½,è□Aƒlƒtƒ ‰ƒCƒ“,É,È,Á,½,è,·,é,Æ□A,à,¤ˆê•û,ÌƒT□[ƒo□[,ªŒÌ□á,µ,½ƒT□[ƒo□[,Ì“®□ì,ðˆø,«Œp,¬,Ü,·□B

Microsoft Transaction Server ,ð MSCS ,Æ‘g,Ý□‡,í,¹,ÄŽg,¤,É,Í□A,Ü,¸ Windows NT Server 4.0 Enterprise Edition ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□BŽŸ,É□AMicrosoft Cluster Server 1.0 ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B□ÅŒã,É□AŽŸ,ÌŽè□‡,É□],Á,Ä Microsoft Transaction Server 2.0 ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B

’□ MTS 2.0 ,Í□AMicrosoft Cluster Server 1.0 ,ÌƒvƒŒƒŠƒŠ□[ƒX”Å,Æ‘g,Ý□‡,í,¹,ÄŽg,¤,± ,Æ,Í,Å,«,Ü,¹,ñ□BMTS 2.0 ,ð Microsoft Cluster Server ,ÌƒvƒŒƒŠƒŠ□[ƒX”Å,Æ‹¤,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é,Æ□AMSCS ,Í□³,µ,“®□ì,µ,Ü,¹,ñ□B

MSCS ,Ì□Ú□×,É,Â,¢,Ä,Í□AWindows NT ,ÌƒIƒ“ƒ‰ƒCƒ“ ƒuƒbƒN,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

▸ Microsoft Cluster Server 1.0 ,Æ‹¤,É Microsoft Transaction Server 2.0 ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,É,Í

1. Windows NT Server 4.0 Enterpirse Edition ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B
2. ƒNƒ‰ƒXƒ^“à,Ì,·,×,Ä,ÌƒRƒ“ƒsƒ…□[ƒ^,É MSCS 1.0 ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B
3. MSCS ƒNƒ‰ƒXƒ^ ƒAƒhƒ~ƒjƒXƒgƒŒ□[ƒ^,ðŽg,Á,Ä□Aƒlƒbƒgƒ□□[ƒN–¼ƒŠƒ\□[ƒX,Æ‹¤—LƒffBƒXƒN ƒŠƒ\ □[ƒX,ðŠÜ,ÞƒOƒ‹□[ƒv,ð□\□¬,µ,Ü,·□B
4. Žè□‡ 3 ,Å□\□¬,µ,½ƒOƒ‹□[ƒv,ð□Š—L,·,éƒm□[ƒh,É Microsoft Transaction Server 2.0 ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B□Ú□×,É,Â,¢,Ä,Í□A□ufZfbfgfAfbfv□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B
5. MTS ƒZfbfgfAfbfv,ªƒVƒXƒef€□ã,É MSCS ,ª, ,é,±,Æ,ðŒŸ□o,·,é,Æ□AMS DTC ,ÌƒCƒ“ƒXƒg□[ƒ‹□æ,Ì ‰¼‘zƒT□[ƒo□[–¼,ðŽw’è,·,é,æ,¤,É‹□,ß,éƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ª•\Ž¦,³,ê,Ü,·□BŽè□‡ 3 ,Å□\ □¬,µ,½ƒlƒbƒgƒ□□[ƒN–¼ƒŠƒ\□[ƒX,Ì–¼‘O,ðŽw’è,µ,Ü,·□B
6. “¯,¶ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□AŽè□‡ 3 ,Å□\□¬,µ,½‹¤—LƒffBƒXƒN□ã,Ì MS DTC ƒ□ƒO ƒtƒ@ƒCƒ‹,Ì□ê□Š,ðŽw’è,µ,Ü,·□B
7. [ŽŸ,Ö] ,ðƒNƒŠƒbƒN,µ,Ä MTS ƒZfbfgfAfbfv,ðŒp‘±,µ,Ü,·□B
8. ƒNƒ‰ƒXƒ^,Ì 2 ,Â,ß,ÌƒRƒ“ƒsƒ…□[ƒ^,É MTS ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□BƒZfbfgfAfbfv’†,É‰¼‘zƒT□[ƒo□[,Æƒ□ƒO ƒtƒ@ƒCƒ‹,Ì□ê□Š,ðŽw’è,·,é,æ,¤,É‹□,ß,éƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Í□A•\Ž¦,³,ê,Ü,¹,ñ□B

2 ,Â,ÌƒNƒ‰ƒXƒ^ ƒm□[ƒh,Å•À□s,µ,Ä MTS ƒZfbfgfAfbfv,ðŽÀ□s,µ,È,¢,Å,,¾,³,¢□B1 ,Â,Ìƒm□[ƒh,É MTS ,ðŠ®‘S,ÉƒCƒ“ƒXƒg□[ƒ‹,µ□A□Å□‰,Ìƒm□[ƒh,ð□Ä‹N“®,¹,¸,É 2 ,Â,ß,Ìƒm□[ƒh,É MTS ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B—¼•û,Ìƒm□[ƒh,É MTS ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,½,ç□A—¼•û,Ìƒm□[ƒh,ð□Ä‹N“®,µ,Ü,·□B

’□ Microsoft Transaction Server 2.0 ,ð Microsoft Cluster Server 1.0 ,Æˆê□□,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é‘O,É□AMicrosoft Transaction Server 1.0 ,ðˆí□œ,·,é•K—v,Í, ,è,Ü,¹,ñ□BMTS 2.0 ,ð MSCS 1.0 ,Æˆê□□,ÉƒCƒ“ƒXƒg□[ƒ‹,Æ,«,ÌŽè□‡,É□],Á,Ä‘€□ì,µ,Ä,,¾,³,¢□B

▸ Microsoft Transaction Server 1.1 ,Æ Microsoft Cluster Server ,Ì‘g,Ý□‡,í,¹,©,ç Microsoft Transaction Server 2.0 ,Æ Microsoft Cluster Server ,Ì‘g,Ý□‡,í,¹,ÉƒAƒbƒvƒOƒŒ□[ƒh,·,é,É,Í

1. Microsoft Transaction Server 1.1 ,ð Microsoft Cluster Server ,Æ‘g,Ý□‡,í,¹,ÄŽg,Á,Ä,¢ ,½,Æ,«,É□ì□¬,µ,½ Microsoft Transaction Server ƒpƒbƒP□[ƒW ƒŠƒ\□[ƒX,ð,·,×,Ä□í□œ,µ,Ü,·□B
2. MS DTC ,ÌƒŠƒ\□[ƒX,ð□Š—L,·,éƒNƒ‰ƒXƒ^ ƒm□[ƒh,Å Microsoft Transaction Server 2.0 ,ÌƒZfbfgfAfbfv ƒvƒ□ƒOƒ‰ƒ€ ,ðŽÀ□s,µ,Ü,·□B,»,Ìƒm□[ƒh,ÅŽ©“®“I,ÉƒAƒbƒvƒOƒŒ□[ƒh,ªŽÀ□s,³,ê,Ü,·□BŽŸ,ÌŽè□‡,É□i,Þ‘O,É□AMicrosoft Transaction Server ,ÌƒZfbfgfAfbfv,ªŠ®—¹,µ,Ä,¢,é,±,Æ,ðŠm”F,µ,Ä,,¾,³,¢□B
3. ƒNƒ‰ƒXƒ^,Ì 2 ,Â,ß,Ìƒm□[ƒh,Å Microsoft Transaction Server 2.0 ,ÌƒZfbfgfAfbfv ƒvƒ□ƒOƒ‰ƒ€ ,ðŽÀ□s,µ,Ü,·□B

4 —¼•û,ìƒNƒ‰ƒXƒ^ ƒm□[ƒh,ð□Ä‹N“®,µ,Ü,·□B

MTS ƒT□[ƒo□[,Ì Microsoft Cluster Server ,ð□í□œ,·,é

ƒNƒ‰ƒXƒ^,Ìƒm□[ƒh,Ì 1 ,Â,©,ç Microsoft Cluster Server ,ð□í□œ,·,é□ê□‡,Í□AŽŸ,ÌŽè□‡,É□],Á,Ä‘€□ì,µ,Ü,·□B

1 ƒNƒ‰ƒXƒ^ ƒAƒhƒ~ƒjƒXƒgƒŒ□[ƒ^,ðŽg,Á,Ä MS DTC ,ÌƒŠƒ\□[ƒX,ðƒlƒtƒ‰ƒCƒ“,É,µ,Ü,·□B

2 ƒm□[ƒh,©,ç Microsoft Cluster Server ,ð□í□œ,µ□Aƒm□[ƒh,ð□Ä‹N“®,µ,Ü,·□B

3 Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AMS DTC ,ÌƒfO ƒtƒ@ƒCƒ‹,Ì□ê□Š,ð•Ï□X,µ,Ü,·□Bƒ□ƒO ƒtƒ@ƒCƒ‹,Í”ñ‹¤—LƒfƒBƒXƒN,É'u,©,È,¯,ê,Î,È,è,Ü,¹,ñ□B”ñ‹¤—LƒfƒBƒXƒN,É'u,©,È,¢,Æ□Aƒ□ƒO ƒtƒ@ƒCƒ‹,ª ‰ó,ê,½,è□AMS DTC ,ÌƒAƒNƒZƒXˆá”½,ª”□¶,µ,½,è,·,é,±,Æ,ª, ,è,Ü,·□B

ƒNƒ‰ƒXƒ^ ƒT□[ƒo□[,Ì MS DTC ƒ□ƒO ƒtƒ@ƒCƒ‹,ðƒŠƒZƒbƒg,·,é

ƒNƒ‰ƒXƒ^ ƒT□[ƒo□[,Ì MS DTC ƒ□ƒO ƒtƒ@ƒCƒ‹,ðƒŠƒZƒbƒg,·,é,É,Í□Aƒ□ƒO ƒtƒ@ƒCƒ‹,ðŠÜ,Þ‹¤—LƒfƒBƒXƒN,ðŒ»□Ý□Š—L,µ,Ä,¢,éƒm□[ƒh,Å MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽÀs,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ƒNƒ‰ƒXƒ^ ƒT□[ƒo□[,Å MS DTC ,ðŠJŽn,Ü,½,Í'âŽ~,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ü,½,Í MSCS ŠÇ—□ƒt□[ƒeƒBƒŠƒeƒB,Ì,¢,¸,ê,©,ðŽg—p,µ,Ä□AƒNƒ‰ƒXƒ^ ƒT□[ƒo□[,Å MS DTC ,ðŠJŽn,Ü,½,Í'âŽ~,Å,«,Ü,·□B"net start msdtc" ƒRƒ}ƒ“ƒh,Ü,½,Í "net stop msdtc" ƒRƒ}ƒ“ƒh,Í‹@”\ ,µ,Ü,¹,ñ□BƒRƒ}ƒ“ƒh ƒ‰ƒCƒ“,©,ç MSCS ƒNƒ‰ƒXƒ^,Ì MS DTC ,ðŠJŽn,Ü,½,Í'âŽ~,·,é,É,Í□A"msdtc -start" ,Æ "msdtc -stop" ,ðŽg,¢,Ü,·□B

MSCS ,ÌƒtƒF□[ƒ‹ƒI□[ƒo□[Œã,É MTS ƒIƒuƒWƒFƒNƒg,ðŒÄ,Ñ□o,·

Microsoft Transaction Server ƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í□A•K,¸□A□áŠQ,ª”□¶,µ,½ƒm□[ƒh,Ì MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ö,ÌŽQ□Æ,ð,·,×,Ä‰ð•ú,µ□AƒRƒ“ƒ| □[ƒlƒ“ƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□Ä□¶□¬,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,Í 2 ,Â,ß,Ìƒm□[ƒh,Å□¶□¬,³,ê,Ü,·□B

ƒNƒ‰ƒXƒ^ ƒT□[ƒo□[,Æ‘g,Ý□‡,í,¹,Ä [ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðŽg,¤

[ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðŽg,Á,Ä Microsoft Cluster Server (MSCS) ,Ìƒm□[ƒh,©,çƒRƒ“ƒ| □[ƒlƒ“ƒg,ðƒvƒ‹,·,é,É,Í□AŽŸ,ÌŽè□‡,É□],Á,Ä‘€□ì,µ,Ü,·□B

1 ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒvƒ‹,·,éƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,Å□A[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ƒAƒCƒRƒ“,ðƒ}ƒEƒX,Ì ‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒpƒefB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

2 [ƒlƒvƒVƒ‡ƒ“] ƒ^ƒu,ðƒNƒŠƒbƒN,µ□A‰¼‘zƒT□[ƒo□[,Ì–¼‘O,ð [ƒŠƒ,□[ƒg ƒT□[ƒo□[–¼] ƒ{ƒbƒNƒX,É“ü—Í,µ,Ü,·□BŽŸ,É□A[OK] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

3 ƒvƒ‹,µ,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìˆê—□æƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,Å□AƒT□[ƒo□[,Ì•¨—□–¼,ðŽg,Á,Ä MSCS ƒRƒ“ƒsƒ…□[ƒ^,ð [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,É'Ç‰Á,µ,Ü,·□B

4 ŽŸ,Ì,¢,¸,ê,©,Ì•û–@,Å□AƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,µ,Ü,·□B

- [ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ð‘I'ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ| ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
- [ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ| ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

5 [ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ÅƒCƒ“ƒXƒg□[ƒ‹,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'I'ð,µ,Ü,·□BƒŠƒ,□[ƒg ƒRƒ“ƒ| □[ƒlƒ“ƒg,ìƒCƒ“ƒXƒg□[ƒ‹,·†,Í□A(ƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒŠƒ,□[ƒg ƒT□[ƒo□[–¼] ƒ{ƒbƒNƒX,ÅŽw'è,µ,½) ƒT□[ƒo□[,Ì‰¼‘z–¼,Å,È,□A•¨—□–¼,ª•\Ž¦,³,ê,é,±,Æ,É'ˆÓ,µ,Ä,,¾,³,¢□B

6 [ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ð‘I'ð,µ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ƒc□[ƒ‹ ƒo□[,Ì [ƒvƒ□ƒpƒefB•\Ž¦] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□Bƒvƒ□ƒpƒefB•\Ž¦,ÌƒEƒBƒ“ƒhƒE,Ì [ƒT□[ƒo□[] —ñ,É‰¼‘zƒT□[ƒo□[–¼,ª•\ Ž¦,³,ê,Ü,·□B

’□ MSCS ƒRƒ“ƒsƒ…□[ƒ^,ìƒŠƒ‚□[ƒg ƒT□[ƒo□[–¼,ð•Ï□X,µ,½□ê□‡,Í□AƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ÉƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□ÄƒCƒ“ƒXƒg□[ƒ‹,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

‹ó,ÌƒpƒbƒP□[ƒW,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“

MTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Å,Í□A‹ó,ÌƒpƒbƒP□[ƒW,ÍƒŒƒvƒŠƒP□[ƒg,Å,«,Ü,¹,ñ□B

MTS ƒpƒbƒP□[ƒW ID ,Í MSCS ƒNƒ‰ƒXƒ^,Ì Administrator ƒOƒ‹□[ƒv,É‘®,µ,Ä,¢ ,È,¯,ê,Î,È,ç,È,¢

MSCS ƒNƒ‰ƒXƒ^□ã,Ì MTS ƒpƒbƒP□[ƒW,Ì□ê□‡□AƒpƒbƒP□[ƒW ID ,É‘Î ‰ž,·,éƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,Í□AƒVƒXƒeƒ€ŠÇ—□ŽÒ,Å, ,é,©□A,Ü,½,Í (cluadmin ,Å□Ý’è,µ,½) Full Control ƒNƒ ‰ƒXƒ^ ƒAƒNƒZƒXŒ ŒÀ,ðŽ□,Á,Ä,¢,é•K—v,ª, ,è,Ü,·□BƒVƒXƒeƒ€ŠÇ—□ŽÒ,Ì ID ,ÌŒ³,ÅƒpƒbƒP□[ƒW,ðŽÀ□s,µ,È,©,Á,½□ê□‡,Í□AMTS ,Í MS DTC ,É□Ú‘±,Å,«,Ü,¹,ñ□B

MTS ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é‘O,É MSCS ƒNƒ‰ƒXƒ^,ðƒVƒƒƒbƒgƒ‚ƒEƒ“,·,é

MTS ƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€,ðŽÀ□s,·,é‘O,É□AMTS ƒZƒbƒgƒAƒbƒv,ª‰e‹¿,ð—^,¦,é (SQL Server ,È,Ç,Ì) ,·,×,Ä,Ì MSCS ƒNƒ‰ƒXƒ^ ƒŠƒ\□[ƒX,ðƒVƒƒƒbƒgƒ‚ƒEƒ“,·,é•K—v,ª, ,è,Ü,·□B

MSCS ƒT□[ƒo□[,Å,Í ƒŠƒ‚□[ƒg ƒT□[ƒo□[–¼,É‰¼‘zƒT□[ƒo□[,ðŽw’è,µ,È,¯,ê,Î,È,ç,È,¢

ƒNƒ‰ƒXƒ^ ƒT□[ƒo□[,Å,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä [ƒŠƒ‚□[ƒg ƒT□[ƒo□[–¼] ƒvƒ□ƒpƒeƒB,É ‰¼‘zƒT□[ƒo□[–¼,ðŽw’è,·,é•K—v,ª, ,è,Ü,·□B,±,ê,É,æ,Á,Ä□AƒŠƒ‚□[ƒg ƒNƒ‰ƒCƒAƒ“ƒg,ÍƒNƒ‰ƒXƒ^“à,Ì MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉƒAƒNƒZƒX,Å,«,Ü,·□BŽ¸”s,µ,½□ê□‡□AƒNƒ‰ƒCƒAƒ“ƒg,ÍƒNƒ‰ƒXƒ^,Ì‘Á’è,Ìƒm□[ƒh,Å,Í,È,ƒNƒ ‰ƒXƒ^,ðŽQ□Æ,µ,Ü,·□B

MTS ,ð Oracle ,ÉƒAƒNƒZƒX,Å,«,é,æ,¤,ÉÝ’è,·,é

Microsoft Transaction Server ƒRƒ“ƒ|[ƒlƒ“ƒg,ÍAŽŸ,ÌƒŠƒ\[ƒX,ÉƒAƒNƒZƒX,Å,«,Ü,·B

- Oracle 7 ,¨,æ,Ñ Oracle 8 ƒf[ƒ^ƒx[ƒX
- Windows NT ,¨,æ,Ñ Unix ƒVƒXƒeƒ€ã,Ì Oracle ƒf[ƒ^ƒx[ƒX
- Oracle Workgroup Servers
- Oracle Enterprise Servers
- Oracle Parallel Servers

,±,±,Å,ÍAŽŸ,ÌƒgƒsƒbƒN,É,Â,¢,Äà–¾,µ,Ü,·B

•K—v,Èƒ\ƒtƒgƒEƒFƒA

Microsoft Transaction Server ƒRƒ“ƒ|[ƒlƒ“ƒg,©,ç Oracle ƒf[ƒ^ƒx[ƒX,ðƒAƒNƒZƒX,·,é,É,ÍAŽŸ,Ìƒ\ƒtƒgƒEƒFƒA,ª•K—v,Å,·B

ƒRƒ“ƒ|[ƒlƒ“ƒg	ƒo[ƒWƒ‡ƒ“
Windows NT ã,Ì Oracle ƒf[ƒ^ƒx[ƒX	7.3.4 ,Ü,½,Í Oracle 8
Unix ã,Ì Oracle ƒf[ƒ^ƒx[ƒX	Œ»ƒo[ƒWƒ‡ƒ“,Æƒpƒbƒ`A,Ü,½,Í Oracle 8
Microsoft Transaction Server 2.0	2.0
Microsoft Oracle ODBC Driver 2.0	02.73.7283.1 ˆÈ~
ActiveX Data Objects (ADO)	1.5 ˆÈ~

d—v: ƒ\ƒtƒgƒEƒFƒA,Ìˆê‘O,Ìƒo[ƒWƒ‡ƒ“,ÍA³,µ,“®ì,µ,Ü,¹,ñB•K,¸A‘O,Ì•\,ÅŽ¦,µ,½ƒo[ƒWƒ‡ƒ“ (,Ü,½,ÍˆÈ~,Ìƒo[ƒWƒ‡ƒ“) ,Ìƒ\ƒtƒgƒEƒFƒA,ðƒCƒ“ƒXƒg[ƒ‹,µ,Ä,,¾,³,¢BMicrosoft Transaction Server ,ð Oracle ,Æ,Æ,à,ÉŽg—p,·,éê‡,É”¶,·,é–â‘è,ÌŒ´ˆö,ÍA‘½,,Ìê‡,±,Ì“_,É ,è,Ü,·B

Windows NT ã,Ì Oracle 7.3 ƒf[ƒ^ƒx[ƒX

MTS ,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|[ƒlƒ“ƒg,ÍAWindows NT ã,Ì Oracle 7.3 ƒf[ƒ^ƒx[ƒX,ÉƒAƒNƒZƒX,Å,«,Ü,·BOracle 7.3.4 Workgroup Server release for Windows NT ,Ü,½,Í Oracle 7.3.4 Enterprise Server release for Windows NT ,ðŽg—p,Å,«,Ü,·B,± ,ê,æ,è‘O,ÌƒŠƒŠ[ƒX,ÍAMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Æ‹¤,ÉŽg—p,·,é,±,Æ,Í,Å,«,Ü,¹,ñB

Windows NT ã,Ì Oracle 8 ƒf[ƒ^ƒx[ƒX

MTS ,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|[ƒlƒ“ƒg,ÍAWindows NT ã,Ì Oracle 8 ƒf[ƒ^ƒx[ƒX,ÉƒAƒNƒZƒX,Å,«,Ü,·B

Oracle 7.3.4 ƒNƒ‰ƒCƒAƒ“ƒg ƒ\ƒtƒgƒEƒFƒA,ð MTS ƒRƒ“ƒ|[ƒlƒ“ƒg,ª ,éƒVƒXƒeƒ€,ÉƒCƒ“ƒXƒg[ƒ‹,·,é•K—v,ª ,è,Ü,·BMicrosoft Oracle ODBC Driver 2.0 ,¾,¯,ªAOracle 7.3 ƒNƒ‰ƒCƒAƒ“ƒg ƒCƒ“ƒ^[ƒtƒFƒCƒX,Æ‹¤,ÉŽg,¤,±,Æ,ª,Å,«,Ü,·BOracle 8 ƒNƒ‰ƒCƒAƒ“ƒg ƒCƒ“ƒ^[ƒtƒFƒCƒX,Æ‹¤,ÉŽg—p,·,é,±,Æ,Í,Å,«,Ü,¹,ñB

Oracle 8 ,ÌƒhƒLƒ…ƒ□ƒ“ƒg,Å,Í□AOracle 8 ,ð Oracle 7 ƒNƒ‰ƒCƒAƒ“ƒg ƒ\ƒtƒgƒEƒFƒA,Æ‹¤'¶,·,é,æ,¤ ,ÉƒCƒ“ƒXƒg□[ƒ‹,µ,Ä□Ý'è,·,é•û–@,ª□à–¾,³,ê,Ä,¢,Ü,·□B□Ú□×,É,Â,¢,Ä,Í□AOracle 8 ,ÌƒhƒLƒ… ƒ□ƒ“ƒg,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

UNIX □ã,Ì Oracle 7 ƒf□[ƒ^ƒx□[ƒX

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,Í□AUNIX □ã,Ì Oracle 7.3 ƒf□[ƒ^ƒx□[ƒX,ðƒAƒNƒZƒX,Å,«,Ü,·□B

Žg—p,·,é UNIX ƒvƒ‰ƒbƒgƒtƒH□[ƒ€—p,Ì Oracle 7.3.3 release ˆÈ□~,ðƒCƒ“ƒXƒg□[ƒ‹,·,é•K—v,ª, ,è,Ü,·□B,Ù,Æ,ñ,Ç,Ì□ê□‡□AOracle 7.3.3 patch release for Oracle on UNIX ,àƒCƒ“ƒXƒg□[ƒ‹,·,é•K—v,ª, ,è,Ü,·□B

Žg—p,·,é UNIX ƒvƒ‰ƒbƒgƒtƒH□[ƒ€,Å Oracle 7.3.3 patch release ,ª•K—v,©,Ç,¤,©,Í□AOracle ƒJƒXƒ^ƒ} ƒTƒ|□[ƒg,ÉŠm”F,·,é•K—v,ª, ,è,Ü,·□BOracle 7.3.4 release on Windows NT ,ª'ñ‹Ÿ,·,é XA ƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒTƒ|□[ƒg‹@"\,ðŽg,Á,Ä□AUNIX □ã,Ì Oracle ƒf□[ƒ^ƒx□[ƒX,ðƒAƒNƒZƒX,µ,æ,¤,Æ,µ,Ä,¢,é,± ,Æ,ð□à–¾,µ,Ä,,¾,³,¢□B

UNIX □ã,Ì Oracle 8 ƒf□[ƒ^ƒx□[ƒX

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,Í□AUNIX □ã,Ì Oracle 8 ƒf□[ƒ^ƒx□[ƒX,ðƒAƒNƒZƒX,Å,«,Ü,·□B

Oracle 7.3.4 ƒNƒ‰ƒCƒAƒ“ƒg ƒ\ƒtƒgƒEƒFƒA,ð□AMTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ª, ,é Windows NT ƒVƒXƒeƒ€ ,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é•K—v,ª, ,è,Ü,·□BMicrosoft Oracle ODBC Driver 2.0 ,Ì,Ý□AOracle 7.3 ƒNƒ‰ƒCƒAƒ“ƒg ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Æ‹¤,ÉŽg—p,Å,«,Ü,·□B,±,ê,Í□AOracle 8 ƒNƒ‰ƒCƒAƒ“ƒg ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Æ‹¤,ÉŽg—p,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

Oracle ƒNƒ‰ƒCƒAƒ“ƒg ƒ\ƒtƒgƒEƒFƒA

Oracle 7.3.4 ƒNƒ‰ƒCƒAƒ“ƒg ƒ\ƒtƒgƒEƒFƒA,ð□AMTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ª, ,é Windows NT ,Ü,½,Í Windows 95/98 ƒVƒXƒeƒ€,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é•K—v,ª, ,è,Ü,·□BMicrosoft Oracle ODBC Driver 2.0 ,Ì,Ý□AOracle 7.3 ƒNƒ ‰ƒCƒAƒ“ƒg ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Æ‹¤,ÉŽg—p,Å,«,Ü,·□B,±,ê,Í□AOracle 8 ƒNƒ‰ƒCƒAƒ“ƒg ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Æ‹¤,ÉŽg—p,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

Oracle OCIW32.DLL

Oracle 7.3 release ,Æ‹¤,ÉŽg—p,Å,«,é Oracle OCIW32.DLL ,ÌŽw'è,³,ê,½ƒo□[ƒWƒ‡ƒ“,ðƒVƒXƒeƒ€ ,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é•K—v,ª, ,è,Ü,·□B,±,Ì DLL ,Í□AOracle 7.3 CD ,Ì "WIN32\V7\RSF73" ƒfƒBƒŒƒNƒgƒŠ,É, ,è,Ü,·□B

Oracle 7.3 CD ,É,Í□AOCIW32.DLL ,ÌŒÃ,¢ƒo□[ƒWƒ‡ƒ“,àŠÜ,Ü,ê,Ä,¢,Ü,·□B,±,Ì DLL ,Í Oracle 7.2 release ,É'Î‰ž,µ□A"WIN32\V7\RSF72" ƒfƒBƒŒƒNƒgƒŠ,ÉŠÜ,Ü,ê,Ä,¢,Ü,·□B,±,Ì DLL ,ð MTS ,Æ‹¤,ÉŽg—p,·,é,± ,Æ,Í,Å,«,Ü,¹,ñ□B

Oracle OCIW32.DLL ,ÌŽŸ,Ìƒo□[ƒWƒ‡ƒ“,Í "WIN32\V7\RSF72" ƒfƒBƒŒƒNƒgƒŠ,É, ,è□AMTS ,Æ‹¤,ÉŽg—p,·,é,ÆŽ¸”s,·,é,±,Æ,ª'm,ç,ê,Ä,¢,Ü,·□B

```
Version 7.x
Thursday, February 01, 1996 12:50:06 AM
Size 36 KB
```

Oracle XA73.DLL

Oracle XA73.DLL ,ÌŽw'è,³,ê,½ƒo□[ƒWƒ‡ƒ“,ð□AMTS ƒVƒXƒeƒ€,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é•K—v,ª, ,è,Ü,·□B

Microsoft Transaction Server 2.0

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŽg,Á,Ä Oracle ƒf□[ƒ^ƒx□[ƒX,ÉƒfƒAƒNƒZƒX,·,é,É,Í□AMicrosoft Transaction Server 2.0 ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é•K—v,ª, ,è,Ü,·□B

Microsoft ODBC Driver for Oracle

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŽg—p,·,é,É,Í□AMicrosoft ODBC Driver 2.0 for Oracle (MSORCL32.DLL) version 02.73.7283.1 ˆÈ□~,ª•K—v,Å,·□BWindows NT 4.0 Option Pack ƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€,Í□A,± ,Ì DLL ,ðŽ©“®“I,ÉƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B

Oracle ƒf□[ƒ^ƒx□[ƒX,ÉƒAƒNƒZƒX,·,é□ê□‡□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒTƒ|□[ƒg,ª•K—v,È,¢□ê□‡,Å,à□V,µ,¢ Microsoft Oracle ODBC Driver 2.0 ,ðŽg—p,·,é,±,Æ,ð‹,□„□§,µ,Ü,·□B,±,Ì□V,µ,¢ƒhƒ‰ƒCƒo,Í□AOracle 1.0 ƒhƒ ‰ƒCƒo,æ,è,à□,,¢ƒpƒtƒH□[ƒ}ƒ“ƒX,ð’ñ‹Ÿ,µ,Ü,·□BOracle 1.0 ƒhƒ‰ƒCƒo,Í□A,·,×,Ä,ÌƒAƒNƒefBƒrƒefB,ðƒhƒ‰ƒCƒo ƒŒƒxƒ‹,ÅƒVƒŠƒAƒ‹‰»,µ□A—v‹□,Íƒhƒ‰ƒCƒo,ð‰î,µ,ÄƒVƒ“ƒOƒ‹ƒXƒŒƒbƒh‰»,³,ê,Ü,·□BOracle 2.0 ƒhƒ ‰ƒCƒo,Í□A,·,×,Ä,ÌƒAƒNƒefBƒrƒefB,ð□Ú‘±ƒŒƒxƒ‹,ÅƒVƒŠƒAƒ‹‰»,µ,Ü,·□B,± ,ê,É,æ,Á,Ä□AˆÙ,È,éƒf□[ƒ^ƒx□[ƒX□Ú‘±,ð“¯Žž,ÉŽg—p,Å,«,Ü,·□B

ActiveX Data Objects (ADO)

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª ADO ,ðŽg—p,·,é□ê□‡□AADO version 1.5 ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é•K—v,ª, ,è,Ü,·□B,»,êˆÈ‘O,Ì ADO ,ÌƒŠƒŠ□[ƒX,Í□A□V,µ,¢ ODBC 3.5 Driver Manager ,Æ,Æ,à,ÉŽg—p,·,é,±,Æ,ª,Å,«,Ü,¹,ñ□BADO 1.5 ,Í□AWindows NT 4.0 Option Pack ,Å’ñ‹Ÿ,³,ê,Ü,·□B

Oracle ƒTƒ|□[ƒg,ð□Ý’è,·,é

MTS ,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð Oracle ,ªƒTƒ|□[ƒg,·,é,æ,¤,É□Ý’è,·,é,É,Í

1. Oracle Database Server ƒ\ƒftƒgƒEƒFƒA,ÌƒCƒ“ƒXƒg□[ƒ‹

   Šù,É□à–¾,µ,½,æ,¤,É□A“K□Ø,È Oracle ƒ\ƒftƒgƒEƒFƒA,ðƒf□[ƒ^ƒx□[ƒX ƒT□[ƒo□[ƒVƒXƒeƒ€ ,ÉƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B

2. Oracle ƒNƒ‰ƒCƒAƒ“ƒg ƒ\ƒftƒgƒEƒFƒA,ÌƒCƒ“ƒXƒg□[ƒ‹

   Oracle 7.3.4 ƒNƒ‰ƒCƒAƒ“ƒg ƒ\ƒftƒgƒEƒFƒA,ð MTS ƒVƒXƒeƒ€,ÉƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B

   □u•K—v,Èƒ\ƒftƒgƒEƒFƒA□v,Ì•\,É‹L□Ú,³,ê,½□³,µ,¢ƒo□[ƒWƒ‡ƒ“,Ì Oracle OCIW32.DLL ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢ ,é,±,Æ,ðŠm”F,µ,Ü,·□B

3. Microsoft Transaction Server 2.0 ,ÌƒCƒ“ƒXƒg□[ƒ‹

   Microsoft Transaction Server 2.0 ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,Æ□AŽŸ,Ìƒ\ƒftƒgƒEƒFƒA,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ü,·□B

   □E Microsoft Transaction Server 2.0 (Microsoft OCI Interface ,ðŠÜ,Þ)

   □E Microsoft ODBC 3.5 Driver Manager

   □E Microsoft ODBC Driver for Oracle 2.0

   □E ADO 1.5

4. DTCXATM.LOG ,Ì□í□œ

   Microsoft Transaction Server 2.0 ,Ìƒx□[ƒ^”Å,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ä,¢,È,¢□ê□‡,Í□AŽŸ,ÌƒXƒefbƒv,É□i,ñ,Å,- ,¾,³,¢□B

   Microsoft Transaction Server 2.0 ,Ìƒx□[ƒ^”Å,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ä,¢,é□ê□‡,Í□AWindows ƒGƒNƒXƒvƒ□□[ƒ‰ ,ðŽg,Á,Ä DTCXATM.LOG ƒtƒ@ƒCƒ‹,ªƒVƒXƒeƒ€,É, ,é,©,Ç,¤,©,ðŠm”F,µ,Ü,·□B,à,µ, ,ê,Î□AMS DTC ƒT□[ƒrƒX,ð’âŽ~,µ,Ä DTCXATM.LOG ƒtƒ@ƒCƒ‹,ð□í□œ,µ,Ü,·□B

   DTCXATM.LOG ƒtƒ@ƒCƒ‹,ð□í□œ,µ,È,¯,ê,Î,È,ç,È,¢,Ì,Í□AMicrosoft Transaction Server 2.0 ƒx□[ƒ^”Å,©,çƒAƒbƒvƒOƒŒ□[ƒh,·,é,Æ,«,¾,¯,Å,·□BDTCXATM.LOG ƒtƒ@ƒCƒ‹,Í□C•œ,É•K—v,È□d— v,È□î•ñ,ðŠÜ,ñ,Å,¢,é‰Â”\□«,ª, ,é,Ì,Å□A,±,êˆÈ□~,Í□â‘Î,É DTCXATM.LOG ƒtƒ@ƒCƒ‹,ð□í□œ,µ,È,¢,Å,- ,¾,³,¢□B

5. Oracle XA ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒTƒ|□[ƒg,ð—LŒø,É,µ,Ü,·□B

   Oracle XA ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒTƒ|□[ƒg,ð—LŒø,É,·,é,É,Í□AŽŸ,Ì‘€□ì,ð□s,¢,Ü,·□B

1 Oracle ƒVƒXƒeƒ€ŠÇ—□ŽÒ,Í V$XATRANS$,Æ,¢,¤ƒrƒ…□[,ð□ì□¬,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□Bƒrƒ…

□[,ð□ì□¬,·,é,É,Í□AOracle ,ª’ñ‹Ÿ,·,éƒXƒNƒŠƒvƒg "xaview.sql" ,ðŽÀ□s,·,é•K—v,ª, ,è,Ü,·□B,± ,Ìƒtƒ@ƒCƒ‹,Í□A’Ê□í C:\ORANT\RDBMS73\ADMIN ,É, ,è,Ü,·□B

2 ƒVƒXƒeƒ€ŠÇ—□ŽÒ,Í□A,±,ê,ç,Ìƒrƒ…□[,ð‘I‘ð (SELECT) ,·,éŒ ŒÀ,ð Public ƒt□[ƒU□[,É— ^,¦,È,¯,ê,Î,È,è,Ü,¹,ñ□B

```
Grant Select on V$XATRANS$ to public.
```

3 Oracle Instance Manager ,Å [•\Ž¦] ƒ□ƒjƒ…□[,Ì [Šg’£ƒ,□[ƒh] ,ðƒNƒŠƒbƒN,µ□AƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Ì [□ ‰Šú‰»ƒpƒ‰ƒ□□[ƒ^] ,ð‘I‘ð,µ,Ü,·□BƒEƒBƒ“ƒhƒE,Ì‰E‘¤,Å [Šg’£ƒ`ƒ…□[ƒjƒ“ƒO] ,ð‘I‘ð,µ□A"distributed_transactions" ƒpƒ‰ƒ□□[ƒ^,ð,Ì’l,ð‘□,â,·,Æ□A,æ,è‘½,,Ì MTS “¯ŽžŽÀ□sƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª“¯Žž,Éƒf□[ƒ^ƒx□[ƒX,ð□X□V,Å,«,é,æ,¤,É,È,è,Ü,·□B

Oracle ,Ì XA ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒTƒ|□[ƒg,ð□\□¬,·,é•û–@,É,Â,¢,Ä,Í□AOracle Server ,ÌƒhƒLƒ… ƒ□ƒ“ƒg,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

“□□‡ƒZƒLƒ…ƒŠƒefB,Ì□\□¬

“□□‡ƒZƒLƒ…ƒŠƒefB,É,æ,Á,Ä□AOracle ƒf□[ƒ^ƒx□[ƒX,Í□Aƒf□[ƒ^ƒx□[ƒX ƒt□[ƒU□[,ðŠm”F,·,é,½,ß,É Windows NT ,Ì”F□Ø,ð—˜—p,·,é,±,Æ,ª,Å,«,Ü,·□B,±,ê,É,æ,Á,Ä□Aƒt□[ƒU□[,Í•Ê,Ìƒ□ƒOƒCƒ“ ID ,Ü,½,Íƒpƒ‰ƒ□□[ƒh,ðŽw’è,µ,È,,Ä,à Oracle ƒf□[ƒ^ƒx□[ƒX,Éƒ□ƒOƒCƒ“,Å,«,Ü,·□Bƒt□[ƒU□[,Í□AWindows NT ,¨,æ,Ñ Oracle ƒf□[ƒ^ƒx□[ƒX,Ì—¼•û,É‘Î,µ,Äƒ□ƒOƒCƒ“ ID ,Æƒpƒ‰ƒ□□[ƒh,ð ,P ,Â,¾,¯•ÛŽ□,·,ê,Î,æ,¢ ,í,¯,Å,·□B

Oracle ƒf□[ƒ^ƒx□[ƒX,É□Ú‘±,·,é,Æ,«,É□AMTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ª□í,Éƒ□ƒOƒCƒ“ ID ,Æƒpƒ‰ƒ□□[ƒh,ðŽw’è,·,é□ê□‡,Í□A“□□‡ƒZƒLƒ…ƒŠƒefB,ÍŽg—p,µ,Ü,¹,ñ□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª DSN ,ð ‰îˆµ,Ä□A’¼□Ú“I,Ü,½,ÍŠÔ□Ú“I,Éƒ□ƒOƒCƒ“ ID ,¨,æ,Ñƒpƒ‰ƒ□□[ƒh,ðŽw’è,·,é□ê□‡,ªŠY“–,µ,Ü,·□B,¢ ,¸,ê,Ì□ê□‡,à□A“□□‡ƒZƒLƒ…ƒŠƒefB,ðŽg,í,È,¢,Ì,Å□A,±,Ìƒfƒbƒv,Í•K—v, ,è,Ü,¹,ñ□B

“□□‡ƒZƒLƒ…ƒŠƒefB,ðŽg,¤□ê□‡□AOracle ƒf□[ƒ^ƒx□[ƒX,É□Ú‘±,·,é,½,ß,É”F□Ø,³,ê,½ƒ□ƒOƒCƒ“ ID ,¨,æ,Ñƒpƒ‰ƒ□□[ƒh,Ì,à,Æ,ÅŽÀ□s,·,é,æ,¤,É MS DTC ,ð□\□¬,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B,±,ê,Í□Aƒf□[ƒ^ƒx□[ƒX,ð ‰ñ•œ,·,éŠÔ,É□AMS DTC ,Í Oracle ƒf□[ƒ^ƒx□[ƒX,ðŠJ,«□AƒCƒ“ ƒ_ƒEƒg,Ìƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŒ‹‰Ê,ð•ñ□□,µ,È,¯,ê,Î,È,ç,È,¢,©,ç,Å,·□B

ŽŸ,ÌŽè□‡,Å MS DTC ,Ìƒ□ƒOƒCƒ“ ID ,ð□Ý’è,Å,«,Ü,·□B[ƒXƒ^□[ƒg] ƒ□ƒjƒ…□[,Ì [□Ý’è] ,ðƒ|ƒCƒ“ƒg,µ,Ä [ƒRƒ“ƒgƒ□□[ƒ‹ ƒpƒlƒ‹] ,ð‘I‘ð,µ,Ü,·□BŽŸ,ÉƒRƒ“ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Ì [ƒT□[ƒrƒX] ,ðŠJŽn,µ,Ü,·□Bˆê——,Ì [MSDTC] ,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ,Ü,·□B[ƒ□ƒOƒlƒ“] ,Ì [ƒAƒJƒEƒ“ƒg] ,ð‘I‘ð,µ,Ä□Aƒ□ƒOƒCƒ“ ID ,Æƒpƒ‰ƒ□□[ƒh,ðŽw’è,µ,Ü,·□BOracle ,ÌƒZƒLƒ…ƒŠƒefBŠÇ—□ƒc□[ƒ‹,ðŽg,Á,Ä□AŽw’è,µ,½ƒ□ƒOƒCƒ“ ID ,ª Oracle ƒf□[ƒ^ƒx□[ƒX,ðŠJ,¯,é,æ,¤,É”F□Ø,³,ê,Ä,¢,é,±,Æ,ðŠm”F,µ,Ü,·□B

Windows NT ƒZƒLƒ…ƒŠƒefB‹@”\,ð“□□‡,µ,½ Oracle ƒf□[ƒ^ƒx□[ƒX,É,Â,¢,Ä,Í□AOracle ,ÌƒhƒLƒ… ƒ□ƒ“ƒg,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

,æ,è‘½,,Ì□Ú‘±,ðƒTƒ|□[ƒg,·,é,æ,¤,É Oracle ,ð□\□¬,·,é

Oracle ƒf□[ƒ^ƒx□[ƒX,É‘Î,·,é□Ú‘±,ð□”□\^È□ã□ì□¬,·,é,É,Í□A,³,ç,É‘½,,Ìƒf□[ƒ^ƒx□[ƒX□Ú‘±,ðƒTƒ|□[ƒg,·,é,æ,¤ ,É Oracle ƒT□[ƒo□[,ð□\□¬,·,é•K—v,ª, ,è,Ü,·□B□Ú□×,É,Â,¢,Ä,Í□A□u‘½,,Ì□Ú‘±,ðƒTƒ|□[ƒg,·,é,æ,¤,É Oracle ,ð□\□¬,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒCƒ“ƒXƒg□[ƒ‹,ðƒeƒXƒg,µ,Ä MTS ,Ì Oracle ƒTƒ|□[ƒg,ð□\□¬,·,é

Oracle ƒTƒ|□[ƒg,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ä□\□¬,µ,½,ç□AMTS ,Æ,Æ,à,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,é Oracle ƒeƒXƒg ƒvƒ□ƒOƒ ‰ƒ€,ðŽg,Á,Ä Oracle ƒCƒ“ƒXƒg□[ƒ‹,ðŠm”F,·,é•K—v,ª, ,è,Ü,·□BOracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,Í□AMTS ,ª Oracle ,Ì OCI XA ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŽg—p,·,é,Æ“¯—l,Ì•û–@,Å□A,±,ê,ç,ðŽg—p,µ,Ü,·□B

Oracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,Í□AOracle ,Ì XA ‹@”\,ðŽg,Á,Ä Oracle ƒf□[ƒ^ƒx□[ƒX,É□Ú‘±,Å,«,é,©,Ç,¤ ,©,ð”»’f,µ,Ü,·□BOracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,Í□A•W□€,Ì Oracle ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Æƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‹@”\ ,ðŽg—p,µ,Ü,·□BMicrosoft Transaction Server ,Ü,½,Í Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒl□[ƒ^,ÍŽg—p,µ,Ü,¹,ñ□B,µ,½,ª,Á,Ä□AƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ÌŽ¸”s,Í□AOracle ƒVƒXƒeƒ€,ª□³,µ,- ƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,È,¢,©□A,Ü,½,Í□³,µ,□\□¬,³,ê,Ä,¢,È,¢,±,Æ,ðŽ¦,µ,Ü,·□BOracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€

,ªŽ¸”s,µ,½□ê□‡,Í□AOracle ,ð□Ä“xƒCƒ“ƒXƒg□[ƒ‹,µ,Ä□\□¬,µ,·,é,©□AOracle ,ÌƒTƒ|□[ƒg•”–å,É–â,¢□‡,í,¹,Ä,-,¾,³,¢□B

Oracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ðŽÀ□s,·,é,É,Í

1. □u•K—v,Èƒ\ƒtƒgƒEƒFƒA□v,Å□à–¾,³,ê,Ä,¢,é,æ,¤,É□A□³,µ,¢ƒo□[ƒWƒ‡ƒ“,Ìƒ\ƒtƒgƒEƒFƒA,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,é,±,Æ,ðŠm”F,µ,Ü,·□B
2. Oracle ƒf□[ƒ^ƒx□[ƒX,ðŽQ□Æ,·,é ODBC DSN ,ð□ì□¬,µ,Ü,·□BDSN ,ª□A□V,µ,¢ Microsoft Oracle ODBC 2.0 ƒhƒ‰ƒCƒo,ðŽg—p,·,é,æ,¤,É,µ,Ü,·□B
3. Oracle XA ƒTƒ|□[ƒg,ðŽg—p,Å,«,é,æ,¤,É,µ,Ü,·□B
4. Šù‘¶,Ì,·,×,Ä,Ì Oracle ƒgƒŒ□[ƒX ƒtƒ@ƒCƒ‹,ð□AOracle ƒf□[ƒ^ƒx□[ƒX,ðƒAƒNƒZƒX,·,é MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ª, ,éƒRƒ“ƒsƒ…□[ƒ^,©,ç□í□œ,µ,Ü,·□B,à,Á,Æ,à—eˆÕ,È•û–@,Í□AWindows ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä *.TRC ƒtƒ@ƒCƒ‹,ª, ,é□ê□Š,ðŠm”F,µ□A,·,×,Ä□í□œ,·,é,±,Æ,Å,·□B
   Oracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ªŽ¸”s,µ,½□ê□‡□A–â‘è,ÌŒ´ˆö,ð”»’f,·,é,½,ß,ÉƒgƒŒ□[ƒX ƒtƒ@ƒCƒ‹,ª–ð—§,Â□ê□‡,ª, ,è,Ü,·□B,·,×,Ä,ÌƒgƒŒ□[ƒX ƒtƒ@ƒCƒ‹,ðŠ®‘S,É□í□œ,·,é,±,Æ,Å□A□V,½,É□ì□¬,³,ê,½ƒtƒ@ƒCƒ‹,ð—eˆÕ,ÉŒ©,Â,¯,é,±,Æ,Å,«,Ü,·□B
5. Microsoft Transaction Server 2.0 ƒx□[ƒ^”Å,ðƒCƒ“ƒXƒg□[ƒ‹,µ,½□ê□‡,Í□AWindows ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä DTCXATM.LOG ƒtƒ@ƒCƒ‹,ªƒVƒXƒefe€,É, ,é,©,Ç,¤,©,ðŠm”F,µ,Ü,·□B,à,µ, ,ê,Î□AMS DTCƒT□[ƒrƒX,ð’âŽ~,µ,Ä DTCXATM.LOG ƒtƒ@ƒCƒ‹,ð□í□œ,µ,Ü,·□B
   Microsoft Transaction Server 2.0 ƒx□[ƒ^”Å,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ä,È,¢□ê□‡,Í□AŽŸ,ÌƒXƒefbƒv,É□i,ñ,Å,-,¾,³,¢□B
   DTCXATM.LOG ƒtƒ@ƒCƒ‹,ð□í□œ,µ,È,¯,ê,Î,È,ç,È,¢,Ì,Í□AMicrosoft Transaction Server 2.0 ƒx□[ƒ^”Å,©,çƒAƒbƒvƒOƒŒ□[ƒh,·,é,Æ,«,¾,¯,Å,·□BDTCXATM.LOG ƒtƒ@ƒCƒ‹,Í‰ñ•œ,É•K—v,È□d—v,È□î•ñ,ðŠÜ,ñ,Å,¢,é‰Â”\□«,ª, ,é,Ì,Å□A,±,êˆÈ□~,Í□â‘Î,É DTCXATM.LOG ƒtƒ@ƒCƒ‹,ð□í□œ,µ,È,¢,Å,-,¾,³,¢□B
6. MS-DOS ƒRƒ}ƒ“ƒh ƒvƒ□ƒ“ƒvƒg,©,ç Oracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€ (TestOracleXaConfig.exe) ,ðŽÀ□s,µ,Ä□AŽŸ,Ì—á,Ì,æ,¤,É Oracle ƒT□[ƒo□[,ÌƒtƒU□[ID□AƒpƒXƒ□□[ƒh□A,¨,æ,ÑƒT□[ƒrƒX–¼,ðŽw’è,µ,Ü,·□B

   ```
   c:>TestOracleXaConfig.exe  -U<user id> -P<Password>
   -S<Service_Name as contained in the TNS file>.
   ```

   ƒpƒ‰ƒ□□[ƒ^,ðŽw’è,µ,È,¢,ÅƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ðŽÀ□s,·,é,Æ□A•K—v,Èƒpƒ‰ƒ□□[ƒ^,ÉŠÖ,·,éƒwƒ‹ƒv□î•ñ,ª•\Ž¦,³,ê,Ü,·□B
   ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,Í□AŽÀ□s,³,ê,é Oracle ,ÌŠe‘€□ì,ÉŠÖ,·,é□î•ñ,¨,æ,Ñ,Ç,Ì‘€□ì,ª□¬Œ÷,µ,½,©,ð•\Ž¦,µ,Ü,·□B
7. Oracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ªƒGƒ‰□[,ð‹N,±,³,¸,É Oracle ƒf□[ƒ^ƒx□[ƒX,É□Ú‘±,Å,«,ê,Î□AMTS ,Í Oracle ,Æ,Æ,à,ÉŽÀ□s,Å,«,é,Æ□l,¦,ç,ê,Ü,·□BOracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ªƒGƒ‰□[,ð•ñ□□,µ,½□ê□‡,Í□AŽŸ,Ì,¢,¸,ê,©,Ì□ì‹Æ,ð□s,¢,Ü,·□B
   - Oracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ª•\Ž¦,µ,½ƒGƒ‰□[ƒ□ƒbƒZ□[ƒW,ð□³Šm,É‹L˜^,µ,Ü,·□B
   - Oracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ÌŽÀ□s’†,É¶□¬,³,ê,½ Oracle ƒgƒŒ□[ƒX ƒtƒ@ƒCƒ‹,ð’²□¸,µ,Ü,·□BOracle ƒgƒŒ□[ƒX□î•ñ,Í□A*.TRC ƒtƒ@ƒCƒ‹,É, ,è,Ü,·□BOracle ƒgƒŒ□[ƒX ƒtƒ@ƒCƒ‹,Í□A–â‘è,ð□f’f,·,é,Ì,É‘å•Ï–ð—§,Â□î•ñ,àŠÜ,ñ,Å,¢,Ü,·□B
   - Oracle ,ÌƒTƒ|□[ƒg•”–å,É–â,¢□‡,í,¹,Ü,·□B

Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽg,Á,Ä Oracle ,ÌƒCƒ“ƒXƒg□[ƒ‹,Æ□\□¬,ðŠm”F,·,é

Oracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ðŽg,Á,Ä Oracle ,ÌƒCƒ“ƒXƒg□[ƒ‹,Æ□\□¬,ðŠm”F,µ,½,ç□AMicrosoft Transaction Server ,Æ,Æ,à,É’ñ‹Ÿ,³,ê,é Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽg,Á,Ä□AMicrosoft Transaction Server ,ª Oracle ƒf□[ƒ^ƒx□[ƒX,ÉƒAƒNƒZƒX,Å,«,é,©,Ç,¤,©,ðŠm”F,·,é•K—v,ª, ,è,Ü,·□B

Sample Bank ,ðŽg,Á,Ä Oracle ƒTƒ|□[ƒg,ðŠm”F,·,é,É,Í

1. MTS ,Å’ñ‹Ÿ,³,ê,é Oracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ðŽg,Á,Ä□AOracle ƒVƒXƒefe€,ª□³,µ,ƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä□\

□¬,³,ê,Ä,¢,é,©,ðŠm”F,µ,Ü,·□BOracle ƒeƒXƒg ƒvƒ□ƒOƒ‰ƒ€,ª‰½,ç,©,ÌƒGƒ
‰□[,ð•ñ□□,µ,½□ê□‡,Í□AŽŸ,É□i,Þ‘O,É□C□³,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

2 Oracle ƒf□[ƒ^ƒx□[ƒX ƒT□[ƒo□[,Å "Account" ,Æ,¢,¤–¼‘O,Ìƒe□[ƒuƒ‹,ð□ì□¬,µ,Ü,·□BŽŸ,Ì—á,Í□AAccount ƒe□[ƒuƒ‹,Ì□Ý’è,ðŽ¦,µ,Ü,·□B

```
□Š—LŽÒ          scott
ƒe□[ƒuƒ‹–¼      Account
—ñ 1 ,Ì–¼‘O      AccountNo    (ƒ^ƒCƒv,Í NUMBER)
—ñ 2 ,Ì–¼‘O      Balance    (ƒ^ƒCƒv,Í NUMBER)
```

3 "Account" ƒe□[ƒuƒ‹,É□,È,,Æ,à 2 □s,ÌƒŒƒR□[ƒh,ð□ì□¬,µ,Ü,·□BŽŽŸ,Ì•\ ,Í□Aƒe□[ƒuƒ‹,É“Ç,Ý□ž,ÞƒŒƒR□[ƒh,ðŽ¦,µ,Ü,·□B

AccountNo	Balance
1	1000
2	1000

4 Oracle ƒf□[ƒ^ƒx□[ƒX ƒT□[ƒo□[,Å "Receipt" ,Æ,¢,¤–¼‘O,Ìƒe□[ƒuƒ‹,ð□ì□¬,µ,Ü,·□BŽŸ,Ì—á,Í□AReceipt ƒe□[ƒuƒ‹,Ì□Ý’è,ðŽ¦,µ,Ü,·□B

```
□Š—LŽÒ          scott
ƒe□[ƒuƒ‹–¼      Receipt
—ñ 1 ,Ì–¼‘O      NextReceipt      (ƒ^ƒCƒv,Í NUMBER)
```

5 "Receipt" ƒe□[ƒuƒ‹,É□,È,,Æ,à 1 □s,ÌƒŒƒR□[ƒh,ð□ì□¬,µ,Ü,·□BŽŽŸ,Ì•\ ,Í□Aƒe□[ƒuƒ‹,É“Ç,Ý□ž,ÞƒŒƒR□[ƒh,ðŽ¦,µ,Ü,·□B

NextReceipt
1000

6 ODBC □\□¬ƒ†□[ƒeƒBƒŠƒeƒB,ðŽg,Á,Ä□Aƒtƒ@ƒCƒ‹ DSN ,ð□ì□¬,µ,Ü,·□Bƒtƒ@ƒCƒ‹ DSN ,Ì–¼‘O,Í□A"MTSSamples" ,Å,·□BŽŸ,É□Aƒtƒ@ƒCƒ‹ DSN ,ðŽè□ì‹Æ,Å□X□V,µ,Ä□Aƒ†□[ƒU□[,ÌƒpƒXƒ□□[ƒh,ð’Ç ‰Á,µ,Ü,·□BŽŸ,Ì—á,Í□Aƒtƒ@ƒCƒ‹ DSN ,Éƒ†□[ƒU□[ƒpƒXƒ□□[ƒh,ð’Ç‰Á,·,é•û–@,ðŽ¦,µ,Ü,·□B

```
[ODBC]
DRIVER=Microsoft ODBC for Oracle
UID=scott
PWD=mypassword
ConnectString=myserver
SERVER=myserver
```

7 ƒtƒ@ƒCƒ‹ DSN ,ð•Û‘¶,µ□ASample Bank ƒNƒ‰ƒCƒAƒ“ƒg,ðŽÀ□s,µ,Ü,·□B

Oracle ,Æ Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒl□[ƒ^,ÌŠÇ—□

Oracle ,Ì□«—ˆ,ÌƒŠƒŠ□[ƒX,Å,Ì DLL –¼,Ì•Ï□X

Oracle ,Í□»•i,Ì□V,µ,¢ƒo□[ƒWƒ‡ƒ“,ªƒŠƒŠ□[ƒX,³,ê,é,Æ□ADLL –¼,ð•Ï□X,·,é,±,Æ,ª, ,è,Ü,·□BMTS ,Í□A,¢,,Â,©,Ì Oracle DLL ,ÌŠù’m,Ì–¼‘O,ðŽg—p,µ,Ü,·□BŒ»□Ý□AMTS ,Í Oracle 7.3 release ,Ì DLL –¼,ðŒŸ□õ,µ,Ü,·□BOracle ,ª,±,ê,ç,Ì DLL ,Ì–¼‘O,ð•Ï□X,µ,½□ê□‡,Í□AŽŸ,ÌƒŒƒWƒXƒgƒŠ ƒL□[,Ì’l,ð•Ï□X,·,é•K—v,ª, ,è,Ü,·□B

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Transaction Server\Local Computer\My Computer
```

ŽŸ,ÌƒL□[,Ì‰ºˆÊ,Í□A2 ,Â,Ì•¶Žš—ñ–¼’l,ª, ,è,Ü,·□B

- OracleXaLib"xa73.dll"
- OracleSqlLib "sqllib18.dll"

MS DTC ,ðŽÀ□s,·,éƒt□[ƒU□[ID ,Ì•Ï□X

Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒl□[ƒ^ (MS DTC) ,í□ANT ƒŒƒWƒXƒgƒŠ,Ì•ÛŒì,³,ê,½•”•ª,Ì□î•ñ,ð•ÛŽç,µ,Ü,·□B,±,ê,ç,Ì□î•ñ,Í□AOracle ,ª'ñ‹Ÿ,·,é XA ŒÝŠ·ƒf□[ƒ^ƒx□[ƒX,ð‰ñ•œ,·,é,Æ,«,ÉŽg,í,ê,Ü,·□BMS DTC ,ðŽÀ□s,·,éƒt□[ƒU□[ID ,ð•Ï□X,µ,½□ê□‡□AƒŒƒWƒXƒgƒŠ,Ì•ÛŒì,³,ê,½•”•ª,É MS DTC ,ªŠi"[,µ,½□î•ñ,ð□AMS DTC ,ªƒAƒNƒZƒX,Å,«,é,±,Æ,ð'□^Ó,µ,ÄŠm"F,µ,Ä,,¾,³,¢□B

MS DTC ,Ìƒt□[ƒU□[ID ,ð•Ï□X,µ□A•ÛŒì,³,ê,½ƒŒƒWƒXƒgƒŠ,Ì□î•ñ,ÉƒAƒNƒZƒX,Å,«,È,-,È,Á,½□ê□‡□AŽŸ,Ìƒ□ƒbƒZ□[ƒW,ª Windows NT ƒCƒxƒ"ƒg ƒ□ƒO,É‹L˜^,³,ê,Ü,·□B

XATM log object failed to set log encryption key

,±,Ì–â'è,Í□AŽŸ,ÌŽè□‡,Å‰ðŒˆ,Å,«,Ü,·□B

1. MS DTC ,ð'âŽ~,µ,Ü,·□B
2. MS DTC ,Ìƒt□[ƒU□[ID ,ð'O,Ì'l,É–ß,µ,Ü,·□B'ã,í,è,É□AŠÇ—□ŽÒƒOƒ‹□[ƒv,Ìƒ□ƒ"ƒo□[,Ìƒt□[ƒU□[ID ,ðŠ„,è"–,Ä,é,±,Æ,à,Å,«,Ü,·□Bƒt□[ƒU□[ID ,¨,æ,ÑƒpƒXƒ□□[ƒh,ð•Ï□X,·,é,É,Í□AƒRƒ"ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Ì [ƒT□[ƒrƒX] ,ðŽÀ□s,µ,Ä□Aˆê——,Ì [MSDTC] ,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ,Ü,·□B[ƒ□ƒOƒIƒ"] ,Ì [ƒAƒJƒEƒ"ƒg] ,Ì'l,ð•Ï□X,µ,Ü,·□B
3. MS DTC ,ð□ÄŠJ,µ,Ü,·□B

'½,,Ì□Ú'±,ðƒTƒ|□[ƒg,·,é,æ,¤,É Oracle ,ð□\□¬,·,é

Oracle ƒf□[ƒ^ƒx□[ƒX,É'Î,·,é□Ú'±,ð□"□\^È□ã□ì□¬,·,é,É,Í□A,³,ç,É'½,,Ìƒf□[ƒ^ƒx□[ƒX□Ú'±,ðƒTƒ|□[ƒg,·,é,æ,¤,É Oracle ƒT□[ƒo□[,ð□\□¬,·,é•K—v,ª, ,è,Ü,·□B

,±,ê,ð□s,í,È,¢,Æ□AŽŸ,Ì,¢,¸,ê,©,ÌƒGƒ‰□[,ª"□¶,·,é‰Â"\□«,ª, ,è,Ü,·□B

- SQLConnect ŒÄ,Ñ□o,µ,ÌŽ¸"s
- ƒlƒufƒWƒFƒNƒg ƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ",ÌŒÄ,Ñ□o,µ,ÅƒGƒ"ƒŠƒXƒg,ÉŽ¸"s,·,é□B,±,Ì–â'è,ª"□¶,µ,½□ê□‡□AOracle ƒgƒŒ□[ƒX ƒtƒ@ƒCƒ‹,ÉŽŸ,Ì,¢,¸,ê,©,ÌƒGƒ‰□[,ÌŒ‹‰Ê,ª‹L˜^,³,ê,Ä,¢,é‰Â"\□«,ª, ,è,Ü,·□B
  - ƒZƒbƒVƒ‡ƒ",ª'½,·,¬,é
  - TNS ƒT□[ƒo□[,ªƒT□[ƒo□[–¼,ÌŒŸ□õ,ÉŽ¸"s,µ,½
  - •ªŽUƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ",ª'½,·,¬,é
- ƒf□[ƒ^ƒx□[ƒX ƒ□ƒbƒN,ð'Ò,Á,Ä,¢,éŠÔ,Éƒ^ƒCƒ€ƒAƒEƒg,É,È,è,Ü,µ,½□B,± ,ê,Í□A□Ý'è,³,ê,½ƒ□ƒbƒN□",ª□A"¯ŽžŽÀ□s,³,ê,éƒAƒNƒeƒBƒu ƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ"□",æ,è•s'«,µ,Ä,¢,é,± ,Æ,ª□I,¦,ç,ê,Ü,·□B
- ƒCƒ" ƒ_ƒEƒg ƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ",ª‹N,±,µ,½ƒ□ƒbƒN,É,æ,Á,Ä□AƒŒƒR□[ƒh,ª□Õ"Ë,µ,Ü,µ,½□B

,±,ê,ç,Ì–â'è,ª"□¶,µ,½□ê□‡,É,Í□AŽŸ,Ì Oracle ƒT□[ƒo□[□\□¬ƒpƒ‰ƒ□□[ƒ^,Ì'l,ð'□,â,µ,Ü,·□B

Oracle □\□¬ƒpƒ‰ƒ□□[ƒ^

sessions

distributed_lock_timeout

distributed_transactions

dml_locks

max_transaction_branches

open_cursors

processes

queuesize

Sessions

sessions ,Ì'l,Í□A'Ê□í□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ",ª□ì□¬,·,é,Æ—\'z,³,ê,éƒf□[ƒ^ƒx□[ƒX□Ú'±,Ì'□□",Ì 3

”{,ì□”,Å,È,¯,ê,Î,È,è,Ü,¹,ñ□B

Queuesize

Oracle Listener ƒvƒ□ƒZƒX,É‘Î,·,éƒLƒ…□[ƒTƒCƒY,ª“K□Ø,Å,È,¢□ê□‡□AListener ,Í□^—□,ªŠÔ,É□‡,í,È,-
,È,èƒf□[ƒ^ƒx□[ƒX,ðŠJ,—v‹□,ð‹'”Û,·,é,±,Æ,ª, ,è,Ü,·□B,±,ê,Í□AOracle Listener ,ª‘½,·,¬,é□Ú‘±—
v‹□,ðŽó,¯Žæ,è□AƒLƒ…□[,ªƒl□[ƒo□[ƒtƒ□□[,·,é□ê□‡,É□A”□¶,µ,Ü,·□B,±,ìƒGƒ‰□[,ðŒŸ□o,µ,½ƒNƒ
‰ƒCƒAƒ“ƒg,Í□Aƒ□ƒbƒZ□[ƒW "ORA-12541: No Listener" ,ð•Ô,µ□AƒNƒ‰ƒCƒAƒ“ƒg ƒ□ƒO,Ü,½,ÍƒgƒŒ□[ƒX
ƒtƒ@ƒCƒ‹,ª EDONREFSED ƒ□ƒbƒZ□[ƒW,ð•\Ž¦,µ,Ü,·□B

□Ú‘±—v‹□,Ì–â‘è,ð‰ðŒˆ,·,é,É,Í

1. Oracle ƒf□[ƒ^ƒx□[ƒX ƒT□[ƒo□[,ðŠÜ,ÞƒVƒXƒeƒ€,Ì Oracle Listener ,ð’âŽ~,µ,Ü,·□B
2. Oracle ƒf□[ƒ^ƒx□[ƒX ƒT□[ƒo□[ƒVƒXƒeƒ€□ã,Ì LISTENER.ORA□ATNSNET.ORA□A,Ü,½,Í NAMES.ORA
   ƒtƒ@ƒCƒ‹,Ì QUEUESIZE ƒpƒ‰ƒ□□[ƒ^,Ì'l,ð‘□,â,µ,Ü,·□BŽQ‰Á,·,é□Ú‘±—
   v‹□,Æ“¯,¶□A,Ü,½,Í‹ß,¢'l,ðŠî,ÉƒLƒ…□[ƒTƒCƒY,ð‘I'ð,µ,Ü,·□B
   QUESIZE ,É,Í'å,«,ß,Ì'l,ð□Ý'è,·,é,±,Æ,ð,¨Š©,ß,µ,Ü,·□B,±,ê,Í□AOracle ƒVƒXƒeƒ€
   ,ª□A’Ê□í□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ªŠJ,Šeƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒf□[ƒ^ƒx□[ƒX□Ú‘±,É‘Î,µ,Ä□A2
   ,Âˆ□È□ã,Ìƒf□[ƒ^ƒx□[ƒX□Ú‘±,ðŠJ,,©,ç,Å,·□BŒ‹‰Ê,Æ,µ,Ä□AOracle Listener ,ìƒLƒ…□[,Í□A,¢,Á,Ï,¢
   ,É,È,èƒl□[ƒo□[ƒtƒ□□[,µ,Ü,·□B
   ,½,Æ,¦,Î□A100 ,Ì□Ú‘±—v‹□,É‘Î‰ž,Å,«,é,æ,¤,É,·,é,É,Í□ALISTENER.ORA ƒtƒ@ƒCƒ‹,ðŽŸ,Ì,æ,¤
   ,É•Ï□X,µ,Ü,·□B

   ```
   QUEUESIZE = 100
   ```

3. Oracle Listener ,ð□ÄŠJ,µ,Ü,·□B

MTS ,Ì Oracle ƒTƒ|□[ƒg,ÉŠÖ,·,éŠù'm,Ì□§ŒÀŽ–□€

ODBC 3.5 ,ðŽg—p,·,é,É,Í ADO 1.5 ,ª•K—v

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª ADO ,ðŽg—p,·,é□ê□‡□A•K,¸ ADO 1.5 ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ä,,¾,³,¢□B□Ú□×,É,Â,¢
,Ä,Í□A□u•K—v,È‚ƒtƒgƒEƒFƒA□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

Oracle ,Í DEC ŽÐ,Ì Alpha ƒvƒ‰ƒbƒgƒtƒH□[ƒ€,ðƒTƒ|□[ƒg,µ,È,¢

Oracle ƒf□[ƒ^ƒx□[ƒX□Ú‘±,Í□AMicrosoft Transaction Server ,ðŽÀ□s,·,é Alpha ƒvƒ‰ƒbƒgƒtƒH□[ƒ€,Å,ÍƒTƒ|
□[ƒg,³,ê,Ü,¹,ñ□BDEC ŽÐ,Ì Alpha ƒvƒ‰ƒbƒgƒtƒH□[ƒ€,Å,Ì Oracle ƒTƒ|□[ƒg,Í□A□«—
ˆ,ÌƒšƒŠ□[ƒX,ÅŽÀŒ»,³,ê,Ü,·□B

Oracle OCIW32.DLL ,Ìƒo□[ƒWƒ‡ƒ“,Ì–â‘è

OCIW32.DLL ,Ì□³,µ,¢ƒo□[ƒWƒ‡ƒ“,ðƒRƒ“ƒsƒ…□[ƒ^,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é,±,Æ,ª□A□d—v,Å,·□BOracle ,Ü,½,Í
Microsoft Transaction Server ,ð□ÄƒCƒ“ƒXƒg□[ƒ‹,·,é□ê□‡,Í□A•K,¸ DLL ,Ìƒo□[ƒWƒ‡ƒ“,ðŠm”F,µ,Ä,,¾,³,¢□B

Oracle ,Ì□«—ˆ,ìƒšƒš□[ƒX,Å,Ì DLL –¼,Ì•Ï□X

Oracle ,Í□»•i,Ì□V,µ,¢ƒo□[ƒWƒ‡ƒ“,ªƒšƒš□[ƒX,³,ê,é,Æ□ADLL –¼,ð•Ï□X,·,é,±,Æ,ª, ,è,Ü,·□BMTS ,Í□A,¢,,Â,©,Ì
Oracle DLL ,ÌŠù'm,Ì–¼'O,ðŽg—p,µ,Ü,·□BŒ»□Ý□AMTS ,Í Oracle version 7.3.3 ,Ì DLL –
¼,ðŒŸ□õ,µ,Ü,·□BMTS ,Í,±,ê,ç,Ì DLL ,Ì□«—ˆ,Ì–¼'O,ð—\'z,Å,«,È,¢,Ì,Å□AOracle
,ðƒAƒbƒvƒOƒŒ□[ƒh,µ,½,Æ,«,É,ÍŽŸ,ÌƒŒƒWƒXƒgƒš,Ì'l,ð•Ï□X,µ,È,¯,ê,Î,È,ç,È,¢□ê□‡,ª, ,è,Ü,·□B

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Transaction Server\Local Computer\My Computer
```

ŽŸ,ÌƒL□[,Ì‰º,É,Í□A2 ,Â,Ì•¶Žš—ñ–¼'l,ª, ,è,Ü,·

- OracleXaLib"xa73.dll"
- OracleSqlLib "sqllib18.dll"

Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒZƒbƒgƒAƒbƒv

Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í□A“ü‹à□A□o‹à□A,¨,æ,Ñ□U,è‘Ö,¦,ð□s,¤‹â□sƒT□[ƒrƒX‹Æ–±ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Å,·□BSample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽÀ□s,·,é,Æ□AMTS ,Æ SQL Server 6.5 ,ð‘g,Ý□‡,í,¹,½ƒCƒ“ƒXƒg□[ƒ‹,ÌŒ‹‰Ê□A,¨,æ,Ñ—û□K—pƒpƒbƒP□[ƒW,Ì”z’u,ÆŠÇ—□,ðƒefXƒg,·,é,±,Æ,ª,Å,«,Ü,·□BSample Bank ,É,Í□AVisual Basic□AVisual C++□A,¨,æ,Ñ Visual J++ ,Å‹L□q,³,ê,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ü,·□BSample Bank ,Í \Program Files\Mts\Samples ƒfƒBƒŒƒNƒgƒŠ,É, ,è,Ü,·□B

□wProgrammer's Guide□x,É,Í□ASample Bank ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□ì□¬,·,é•û–@,ð□Ú,µ,□à–¾,·,éƒ`ƒ…□[ƒgƒŠƒAƒ‹,ª‹L□q,³,ê,Ä,¢,Ü,·□B

Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽÀ□s,·,é,É,Í□AŽŸ,Ì□ì‹Æ,ð,ð□s,í,È,¯,ê,Î,È,è,Ü,¹,ñ□B

- ƒZƒbƒgƒAƒbƒv,Ì [ƒJƒXƒ^ƒ€] ƒIƒvƒVƒ‡ƒ“,ð‘I‘ð,µ□A,·,×,Ä,Ì MTS ƒTƒuƒRƒ“ƒ|□[ƒlƒ“ƒg,ð‘I‘ð,µ,Ü,·□B□Ú□×,É,Â,¢,Ä,Í□A□uŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒ□ƒ“ƒg,ÌƒCƒ“ƒXƒg□[ƒ‹□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B
- DSN ,ð□Ý’è,·,é□B
- Sample Bank ƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,·,é□B
- MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å Sample Bank ƒpƒbƒP□[ƒW,ðŠÄŽ‹,Å,«,é,æ,¤,É□Ý’è,·,é□B
- Bank ƒNƒ‰ƒCƒAƒ“ƒg,ðŽÀ□s,·,é□B

Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□Ý’è,·,é

MTS ,ÍƒZƒbƒgƒAƒbƒv’†,ÉŽ©“®“I,É Sample Bank ,Ì ODBC ƒf□[ƒ^ ƒ\□[ƒX,ð□\□¬,µ,Ü,·□BŠù’è,Ì□Ý’è,ÅƒÅƒ□□[ƒJƒ‹ƒ}ƒVƒ“,ªŽg—p,³,ê,é,Ì,Å□AƒÅƒ□□[ƒJƒ‹ ƒ}ƒVƒ“,É SQL Server 6.5 ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ä,¨,©,È,¯,ê,Î,È,è,Ü,¹,ñ□B

Šù’è,Ì□Ý’è,Å,Í□AMTS DSN ,Í SQL Server 6.5 ,ðŽw,µ,Ü,·□BSQL Server 6.5 ˆÈŠO,Ìƒf□[ƒ^ƒx□[ƒX,ðŽg—p,µ,Ä,¢,é□ê□‡,Í□ADSN ,ð□í□œ,µ□AŽg—p,µ,Ä,¢,éƒf□[ƒ^ƒx□[ƒX,ðŽw,· MTSSamples ,Æ,¢,¤□V,µ,¢ DSN ,ð’Ç‰Á,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

,Ù,©,ÌƒRƒ“ƒsƒ…□[ƒ^,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,é SQL Server ,ðŽg,¤□ê□‡,Í□AƒRƒ“ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Ì [ODBC] ƒAƒCƒRƒ“,ðŽg,Á,Äƒf□[ƒ^ ƒ\□[ƒX,ðŽŸ,ÌŽè□‡,Å•Ï□X,µ,Ü,·□B

- [ODBC ƒf□[ƒ^ ƒ\□[ƒX ƒAƒhƒ~ƒjƒXƒgƒŒ□[ƒ^] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å [ƒtƒ@ƒCƒ‹ DSN] ƒ^ƒu,ðƒNƒŠƒbƒN,µ□AMTSSamples ƒf□[ƒ^ ƒ\□[ƒX,ð‘I‘ð,µ,Ü,·□B
- [□\□¬] ,ðƒNƒŠƒbƒN,µ□AŽg—p,·,éƒT□[ƒo□[,Ì–¼‘O,ð“ü—Í,µ,Ü,·□B

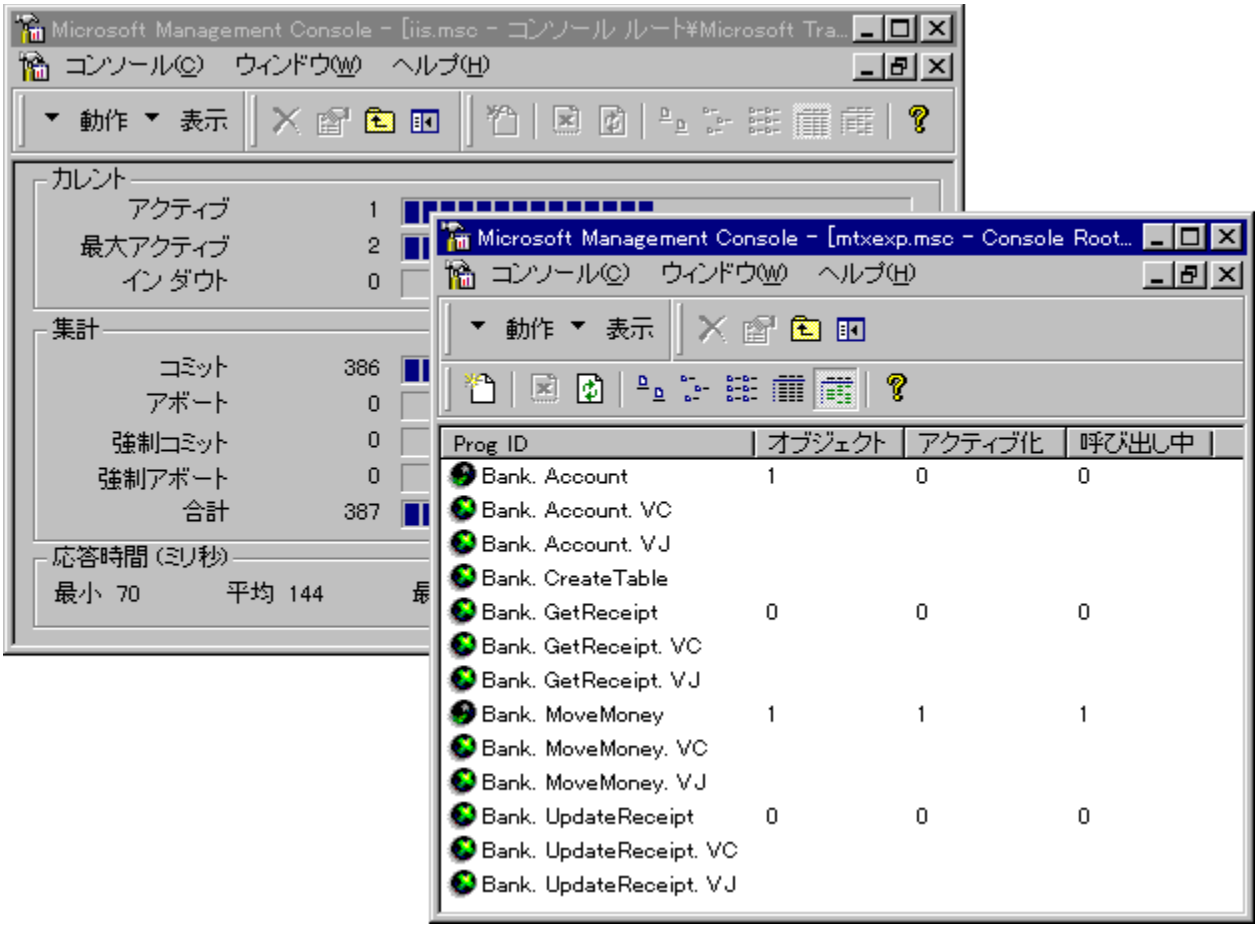
MTSSamples.dsn ,ÅŽw’è,³,ê,Ä,¢,éƒ□ƒOƒCƒ“ ID ,ÆƒpƒXƒ□□[ƒh,Í□ASample Bank ,Å,ÍŽg,í,ê,È,¢,±,Æ,É’□ˆÓ,µ,Ä,,¾,³,¢□BSample Bank ,Å,Í "sa" ,Æ,¢,¤ƒAƒJƒEƒ“ƒg,Æ null ƒpƒXƒ□□[ƒh,ªŽg—p,³,ê,Ü,·□BƒVƒXƒefŠÇ—□ŽÒ,ÌƒpƒXƒ□□[ƒh,ª null ,Å,È,¢□ê□‡□A,Ü,½,Í•Ê,Ìƒ□ƒOƒCƒ“ ID ,ðŽw’è,·,é□ê□‡,Í□ASample Bank ,Ìƒ\□[ƒX ƒR□[ƒh,Ì ODBC □Ú‘±•¶Žš—ñ,ð•Ï□X,·,é•K—v,ª, ,è,Ü,·□B

Sample Bank ƒpƒbƒP□[ƒW,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Æƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŠÄŽ‹,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì‰E‘¤,Å□A[Sample Bank] ƒAƒCƒRƒ“,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ,Ü,·□B
2. [ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ,Ü,·□B
3. [•\Ž¦] ƒ□ƒjƒ…□[,Ì [□ó‘Ô] ,ðƒNƒŠƒbƒN,µ,ÄƒpƒbƒP□[ƒW,Ì,³,Ü,´,Ü,ÈƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŽg—p□ó‹µ,ÉŠÖ,·,é□î•ñ,ð•\Ž¦,µ,Ü,·□B
4. [ƒEƒBƒ“ƒhƒE] ƒ□ƒjƒ…□[,Ì [□V,µ,¢ƒEƒBƒ“ƒhƒE,ðŠJ,] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
5. 2 ,Â,ÌƒEƒBƒ“ƒhƒE,ª□d,È,ç,È,¢,æ,¤,É□A□V,µ,¢ƒEƒBƒ“ƒhƒE,Ìˆê’u,ð’²□®,µ,Ü,·□B[ƒEƒBƒ“ƒhƒE] ƒ□ƒjƒ…□[,Ì [□d,Ë,Ä•\Ž¦] ,Ü,½,Í [□ã‰º,É•À,×,Ä•\Ž¦] ,ðƒNƒŠƒbƒN,µ,Ä□AƒEƒBƒ“ƒhƒE,ð□®—ñ,·,é,±,Æ,à,Å,«,Ü,·□B
6. □V,µ,¢ƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Å□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
7. [“®□ì] ƒ□ƒjƒ…□[,Ì [ƒXƒR□[ƒv ƒEƒBƒ“ƒhƒE] ,ðƒNƒŠƒbƒN,µ,ÄƒEƒBƒ“ƒhƒE,Ì□¶‘¤,ð”ñ•\Ž¦,É,µ,Ü,·□BƒEƒBƒ“ƒhƒE,Ì□¶‘¤,ª•\Ž¦,³,ê,Ä,¢,é,Æ,«,Í□A,±,ÌƒRƒ}ƒ“ƒh,Éƒ`ƒFƒbƒN ƒ}□[ƒN,ª•t,¢,Ä,¢,Ü,·□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŽg—p,³,ê,é,Æ□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ““□Œv□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

Bank ƒNƒ‰ƒCƒAƒ“ƒg,ðŽÀ□s,·,é,É,Í

1. Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒl□[ƒ^ □iMS DTC) ,ª“®□ì,µ,Ä,¢,é,± ,Æ,ðŠm”F,µ,Ü,·□BTransaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Å [ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ,ð‘I‘ð,µ,Ü,·□B[“®□ì] ƒ□ƒjƒ…□[,Ì [MS DTC ŠJŽn] ,ª•\Ž¦,³,ê,Ä,¢,ê,Î□A,±,ÌƒRƒ}ƒ“ƒh,ðƒNƒŠƒbƒN,µ,Ü,·□B
2. SQL Server ,ª“®□ì,µ,Ä,¢,é,±,Æ,ðŠm”F,µ,Ü,·□BSQL Server ,ÍƒRƒ“ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Å‹N“®,Å,«,Ü,·□B
3. [ƒXƒ^□[ƒg] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒOƒ‰ƒ€] ,ðƒ|ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[Windows NT 4.0 Option Pack] ,ðƒ|ƒCƒ“ƒg,µ□A[Microsoft Transaction Server] ,ðƒ|ƒCƒ“ƒg,µ,Ä [Bank ƒNƒ‰ƒCƒAƒ“ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□BBank ƒNƒ‰ƒCƒAƒ“ƒg,ÌƒEƒBƒ“ƒhƒE,ª MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Æ□d,È,ç,È,¢ ,æ,¤,É□ABank ƒNƒ‰ƒCƒAƒ“ƒg,ÌƒEƒBƒ“ƒhƒE,Ìˆê'u,ð'²□®,µ,Ü,·□B
4. ƒtƒH□[ƒ€,Í□AŒû□À”Ô□† (Account Number) 1 ,Ö 1 ƒhƒ‹“ü‹à (Credit) ,·,é,æ,¤,É, ,ç,©,¶,ß□Ý'è,³,ê,Ä,¢ ,Ü,·□B[Submit] ,ðƒNƒŠƒbƒN,µ,Ü,·□BŽc□,,ª•Ï,í,é,±,Æ,ªŠm”F,Å,«,Ü,·□B
5. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì•\Ž¦,É'□ˆÓ,µ,Ä,,¾,³,¢□BƒRƒ“ƒ|[ƒlƒ“ƒgŽg—p□ó‹µ,Æƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ““□Œv□î•ñ,Ì•\Ž¦,ª□X□V,³,ê,Ä,¢,é,±,Æ,ª,í,©,è,Ü,·□B
6. ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŽí—Þ□AƒT□[ƒo□[□A,¨,æ,ÑŒJ,è•Ô,µ (Iterations) ,Ì,³,Ü,´,Ü,È□Ý'è,ðŽg,Á,Ä Bank ƒNƒ‰ƒCƒAƒ“ƒg,ðŽÀ□s,µ□A“□Œv□î•ñ,ðŠm”F,µ,Ä,,¾,³,¢□BŽŸ,Ì—□—R,É,æ,è□A□Å□‰,Ìƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Å,Í 2 ‰ñ–Ú^È□~,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,æ,èŽžŠÔ,ª,©,©,è,Ü,·□B
   - □Å□‰,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Å,Í Sample Bank ,Ìƒf□[ƒ^ƒx□[ƒX fe□[ƒuƒ‹,ª□ì□¬,³,ê□Aˆê□ŽžŽ“I,ÈƒŒƒR□[ƒh,ª'}“ü,³,ê,é□B
   - ƒT□[ƒo□[ƒvƒ□ƒZƒX,ÌŠJŽn,Í□AƒVƒXƒeƒ€ ƒŠƒ\□[ƒX,ð□Á”ï,·,é□B
   - □Å□‰,Ìƒf□[ƒ^ƒx□[ƒX□Ú'±,ÌŠJŽn,Í□AƒT□[ƒo□[,É,Æ,Á,Ä•‰‰×,ª□d,¢'€□ì,Å, ,é□B



Tic-Tac-Toe ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ÌƒZƒbƒgƒAƒbƒv

Tic-Tac-Toe ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚Í□AƒRƒ“ƒsƒ…□[ƒ^‚Ü‚½‚ÍŠƒ‚□[ƒg‚Ì MTS ƒRƒ“ƒsƒ…□[ƒ^□ã‚Ì‚Ù‚©‚Ìƒt□[ƒU□[‚Æ‘Î□í‚·‚éƒQ□[ƒ€‚Å‚·□BTic-Tac-Toe ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ðŽÀ□s‚·‚é‚ÆASQL Server ‚ðƒCƒ“ƒXƒg□[ƒ‹‚¹‚¸‚É MTS ‚ðƒCƒ“ƒXƒg□[ƒ‹‚µ‚½Œ‹‰Ê□A‚¨‚æ‚Ñ—û□K—pƒpƒbƒP□[ƒW‚Ì”z’u‚ÆŠÇ—□‚ðƒeƒXƒg‚·‚é‚±‚Æ‚ª‚Å‚«‚Ü‚·□BTic-Tac-Toe ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚Í \Program Files\Mts\Samples ƒfƒBƒŒƒNƒgƒŠ‚É‚ ‚è‚Ü‚·□B

Tic-Tac-Toe ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ðŽÀ□s‚·‚é‚É□AŽŸ‚Ì□ì‹Æ‚ð□s‚¤•K—v‚ª ‚è‚Ü‚·□B

- Microsoft Transaction Server ‚ðƒCƒ“ƒXƒg□[ƒ‹‚·‚é□B
- Tic-Tac-Toe ƒNƒ‰ƒCƒAƒ“ƒg‚ðŽÀ□s‚µ□AƒRƒ“ƒsƒ…□[ƒ^‚Ü‚½‚Í‚Ù‚©‚Ìƒt□[ƒU□[‚Æ‘Î□í‚·‚é□B

Tic-Tac-Toe ƒNƒ‰ƒCƒAƒ“ƒg‚ð‹N“®‚·‚é‚É

1. [ƒXƒ^□[ƒg] ƒ{ƒ^ƒ“‚ðƒNƒŠƒbƒN‚µ□A[ƒvƒ□ƒOƒ‰ƒ€] ‚ðƒ|ƒCƒ“ƒg‚µ‚Ü‚·□BŽŸ‚É□A[Microsoft Transaction Server] ‚ðƒ|ƒCƒ“ƒg‚µ□A[Tic-Tac-Toe ƒNƒ‰ƒCƒAƒ“ƒg] ‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□B
2. [Your Name is] ƒ{ƒbƒNƒX‚ÉŽ©•ª‚Ì–¼‘O‚ð“ü—Í‚µ□[ƒ^ (Deep Viper) ‚Ü‚½‚Í‚Ù‚©‚Ìƒt□[ƒU□[‚ð‘Î□í‚ŠŽè‚Æ‚µ‚Ä‚I‚Ñ‚Ü‚·□BƒŠƒ‚□[ƒg‚Ìƒt□[ƒU□[‚Æ‘Î□í‚·‚é•û–@‚É‚Â‚¢‚Ä‚Í□A□uƒŠƒ‚□[ƒg MTS ƒRƒ“ƒsƒ…□[ƒ^‚ðŽg‚¤□v‚ðŽQ□Æ‚µ‚Ä‚,,¾‚³‚¢‚□B

ƒQ□[ƒ€‚ðŠJŽn‚µ‚½‚ç□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰‚É–ß‚è‚Ü‚·□BTic-Tac-Toe ƒT□[ƒo□[ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ÌƒAƒCƒRƒ“‚ª‰ñ“]‚µ‚Ä‚¢‚é‚±‚Æ‚É’ˆÓ‚µ‚Ä‚,,¾‚³‚¢□BƒAƒCƒRƒ“‚Ì‰ñ“]‚Í□ATic-Tac-Toe ƒT□[ƒo□[ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ªƒAƒNƒeƒBƒu‰»‚³‚ê‚Ä‚¢‚Ä□AMTS ‚ª□³‚µ‚ƒCƒ“ƒXƒg□[ƒ‹‚ê‚Ä‚¢‚é‚±‚Æ‚ðŽ¦‚µ‚Ä‚¢‚Ü‚·□BƒQ□[ƒ€‚ðŽ~‚·‚é‚ÆATic-Tac-Toe ƒNƒ‰ƒCƒAƒ“ƒg‚ª Tic-Tac-Toe ƒT□[ƒo□[ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ðŽg—p‚µ‚È‚,‚È‚é‚Å‚Å□AƒAƒCƒRƒ“‚Ì‰ñ“]‚ªŽ~‚Ü‚è‚Ü‚·□B

[•\Ž¦] ƒ□ƒjƒ…□[‚Ì [□ó‘Ô] ‚ðƒNƒŠƒbƒN‚·‚é‚ÆATic-Tac-Toe ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ÌŽg—p□ó‹µ‚ðŠm”F‚Å‚«‚Ü‚·□B

ŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ÌƒZƒbƒgƒAƒbƒv

ŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,Å,Í□AVisual Basic Scripting Edition (VBScript) ,È,Ç,ÌƒI□[ƒgƒ□□[ƒVƒ‡ƒ“ŒÝŠ·,ÌŒ¾Œê,ðŽg,Á,Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì□^—□,ðŽ©“®‰»,·,éƒXƒNƒŠƒvƒg,ð‹L□q,·,é•û–@,ðŽ¦,µ,Ü,·□BƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,Í Sample Bank ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì”z’u,ðŽ©“®‰»,µ,Ü,·□B

ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ðŽÀ□s,·,é,É,Í□A,Ü,¸□AWindows Scripting Host (WSH) ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B,±,ê,ç,ÌƒXƒNƒŠƒvƒg,Í□AMicrosoft Windows NT ƒvƒ‰ƒbƒgƒtƒH□[ƒ€ ,¨,æ,Ñ Alpha ƒvƒ‰ƒbƒgƒtƒH□[ƒ€□ã,Å,¾,¯“®□ì,·,é,±,Æ,É’□^Ó,µ,Ä,,¾,³,¢□B

Windows NT 4.0 Option Pack ,ðŽg,Á,Ä Windows Scripting Host ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,É,Í

1. WSH ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,È,¢□ê□‡,Í□A[ƒXƒ^□[ƒg] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒOƒ‰ƒ€] ,ðƒ|ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[Microsoft ƒCƒ“ƒ^□[ƒlƒbƒg ƒT□[ƒo□[(‹¤’Ê)] ,ðƒ|ƒCƒ“ƒg,µ□A[Internet Information Server ƒZƒbƒgƒAƒbƒv] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
2. ƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€,Ì ’Ç‰Á/□í□œ] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
3. ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìˆê——,Ì’†,©,ç [Windows Scripting Host] ,ð’T,µ□Aƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒ“,É,µ,Ü,·□B
4. [Š®—¹] ,ðƒNƒŠƒbƒN,µ,Ü,·□BWindows NT 4.0 Option Pack ƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€,É,æ,Á,ÄƒRƒ“ƒsƒ…□[ƒ^,É WSH ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ü,·□B

WSH ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,é,Æ□AŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ðŽg—p,Å,«,é,æ,¤ ,É,È,è,Ü,·□BŽŸ,ÌƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ª \Program Files\Mts\Samples\wsh ƒfƒBƒŒƒNƒgƒŠ,É, ,è,Ü,·□B

- InstDLL.vbs

  ,±,ÌƒXƒNƒŠƒvƒg,Í□ASample Bank ,ÌŠù‘¶,Ìƒo□[ƒWƒ‡ƒ“,ð□í□œ,µ□ASample Bank ,Æ,¢,¤–¼‘O,Ì□V,µ,¢ƒpƒbƒP□[ƒW,ð□ì□¬,µ,Ü,·□BŽŸ,É□ASample Bank ,Ì Visual Basic DLL□AVisual C++ DLL□A,¨,æ,Ñ Visual J++ DLL ,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□V,µ,¢ƒpƒbƒP□[ƒW,ÉƒCƒ“ƒXƒg□[ƒ‹,µ□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘®□«,ð•Ï□X,µ,Ü,·□BŽŸ,É□A□V,µ,¢ƒ□□[ƒ‹,ð’Ç‰Á,µ,Ü,·□BƒXƒNƒŠƒvƒg,ðŽÀ□s,·,é‘O,É□Aƒtƒ@ƒCƒ‹,ìƒpƒX,ð•Ï□X,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

  ,±,ÌƒXƒNƒŠƒvƒg,Å,Í□AƒRƒ“ƒsƒ…□[ƒ^,ìŒƒWƒXƒgƒŠ,É,·,×,Ä,ìƒRƒ“ƒ|□[ƒlƒ“ƒg,ìƒvƒ□ƒOƒ‰ƒ€ ID (prgID) ,ªŠÜ,Ü,ê,Ä,¢,é•K—v,ª, ,é,±,Æ,É’□^Ó,µ,Ä,,¾,³,¢□B,µ,½,ª,Á,Ä□A,±,ÌƒXƒNƒŠƒvƒg,ðŽÀ□s,·,é‘O,É□AVisual Studio™ 97 ActiveX™ ƒEƒBƒU□[ƒh,ðŽg,Á,Ä Java ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð“o˜^,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

- InstPak.vbs

  ,±,ÌƒXƒNƒŠƒvƒg,Í□ASample Bank ƒpƒbƒP□[ƒW,ð MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÉƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□BƒXƒNƒŠƒvƒg,ðŽÀ□s,·,é‘O,É□Aƒtƒ@ƒCƒ‹,ìƒpƒX,ð•Ï□X,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

- Uninst.vbs

  ,±,ÌƒXƒNƒŠƒvƒg,Í□ASample Bank ƒpƒbƒP□[ƒW,ð MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,©,ç□í□œ,µ,Ü,·□BƒXƒNƒŠƒvƒg,ðŽÀ□s,·,é‘O,É□Aƒtƒ@ƒCƒ‹,ìƒpƒX,ð•Ï□X,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

- InstDllCLI.vbs

  ,±,ÌƒXƒNƒŠƒvƒg,ðŽg,¤,Æ□AInstDll.vbs ƒXƒNƒŠƒvƒg,ðƒRƒ}ƒ“ƒh ƒvƒ□ƒ“ƒvƒg,©,çŽÀ□s,µ□Aƒtƒ@ƒCƒ‹ ƒpƒX,ðƒpƒ‰ƒ□□[ƒ^,Æ,µ,Ä“ü—Í,·,é,±,Æ,ª,Å,«,Ü,·□BInstDll.vbs ƒXƒNƒŠƒvƒg,Í Sample Bank ,ÌŠù‘¶,Ìƒo□[ƒWƒ‡ƒ“,ð□í□œ,µ□ASample Bank ,Æ,¢,¤–¼‘O,Ì□V,µ,¢ƒpƒbƒP□[ƒW,ð□ì□¬,µ,Ü,·□BŽŸ,É□ASample Bank ,Ì Visual Basic DLL□AVisual C++ DLL□A,¨,æ,Ñ Visual J++ DLL ,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□V,µ,¢ƒpƒbƒP□[ƒW,ÉƒCƒ“ƒXƒg□[ƒ‹,µ□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘®□«,ð•Ï□X,µ,Ü,·□BŽŸ,É□A□V,µ,¢ƒ□□[ƒ‹,ð’Ç‰Á,µ,Ü,·□B

- InstPakCLI.vbs

  ,±,ÌƒXƒNƒŠƒvƒg,ðŽg,¤,Æ□AInstPak.vbs ƒXƒNƒŠƒvƒg,ðƒRƒ}ƒ“ƒh ƒvƒ□ƒ“ƒvƒg,©,çŽÀ□s,µ□Aƒtƒ@ƒCƒ‹ ƒpƒX,ðƒpƒ‰ƒ□□[ƒ^,Æ,µ,Ä“ü—Í,·,é,±,Æ,ª,Å,«,Ü,·□BInstPak.vbs ƒXƒNƒŠƒvƒg,Í Sample Bank

ƒpƒbƒP□[ƒW,ð MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÉƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B

ŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ÆƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,Ì□Ú□×,É,Â,¢,Ä,Í□A□wŠÇ—□ŽÒƒKƒCƒh□x,Ì□uMTS ŠÇ—□,ÌŽ©“®‰»□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,Ì§–ñ

- InstDLL.vbs ƒXƒNƒŠƒvƒg,ðŽÀ□s,·,é‘O,É□AVisual Studio 97 ActiveX ƒEƒBƒU□[ƒh,ðŽg,Á,Ä Java ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð“o˜^,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B
- ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ðŽg,Á,½ Java ƒRƒ“ƒ|□[ƒlƒ“ƒg,ìƒCƒ“ƒXƒg□[ƒ‹,Í□AAlpha ƒRƒ“ƒsƒ…□[ƒ^,Ü,½,Í i386 ƒRƒ“ƒsƒ…□[ƒ^□ã,Å,¾,¯ƒTƒ|□[ƒg,³,ê,Ü,·□BAlpha ƒRƒ“ƒsƒ…□[ƒ^,ìƒt□[ƒU□[,Í□AƒXƒNƒŠƒvƒg,ðŽÀ□s,·,é‘O,É□AInstDLL.vbs ,©,ç Java ƒRƒ“ƒ|□[ƒlƒ“ƒg,ìƒCƒ“ƒ|□[ƒg—p,ìƒR□[ƒh,ð□í□œ,µ,Ä,,¾,³,¢□B

MTS ,ÉŠÖ,·,é□î•ñ,Ì“üŽè

MTS ,ÉŠÖ,·,é□î•ñ,Í□AŽŸ,Ì•û–@,Å“üŽè,Å,«,Ü,·□B

- ƒhƒLƒ…ƒ□ƒ“ƒg,ð’²,×,é□B
- Microsoft Transaction Server ,Ìƒz□[ƒ€ ƒy□[ƒW http://www.microsoft.com/japan/products/ntserver/ ,ÉƒAƒNƒZƒX,·,é□B
- □»•iƒTƒ|□[ƒg ƒT□[ƒrƒX,É–â,¢□‡,í,¹,é□B

ƒhƒLƒ…ƒ□ƒ“ƒg,ð’²,×,é

MTS ,Ìƒhƒ‰ƒCƒo[,É,Í□AMTS ,Ì‹@”\,ÉŠÖ,·,éŠT”O□à–¾□A‘€□ìŽè□‡□AƒŠƒtƒ@ƒŒƒ“ƒX□î•ñ,ª‹L□Ú,³,ê,Ä,¢ ,Ü,·□B,Ü,½□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,âƒvƒ□ƒOƒ‰ƒ~ƒ“ƒOŒ¾Œê,ÌƒL□[ƒ□□[ƒh,ðŽg—p,µ,Ä,¢,é,Æ,«,É□A**F1** ƒL□[,ð‰Ÿ,·,Æ□A□ó‹µˆË‘¶,Ìƒwƒ‹ƒv,ð•\Ž¦,·,é,±,Æ,ª,Å,«,Ü,·□B"□ó‹µˆË‘¶" ,Æ,Í□A[ƒwƒ‹ƒv] ƒ□ƒjƒ… □[,ðŽg,í,¸,É□A**F1** ƒL□[,ð‰Ÿ,·,¾,¯,Å’¼□Úƒwƒ‹ƒv,ð•\Ž¦,Å,«,é‹@”\,Ì,±,Æ,Å,·□B

ŠÖ˜A□€–Ú ƒhƒLƒ…ƒ□ƒ“ƒg,Ì□\□¬

MTS ŠÇ—□ŽÒƒKƒCƒh

□wŠÇ—□ŽÒƒKƒCƒh□x,Å,Í□AMicrosoft® Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ð,¢,Â,Ç,Ì,æ,¤,ÉŽg—p,µ□Aƒpƒbƒ□[ƒW,Ì□ì□¬□AƒCƒ“ƒXƒg□[ƒ‹□A”z•z□A,¨,æ,Ñ•ÛŽç,ð□s,¤,©,ð□à–¾,µ,Ü,·□B,± ,ÌƒKƒCƒh,É,Í□AŽŸ,Ìƒt□[ƒU□[,ð‘Î□Û,Æ,·,é□î•ñ,ª‹L□q,³,ê,Ä,¢,Ü,·□B

- ƒVƒXƒeƒ€ŠÇ—□ŽÒ
- Web ŠÇ—□ŽÒ
- ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŠJ”ŽÒ

ŠJ”ŽÒ,Í□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì□ì□¬□A”z’u□A,¨,æ,Ñ”z•z,ð□s,¤ ,½,ß,Ì‘€□ìŽè□‡,ðŽQ□Æ,Å,«,Ü,·□BƒVƒXƒeƒ€ŠÇ—□ŽÒ,Æ Web ŠÇ—□ŽÒ,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì”z’u□AŠÇ—□□A,¨,æ,Ñ•ÛŽç,ð□s,¤,½,ß,Ì‘€□ìŽè□‡,ðŽQ□Æ,Å,«,Ü,·□B

ŽŸ,Ìƒgƒsƒbƒ‰,Å,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,½ƒpƒbƒP□[ƒW,Ì”z’u,âŠÇ—□,É,Â,¢,Ä□à–¾,µ□A,æ,è□Ú□×,ÈŽè□‡,ª‹L□q,³,ê,½ƒgƒsƒbƒ‰,Ö,ÌƒŠƒ“ƒN,ð’ñ‹Ÿ,µ,Ä,¢,Ü,·□B

MTS ƒpƒbƒP□[ƒW,Ì□ì□¬,É,Â,¢,Ä

MTS ƒpƒbƒP□[ƒW,Ì”z•z,É,Â,¢,Ä

MTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,É,Â,¢,Ä

MTS ƒpƒbƒP□[ƒW,Ì•ÛŽç,É,Â,¢,Ä

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□,É,Â,¢,Ä

MTS ,ÌŠÇ—□,ÌŽ©“®‰»,É,Â,¢,Ä

Windows 95/98 □ã,Ì MTS ƒGƒNƒXƒvƒ□□[ƒ‰

Windows® 95/98 ƒIƒyƒŒ□[ƒeƒBƒ“ƒO ƒVƒXƒeƒ€□ã,Å MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä MTS ƒpƒbƒP□[ƒW,ðŠÇ—□,·,é,±,Æ,ª,Å,«,Ü,·□B,½,¾,µ□AWindows 95/98 □ã,Å,Ì MTS ,ÌŠÇ—□,É,Í□AŽŸ,Ì□§–ñ,ª, ,è,Ü,·□B

- Windows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,ÅƒŠƒ,□[ƒg ƒŒƒWƒXƒgƒŠ ƒT□[ƒrƒX,ðŽÀ□s,µ,Ä,¢,é□ê□‡,Í□AWindows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,ðŽg,Á,ÄƒŠƒ,□[ƒg,©,ç Windows NT ƒRƒ“ƒsƒ…□[ƒ^,ðŠÇ—□,·,é,±,Æ,ª,Å,«,Ü,·□BƒŠƒ,□[ƒg ƒŒƒWƒXƒgƒŠ ƒT□[ƒrƒX,ðŽg—p,·,é,Æ□A(“K□Ø,ÈƒAƒNƒZƒXŒ ,ª, ,ê,Î) ƒŠƒ,□[ƒg,Ì Windows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,ÌƒŒƒWƒXƒgƒŠ ƒGƒ“ƒgƒŠ,ð•Ï□X,Å,«,Ü,·□B

  ƒŠƒ,□[ƒg ƒŒƒWƒXƒgƒŠ ƒT□[ƒrƒX,Í□AWindows 95/98 ,Ì CD-ROM ,Ì \Admin\Nettols\Remotreg ƒfƒBƒŒƒNƒgƒŠ,É, ,è,Ü,·□BRegserv.txt ƒtƒ@ƒCƒ‹,ÌƒŠƒ,□[ƒg ƒŒƒWƒXƒgƒŠ ƒT□[ƒrƒX,ÌƒCƒ“ƒXƒg□[ƒ‹Žè□‡,ðŽQ□Æ,µ□AƒŠƒ,□[ƒg ƒŒƒWƒXƒgƒŠ ƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€ (Regserv.exe) ,ðŽÀ□s,µ,Ü,·□B
- Windows 95/98 □ã,ÅŽÀ□s,µ,Ä,¢,é MTS ,ð□AƒŠƒ,□[ƒg,Ì•Ê,Ì Windows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,Ü,½,Í Windows NT ƒRƒ“ƒsƒ…□[ƒ^,©,çŠÇ—□,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B
- MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,ÉƒcƒŠ□[□\‘¢,Í•\Ž¦,³,ê,Ü,¹,ñ□BŠK‘w,ð ‰º,É^Ú“®,·,é,É,Í□AƒAƒCƒRƒ“,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ□A□ã,É^Ú“®,·,é,É,Í□Afc□[ƒ‹ ƒo□[,Ì [1 ,Â□ã,ÌƒŒƒxƒ‹] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B
- Windows 95/98 ,Å,ÍƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹ ƒt□[ƒeƒBƒŠƒeƒB,ªƒTƒ|□[ƒg,³,ê,Ä,¢,È,¢ ,½,ß□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□¶□¬,·,é,± ,Æ,Í,Å,«,Ü,¹,ñ□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹ ƒt□[ƒeƒBƒŠƒeƒB,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□¶□¬,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B
- Windows 95/98 □ã,ÌŠÇ—□,Å,Í□AMTS ƒZƒLƒ…ƒŠƒeƒB ƒvƒ□ƒpƒeƒB,Ü,½,Í MTS ƒ□□[ƒ‹,ªƒTƒ|□[ƒg,³,ê,Ä,¢ ,Ü,¹,ñ□B,µ,½,ª,Á,Ä□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å [ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_□A[ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_□A,¨,æ,Ñ [ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_,ð•\Ž¦,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B
- Windows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,Åƒt□[ƒU□[,ðƒ□□[ƒ‹,ÉŠ„,è“–,Ä,é,É,Í□AWindows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,Éƒt□[ƒU□[ƒŒƒxƒ‹,ÌƒAƒNƒZƒXŠÇ—□,ð□Ý’è,·,é•K—v,ª, ,è,Ü,·□BWindows 95/98 ƒRƒ“ƒgƒ□□[ƒ‹

ƒpƒlƒ‹,ðŠJ,¢,Ä [ƒlƒbƒgƒ□□[ƒN] ,ðƒNƒŠƒbƒN,µ,Ü,·□B[ƒAƒNƒZƒXŒ ,ÌŠÇ—□] ƒ^ƒu,ðƒNƒŠƒbƒN,µ,Ä [ƒ†□[ƒU□[ƒŒƒxƒ‹,ÌƒAƒNƒZƒXŠÇ—□] ,ð'I'ð,µ,Ü,·□B,±,Ì□Ý'è,Ì□Ú□×,É,Â,¢,Ä,Í□AWindows 95/98 ,ÌƒŠƒ\□[ƒXƒLƒbƒg ,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

- ƒŠƒ‚□[ƒg,Ì•Ê,ÌƒRƒ"ƒsƒ…□[ƒ^□ã,ÌƒNƒ‰ƒCƒAƒ"ƒg,©,ç Windows 95/98 □ã,ÅŽÀ□s,µ,Ä,¢,éƒRƒ"ƒ|□[ƒlƒ"ƒg,ÉƒAƒNƒZƒX,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B
- Windows 95/98 ,É,ÍƒVƒXƒeƒ€ ƒCƒxƒ"ƒg ƒ□ƒO,ª,È,¢,Ì,Å□AƒCƒxƒ"ƒg ƒ□ƒO,ÌƒGƒ"ƒgƒŠ,Í□AWindows ƒfƒBƒŒƒNƒgƒŠ,Ì \MTSLogs ƒTƒuƒfƒBƒŒƒNƒgƒŠ,É, ,é Transaction Server.html ,Æ,¢,¤–¼'O,Ì HTML ƒtƒ@ƒCƒ‹,É□',«□ž,Ü,ê,Ü,·□B,±,Ì HTML ƒtƒ@ƒCƒ‹,ðŽg,Á,Ä□AƒVƒXƒeƒ€,Ü,½,Íƒvƒ□ƒOƒ‰ƒ€,Å"-□¶,µ,½□d'å,ÈƒCƒxƒ"ƒg,ðŠÄŽ‹,·,é,±,Æ,ª,Å,«,Ü,·□B
- MTS ƒT□[ƒo□[ƒvƒ□ƒZƒX,ª (ƒtƒFƒCƒ‹ƒtƒ@□[ƒXƒg□AƒAƒT□[ƒg□A,Ü,½,ÍƒAƒNƒZƒXˆá"½,É,æ,Á,Ä) —\Šú,¹,¸□I—¹,µ,½□ê□‡□A,»,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ",ð□ÄŠJ,µ,æ,¤,Æ,·,é,Æ□A•¡□",Ì MTS ƒvƒ□ƒZƒX,ª□¶□¬,³,ê,é,Æ,¢ ,¤Šù'm,Ì–â'è,ª, ,è,Ü,·□B□³,µ,,Í□AƒpƒbƒP□[ƒW,²,Æ,É MTS ƒvƒ□ƒZƒX,ª 1 ,Â,¾,¯"®□ì,·,é•K—v,ª, ,è,Ü,·□BƒT□[ƒo□[ƒvƒ□ƒZƒX,ª—\Šú,¹,¸□I—¹,µ,½□ê□‡□AWindows 95/98 ƒRƒ"ƒsƒ…□[ƒ^,ð□Ä‹N"®,·,é•K—v,ª, ,è,Ü,·□B

ŠÖ˜A□€–Ú

□wMicrosoft Transaction Server ,ÌƒNƒCƒbƒN ƒcƒA□[□x□A□wMicrosoft Transaction Server ƒtƒ@□[ƒXƒg ƒXƒeƒbƒv ƒKƒCƒh□x

MTS ƒpƒbƒP□[ƒW,Ì□ì□¬,É,Â,¢,Ä

ƒpƒbƒP□[ƒW,Ì□ì□¬,Í□AŠJ”‰ß'ö,Ì□Å□I'iŠK,Å□s,í,ê,Ü,·□BƒpƒbƒP□[ƒWŠJ”ŽÒ□A□ã‹‰ƒVƒXƒeƒ€ŠÇ—□ŽÒ□A,¨,æ,Ñ□ã‹‰ Web ŠÇ—□ŽÒ,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒpƒbƒP□[ƒW,Ì□ì□¬,Æ”z'u,ð□s,¢,Ü,·□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŠJ”Žž,ÉŒˆ'è,³,ê,éƒpƒbƒP□[ƒW,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□\□¬,ð MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄŽÀŒ»,µ,Ü,·□B

MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□ÝŒv,µ□A□ì□¬,·,é•û–@,Ì□Ú□×,É,Â,¢,Ä,Í□A□wProgrammer's Guide□x,Ì□uBuilding MTS Applications (ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì□ì□¬)□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

'€□ìŽè□‡

‹ó,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

MTS ƒpƒbƒP□[ƒW,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,·,é

MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,ÉƒCƒ“ƒ|□[ƒg,·,é

MTS ƒpƒbƒP□[ƒW,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é

ƒGƒNƒXƒ|□[ƒg—p,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

MTS ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,ð□Ý'è,·,é

MTS ƒAƒNƒeƒBƒu‰»,Ìƒvƒ□ƒpƒeƒB,ð□Ý'è,·,é

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìƒvƒ□ƒpƒeƒB,ð□Ý'è,·,é

MTS ”F□ØƒCŒfxƒ‹,ð□Ý'è,·,é

MTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é

ŠÖ˜A□€–Ú

□wMicrosoft Transaction Server ,ÌƒNƒCƒbƒN ƒcƒA□[□x□A□wMicrosoft Transaction Server ƒtƒ@□[ƒXƒg ƒXƒeƒbƒv ƒKƒCƒh □x□AMTS ƒpƒbƒP□[ƒW,Ì”z•z,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,Ì•ÛŽç,É,Â,¢,Ä□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□,É,Â,¢,Ä□AMTS ,ÌŠÇ—□,ÌŽ©“®‰»,É,Â,¢,Ä

MTS ƒpƒbƒP□[ƒW,Ì”z•z,É,Â,¢,Ä

MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□ì□¬,µ□AƒpƒbƒP□[ƒW,É‘g,Ý□ž,ñ,¾,ç□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒNƒ‰ƒCƒAƒ“ƒg,É”z•z,µ,Ü,·□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð”z•z,·,é,É,ÍŽŸ,Ì•û–@,ª, ,è,Ü,·□B

- MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AŽg—p,µ,Ä,¢,éƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,©,çƒVƒXƒef€ŠÇ—□ŽÒ,Ü,½,Í Web ƒTƒCƒgŠÇ—□ŽÒ,ÌƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,ÖƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒvƒbƒVƒ…,·,é•û–@□B,±,Ì□ê□‡,Í□A—¼•û,ÌƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,Å MTS ,ðŽÀ□s,µ,Ä,¢,é•K—v,ª, ,è,Ü,·□B
- MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÅƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹ ƒt□[ƒeƒBƒŠƒeƒB,ðŽg,Á,Ä□AƒŠƒ,□[ƒg ƒT□[ƒo□[,ðŽQ□Æ,·,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ðŽ©“®“I,É□¶□¬,·,é•û–@□BƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Å MTS ,ðŽÀ□s,µ,Ä,¢,é•K—v,Í, ,è,Ü,¹,ñ□B

ƒNƒ‰ƒCƒAƒ“ƒg,ÌƒRƒ“ƒsƒ…□[ƒ^,É,Í□AMTS ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,é,±,Æ,à□AƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,È,¢,±,Æ,à, ,é,Ì,Å□AƒNƒ‰ƒCƒAƒ“ƒg,ÉƒT□[ƒo□[ƒpƒbƒP□[ƒW,ð”z•z,·,é,Æ,«,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹ ƒt□[ƒeƒBƒŠƒeƒB,ðŽg,¤,±,Æ,ð,¨Š©,ß,µ,Ü,·□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,Í□AƒŠƒ,□[ƒg,Ì MTS ƒT□[ƒo□[,ÅŽÀ□s,µ,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉƒAƒNƒZƒX,·,é,æ,¤,ÉƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ðŽ©“®“I,É□\□¬,µ,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŽè“®,Å□\□¬,·,é,±,Æ,à,Å,«,Ü,·□B

ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO,Ì’mŽ¯,ª,È,,Ä,àƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹ ƒt□[ƒeƒBƒŠƒeƒB,ÍŽg—p,Å,«,Ü,·,ª□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð”z•z,·,é,É,Í□AƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÆƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒpƒbƒP□[ƒW‰»,µ□A’ñ‹Ÿ,·,é,±,Æ,Ì^Ó–¡,É,Â,¢,Ä□n’m,µ,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□B,½,Æ,¦,Î□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□³,µ,ƒpƒbƒP□[ƒW‰»,Å,«,È,¢,Æ□AƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ÉƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒR□[ƒh,ð‘}“ü,µ,ÄƒNƒ‰ƒCƒAƒ“ƒg,É‹Ÿ‹‹,µ,Ä,µ,Ü,¤,± ,Æ,à, ,è,Ü,·□B

‘€□ìŽè□‡

ƒŠƒ,□[ƒg MTS ƒRƒ“ƒsƒ…□[ƒ^,ðŽg,¤

MTS ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg

MTS ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□¶□¬,·,é

ŠÖ˜A□€–Ú

□wMicrosoft Transaction Server ,ÌƒNƒCƒbƒN ƒcƒA□[□x□A□wMicrosoft Transaction Server ƒtƒ@□[ƒXƒg ƒXƒeƒbƒv ƒKƒCƒh □x□AMTS ƒpƒbƒP□[ƒW,Ì□ì□¬,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,Ì•ÛŽç,É,Â,¢,Ä□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□,É,Â,¢,Ä□AMTS ,ÌŠÇ—□,ÌŽ©“®‰»,É,Â,¢,Ä

MTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,É,Â,¢,Ä

ƒpƒbƒP□[ƒW,ð□ì□¬,µ,½Œã,Í□AƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,Æ”z’u,ð□s,¢,Ü,·,ª□A,»,ê,É,ÍƒpƒbƒP□[ƒW,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒpƒeƒB,ÉŠÖ,·,é’mŽ¯,ª•K—v,Å,·□B,½,Æ,¦,Î□AƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,µ,½,ç□AƒVƒXƒeƒ€ŠÇ—□ŽÒ,Ü,½,Í Web ŠÇ—□ŽÒ,Í□AƒpƒbƒP□[ƒW,ÉŠÖ˜A•t,¯,ç,ê,Ä,¢,éƒ□□[ƒ‹,É Windows NT ƒt□[ƒU□[,ðƒ}ƒbƒv,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BƒVƒXƒeƒ€ŠÇ—□ŽÒ,Ü,½,Í Web ŠÇ—□ŽÒ,Í□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽg—p,·,éƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,ðƒ□□[ƒ‹,Éƒ}ƒbƒv,·,é‘O,É□Aƒ□□[ƒ‹,ÉŠî,Ã,□éŒ¾,É,æ,éƒZƒLƒ… ƒŠƒeƒB,ÆƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉŠÖ˜A•t,¯,ç,ê,Ä,¢,éƒ□□[ƒ‹,É,Â,¢,Ä□A□\•ª,É—□‰ð,µ,Ä,,¾,³,¢□B

‘€□ìŽè□‡

ŠÖ˜A□€–Ú

□wMicrosoft Transaction Server ,ÌƒNƒCƒbƒN ƒcƒA□[□x□A□wMicrosoft Transaction Server ƒtƒ@□[ƒXƒg ƒXƒeƒbƒv ƒKƒCƒh □x□AMTS ƒpƒbƒP□[ƒW,Ì□ì□¬,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,Ì”z•z,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,Ì•ÛŽç,É,Â,¢,Ä□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□,É,Â,¢,Ä□AMTS ,ÌŠÇ—□,ÌŽ©“®‰»,É,Â,¢,Ä

MTS ƒpƒbƒP□[ƒW,Ì•ÛŽç,É,Â,¢,Ä

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,Ì□ó‘Ô,ðŠÄŽ‹,µ□A•K—v,É‰ž,¶,ÄƒRƒ“ƒ|□[ƒlƒ“ƒg,ÆƒpƒbƒP□[ƒW,lƒvƒ□ƒpƒeƒB,ð□Ä□\□¬,·,é,±,Æ,É,æ,Á,Ä□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð•ÛŽç,·,é,±,Æ,ª,Å,«,Ü,·□B,±,±,Å,Í□AŠù,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê□A”z’u,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,ð□Ä□\□¬,·,é•û–@,É,Â,¢,Ä□à–¾,µ,Ü,·□B

‘€□ìŽè□‡

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□ó‘Ô,Æƒvƒ□ƒpƒeƒB,ðŠÄŽ‹,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,¤

MTS ƒ□□[ƒ‹,ÌƒtƒU□[,ðŠÇ—□,·,é

MTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽg,¤

ŠÖ˜A□€–Ú

□wMicrosoft Transaction Server ,ÌƒNƒCƒbƒN ƒcƒA□[□x□A□wMicrosoft Transaction Server ƒtƒ@□[ƒXƒg ƒXƒeƒbƒv ƒKƒCƒh □x□AMTS ƒpƒbƒP□[ƒW,Ì□ì¬,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,Ì”z•z,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,lƒCƒ“ƒXƒg□[ƒ‹,É,Â,¢,Ä□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□,É,Â,¢,Ä□AMTS ,ÌŠÇ—□,ÌŽ©“®‰»,É,Â,¢,Ä

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÇ—□‚É‚Â‚¢‚Ä

MTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ðŽg‚Á‚Ä‚Ç‚Ì‚æ‚¤‚È□ó‹µ‚Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠÇ—□‚·‚é‚©‚ð—□‰ð·‚é‚½ß‚É□AŠÇ—□ŽÒ‚Í□A•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª‚Ç‚Ì‚æ‚¤‚É“®□ì‚·‚é‚©‚ð—□‰ð‚µ‚Ä‚¾‚³‚¢□B‚±‚±‚Å‚Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó‘Ô□A¨‚æ‚ÑŠÇ—□‚Ì‘Î□Û‚É‚È‚Á‚Ä‚¢‚éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠÄŽ‹‚µ□AŠÇ—□‚·‚é•û–@‚É‚Â‚¢‚Ä□à–¾‚µ‚Ü‚·□B

‘€□ìŽè□‡

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚É‚Â‚¢‚Ä

MS DTC ‚ÌŠÇ—□

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÄŽ‹

Windows 95/98 ‚Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÄŽ‹

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó‘Ô‚É‚Â‚¢‚Ä

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì‰ðŒˆ

ŠÖ˜A□€–Ú

□wMicrosoft Transaction Server ‚ÍƒNƒCƒbƒN ƒcƒA□[□x□A□wMicrosoft Transaction Server ƒtƒ@□[ƒXƒg ƒXƒeƒbƒv ƒKƒCƒh □x□AMTS ƒpƒbƒP□[ƒW‚Ì□ì¬‚É‚Â‚¢‚Ä□AMTS ƒpƒbƒP□[ƒW‚Ì”z•z‚É‚Â‚¢‚Ä□AMTS ƒpƒbƒP□[ƒW‚ÌƒCƒ“ƒXƒg□[ƒ‹‚É‚Â‚¢‚Ä□AMTS ƒpƒbƒP□[ƒW‚Ì•ÛŽç‚É‚Â‚¢‚Ä□AMTS ‚ÌŠÇ—□‚ÌŽ©“®‰»‚É‚Â‚¢‚Ä

MTS ,ÌŠÇ—□,ÌŽ©“®‰»,É,Â,¢,Ä

ŠÇ—□ƒXƒNƒŠƒvƒg,ð—˜—p,·,é,±,Æ,É,æ,Á,Ä□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,É,æ,éƒpƒbƒP□[ƒW,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□\□¬,ðŽ©“®‰»,·,é,±,Æ,ª,Å,«,Ü,·□BMicrosoft Visual Basic ,È,Ç,ÌƒXƒNƒŠƒvƒgŒ¾Œê,Ì’mŽ¯,ª, ,ê,Î□AƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠÇ—□□ì‹Æ,ðŽ©“®‰»,Å,«,Ü,·□B,½,Æ,¦,Î□Aƒc□[ƒ‹ŠJ”ŽÒ,Í□AƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä□AƒŠƒ,□[ƒg ƒNƒ‰ƒCƒAƒ“ƒg,ðŽ©“®“I,É□\□¬,·,éƒIƒuƒWƒFƒNƒg,È,Ç,Ì MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒAƒhƒCƒ“,ð□ì□¬,·,é,± ,Æ,à,Å,«,Ü,·□B

’□ ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŽg— p,·,é,É,Í□AƒI□[ƒgƒ□□[ƒVƒ‡ƒ“ŒÝŠ·,ÌƒXƒNƒŠƒvƒgŒ¾Œê,ÌŽg,¢•û,ÉŠÖ,·,é’mŽ¯,ª•K—v,Å,·□B

ŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,Æ□AMicrosoft Visual Basic ,¨,æ,Ñ Microsoft Visual C++® API ,ÉŠÖ,·,éƒŠƒtƒ@ƒŒƒ“ƒX□î•ñ,ÆƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ÌƒTƒ“ƒvƒ‹ ƒR□[ƒh,ª‹L□Ú,³,ê,Ä,¢,é MTS ŠÇ—□ƒŠƒtƒ@ƒŒƒ“ƒX,ðŽQ□Æ,·,é,É,Í□A"MTS ŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒ□ƒ“ƒg" ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,Í Visual Basic Script ,Å‹L□q,³,ê,Ä,¨,è□AWindows® NT 4.0 Option Pack ƒZƒbƒgƒAƒbƒv ƒvƒ□ƒOƒ‰ƒ€,ÅƒCƒ“ƒXƒg□[ƒ‹,³,ê,é Windows® Scripting Host ,ðŽg—p,µ,Ü,·□BMTS ŠÇ—□ƒXƒNƒŠƒvƒg,ÌŽg,¢•û,Ì□Ú□×,É,Â,¢,Ä,Í□A□uŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ÌƒZƒbƒgƒAƒbƒv□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

‘€□ìŽè□‡

MTS ŠÇ—□ƒIƒuƒWƒFƒNƒg

MTS ŠÇ—□Ž©“®‰» Visual Basic Script ƒTƒ“ƒvƒ‹

MTS ŠÇ—□Ž©“®‰» Visual Basic ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“

Visual Basic ,É,æ,é MTS ŠÇ—□,ÌŽ©“®‰»

Visual Basic ,É,æ,é□,“x,È MTS ŠÇ—□,ÌŽ©“®‰»

ŠÖ˜A□€–Ú

□wMicrosoft Transaction Server ,ÌƒNƒbƒN ƒcƒA□[□x□A□wMicrosoft Transaction Server ƒtƒ@□[ƒXƒg ƒXƒeƒbƒv ƒKƒCƒh □x□AMicrosoft Transaction Server ƒtƒ@□[ƒXƒg ƒXƒeƒbƒv ƒKƒCƒh□AMTS ƒpƒbƒP□[ƒW,Ì□ì□¬,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,Ì”z•z,É,Â,¢,Ä□AMTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,É,Â,¢ ,Ä□AMTS ƒpƒbƒP□[ƒW,Ì•ÛŽç,É,Â,¢,Ä□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□,É,Â,¢,Ä

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK‘w

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Í□AMTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÅŽÀ□s,µ,Ä,¢,é MTS ƒpƒbƒP□[ƒW,Æ MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŠÇ—□,·,é,½,ß,ÌƒrƒWƒ…ƒAƒ‹ ƒc□[ƒ‹,Å,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒpƒbƒP□[ƒW,Ö,Ì‘g,Ý□ž,Ý,©,çƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì□ó‘Ô,ÌŠÄŽ‹,Ü,Å,Ì□ì‹Æ,ðŽÀ□s,µ,Ü,·□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Í□AWindows NT □ã,Å,Í Microsoft Management Console (MMC) ,ðƒzƒXƒg,Æ,·,éƒXƒiƒbƒvƒCƒ“,Å, ,è□AWindows 95/98 □ã,Å,ÍŽÀ□s‰Â”\ftƒ@ƒCƒ‹ (mtxpd.exe) ,Å,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,ÌŠK‘wƒcƒŠ□[,É,Í□AŽŸ,Ì□€–Ú,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

- **[ƒRƒ“ƒsƒ…□[ƒ^]**

  ,±,ÌƒT□[ƒo□[,©,çŠÇ—□,·,éƒRƒ“ƒsƒ…□[ƒ^,ªŠÜ,Ü,ê,Ä,¢,Ü,·□Bƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^,É,Í "ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^" ,Æ,¢,¤–¼‘O,ª•t,¢,Ä,¢,Ü,·□B
- **[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW]**

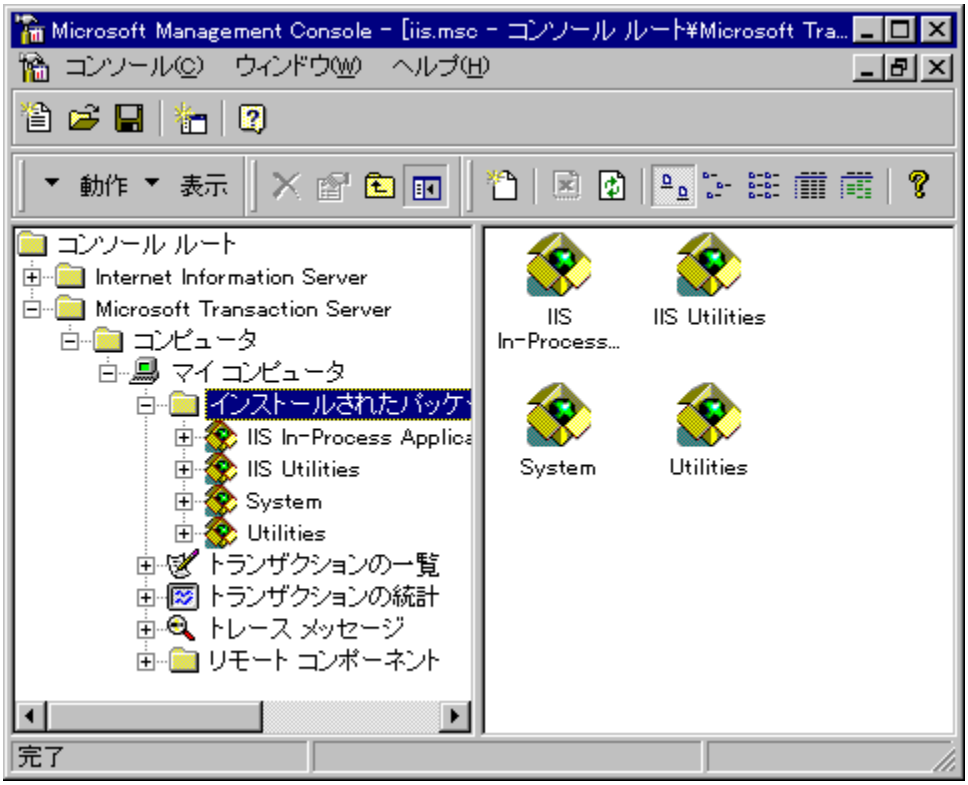
  ƒRƒ“ƒsƒ…□[ƒ^,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B
- **[ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg]**

  ƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^□ã,ÌƒpƒbƒP□[ƒW,ªŽg—p,·,éƒŠƒ‚□[ƒg ƒRƒ“ƒsƒ…□[ƒ^□ã,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B
- **[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ì^ê——]**

  MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ªŽQ‰Á,µ,Ä,¢,éŒ»□Ý,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª•\Ž¦,³,ê,Ü,·□B
- **[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ì“□Œv]**

  ƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^,ªŽQ‰Á,µ,Ä,¢,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv□î•ñ,ª•\Ž¦,³,ê,Ü,·□B
- **[ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW]**

  Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒffƒBƒl□[ƒ^ (MS DTC) ,ª”□s,µ,Ä,¢,éŒ»□Ý,ÌƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW,Ìˆê——,ª•\Ž¦,³,ê,Ü,·□B



’□　Windows 95/98 □ã,Ì MTS ƒGƒNƒXƒvƒ□□[ƒ‰,É,Í□AƒEƒBƒ“ƒhƒE—Ìˆæ,ª 1 ,Â,µ,©, ,è,Ü,¹,ñ□BWindows

95/98 □ã,Å MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK‘w,ð
‰º,É^Ú“®,·,é,É,Í□AƒAƒCƒRƒ“,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ□AŠK‘w,ð□ã,É^Ú“®,·,é,É,Í□Aƒc□[ƒ‹ ƒo□[,Ì [1
,Â□ã,ÌƒŒƒxƒ‹] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B

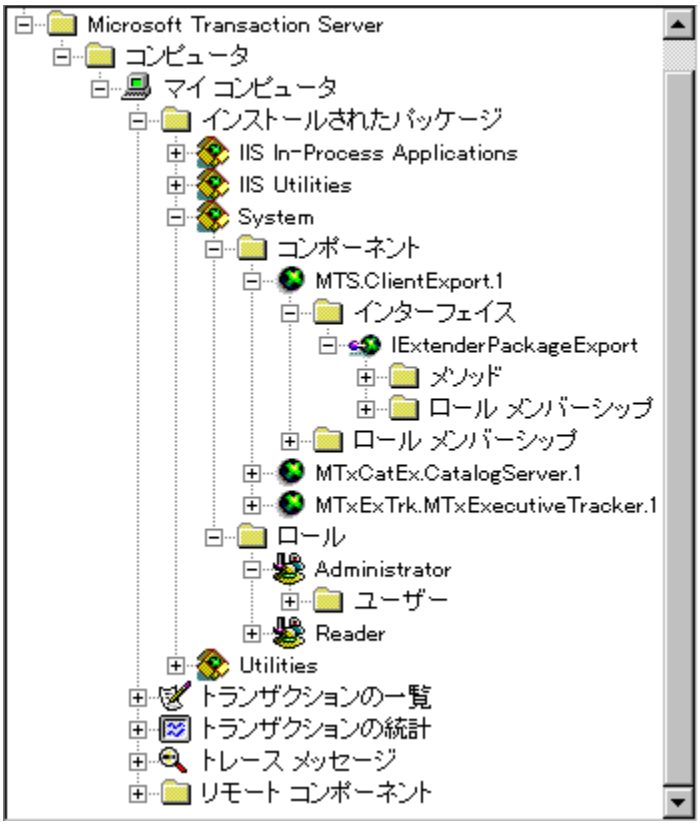
,Ü,½□AWindows 95/98 □ã,Å,Í□A[ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,¨,æ,ÑƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\
ƒtƒ@ƒCƒ‹ ƒt□[ƒefBƒŠƒefB,ÍŽg—p,Å,«,Ü,¹,ñ□B

Windows 95/98 □ã,Ì MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì□§–ñ,É,Â,¢,Ä,Í□A□wŠÇ—□ŽÒƒKƒCƒh□x,Ì□uWindows 95/98 □ã,Ì MTS ƒGƒNƒXƒvƒ□□[ƒ‰□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ªŠÇ—□,·,éƒŠƒ,□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,Å,Í□AƒRƒ“ƒsƒ…□[ƒ^,ÌƒAƒCƒRƒ“,Ì‰º,ÌŠK‘w,É
[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_□A[ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê
——]□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv]□A,¨,æ,Ñ [ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ,ÌŠeƒEƒBƒ“ƒhƒE,Ìˆê——,ª•\
Ž¦,³,ê,Ü,·□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ŒÂ□X,ÌƒpƒbƒP□[ƒW,É,Í□AŽŸ,ÌƒTƒuƒtƒHƒ‹ƒ_,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

- **[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_**
  ƒpƒbƒP□[ƒW,É‘g,Ý□ž,Ü,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B
- **[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_**
  ƒpƒbƒP□[ƒW“à,ÅŽg—p,Å,«,éƒ□□[ƒ‹,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B
- **[ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_**
  ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ü,½,ÍƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒCƒ“ƒ^□[ƒtƒFƒCƒX,ÉŠ„,è“–,Ä,ç,ê,Ä,¢,éƒ□□[ƒ‹,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B
- **[ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_**
  ƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,Éƒ}ƒbƒv,³,ê,Ä,¢,éƒ†□[ƒU□[,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B
- **[ƒCƒ“ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_**
  ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒCƒ“ƒ^□[ƒtƒFƒCƒX,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B
- **[ƒ□ƒ\ƒbƒh] ƒtƒHƒ‹ƒ_**
  ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ìƒ□ƒ\ƒbƒh,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B



ŠÖ˜A□€-Ú

□wMicrosoft Transaction Server ,ìfNfCfbfN fcfA□[□x

[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_

[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,É,Í□A[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ,Æ□Aƒ†□[ƒU□[,ª [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,É'Ç‰Á,µ,½,»,Ì,Ù,©,ÌƒRƒ“ƒsƒ…□[ƒ^,ªŠÜ,Ü,ê,Ä,¢,Ü,·□BŠù'è,Ì□Ý'è,Å,Í□AMTS ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒ□□[ƒJƒ‹ƒRƒ“ƒsƒ…□[ƒ^,ª [ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ,É'Š"–,µ,Ü,·□B

[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ÉƒRƒ“ƒsƒ…□[ƒ^,ð'Ç‰Á,·,é,É,Í□AŽŸ,Ì,¢,¸,ê,©,Ì'€□ì,ðŽÀ□s,µ,Ü,·□B

- [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ",ÅƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ|ƒCƒ"ƒg,µ,Ü,·□BŽŸ,É□A[ƒRƒ“ƒsƒ…□[ƒ^] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
- [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ð'I'ð,µ□Aƒc□[ƒ‹ ƒo□[,Ì [□V,µ,¢ƒIƒuƒWƒFƒNƒg,Ì□ì□¬] ƒ{ƒ^ƒ",ðƒNƒŠƒbƒN,µ,Ü,·□B
- [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ð'I'ð,µ□A["®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ"ƒg,µ,Ü,·□BŽŸ,É□A[ƒRƒ“ƒsƒ…□[ƒ^] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

[ƒRƒ“ƒsƒ…□[ƒ^,Ì'Ç‰Á] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□AŽg—p,µ,Ä,¢,éƒRƒ“ƒsƒ…□[ƒ^,©,çŠÇ—□,·,éƒT□[ƒo□[,Ì–¼'O,ð"ü—Í,µ,Ü,·□B□V,µ,¢ƒT□[ƒo□[,ª [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,É'Ç‰Á,³,ê□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK'w,Ì [ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ƒAƒCƒRƒ",Æ"¯,¶ƒŒƒxƒ‹,É•\Ž¦,³,ê,Ü,·□B

ŠÖ˜A□€–Ú

[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^]□AƒRƒ“ƒsƒ…□[ƒ^,Ìƒvƒ□ƒpƒefB

[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^]

[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ,Í□AMTS ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^,Å,·□B

[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ÌƒCƒxƒ‹,ÅŽÀ□s,Å,«,é'€□ì,É,Â,¢,Ä,Í□AŽŸ,ÌƒgƒsƒbƒN,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

MTS ,ÌƒCƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽg,¤

MS DTC ,ÌŠÇ—□

,·,×,Ä,Ì [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ÉŽŸ,Ì□€–Ú,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW]

,±,ÌƒRƒ“ƒsƒ…□[ƒ^,ªŠÇ—□,µ,Ä,¢,éƒVƒXƒeƒ€
ƒpƒbƒP□[ƒW,Æ□Aƒ†□[ƒU□[,ªƒCƒ“ƒXƒg□[ƒ‹,µ,½ƒpƒbƒP□[ƒW,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

[ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg]

ƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^□ã,ÌƒpƒbƒP□[ƒW,ªŒÄ,Ñ□o,·ƒŠƒ‚□[ƒg ƒRƒ“ƒsƒ…□[ƒ^□ã,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢
,Ü,·□B

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——]

,±,ÌƒRƒ“ƒsƒ…□[ƒ^,ªŠÇ—□,µ,Ä,¢,éŒ»□Ý,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——,ª•\Ž¦,³,ê,Ü,·□B

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv]

ƒRƒ“ƒsƒ…□[ƒ^,ªŽQ‰Á,µ,Ä,¢,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

[ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW]

Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒƒBƒl□[ƒ^ (MS DTC),ª"□s,µ,Ä,¢,éŒ»□Ý,ÌƒgƒŒ□[ƒX
ƒ□ƒbƒZ□[ƒW,Ì^ê——,ª•\Ž¦,³,ê,Ü,·□B

[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_"à,Ì,·,×,Ä,ÌƒRƒ“ƒsƒ…□[ƒ^,Æ"¯—l,É□A[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^]
,Ìƒvƒ□ƒpƒeƒB,àŽŸ,Ìƒ^ƒu,ðŽg,Á,Ä□\□¬,·,é,±,Æ,ª,Å,«,Ü,·□B

- [‘S"Ê] ƒ^ƒu
- [ƒlƒvƒVƒ‡ƒ“] ƒ^ƒu
- [□Ú□×□Ý'è] ƒ^ƒu

ŠÖ˜A□€–Ú

[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,É,Í□Aƒ†□[ƒU□[,ªƒRƒ“ƒsƒ…□[ƒ^,É’Ç‰Á,µ,½,·,×,Ä,Ìƒpƒbƒ P□[ƒW,Ì^ê——,ª•\Ž¦,³,ê,Ü,·□Bƒpƒbƒ P□[ƒW ƒŒƒxƒ‹,Å,Í□AŽŸ,Ì‘€□ì,ðŽÀ□s,Å,«,Ü,·□B

‹ó,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

Šù□¬,Ì MTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹

MTS ƒpƒbƒP□[ƒW,ÌƒAƒbƒvƒOŒ□[ƒh

MTS ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é

MTS ƒAƒNƒeƒBƒu‰»,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é

MTS ƒpƒbƒP□[ƒW ID ,Ì□Ý’è

MTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é

MTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é

MTS ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□ó‘Ô,Æƒvƒ□ƒpƒeƒB,ðŠÄŽ‹,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,¤

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,É,Í□AŽŸ,ÌƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW,àŠÜ,Ü,ê,Ä,¢,Ü,·□B

System ƒpƒbƒP□[ƒW

Utilities ƒpƒbƒP□[ƒW

’□ ,±,ê,ç,Ìƒpƒbƒ P□[ƒW,Í MTS ,Ì“à•”ƒpƒbƒP□[ƒW,Å, ,è□A’Ê□í,Í•Ï□X,µ,½,è□\□¬,µ,½,è,µ,È,¢,Å,- ,¾,³,¢□B,½,¾,µ□AMTS ƒT□[ƒo□[,ÌŒ ŒÀ,ðŠg’£,µ,½,è□§ŒÀ,µ,½,è,·,é,É,Í□AMTS ,Ì“à•”ƒpƒbƒP□[ƒW,ð□\ □¬,·,é•K—v,ª, ,é□ê□‡,ª, ,è,Ü,·□B,½,Æ,¦,Î□AMTS ƒT□[ƒo□[□ã,Ìƒ†□[ƒU□[,ÌŠÇ— □Œ ŒÀ,ð□Ý’è,·,é,É,Í□ASystem ƒpƒbƒP□[ƒW,Ì Administrator ƒ□□[ƒ‹,Éƒ†□[ƒU□[,ð’Ç‰Á,µ,Ü,·□B

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,É,Í□A,¢,,Â,Å,àƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,± ,Æ,ª,Å,«,Ü,·□BƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ŒÃ□X,ÌƒpƒbƒP□[ƒW,É,Í□AŽŸ,ÌƒTƒuƒtƒHƒ‹ƒ_,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

- [fRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_
  ƒpƒbƒP□[ƒW,É‘g,Ý□ž,Ü,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B
- [ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_
  ƒpƒbƒP□[ƒW,ÉŠ„,è“–,Ä,ç,ê,Ä,¢,éƒ□□[ƒ‹,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

ŽŸ,Ìƒ^ƒu,ðŽg,Á,ÄƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,ð□\□¬,·,é,±,Æ,ª,Å,«,Ü,·□B

- [‘S”Ê] ƒ^ƒu
- [ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu
- [□Ú□×□Ý’è] ƒ^ƒu
- [ID] ƒ^ƒu
- [ƒAƒNƒeƒBƒu‰»] ƒ^ƒu

MTS ,ð Internet Information Server (IIS) Version 4.0 ,Æ‘g,Ý□‡,í,¹,ÄŽg—p,µ,Ä,¢ ,é□ê□‡,Í□A[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ÉŽŸ,Ì IIS ŒÅ—L,ÌƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW,àŠÜ,Ü,ê,Ä,¢ ,Ü,·□B

- IIS ƒCƒ“ƒvƒ□ƒZƒX ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“
  [IIS In-Process Applications] ƒtƒHƒ‹ƒ_,É,Í□AInternet Information Server (IIS) ,Ìƒvƒ□ƒZƒX,ÅŽÀ□s,³,ê,Ä,¢ ,éŠe IIS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□BIIS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í□AIIS

,Ìƒvƒ□ƒZƒX,Ü,½,Í•Ê,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìƒvƒ□ƒZƒX,ÅŽÀ□s,Å,«,Ü,·□BIIS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª IIS ,Ìƒvƒ□ƒZƒX,ÅŽÀ□s,³,ê,Ä,¢,é□ê□‡□AIIS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í [IIS In-Process Applications] ƒtƒHƒ‹ƒ_“à,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,Æ,µ,Ä•\Ž¦,³,ê,Ü,·□BIIS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª•Ê,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìƒvƒ□ƒZƒX,ÅŽÀ□s,³,ê,Ä,¢,é□ê□‡□AIIS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK‘w,É“Æ—§,µ,½ƒpƒbƒP□[ƒW,Æ,µ,Ä•\Ž¦,³,ê,Ü,·□B

- IIS ƒt□[ƒeƒBƒŠƒeƒB

  [IIS Utilities] ƒtƒHƒ‹ƒ_,É,Í□AActive Server Pages (ASP) ,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð—LŒø,É,·,é,½,ß,É•K—v,È ObjectContext ƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ASP ,Ì□Ú□×,É,Â,¢,Ä,Í□AInternet Information Server (IIS) ,ÌƒhƒLƒ…ƒ□ƒ“ƒg,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

  Microsoft Transaction Server ,Í□A“à•”ŠÖ□”,Å ObjectContext ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŽg—p,µ,Ü,·□B,±,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒpƒeƒB,ð•\Ž¦,·,é,±,Æ,Í,Å,«,Ü,·,ª□A□Ý’è,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

’□ ƒvƒ□ƒZƒX,ð•ª—£,µ,Ä•ÛŒì,³,ê,½ IIS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□ì□¬,·,é□ê□‡□AIIS ,Í IWAN_<*computer name*> ,Æ,µ,ÄŽÀ□s,³,ê,é,æ,¤,É MTS ƒpƒbƒP□[ƒW,ð□ì□¬,µ,Ü,·□B,±,ê,ç,ÌƒpƒbƒP□[ƒW,Ì ID ,ð•Ï□X,·,é□ê□‡□A□V,µ,¢ƒpƒbƒP□[ƒW ID ,ð "MTS Impersonators" ƒOƒ‹□[ƒv,É,à’Ç‰Á,·,é•K—v,ª, ,è,Ü,·□B,³,à,È,¯,ê,Î□AƒpƒbƒP□[ƒW,ªŒÄ,Ñ□o,·ƒvƒ□ƒZƒXŠO,Ì,Ù,©,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Å,Í□AƒZƒLƒ…ƒŠƒeƒB‹@”\,ª□³,µ,‹@”\,µ,Ü,¹,ñ□B,±,ê,ð□s,í,È,¢,Æ□AŒÄ,Ñ□o,µŒ³,Í□AIIS ,ÌŽÀ□Û,ÌƒNƒ‰ƒCƒAƒ“ƒg,Å,Í,È,-□AIWAN_<*computer name*> ,ÆŒ©,È,³,ê,Ü,·□B

Utilities ƒpƒbƒP□[ƒW

Utilities ƒpƒbƒP□[ƒW,É,Í□A**TransactionContext** ,Æ **TransactionContextEx** ,Ì 2 ,Â,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□Bƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg,Å TransactionContext/TransactionContextEx ,ðŽg—p,µ,Ä 1 ,Â,Ü,½,Í•¡□”,Ì Microsoft Transaction Server ƒIƒuƒWƒFƒNƒg,Ì“®□ì,ð‘g,Ý□‡,í,¹,½ƒAƒgƒ~ƒbƒN ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð□ì□¬,µ□A,»,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒAƒ{□[ƒg,µ,½,è□AƒRƒ~ƒbƒg,µ,½,è,·,é,±,Æ,ª,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□

System ƒpƒbƒP□[ƒW

System ƒpƒbƒP□[ƒW,É,Í□A•Ï□X,Å,«,È,¢ƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□BMicrosoft Transaction Server ,Í□A,±,ê,ç,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ð“à•”ŠÖ□”,ÅŽg—p,µ,Ü,·□BƒVƒXƒef€ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒvƒOƒpƒeƒB,ð•\Ž¦,·,é,± ,Æ,Í,Å,«,Ü,·,ª□A□Ý’è,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,É,æ,Á,ÄŠÇ—□,³,ê,Ä,¢ ,éƒpƒbƒP□[ƒW,ðƒt□[ƒU□[,ª□í□œ,µ,½,è•Ï□X,µ,½,è,·,é,É,Í□A,»,Ìƒt□[ƒU□[,Í System ƒpƒbƒP□[ƒW,Ì Administrator ƒ□□[ƒ‹,Éƒ}ƒbƒv,³,ê,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□Bƒvƒ‰ƒCƒ}ƒŠ ƒhƒ□ƒCƒ“ ƒRƒ“ƒgƒ□□[ƒ‰ ,Ü,½,ÍƒoƒbƒNƒAƒbƒv ƒhƒ□ƒCƒ“ ƒRƒ“ƒgƒ□□[ƒ‰,Ì–ðŠ„,ð‰Ê,½,µ,Ä,¢,éƒT□[ƒo□[,É MTS ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢ ,é□ê□‡□Aƒt□[ƒU□[,ª MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Åƒpƒbƒ P□[ƒW,ðŠÇ—□,·,é,É,Í□A,»,Ìƒt□[ƒU□[,Íƒhƒ□ƒCƒ“ŠÇ— □ŽÒ,É,È,Á,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

MTS ƒT□[ƒo□[,Ì□\□¬□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é□AMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é□A[ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_

[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_

[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,É,Í□A'I'ð,³,ê,Ä,¢,éƒpƒbƒP□[ƒW"à,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒŒƒxƒ‹,Å,Ì'€□ìŽè□‡,É,Â,¢,Ä,Í□AŽŸ,ÌƒgƒsƒbƒN,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

MTS ƒpƒbƒP□[ƒW,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,·,é

MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,ÉƒCƒ“ƒ|□[ƒg,·,é

MTS ƒpƒbƒP□[ƒW,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□ó'Ô,Æƒvƒ□ƒpƒeƒB,ðŠÄŽ‹,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,¤

MTS ƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ",Ìƒvƒ□ƒpƒeƒB,ð□Ý'è,·,é

MTS "F□ØƒCƒxƒ‹,ð□Ý'è,·,é

[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,É,Í□AŽŸ,ÌƒTƒuƒtƒHƒ‹ƒ_,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

- [ƒCƒ"ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_
  ƒRƒ"ƒ|□[ƒlƒ"ƒg,ÉŠÖ˜A•t,¯,ç,ê,Ä,¢,éƒCƒ"ƒ^□[ƒtƒFƒCƒX,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B
- [ƒ□□[ƒ‹ ƒ□ƒ"ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_ (ƒRƒ"ƒ|□[ƒlƒ"ƒg)
  ƒRƒ"ƒ|□[ƒlƒ"ƒg,Ìƒ□□[ƒ‹,ÉŠÖ˜A•t,¯,ç,ê,Ä,¢,éƒ□□[ƒ‹,Æƒ†□[ƒU□[,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

ŽŸ,Ìƒ^ƒu,ðŽg,Á,ÄƒRƒ"ƒ|□[ƒlƒ"ƒg,Ìƒvƒ□ƒpƒeƒB,ð□\□¬,·,é,±,Æ,ª,Å,«,Ü,·□B

- ['S"Ê] ƒ^ƒu
- [ƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ"] ƒ^ƒu
- [ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu

ŠÖ˜A□€–Ú

MTS ƒAƒNƒeƒBƒu‰»,Ìƒvƒ□ƒpƒeƒB,ð□Ý'è,·,é

[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_

[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,É,Í□A'I'ð,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,ÉŠ„,è"–,Ä,ç,ê,Ä,¢,éƒ□□[ƒ‹,ªŠÜ,Ü,ê,Ä,¢,Ü,·□BMTS ,Å,Í□AƒpƒbƒP□[ƒW□AƒRƒ"ƒ| □[ƒlƒ"ƒg□A,Ü,½,ÍƒCƒ"ƒ^□[ƒtƒFƒCƒX,Ö,Ìƒ†□[ƒU□[,ÌƒAƒNƒZƒX,ð□§Œä,·,éƒ□□[ƒ‹,ð'è‹`,·,é,± ,Æ,ª,Å,«,Ü,·□BƒpƒbƒP□[ƒW,Ì [ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,É'Ç‰Á,·,éƒ□□[ƒ‹,Í□A,¢,¸,ê,à,»,ÌƒpƒbƒP□[ƒW"à,ÌƒRƒ"ƒ| □[ƒlƒ"ƒg,Ü,½,ÍƒCƒ"ƒ^□[ƒtƒFƒCƒX,Ì [ƒ□□[ƒ‹ ƒ□ƒ"ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_,É,à'Ç‰Á,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒ□□[ƒ‹ ƒŒƒxƒ‹,Å,Í□AŽŸ,Ì'€□ì,ðŽÀ□s,·,é,±,Æ,ª,Å,«,Ü,·□B

MTS ƒ□□[ƒ‹,ðƒ†□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

MTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é

□V,µ,¢ MTS ƒ□□[ƒ‹,Ì'Ç‰Á

[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,É,Í□AŽŸ,ÌƒTƒuƒtƒHƒ‹ƒ_,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

- [ƒ†□[ƒU□[] ƒtƒHƒ‹ƒ_

ŽŸ,Ìƒ^ƒu,ðŽg,Á,Äƒ□□[ƒ‹,Ìƒvƒ□ƒpƒeƒB,ð□\□¬,·,é,±,Æ,ª,Å,«,Ü,·□B

- ['S"Ê] ƒ^ƒu

ŠÖ˜A□€–Ú

MTS ƒT□[ƒo□[,Ì□\□¬□ASystem ƒpƒbƒP□[ƒW

[ƒCƒ“ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_

[ƒCƒ“ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_,É,Í□A‘I‘ð,³,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,É’è‹`,³,ê,Ä,¢,éƒCƒ“ƒ^□[ƒtƒFƒCƒX,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

[ƒCƒ“ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_,É,Í□AŽŸ,Ì 2 ,Â,ÌƒTƒuƒtƒHƒ‹ƒ_,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

- [ƒ□ƒ\ƒbƒh] ƒtƒHƒ‹ƒ_
- [ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_

ŽŸ,Ìƒ^ƒu,ðŽg,Á,ÄƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ìƒvƒ□ƒpƒefB,ð•\Ž¦,·,é,±,Æ,ª,Å,«,Ü,·□B

- [‘S”Ê] ƒ^ƒu
- [ƒvƒ□ƒLƒV] ƒ^ƒu

[‘S”Ê] ƒ^ƒu,ÉƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ì□à–¾,ð’Ç‰Á,·,é□ê□‡ˆÈŠO,Í□AƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ìƒvƒ□ƒpƒefB,ð□\□¬,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

MTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é

[ƒ□ƒ\ƒbƒh] ƒtƒHƒ‹ƒ_

[ƒ□ƒ\ƒbƒh] ƒtƒHƒ‹ƒ_,É,Í□A'I'ð,³,ê,Ä,¢,éƒCƒ"ƒ^□[ƒtƒFƒCƒX,Å'è‹`,³,ê,Ä,¢,éƒ□ƒ\ƒbƒh,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

ƒ□ƒ\ƒbƒh,Ìƒvƒ□ƒpƒeƒB,É,Í□AŽŸ,Ìƒ^ƒu,ª, ,è,Ü,·□B

- [‘S"Ê] ƒ^ƒu

[‘S"Ê] ƒ^ƒu,Éƒ□ƒ\ƒbƒh,Ì□à–¾,ð'Ç‰Á,·,é□ê□‡ˆÈŠO,Í□Aƒ□ƒ\ƒbƒh,ð□\□¬,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

[ƒCƒ"ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_

[ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_

[ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_,É,Í□Aƒ†□[ƒU□[,ªƒRƒ“ƒ|
□[ƒlƒ“ƒg,Ü,½,ÍƒCƒ“ƒ^□[ƒtƒFƒCƒX,ÉŠÖ˜A•t,¯,½ƒ□□[ƒ‹,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B,±,ê,ç,Ìƒ□□[ƒ‹,ðƒpƒbƒP□[ƒW,Ì [ƒ□□[ƒ‹]
ƒtƒHƒ‹ƒ_,©,ç [ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_,É'Ç‰Á,·,é,Æ□AƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ü,½,ÍƒRƒ“ƒ|
□[ƒlƒ“ƒg,ÉƒAƒNƒZƒX,Å,«,éƒ†□[ƒU□[,ð□§Œä,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv,ÌƒvƒOƒpƒeƒB,Í□A[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_
ƒŒƒxƒ‹,Å□Ý'è,µ,Ü,·□B,½,Æ,¦,Î□AƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,Ì□à–¾,ð"ü—Í,·,é,Æ□A,»,Ì□à–¾,Í□Aƒ□□[ƒ‹
ƒ□ƒ“ƒo□[ƒVƒbƒv ƒŒƒxƒ‹,Åƒ□□[ƒ‹,Ì□à–¾,Æ,µ,Ä•\Ž¦,³,ê,Ü,·□B

□éŒ¾,É,æ,éƒZƒLƒ…ƒŠƒeƒB,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—
LŒø,É,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜A□€–Ú

[ƒCƒ“ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_

[ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_

[ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_,É,Í□AŠÇ—□ŽÒ,ªƒ□□[ƒ‹,ÉŠÖ˜A•t,¯,é Windows NT ƒt□[ƒU□[,Ü,½,ÍƒOƒ‹□[ƒv,ªŠÜ,Ü,ê,Ä,¢,Ü,·□BŠeƒt□[ƒU□[,Í□AŠÇ—□ŽÒ,ªƒpƒbƒP□[ƒW,Ì [ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,É'Ç‰Á,·,é Windows NT ƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,ð•\,µ,Ü,·□B[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,É Windows NT ƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,Ü,½,ÍƒOƒ‹□[ƒv,ð'Ç‰Á,·,é,Æ□AƒpƒbƒP□[ƒW□AƒRƒ“ƒ|□[ƒlƒ“ƒg□A,¨,æ,ÑƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ö,ÌƒAƒNƒZƒX,ð□§Œä,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒt□[ƒU□[ƒŒƒxƒ‹,Å,Í□AŽŸ,Ì'€□ì,ðŽÀ□s,·,é,±,Æ,ª,Å,«,Ü,·□B

- ƒ□□[ƒ‹,É□V,µ,¢ƒt□[ƒU□[,âƒOƒ‹□[ƒv,ð'Ç‰Á,·,é□B
- ƒ□□[ƒ‹,©,çƒt□[ƒU□[,âƒOƒ‹□[ƒv,ð□í□œ,·,é□B

[ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_,ÉŠÖ˜A,·,é [ƒvƒ□ƒpƒefB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Í, ,è,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

MTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é□AMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

[ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_

[ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,É,Í□A•Ê,ìƒRƒ“ƒsƒ…□[ƒ^□ã,ÅƒŠƒ‚□[ƒg,©,çŽÀ□s,·,é,½,ß,É□A,±,ìƒRƒ“ƒsƒ…□[ƒ^,Éƒ□□[ƒlƒ‹,Å“o˜^,³,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B[ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðŽg—p,·,é,É,Í□A□\□¬,·,éƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,É MTS ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,é•K—v,ª, ,è,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒŠƒ‚□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,ðŽè“®,Å□\□¬,·,é□ê□‡,Í□AƒŠƒ‚□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,ªƒAƒNƒZƒX,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ð [ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,É’Ç‰Á,µ,Ü,·□B

ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□\□¬,·,é‘O,É□AƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŽÀ□s,·,é,·,×,Ä,ìƒT□[ƒo□[,ðŽ©•ª,Ì [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,É’Ç‰Á,µ,Ä,,¾,³,¢□B

MTS ,ðŽÀ□s,µ,Ä,¢,éƒŠƒ‚□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,ÌŽg,¢•û,Ì□Ú□×,É,Â,¢,Ä,Í□AŽŸ,ìƒgƒsƒbƒN,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

MTS ƒpƒbƒP□[ƒW,ìƒGƒNƒXƒ|□[ƒg

ƒŠƒ‚□[ƒg MTS ƒRƒ“ƒsƒ…□[ƒ^,ðŽg,¤

’□　Windows 95/98 □ã,ÅŽÀ□s,µ,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,É•Ê,ìƒRƒ“ƒsƒ…□[ƒ^□ã,ìƒNƒ‰ƒCƒAƒ“ƒg,©,çƒŠƒ‚□[ƒg,ÅƒAƒNƒZƒX,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B□Ú□×,É,Â,¢,Ä,Í□A□wŠÇ—□ŽÒƒKƒCƒh□x,Ì□uWindows 95/98 □ã,Ì MTS ƒGƒNƒXƒvƒ□□[ƒ‰□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜A□€–Ú

MTS ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□¶□¬,·,é

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE‚É‚Í□AŽŸ‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠÜ‚Þ□A±‚Ìƒ‰ƒ“ƒsƒ…□[ƒ^‚ªŽQ‰Á‚µ‚Ä‚¢‚éŒ»□Ý‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª•\Ž¦‚³‚ê‚Ü‚·□B

- ƒCƒ“ƒ_ƒEƒg‚Ì□ó‘Ô‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□B
- ƒRƒ“ƒsƒ…□[ƒ^‚Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX‚Ì [□Ú□×□Ý’è] ƒ^ƒu‚ÅŽw’è‚µ‚½ŽžŠÔ‚ÉŠY“–‚·‚éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□B

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE‚É•\Ž¦‚³‚êƒfƒAƒCƒRƒ“‚Ì□à–¾‚É‚Â‚¢‚Ä‚Í□A□uƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ƒAƒCƒRƒ“□v‚ðŽQ□Æ‚µ‚Ä‚,,¾‚³‚¢□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÇ—□‚ÌŠT—v‚É‚Â‚¢‚Ä‚Í□AŽŸ‚Ìƒgƒsƒbƒ‚N‚ðŽQ□Æ‚µ‚Ä‚,,¾‚³‚¢□B

- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚É‚Â‚¢‚Ä
- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó‘Ô‚É‚Â‚¢‚Ä

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE‚ðŽg‚Á‚Ä□AŽŸ‚Ì□ì‹Æ‚ð□s‚¤‚±‚Æ‚ª‚Å‚«‚Ü‚·□B

- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÄŽ‹
- Windows 95/98 ‚Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÄŽ‹
- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì‰ðŒˆ

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒ}ƒEƒ{ƒ^ƒ“‚ÅNƒŠƒbƒN‚µ□A[ƒvƒ□ƒpƒeƒB] ‚ðƒNƒŠƒbƒN‚µ‚Äƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒvƒ□ƒpƒeƒB‚ð•\Ž¦‚·‚é‚±‚Æ‚à‚Å‚«‚Ü‚·□B

ŠÖ˜A□€–Ú

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒAƒCƒRƒ“□AMS DTC ‚ÌŠÇ—□

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv] ƒEƒBƒ“ƒhƒE

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv] ƒEƒBƒ“ƒhƒE‚É‚Í□AƒRƒ“ƒsƒ…□[ƒ^‚ªŽQ‰Á‚µ‚Ä‚¢‚éƒgƒ‰ƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv□î•ñ‚ª•\Ž¦‚³‚ê‚Ü‚·□B“□Œv□î•ñ‚É‚Í□A—Ý□Ï'l‚ðŽ¦‚·‚à‚Ì‚Æ□AŒ»□Ý‚ÌƒpƒtƒH□[ƒ}ƒ“ƒX‚ð”½‰f‚·‚é‚à‚ª‚ ‚è‚Ü‚·□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÇ—□‚ÌŠT—v‚É‚Â‚¢‚Ä‚Í□AŽŸ‚ÌƒgƒsƒbƒN‚ðŽQ□Æ‚µ‚Ä‚,,¾‚³‚¢□B

- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚É‚Â‚¢‚Ä
- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó'Ô‚É‚Â‚¢‚Ä

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv] ƒEƒBƒ“ƒhƒE‚ðŽg‚Á‚Ä□AŽŸ‚Ì□ì‹Æ‚ð□s‚¤‚±‚Æ‚ª‚Å‚«‚Ü‚·□B

- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÄŽ‹
- Windows 95/98 ‚Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÄŽ‹
- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì‰ðŒˆ

Windows NT □ã‚Å MTS ƒGƒNƒXƒvƒ□□[ƒ‰ðŽg—p‚µ‚Ä‚¢‚ê□ê□‡‚Í□A1 ‚Â‚ÌƒT□[ƒo□[‚É‚Â‚¢‚Ä 1 ‚Â‚¾‚¯ [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv] ƒEƒBƒ“ƒhƒE‚ðŠJ‚±‚Æ‚ª‚Å‚«‚é‚±‚Æ‚É'□ˆÓ‚µ‚Ä‚,,¾‚³‚¢□B

[ƒJƒŒƒ“ƒg]

- [ƒAƒNƒeƒBƒu] „Ÿ 2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg ƒvƒ□ƒgƒRƒ‹‚ðŠ®—¹‚µ‚Ä‚¢‚È‚¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŒ»□Ý‚Ì□”‚ðŽ¦‚·‚µ‚Ü‚·□B
- [□Å'åƒAƒNƒeƒBƒu] „Ÿ Œ»□Ý‚Ì Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒffƒBƒl□[ƒ^ (MS DTC) ƒZƒbƒVƒ‡ƒ''†‚Ì"C^Ó‚ÌŽž"_‚Å‚ÌƒAƒNƒeƒBƒu ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□Å'å□”‚ðŽ¦‚µ‚Ü‚·□B
- [ƒCƒ“ ƒ_ƒEƒg] „Ÿ ƒ□□[ƒJƒ‹ ƒf□[ƒ^ƒx□[ƒX ƒT□[ƒo□[‚ÆƒRƒ~ƒbƒg ƒR□[ƒffƒBƒl□[ƒ^‚Æ‚ÌŠÔ‚Å'Ê□M□áŠQ‚ª"-□¶‚µ‚½‚½‚ß‚ÉƒRƒ~ƒbƒg‚Å‚«‚È‚¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŒ»□Ý‚Ì□”‚ðŽ¦‚µ‚Ü‚·□B

[□WŒv]

- [ƒRƒ~ƒbƒg] „Ÿ ƒRƒ~ƒbƒg‚³‚ê‚½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì—Ý□Ï□‡Œv□”‚ðŽ¦‚µ‚Ü‚·□B½‚¾µ□A<□§ (Žè"®‚Å‰ðŒˆ‚³‚ê‚½) ƒRƒ~ƒbƒg‚ÍŠÜ‚Ü‚ê‚Ü‚¹‚ñ□B
- [ƒAƒ{□[ƒg] „Ÿ ƒAƒ{□[ƒg‚³‚ê‚½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì—Ý□Ï□‡Œv□”‚ðŽ¦‚µ‚Ü‚·□B½‚¾µ□A<-□§ƒAƒ{□[ƒg‚ÍŠÜ‚Ü‚ê‚Ü‚¹‚ñ□B
- [‹□§ƒRƒ~ƒbƒg] „Ÿ Žè"®‚ÅƒRƒ~ƒbƒg‚³‚ê‚½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì—Ý□Ï□‡Œv□”‚ðŽ¦‚µ‚Ü‚·□B
- [‹□§ƒAƒ{□[ƒg] „Ÿ Žè"®‚ÅƒAƒ{□[ƒg‚³‚ê‚½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì—Ý□Ï□‡Œv□”‚ðŽ¦‚µ‚Ü‚·□B
- [□‡Œv] „Ÿ ‚·‚×‚Ä‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì—Ý□Ï□‡Œv□”‚ðŽ¦‚µ‚Ü‚·□B

[‰ž"šŽžŠÔ (ƒ~ƒŠ•b)]

‚±‚ÌƒOƒ‹□[ƒv‚É‚Í□Aƒgƒransƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□Å□¬□A•½‹Ï□A¨‚æ‚Ñ□Å'å‚Ì‰ž"šŽžŠÔ‚ªƒ~ƒŠ•b'P^Ê‚Å•\Ž¦‚³‚ê‚Ü‚·□B‰ž"šŽžŠÔ‚Æ‚Í□Aƒgƒransƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠJŽn‚©‚çƒRƒ~ƒbƒg‚Ü‚Å‚Ì□Š—vŽžŠÔ‚Å‚·□B

[MS DTC ŠJŽn/MS DTC 'âŽ~]

‚±‚ÌƒOƒ‹□[ƒv‚É‚Í□AŒ»□Ý‚Ì MS DTC ƒZƒbƒVƒ‡ƒ“‚ªŠJŽn‚³‚ê‚½ú•t‚ÆŽžž□□‚ª•\Ž¦‚³‚ê‚Ü‚·□BMS DTC ‚ªŠJŽn‚³‚ê‚È‚¢‚Æ“ú•t‚ÆŽžž□□‚Í•\Ž¦‚³‚ê‚Ü‚¹‚ñ□B

"□Œv□î•ñ‚É‚Í□A—Ý□Ï'l‚ðŽ¦‚·‚à‚Ì‚Æ□AŒ»□Ý‚ÌƒpƒtƒH□[ƒ}ƒ“ƒX‚ð”½‰f‚·‚é‚à‚ª‚ ‚è‚Ü‚·□B

MS DTC ƒT□[ƒrƒX‚ð'âŽ~‚·‚é‚Æ□A—Ý□Ï"□Œv□î•ñ‚Ì'l‚Í‚·‚×‚Ä 0 ‚ÉŠ‚ƒZƒbƒg‚³‚ê‚Ü‚·□B

ŠÖ˜A□€–Ú

MS DTC ,ÌŠÇ—

[ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ƒEƒBƒ“ƒhƒE

[ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ƒEƒBƒ“ƒhƒE,É,Í□AMicrosoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒƒƒBƒl□[ƒ^ (MS DTC) ,ª”-□s,µ,Ä,¢,éŒ»□Ý,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒƒƒBƒl□[ƒ^,Ìˆê——,ª•\Ž¦,³,ê,Ü,·□BƒgƒŒ□[ƒX,ð□s,¤ ,Æ□AƒXƒ^□[ƒgƒAƒbƒv,âƒVƒƒƒbƒgƒ‚ƒEƒ“,È,Ç□AMS DTC ,Ì,³,Ü,´,Ü,ÈƒAƒNƒefƒBƒrƒefB,ÌŒ»□Ý,Ì□ó‘Ô,ð•\Ž¦,µ,½,è□AƒffƒoƒbƒO□î•ñ,ÉŠî,Ã,¢,Ä□ö□Ý“I,È–â'è,ð'Ç□Õ,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□,ÌŠT—v,É,Â,¢,Ä,Í□AŽŸ,Ìƒgƒsƒbƒ‰N,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,É,Â,¢,Ä
- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì□ó‘Ô,É,Â,¢,Ä

[ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ƒEƒBƒ“ƒhƒE,ðŽg,Á,Ä□AŽŸ,Ì□ì‹Æ,ð□s,¤,±,Æ,ª,Å,«,Ü,·□B

- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹
- Windows 95/98 ,Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹

ƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒvƒ□[ƒpƒeƒB] ƒ_ƒCƒAƒ□[ƒO ƒ{ƒbƒNƒX,Ì [□Ú□×□Ý'è] ƒ^ƒu,Å□A[ƒgƒŒ□[ƒX] ,Ì,Â,Ü,Ý,ðŽg,Á,Ä□A,± ,ÌƒEƒBƒ“ƒhƒE,É•\Ž¦,³,ê,éƒgƒŒ□[ƒX,ÌƒŒƒxƒ‹,ðŽw'è,·,é,±,Æ,ª,Å,«,Ü,·□B

[□d'å,³]

ƒAƒCƒRƒ“	□à–¾
	ƒGƒ‰ƒ□[
	MS DTC ,Ì□Ä‹N“®,ª•K—v,È–â'è,ª”-□¶,µ,Ü,µ,½□B,½,Æ,¦,Î□Aƒ□ƒO ƒtƒ@ƒCƒ‹,Ì"j'¹,ªŒŸ□o,³,ê,½□ê□‡,Å,·□B
	Œx□□
	,Ü,à,È,^Ù□í,ª”□¶,·,é‰Â"\□«,ª, ,è,Ü,·□B
	□î•ñ
	ƒXƒ^□[ƒgƒAƒbƒv,âƒVƒƒƒbƒgƒ‚ƒEƒ“,È,Ç□A•p“x,Ì'á,¢ƒCƒxƒ“ƒg,ÉŠÖ,·,é□î•ñ,ª•\Ž¦,³,ê,Ü,·□B
	ƒgƒŒ□[ƒX
	□V,µ,¢ƒNƒ‰ƒCƒAƒ“ƒg□Ú'±□AƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,Ì"o˜^,È,Ç,ÌƒCƒxƒ“ƒg,ÉŠÖ,·,éƒffƒoƒbƒO□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

[ƒ\□[ƒX]

ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW,Ì"□sŒ³,ªŽŸ,Ì,æ,¤,É•\Ž¦,³,ê,Ü,·□B

- [SVC] „Ÿ MS DTC ƒT□[ƒrƒX,ªƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW,Ì"□sŒ³,Å,·□B
- [LOG] „Ÿ MS DTC ƒ□ƒO,ªƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW,Ì"□sŒ³,Å,·□B
- [CM] „Ÿ MS DTC ƒIƒbƒgƒ□□[ƒN□Ú'±ƒ}ƒl□[ƒWƒƒ,ªƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW,Ì"□sŒ³,Å,·□B

ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW,Æ Windows NT ƒCƒxƒ“ƒg ƒ□ƒO,Ìƒ□ƒbƒZ□[ƒW,É,Í□A"□sŒ³,Ìƒ^ƒO,ª•t,«,Ü,·□B

[ƒ□ƒbƒZ□[ƒW]

ƒ□ƒbƒZ□[ƒW,ª•\Ž¦,³,ê,Ü,·□B

ŠÖ˜A□€–Ú

MS DTC ,ÌŠÇ—□

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒAƒCƒRƒ“

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——] ƒEƒBƒ“ƒhƒE,É,ÍŽŸ,ÌƒAƒCƒRƒ“,ª•\Ž¦,³,ê,Ü,·□B

ƒAƒCƒRƒ“	□à–¾
	**ƒAƒNƒeƒBƒu** ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªŠJŽn,³,ê,Ü,µ,½□B
	**ƒAƒ{□[ƒg'†** ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒAƒ{□[ƒg,µ,Ä,¢,Ü,·□BMS DTC ,Í□A,·,×,Ä,ÌŠÖ˜A•”•ª,É□Aƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒAƒ{□[ƒg,µ,È,¯,ê,Î,È,ç,È,¢,± ,Æ,ð'Ê'm,µ,Ä,¢,Ü,·□B ,±,ÌŽž“_,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŒ‹‰Ê,ð•Ï□X,·,é,± ,Æ,Í,Å,«,Ü,¹,ñ□B
	**ƒAƒ{□[ƒg** ƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒAƒ{□[ƒg,³,ê,Ü,µ,½□B,·,×,Ä,ÌŠÖ˜A•”•ª,Í □A'Ê'm,ðŽó,¯,Ü,µ,½□BƒAƒ{□[ƒg,³,ê,½ƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——] ƒEƒBƒ“ƒhƒE,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê—— ,©,ç,·,®,É□í□œ,³,ê,Ü,·□B ,±,ÌŽž“_,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŒ‹‰Ê,ð•Ï□X,·,é,± ,Æ,Í,Å,«,Ü,¹,ñ□B
	**□€"õ'†** ƒNƒ‰ƒCƒAƒ"ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ªƒRƒ~ƒbƒg—v‹□,ð"-□s,µ,Ü,µ,½□BMS DTC ,Í□A,·,×,Ä,ÌŠÖ˜A•"•ª,©,ç□€"õŠ®—¹,Ì‰ž"š,ð□W,ß,Ä,¢ ,Ü,·□B
	**□€"õŠ®—¹** ,·,×,Ä,ÌŠÖ˜A•"•ª,ª□€"õŠ®—¹,Ì‰ž"š,ð•Ô,µ,Ü,µ,½□B
	**ƒCƒ" ƒ_ƒEƒg** ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª•Ê,Ì MS DTC ,É,æ,Á,Ä□€"õ,³,ê□A'²□®,³,ê,Ä,¨,è□A,»,Ì'²□®–ð,Ì MS DTC ,ªƒAƒNƒZƒX,Å,«,È,,È,Á,Ä,¢,Ü,·□BƒVƒXƒeƒ€ŠÇ— □ŽÒ,Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——] ƒEƒBƒ"ƒhƒE"à,Åƒ}ƒEƒX,Ì ‰Eƒ{ƒ^ƒ",ðƒNƒŠƒbƒN,µ□A[‰ðŒˆ] ,ðƒ| ƒCƒ"ƒg,µ,Ü,·□BŽŸ,É□A[ƒRƒ~ƒbƒg] ,Ü,½,Í [ƒAƒ{□[ƒg] ,ðƒNƒŠƒbƒN,·,é,Æ□A‹□§'I,Éƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒRƒ~ƒbƒg,Ü,½,ÍƒAƒ{□[ƒg,³,¹,é,± ,Æ,ª,Å,«,Ü,·□B‹□§'I,É□ˆ—□,ð□s,¤,Æ□A,»,Ìƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Í [‹□§ƒRƒ~ƒbƒg] ,Ü,½,Í [‹□§ƒAƒ{□[ƒg] ,Æ,µ,Ä•\Ž¦,³,ê,Ü,·□B **Œx□□** ƒCƒ" ƒ_ƒEƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŽè"®,Å‹-□§'I,É□ˆ—□,·,é'O,É□A•K,¸□u MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì ‰ðŒˆ□v,ð,¨"Ç,Ý,,¾,³,¢□B
	**‹□§ƒRƒ~ƒbƒg**

ŠÇ—□ŽÒ,ª□AƒCƒ“ ƒ_ƒEƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‹-
□§“I,ÉƒRƒ~ƒbƒg,µ,Ü,µ,½ (□uMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì
‰ðŒˆ□v,ðŽQ□Æ,µ,Ä,,¾,³,¢)□B

‹□§ƒAƒ{□[ƒg

ŠÇ—□ŽÒ,ª□AƒCƒ“ ƒ_ƒEƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‹-
□§“I,ÉƒAƒ{□[ƒg,µ,Ü,µ,½ (□uMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì
‰ðŒˆ□v,ðŽQ□Æ,µ,Ä,,¾,³,¢)□B

ƒRƒ~ƒbƒg’†

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª□³,µ,□€”õ,³,ê□AMS DTC
,ªŠÖ˜A•”•ª,É□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒRƒ~ƒbƒg,³,ê,½,±
,Æ,ð’Ê’m,µ,Ä,¢,Ü,·□BMS DTC
,Í□A,·,×,Ä,ÌŠÖ˜A•”•ª,ªƒRƒ~ƒbƒg—v‹□,ÌŽó□M
(,¨,æ,Ñƒ□ƒO,Ì‹L˜^) ,ðŠm”F,·,é,Ü,Å□A,»,Ìƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð□I—¹,µ,Ü,¹,ñ□B

,±,ÌŽž“_,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŒ‹‰Ê,ð•Ï□X,·,é,±
,Æ,Í,Å,«,Ü,¹,ñ□B

ƒAƒ{□[ƒg,ð’Ê’m,Å,«,Ü,¹,ñ

MS DTC ,Í□A,·,×,Ä,Ì□Ú‘±,³,ê,Ä,¢,éŠÖ˜A•”•ª,É□Aƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒAƒ{□[ƒg,³,ê,½,±
,Æ,ð’Ê’m,µ,Ü,µ,½□B’Ê’m,ðŽó,¯,Ä,¢,È,¢
,Ì,Í□AŒ»□ÝƒAƒNƒZƒX,Å,«,È,¢ŠÖ˜A•”•ª,¾,¯,Å,·□B

,±,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□ó‘Ô,Í□AMS DTC ,ª,¢,¸,ê,©,ÌƒŠƒ\
□[ƒX ƒ}ƒl□[ƒWƒƒ (IBM LU 6.2 ƒVƒXƒeƒ€,È,Ç) ,Éƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒAƒ{□[ƒg,³,ê,½,±
,Æ,ð’Ê’m,µ,È,¯,ê,Î,È,ç,È,¢,Ì,É□A,»,ÌƒVƒXƒeƒ€,Æ,Ì□Ú‘±
,ª□Ø’f,³,ê,Ä,¢,é,½,ß□A’Ê’m,Å,«,È,¢,Æ,«,É”□¶,µ,Ü,·□B

ƒVƒXƒeƒ€ŠÇ—□ŽÒ,Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——]
ƒEƒBƒ“ƒhƒE“à,Åƒ}ƒEƒX,Ì
‰ðEƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ□A[‰ðŒˆ] ,ðƒ|
ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[”jŠü] ,ðƒNƒŠƒbƒN,·,é,Æ□AMS DTC
,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‹□§“I,É”jŠü,³,¹,é,±,Æ,ª,Å,«,Ü,·□B

Œx□□ ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŽè“®,Å”jŠü,·,é‘O,É□A•K,¸□uMTS ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì‰ðŒˆ□v,ð,¨“Ç,Ý,,¾,³,¢□B

ƒRƒ~ƒbƒg,ð’Ê’m,Å,«,Ü,¹,ñ

MS DTC ,Í□A,·,×,Ä,Ì□Ú‘±,³,ê,Ä,¢,éŠÖ˜A•”•ª,É□Aƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒRƒ~ƒbƒg,³,ê,½,±
,Æ,ð’Ê’m,µ,Ü,µ,½□B’Ê’m,ðŽó,¯,Ä,¢,È,¢
,Ì,Í□AŒ»□ÝƒAƒNƒZƒX,Å,«,È,¢ŠÖ˜A•”•ª,¾,¯,Å,·□B

ƒVƒXƒeƒ€ŠÇ—□ŽÒ,Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——]
ƒEƒBƒ“ƒhƒE“à,Åƒ}ƒEƒX,Ì
‰ðEƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ□A[‰ðŒˆ] ,ðƒ|
ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[”jŠü] ,ðƒNƒŠƒbƒN,·,é,Æ□AMS DTC
,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‹□§“I,É”jŠü,³,¹,é,±,Æ,ª,Å,«,Ü,·□B

Œx□□ ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŽè“®,Å”jŠü,·,é‘O,É□A•K,¸□uMTS ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì‰ðŒˆ□v,ð,¨“Ç,Ý,,¾,³,¢□B

ƒRƒ~ƒbƒg

ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ªƒRƒ~ƒbƒg‚³‚ê‚A‚·‚×‚Ä‚ÌŠÖ˜A•”•ª‚ª Ê’m‚ðŽó‚¯Ü‚µ‚½‚BƒRƒ~ƒbƒg‚³‚ê‚½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Í‚A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì^ê——] ƒEƒBƒ“ƒhƒE‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì^ê——‚©‚ç‚·‚®‚É‚í‚œ‚³‚ê‚Ü‚·‚B

‚±‚Ìƒ_ƒCƒ‰ƒOƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ŒŠ‹‰Ê‚ð•Ï‚X‚·‚é‚±‚Æ‚Í‚Å‚«‚Ü‚¹‚ñB

ŠÖ˜A€–Ú

MS DTC ‚ÌŠŽÇ—

ƒRƒ“ƒsƒ…□[ƒ^‚Ìƒvƒ□ƒpƒeƒB

ƒRƒ“ƒsƒ…□[ƒ^‚Ìƒvƒ□ƒpƒeƒB‚Å‚Í□AƒRƒ“ƒsƒ…□[ƒ^‚ÉŠÖ‚·‚é‘S”Ê“I‚È□î•ñ‚ð□Ý’è‚µ□AMicrosoft Transaction Server ‚ªƒRƒ“ƒsƒ…□[ƒ^‚ð□X□V‚·‚é•û–@‚ð□§Œä‚µ□Ü‚·□B

- [‘S”Ê] ƒ^ƒu
- [ƒlƒtƒvƒVƒ‡ƒ“] ƒ^ƒu
- [□Ú□×□Ý’è] ƒ^ƒu

ŠÖ˜A□€–Ú

[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_

[‘S”Ê] ƒ^ƒu (ƒRƒ“ƒsƒ…□[ƒ^)

[‘S”Ê] ƒ^ƒu,Å,Í□AƒRƒ“ƒsƒ…□[ƒ^,Ì–¼‘O,Æ□à–¾,ð‘è‹`,µ,Ü,·□B

–¼‘O

ƒRƒ“ƒsƒ…□[ƒ^,Ì–¼‘O,ª•\Ž¦,³,ê,Ü,·□B

[□à–¾]

ƒRƒ“ƒsƒ…□[ƒ^,Ì□à–¾,ª•\Ž¦,³,ê,Ü,·□BƒRƒ“ƒsƒ…□[ƒ^,ÌŽ¯•Ê,ÆŠÇ—□,É–ð—§,Â□à–¾,ð“ü—Í,·,é,±,Æ,ª,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_

[ƒIƒvƒVƒ‡ƒ“] ƒ^ƒu (ƒRƒ“ƒsƒ…□[ƒ^)

[ƒIƒvƒVƒ‡ƒ“] ƒ^ƒu,Í□AƒRƒ“ƒsƒ…□[ƒ^,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ^ƒCƒ€ƒAƒEƒg ƒvƒ□ƒpƒeƒB,ÆƒCŒƒvƒŠƒP□[ƒVƒ‡ƒ“□î•ñ,ð□Ý'è,·,é,Æ,«,ÉŽg,¢,Ü,·□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ^ƒCƒ€ƒAƒEƒg'I,Í•b'PˆÊ,ÅŒv'ª,³,ê□A,±,ÌƒRƒ“ƒsƒ…□[ƒ^,ÅŠJŽn,³,ê,½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒAƒNƒeƒBƒu,È□ó'Ô,ðˆÛŽ□,Å,«,é□Å'åŒÀ,ÌŽžŠÔ,ðŽ¦,µ,Ü,·□BŽw'è,µ,½ŽžŠÔ,ð‰ß,¬,Ä,àƒAƒNƒeƒBƒu,É,È,Á,Ä,¢,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Í□AƒVƒXƒef€,ªŽ©“®“I,ÉƒAƒ{□[ƒg,µ,Ü,·□BŠù'è'I,Í 60 •b,Å,·□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ^ƒCƒ€ƒAƒEƒg,ð–³Œø,É,·,é,É,Í□A'I,ð 0 ,É□Ý'è,µ,Ü,·□B–³Œø,É,·,é,Æ□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ìƒffƒofbfOŽž,É•Ö—˜,Å,·□B

[ƒIƒvƒVƒ‡ƒ“] ƒ^ƒu,Ì [ƒCƒfƒvƒŠƒP□[ƒVƒ‡ƒ“] ,ðŽg,Á,Ä□AŽg—p,µ,Ä,¢,é MTS ƒRƒ“ƒsƒ… □[ƒ^,ÌƒfŒƒvƒŠƒP□[ƒVƒ‡ƒ“□î•ñ,ð“ü—Í,µ,Ü,·□BƒtƒF□[ƒ‹ƒ□[ƒo□[ƒTƒ|□[ƒg,ð— LŒø,É,·,é,É,Í□A[ƒCƒfƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì‹¤—LƒffBƒCƒNƒgƒŠ] ƒ{ƒbƒNƒX,É Microsoft Cluster Server (MSCS) ,Ì‰¼‘zƒT□[ƒo□[–¼,ð“ü—Í,µ,Ü,·□BWindows 95/98 ƒRƒ“ƒsƒ…□[ƒ^□ã,Å MTS ƒJƒ^ƒ□ƒO,ðƒCƒvƒŠƒP□[ƒg,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

ƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â"\ƒtƒ@ƒCƒ‹,ÉƒAƒNƒZƒX,³,¹,éƒRƒ“ƒsƒ…□[ƒ^,ðŽw'è,·,é,± ,Æ,à,Å,«,Ü,·□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â"\ƒtƒ@ƒCƒ‹,ð□¶□¬,·,é'O,É□AƒNƒ‰ƒCƒAƒ“ƒg,ªƒAƒNƒZƒX,·,é•¨—□ƒT□[ƒo□[,Ì–¼'O,ð [ƒŠƒ,□[ƒg ƒT□[ƒo□[–¼] ƒ{ƒbƒNƒX,É"ü—Í,µ,Ü,·□B,±,Ìƒ{ƒbƒNƒX,É‰½,à"ü—Í,µ,È,¢ ,Æ□AƒGƒNƒXƒ|□[ƒg,ð□s,Á,½ƒRƒ“ƒsƒ…□[ƒ^,Ì•¨—□ƒRƒ“ƒsƒ…□[ƒ^–¼,ªŽg,í,ê,Ü,·□B•¶Žš—ñ,Æ,µ,ÄƒŠƒ,□[ƒg ƒT□[ƒo□[,Ì–¼'O,ð"ü—Í,·,é,Æ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ª□¶□¬,·,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â"\ ƒtƒ@ƒCƒ‹,Í□A,»,ÌƒŠƒ,□[ƒg ƒT□[ƒo□[–¼,ðŽw,µ,Ü,·□B

ŠÖ˜A□€–Ú

MTS ,ìƒCƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽg,¤□AMTS ŽÀ□s‰Â"\ƒtƒ@ƒCƒ‹,ð□¶□¬,·,é□A[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_

[□Ú□×□Ý’è] ƒ^ƒu (ƒRƒ“ƒsƒ…□[ƒ^)

[□Ú□×□Ý’è] ƒ^ƒu,Í□AMicrosoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒl□[ƒ^ (MS DTC) ,Ìƒvƒ□ƒpƒeƒB,ð□\□¬,·,é,Æ,«,ÉŽg,¢,Ü,·□B,±,Ìƒ^ƒu,Ì□Ý’è,Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——]ƒEƒBƒ“ƒhƒE□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv] ƒEƒBƒ“ƒhƒE□A,¨,æ,Ñ [ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ƒEƒBƒ“ƒhƒE,É,¾,¯“K—p,³,ê,Ü,·□B

’□ [ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^,Ìƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [□Ú□×□Ý’è] ƒ^ƒu,©,çƒwƒ‹ƒv,ð•\Ž¦,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

[•\Ž¦]

- [□X□V•p“x] „Ÿ [’á] ,©,ç [□,] ,Ü,Å,Ì”Í^Í,ð,Â,Ü,Ý,Å□Ý’è,µ,Ü,·□B[’á] ,Å,Íƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒEƒBƒ“ƒhƒE,ª 20 •b,²,Æ,É□X□V,³,ê□A[□,] ,Å,Í 1 •b,²,Æ,É□X□V,³,ê,Ü,·□B□X□V,Ì•p“x,ð□ã,°,é,Æ□AŽÀ□s’†,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□ƒl□[ƒo□[ƒwƒbƒh,Í‘□‰Á,µ,Ü,·,ª□A□Å□V,Ì□î•ñ,ð•\Ž¦,Å,«,Ü,·□B
- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“•\Ž¦] „Ÿ [‹Œ] ,©,ç [□V + ‹Œ] ,Ü,Å,Ì”Í^Í,ð,Â,Ü,Ý,Å□Ý’è,µ,Ü,·□B[‹Œ] ,É□Ý’è,·,é,Æ 5 •ª^È□ãƒAƒNƒeƒBƒu,É,È,Á,Ä,¢,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,¾,¯,ª•\Ž¦,³,ê□A[□V + ‹Œ] ,É□Ý’è,·,é,Æ 1 •b^È□ãƒAƒNƒeƒBƒu,É,È,Á,Ä,¢,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª•\Ž¦,³,ê,Ü,·□B
- [ƒgƒŒ□[ƒX] „Ÿ [□ (□,‘¬ MS DTC)] ,©,ç [‘½ (’á‘¬ MS DTC)] ,Ü,Å,Ì”Í^Í,ð,Â,Ü,Ý,Å□Ý’è,µ,Ü,·□B□¶‘¤,Ì–Ú□·,è,©,ç□‡,É□AŽŸ,Ì□Ý’è,É‘Î‰ž,µ,Ü,·□B
  - ƒgƒŒ□[ƒX,ð‘—□M,µ,È,¢□B
  - ƒGƒ‰□[ƒgƒŒ□[ƒX,¾,¯,ð‘—□M,·,é□B
  - ƒGƒ‰□[ƒgƒŒ□[ƒX,ÆŒx□□ƒgƒŒ□[ƒX,ð‘—□M,·,é□B
  - ƒGƒ‰□[ƒgƒŒ□[ƒX□AŒx□□ƒgƒŒ□[ƒX□A,¨,æ,Ñ□î•ñƒgƒŒ□[ƒX,ð‘—□M,·,é (Šù’è,Ì□Ý’è)□B
  - ,·,×,Ä,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‘—□M,·,é□B

[ƒ□ƒO]

- [□ê□Š] „Ÿ ƒ□ƒO ƒtƒ@ƒCƒ‹,ðŠi”[,·,é□ê□Š,ðŽw’è,µ,Ü,·□B
- [—e—Ê] „Ÿ ƒ□ƒO ƒtƒ@ƒCƒ‹,Ì□Å‘å,ÌƒTƒCƒY,ðŽw’è,µ,Ü,·□B

[ƒ□ƒO,ÌƒŠƒZƒbƒg]

‰½,ç,©,Ì•Ï□X,ª‰Á,¦,ç,ê,½Œã,Åƒ□ƒO ƒtƒ@ƒCƒ‹,ð□X□V,·,é,Æ,«,ÉŽg,¢,Ü,·□B

Œx□□ –¢‰ðŒˆ,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŽÀ□s’†,Í□AMS DTC ,Ìƒ□ƒO ƒtƒ@ƒCƒ‹,ð□X□V,µ,È,¢,Å,¸,¾,³,¢□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□,ÌŠT—v,É,Â,¢,Ä,Í□AŽŸ,Ìƒgƒƒsƒbƒ‰N,ðŽQ□Æ,µ,Ä,¸,¾,³,¢□B

- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,É,Â,¢,Ä
- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì□ó‘Ô,É,Â,¢,Ä

MTS ƒGƒfƒXƒvƒ□□[ƒ‰,Å [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“□Œv] ƒEƒBƒ“ƒhƒE□A[ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ƒEƒBƒ“ƒhƒE□A,¨,æ,Ñ [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——] ƒEƒBƒ“ƒhƒE,ðŽg—p,·,é•û–@,É,Â,¢,Ä,Í□AŽŸ,Ìƒgƒƒsƒbƒ‰N,ðŽQ□Æ,µ,Ä,¸,¾,³,¢□B

- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹
- Windows 95/98 ,Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹

ŠÖ˜A□€–Ú

MS DTC ,ÌŠÇ—□

ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB

ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,Å,Í□AƒpƒbƒP□[ƒW,ÉƒAƒNƒZƒX,·,é•û–@,ð□§Œä,µ,Ü,·□B

- [‘S”Ê] ƒ^ƒu
- [ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu
- [□Ú□×□Ý’è] ƒ^ƒu
- [ID] ƒ^ƒu
- [ƒAƒNƒeƒBƒu‰»] ƒ^ƒu

□**d—v**　ƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,Á,Ä□A[ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu,Ü,½,Í [ID] ƒ^ƒu,Ìƒvƒ□ƒpƒeƒB,ð•Ï□X,·,é (,Ü,½,Íƒpƒbƒ P□[ƒW,ðƒVƒƒƒbƒgƒ_ƒEƒ“,·,é) ,±,Æ,Í,Å,«,Ü,¹,ñ□B

ƒ□□[ƒ‹,ÉŠî,Ã,ƒZƒLƒ…ƒŠƒeƒB,Í Windows NT ,¾,¯,ÅƒTƒ|□[ƒg,³,ê,Ä,¢,é,½,ß□AWindows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,Å,Íƒpƒbƒ P□[ƒW,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,É [ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu,ª, ,è,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_

[‘S”Ê] ƒ^ƒu (ƒpƒbƒP□[ƒW)

[‘S”Ê] ƒ^ƒu,É,Í□A‘I‘ð,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,ÉŠÖ,·,é‘S”Ê“I,È□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

–¼‘O

ƒpƒbƒP□[ƒW,Ì–¼‘O,ª•\Ž¦,³,ê,Ü,·□B

[□à–¾]

ƒpƒbƒP□[ƒW,Ì□à–¾,ª•\Ž¦,³,ê,Ü,·□BƒpƒbƒP□[ƒW,ÌŽ¯•Ê,ÆŠÇ—□,É–ð—§,Â□à–¾,ð“ü—Í,·,é,±,Æ,ª,Å,«,Ü,·□B

[ƒpƒbƒP□[ƒW ID]

ƒpƒbƒP□[ƒW ID ”Ô□†,ª•\Ž¦,³,ê,Ü,·□BƒpƒbƒP□[ƒW ID
”Ô□†,Í□AƒpƒbƒP□[ƒW,Ì□ì□¬Žž,É□¶□¬,³,ê,éˆê‘Ó,Ì”Ô□†,Å,·□BƒpƒbƒP□[ƒW ID ,ðŽg,Á,ÄƒfRƒ“ƒsƒ…□[ƒ^□ã,Ì“Á’è,Ìƒo□[ƒWƒ‡ƒ“,ìƒpƒbƒP□[ƒW,ðŽ¯•Ê,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_□AƒpƒbƒP□[ƒW,ÌƒvƒO□ƒpƒeƒB

[ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu (ƒpƒbƒP□[ƒW)

[ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu,É,Í□A'I'ð,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,ÉŠÖ,·,éƒZƒLƒ…ƒŠƒeƒB□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

["F□Ø,ÌŠm"F,ðŽÀ□s,·,é]

,±,ìƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒ",É,·,é,Æ□AƒpƒbƒP□[ƒW,ðŒÄ,Ñ□o,·,·,×,Ä,ÌƒNƒ‰ƒCƒAƒ"ƒg,ìƒZƒLƒ…ƒŠƒeƒB□î•ñ,ªƒ`ƒFƒbƒN,³,ê,Ü,·□BŠù'è,Ì□Ý'è,Å,Í□A"F□Ø,ÌŠm"F,ÍŽÀ□s,³,ê,Ü,¹,ñ□B

[ŒÄ,Ñ□o,µ,Ì"F□ØƒCŒƒxƒ‹]

ƒpƒbƒP□[ƒW,ðŒÄ,Ñ□o,·ƒNƒ‰ƒCƒAƒ"ƒg,Ì"F□ØƒCŒƒxƒ‹,ð□Ý'è,µ,Ü,·□B

ŠÖ˜A□€–Ú

[ƒCƒ"ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_□AƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB□AMTS "F□ØƒCŒƒxƒ‹,ð□Ý'è,·,é□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é

[□Ú□×□Ý’è] ƒ^ƒu (ƒpƒbƒP□[ƒW)

[□Ú□×□Ý’è]
ƒ^ƒu,Å,Í□AƒpƒbƒP□[ƒW,ÉŠÖ˜A•t,¯,ç,ê,½ƒT□[ƒo□[ƒvƒ□ƒZƒX,ð□í,ÉŽÀ□s,·,é,©□A,Ü,½,Íˆê’è,ÌŽžŠÔ,ª
‰ß,¬,½,çƒVƒƒƒbƒgƒ_ƒEƒ“,·,é,©,ð□Ý’è,µ,Ü,·□B

ƒpƒbƒP□[ƒW,ª”ñƒAƒNƒeƒBƒu,Ì,Ü,Ü^ê’è,ÌŽžŠÔ,ª
‰ß,¬,½,çŽ©“®“I,ÉƒVƒƒƒbƒgƒ_ƒEƒ“,³,¹,é□ê□‡,Í□A[^ê’èŽžŠÔƒAƒCƒhƒ‹,µ,½Œã,ÅƒVƒƒƒbƒgƒ_ƒEƒ“,·,é]
,ð‘I‘ð,µ□AƒT□[ƒo□[ƒvƒ□ƒZƒX,ªƒVƒƒƒbƒgƒ_ƒEƒ“,·,é,Ü,Å,ÌŽžŠÔ,ð□Ý’è,µ,Ü,·□B

ƒT□[ƒo□[ƒvƒ□ƒZƒX,ð□í,É—˜—p‰Â”\,È□ó‘Ô,É,·,é□ê□‡,Í□A[ƒAƒCƒhƒ‹□ó‘Ô,Ì,Æ,«,àŽÀ□s,ð‘±,¯,é] ,ð‘I‘ð,µ,Ä,-
,¾,³,¢□B

[“®□ì] ƒ□ƒjƒ…□[,Ì [ƒT□[ƒo□[ƒvƒ□ƒZƒX,ÌƒVƒƒƒbƒgƒ_ƒEƒ“] ƒRƒ}ƒ“ƒh,ðŽg,Á,Ä□A‘I‘ð,³,ê,Ä,¢,éƒRƒ“ƒsƒ…
□[ƒ^,ÅŽÀ□s,µ,Ä,¢,é,·,×,Ä,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,ðƒVƒƒƒbƒgƒ_ƒEƒ“,·,é,±,Æ,ª,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_

[ID] ƒ^ƒu (ƒpƒbƒP□[ƒW)

[ID] ƒ^ƒu,Í□AƒpƒbƒP□[ƒW,ÅŽÀ□s,µ,Ä,¢,é,·,×,Ä,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒ†□[ƒU□[ID ,ð□Ý’è,·,é,Æ,«,ÉŽg,¢ ,Ü,·□BŠù’è’l,Í ['Î˜bƒ†□[ƒU□[- Œ»□Ýƒ□ƒOƒlƒ“,µ,Ä,¢,éƒ†□[ƒU□[] ,Å,·□B,±,Ìƒ†□[ƒU□[,Í□AWindows NT Server ,ÌƒAƒJƒEƒ“ƒg,Éƒ□ƒOƒlƒ“,µ,½ƒ†□[ƒU□[,Å,·□B•Ê,Ìƒ†□[ƒU□[,ð'l'ð,·,é□ê□‡,Í□A[,±,Ìƒ†□[ƒU□[] ,ð'l'ð,µ□AƒAƒJƒEƒ“ƒg–¼,ÆƒpƒXƒ□□[ƒh,ðŽw’è,µ,Ü,·□B

□d—v ƒpƒbƒP□[ƒW ID ,ð□Ý’è,µ,½,Æ,«□AMTS ,Í“ü—Í,³,ê,½ƒpƒXƒ□□[ƒh,ðŠm”F,µ,Ü,·□B,½,¾,µ□AƒAƒJƒEƒ“ƒg,ÌƒpƒXƒ□□[ƒh,ð•Ï□X,µ,½,Æ,«,É MTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ÅƒpƒXƒ□□[ƒh,ð□X□V,µ,È,©,Á,½□ê□‡,Í□AƒpƒbƒP□[ƒW,ÍŽÀ□s,Å,«,Ü,¹,ñ□B•Ê,Ìƒ†□[ƒU□[,ÆƒpƒXƒ□□[ƒh,ðŽw’è,·,é□ê□‡□AƒpƒXƒ□□[ƒh,ÌŽw’èŽž,É,ÍƒpƒXƒ□□[ƒh,ÌŠm”F,ª□s,í,ê,Ü,·□B,½,¾,µ□AƒAƒJƒEƒ“ƒg,ÌƒpƒXƒ□□[ƒh,ð•Ï□X,µ,½□ê□‡,É MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒpƒXƒ□□[ƒh,ð□X□V,µ,È,¢,Æ□AƒpƒbƒP□[ƒW,ðŽÀ□s,Å,«,Ü,¹,ñ□B–³Œø,ÈƒpƒXƒ□□[ƒh,ðŽg,Á,ÄƒpƒbƒP□[ƒW,ðŽÀ□s,·,é,Æ□Aƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰□[,ª”¶,µ□Aƒ□ƒbƒZ□[ƒW,ªƒCƒxƒ“ƒg ƒ□ƒO,É‹L˜^,³,ê,Ü,·□B

[,±,Ìƒ†□[ƒU□[] ,ð'l'ð,µ,Äƒ†□[ƒU□[,ð□Ý’è,·,é,É,Í□A,»,Ìƒ†□[ƒU□[,Éƒ}ƒbƒv,³,ê,Ä,¢,éƒRƒ“ƒsƒ… □[ƒ^,Éƒ□ƒOƒlƒ“,µ,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

MTS ”F□ØƒCŒƒxƒ‹,ð□Ý’è,·,é□A[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_

[ƒAƒNƒeƒBƒu‰»] ƒ^ƒu (ƒpƒbƒP□[ƒW)

[ƒAƒNƒeƒBƒu‰»] ƒ^ƒu,Í□AƒpƒbƒP□[ƒW“à,ÅƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒAƒNƒeƒBƒu‰»,·,é•û–@,ð□Ý’è,·,é,Æ,«,ÉŽg,¢,Ü,·□B

ƒpƒbƒP□[ƒW,Í□AƒpƒbƒP□[ƒW,ðŒÄ,Ñ□o,µ,½ƒNƒ‰ƒCƒAƒ“ƒg,ÌƒvƒOƒ‰ƒZƒX“à,Å (ƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,Æ,µ,Ä) ŽÀ□s,·,é,æ,¤,É□Ý’è,·,é,©□A,Ü,½,ÍƒpƒbƒP□[ƒW□ê—p,Ìƒ□□[ƒJƒ‹ ƒvƒOƒ‰ƒZƒX“à,Å (ƒT□[ƒo□[ƒpƒbƒP□[ƒW,Æ,µ,Ä) ŽÀ□s,·,é,æ,¤,É□Ý’è,·,é,±,Æ,ª,Å,«,Ü,·□B

[ƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW]

ƒpƒbƒP□[ƒW,ðƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,Æ,µ,ÄŽÀ□s,·,é,É,Í□A,±,ÌƒIƒvƒVƒ‡ƒ“,ð‘I‘ð,µ,Ü,·□Bƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,Í□AƒpƒbƒP□[ƒW,Ì□ì□¬Œ³ƒNƒ‰ƒCƒAƒ“ƒg,ÌƒvƒOƒ‰ƒZƒX“à,ÅŽÀ□s,³,ê,Ü,·□B,±,ÌƒIƒvƒVƒ‡ƒ“,Í□AƒpƒbƒP□[ƒW,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê□A□\□¬,³,ê,Ä,¢,éƒRƒ“ƒsƒ…□[ƒ^□ã,ÌƒNƒ‰ƒCƒAƒ“ƒg,Å,¾,¯‘I‘ð,Å,«,Ü,·□Bƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì’Ç□Õ□Aƒ□□[ƒ‹,Ìƒ`ƒFƒbƒN□Aƒvƒ□ƒZƒX,Ì•ª—£,ð□s,¢,Ü,¹,ñ□B

[ƒT□[ƒo□[ƒpƒbƒP□[ƒW]

ƒpƒbƒP□[ƒW,ðƒT□[ƒo□[ƒpƒbƒP□[ƒW,Æ,µ,ÄŽÀ□s,·,é,É,Í□A,±,ÌƒIƒvƒVƒ‡ƒ“,ð‘I‘ð,µ,Ü,·□BƒT□[ƒo□[ƒpƒbƒP□[ƒW,Í□Aƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^□ã,ÌƒT□[ƒo□[ƒpƒbƒP□[ƒWŽ©‘Ì,Ìƒvƒ□ƒZƒX“à,ÅŽÀ□s,³,ê,Ü,·□BƒT□[ƒo□[ƒpƒbƒP□[ƒW,Í□Aƒ□□[ƒ‹,ÉŠî,Ã,-ƒZƒLƒ…ƒŠƒeƒB□AƒŠƒ\□[ƒX,Ì‹¤—L□Aƒvƒ□ƒZƒX,Ì•ª—£□A(ƒpƒbƒP□[ƒW,Ì’Ç□Õ,È,Ç,Ì) ƒvƒ□ƒZƒXŠÇ—□,ðƒTƒ|□[ƒg,µ,Ü,·□B

ŠÖ˜A□€–Ú

ƒfCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_□Aƒpƒbƒ P□[ƒW,Ìƒvƒ□ƒpƒeƒB

ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒvƒ□ƒpƒeƒB

ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒvƒ□ƒpƒeƒB‚Å‚Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌƒTƒ|□[ƒg‚ÆƒZƒLƒ…ƒŠƒeƒB‚Ì□Ý’è‚ð□§Œä‚µ‚Ü‚·□B['S"Ê]ƒ^ƒu‚ðŽg‚Á‚Ä□A–¼‘O□Aƒvƒ□ƒOƒ‰ƒ€ ID (ProgID)□A‚¨‚æ‚ÑƒNƒ‰ƒX ID (CLSID) ‚È‚Ç□AƒRƒ“ƒ|□[ƒlƒ“ƒg‚ðŽ¯•Ê‚·‚é‚½‚ß‚Ìƒvƒ□ƒpƒeƒB‚ð•\Ž¦‚·‚é‚±‚Æ‚à‚Å‚«‚Ü‚·□B

- ['S"Ê] ƒ^ƒu
- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“] ƒ^ƒu
- [ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu

ŠÖ˜A□€–Ú

[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB‚ð—LŒø‚É‚·‚é

[‘S”Ê] ƒ^ƒu (ƒRƒ“ƒ|□[ƒlƒ“ƒg)

[‘S”Ê] ƒ^ƒu,É,Í□A‘I‘ð,³,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉŠÖ,·,é‘S”Ê“I,È□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì Prog ID

ƒvƒ□ƒOƒ‰ƒ€ ID (ProgID) ,ª•\Ž¦,³,ê,Ü,·□B

[□à–¾]

ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□à–¾,ª•\Ž¦,³,ê,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŽ¯•Ê,ÆŠÇ—□,É–ð—§,Â□à–¾,ð“ü—Í,·,é,±,Æ,ª,Å,«,Ü,·□B

[DLL]

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒNƒ‰ƒX,ÆƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ì’è‹`,ðŠÜ,Þ DLL ,ÌƒpƒX,ª•\Ž¦,³,ê,Ü,·□B

[CLSID]

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒNƒ‰ƒX ID (CLSID) ,ª•\Ž¦,³,ê,Ü,·□BƒRƒ□[ƒh,Ì’†,ÅƒNƒ‰ƒX ID (CLSID) ,ðŽg,Á,Ä□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŽ¯•Ê,µ,½,è□AƒAƒNƒZƒX,µ,½,è,·,é,±,Æ,ª,Å,«,Ü,·□B

[ƒpƒbƒP□[ƒW]

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,Ì–¼‘O,ª•\Ž¦,³,ê,Ü,·□B

ŠÖ˜A□€–Ú

[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_□Aƒ‚ƒWƒ…ƒ[ƒ‹,Ìƒvƒ□ƒpƒefB

[ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu (ƒRƒ“ƒ|□[ƒlƒ“ƒg)

[ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu,Í□A‘I‘ð,³,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ìƒZƒLƒ…ƒŠƒeƒB,ð□\□¬,·,é,Æ,«,ÉŽg,¢,Ü,·□B□éŒ¾,É,æ,éƒZƒLƒ…ƒŠƒeƒB,É,Â,¢,Ä,Í□A□uMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

[”F□Ø,ÌŠm”F,ðŽÀ□s,·,é]

,±,ìƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒ“,É,·,é,Æ□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŒÄ,Ñ□o,·,·,×,Ä,ìƒNƒ‰ƒCƒAƒ“ƒg,ìƒZƒLƒ…ƒŠƒeƒB□î•ñ,ªƒ`ƒFƒbƒN,³,ê,Ü,·□B

ŠÖ˜A□€–Ú

[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ìƒvƒ□[ƒpƒeƒB

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“] ƒ^ƒu (ƒRƒ“ƒ|□[ƒlƒ“ƒg)

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“] ƒ^ƒu,Å,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ªƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒTƒ|□[ƒg,·,é•û–@,ðŽw’è,µ,Ü,·□B

- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª•K—v] „Ÿ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒlƒuƒWƒFƒNƒg,ðƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌƒXƒR□[ƒv“à,ÅŽÀ□s,µ,È,¯,ê,Î,È,ç,È,¢□ê□‡,É□A,±,ÌƒlƒvƒVƒ‡ƒ“,ð‘I‘ð,µ,Ü,·□B□V,µ,¢ƒlƒuƒWƒFƒNƒg,ª□ì□¬,³,ê,é,Æ□A,»,ÌƒlƒuƒWƒFƒNƒg ƒRƒ“ƒeƒLƒXƒg,ÍƒNƒ‰ƒCƒAƒ“ƒg,ÌƒRƒ“ƒeƒLƒXƒg,©,çƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŒp□³,µ,Ü,·□BƒNƒ‰ƒCƒAƒ“ƒg,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª,È,¢□ê□‡□AMTS ,ÍŽ©“®“I,ÉƒlƒuƒWƒFƒNƒg,Ì□V,µ,¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð□ì□¬,µ,Ü,·□B
- [□V,µ,¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª•K—v] „Ÿ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒlƒuƒWƒFƒNƒg,ðƒlƒuƒWƒFƒNƒgŽ©‘Ì,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ““à,ÅŽÀ□s,µ,È,¯,ê,Î,È,ç,È,¢□ê□‡,É□A,±,ÌƒlƒvƒVƒ‡ƒ“,ð‘I‘ð,µ,Ü,·□B□V,µ,¢ƒlƒuƒWƒFƒNƒg,ª□ì□¬,³,ê,é,Æ□AƒNƒ‰ƒCƒAƒ“ƒg,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª, ,é,© ,Ç,¤,©,ÉŠÖŒW,È,□AMTS ,Í "Ž©“®“I,É" ƒlƒuƒWƒFƒNƒg,Ì□V,µ,¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð□ì□¬,µ,Ü,·□B
- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒTƒ|□[ƒg,·,é] „Ÿ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒlƒuƒWƒFƒNƒg,ðƒNƒ‰ƒCƒAƒ“ƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌƒXƒR□[ƒv“à,ÅŽÀ□s,Å,«,é□ê□‡,É□A,±,ÌƒlƒvƒVƒ‡ƒ“,ð‘I‘ð,µ,Ü,·□B□V,µ,¢ƒlƒuƒWƒFƒNƒg,ª□ì□¬,³,ê,é,Æ□A,»,ÌƒlƒuƒWƒFƒNƒg ƒRƒ“ƒeƒLƒXƒg,ÍƒNƒ‰ƒCƒAƒ“ƒg,ÌƒRƒ“ƒeƒLƒXƒg,©,çƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŒp□³,µ,Ü,·□BƒNƒ‰ƒCƒAƒ“ƒg,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª,È,¢□ê□‡,Í□A□V,µ,¢ƒRƒ“ƒeƒLƒXƒg,àƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,È,µ,Å□ì□¬,³,ê,Ü,·□B
- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒTƒ|□[ƒg,µ,È,¢] „Ÿ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒlƒuƒWƒFƒNƒg,ðƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌƒXƒR□[ƒv“à,ÅŽÀ□s,µ,Ä,Í,È,ç,È,¢□ê□‡,É□A,±,ÌƒlƒvƒVƒ‡ƒ“,ð‘I‘ð,µ,Ü,·□B□V,µ,¢ƒlƒuƒWƒFƒNƒg,ª□ì□¬,³,ê,é,Æ□AƒNƒ‰ƒCƒAƒ“ƒg,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª, ,é,© ,Ç,¤,©,ÉŠÖŒW,È,□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì,È,¢ƒlƒuƒWƒFƒNƒg ƒRƒ“ƒeƒLƒXƒg,ª□ì□¬,³,ê,Ü,·□B

ŠÖ˜A□€–Ú

[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒvƒfƒpƒeƒB□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÇ—□

ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒvƒ□ƒpƒeƒB

ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒvƒ□ƒpƒeƒB‚Í□A[ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_[‚É’Ç‰Á‚³‚ê‚½ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ÉŠÖ˜A‚é□î•ñ‚ð•\Ž¦‚·‚é‚Æ‚«‚ÉŽg‚¢‚Ü‚·□B['S"Ê] ƒ^ƒu‚ÉƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ì□à–¾‚ð“ü—Í‚·‚é□ê□‡ˆÈŠO‚Í□AƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒvƒ□ƒpƒeƒB‚ð□\□¬‚·‚é‚±‚Æ‚Í‚Å‚«‚Ü‚¹‚ñ□B

['S"Ê] ƒ^ƒu

ŠÖ˜A□€–Ú

[ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_□AMTS ƒpƒbƒP□[ƒW‚Ì"z•z

[‘S”Ê] ƒ^ƒu (ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg)

[‘S”Ê] ƒ^ƒu,É,Í□A‘I‘ð,³,ê,Ä,¢,éƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉŠÖ,·,éŽ¯•Ê□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

–¼‘O

ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì–¼‘O,ª•\Ž¦,³,ê,Ü,·□B

[□à–¾]

ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□à–¾,ª•\Ž¦,³,ê,Ü,·□BƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŽ¯•Ê,ÆŠÇ—□,É–ð—§,Â□à–¾,ð“ü—Í,·,é,±,Æ,ª,Å,«,Ü,·□B

[CLSID]

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒfNƒ‰ƒX ID ,Å,·□B

[ŽÀ□sŠÂ‹«]

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒfCƒ“ƒXƒg□[ƒ‹Œ³,ÌƒRƒ“ƒsƒ…□[ƒ^–¼,ðŽ¯•Ê,µ,Ü,·□B

ŠÖ˜A□€–Ú

[ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_

ƒ□□[ƒ‹,ìƒvƒ□ƒpƒeƒB

ƒ□□[ƒ‹,ìƒvƒ□ƒpƒeƒB,Í□A“Á'è,ìƒ□□[ƒ‹,Ì–¼'O□A□à–¾□A,¨,æ,Ñƒ□□[ƒ‹ ID ,ð•\Ž¦,·,é,Æ,«,ÉŽg,¢,Ü,·□B

[‘S”Ê] ƒ^ƒu

ŠÖ˜A□€–Ú

[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é□AMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

[‘S”Ê] ƒ^ƒu (ƒ□□[ƒ‹)

[‘S”Ê] ƒ^ƒu,É,Í□A‘I‘ð,³,ê,Ä,¢,éƒ□□[ƒ‹,ÉŠÖ,·,é‘S”Ê“I,È□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

–¼‘O

ƒ□□[ƒ‹,Ì–¼‘O,ª•\Ž¦,³,ê,Ü,·□B

[□à–¾]

ƒ□□[ƒ‹,Ì□à–¾,ª•\Ž¦,³,ê,Ü,·□Bƒ□□[ƒ‹,ÌŽ¯•Ê,ÆŠÇ—□,É–ð—§,Â□à–¾,ð“ü—Í,·,é,±,Æ,ª,Å,«,Ü,·□B

[ƒ□□[ƒ‹ ID]

ƒ□□[ƒ‹,ð’Ç‰Á,·,é,Æ,«,É MTS ,ª□¶□¬,·,éƒ□□[ƒ‹ ID ”Ô□†,ª•\Ž¦,³,ê,Ü,·□B,±
,Ì”Ô□†,ðŽg,Á,Ä□A•Ê,Ìƒpƒbƒp□[ƒW,ÉŽw’è,µ,½“¯,¶–¼‘O,Ìƒ□□[ƒ‹,ð‹æ•Ê,·,é,±,Æ,ª,Å,«,Ü,·□B

[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_“à,ÌƒIƒuƒWƒFƒNƒg,Æƒ□□[ƒ‹ ƒŒƒxƒ‹,ÅŽÀ□s,·,é‘€□ì,ÌŠT—v,É,Â,¢,Ä,Í□A□u[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜A□€–Ú

[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_□Aƒ□□[ƒ‹,Ìƒvƒ□ƒpƒeƒB□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é□AMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

ƒ□□[ƒ‹ ƒ□ƒ“ƒo,ìƒvƒ□ƒpƒeƒB

ƒ□□[ƒ‹ ƒ□ƒ“ƒo,ìƒvƒ□ƒpƒeƒB,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ü,½,ÍƒCƒ“ƒ^□[ƒtƒFƒCƒX,É'Ç‰Á,³,ê,½ƒ□□[ƒ‹,ÉŠÖ,·,é□î•ñ,ð•\Ž¦,·,é,Æ,«,ÉŽg,¢,Ü,·□B

- [‘S”Ê] ƒ^ƒu

ŠÖ˜A□€–Ú

[ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é□AMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

[‘S”Ê] ƒ^ƒu (ƒ□□[ƒ‹ ƒ□ƒ“ƒo)

[‘S”Ê] ƒ^ƒu,É,Í□A‘I‘ð,³,ê,Ä,¢,éƒ□□[ƒ‹,ÉŠÖ,·,é‘S”Ê“I,È□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

–¼‘O

ƒ□□[ƒ‹,Ì–¼‘O,ª•\Ž¦,³,ê,Ü,·□B

[□à–¾]

ƒ□□[ƒ‹,Ì□à–¾,ª•\Ž¦,³,ê,Ü,·□B

[ƒ□□[ƒ‹ ID]

ƒ□□[ƒ‹,ð’Ç‰Á,·,é,Æ,«,É MTS ,ª□¶□¬,·,éƒ□□[ƒ‹ ID ”Ô□†,ª•\Ž¦,³,ê,Ü,·□B,±
,Ì”Ô□†,ðŽg,Á,Ä□A•Ê,Ìƒpƒbƒ□[ƒW,ÉŽw’è,µ,½“¯,¶–¼‘O,Ìƒ□□[ƒ‹,ð‹æ•Ê,·,é,±,Æ,ª,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

[ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_□Aƒ□□[ƒ‹,Ìƒvƒ□ƒpƒeƒB□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é□AMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ìƒvƒ□ƒpƒeƒB

ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ìƒvƒ□ƒpƒeƒB,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŒöŠJ,·,éƒCƒ“ƒ^□[ƒtƒFƒCƒX,ÉŠÖ,·,é□î•ñ,ð•\Ž¦,·,é,Æ,«,ÉŽg,¢,Ü,·□B

- [‘S”Ê] ƒ^ƒu
- [ƒvƒ□ƒfƒLƒVƒ] ƒ^ƒu

ŠÖ˜A□€–Ú

[ƒCƒ“ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_

[‘S”Ê] ƒ^ƒu (ƒCƒ“ƒ^□[ƒtƒFƒCƒX)

[‘S”Ê] ƒ^ƒu,É,Í□A‘I‘ð,³,ê,Ä,¢,éƒCƒ“ƒ^□[ƒtƒFƒCƒX,ÉŠÖ,·,é‘S”Ê“I,È□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

–¼‘O

ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ì–¼‘O,ª•\Ž¦,³,ê,Ü,·□B

[□à–¾]

ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ì□à–¾,ª•\Ž¦,³,ê,Ü,·□BƒCƒ“ƒ^□[ƒtƒFƒCƒX,ÌŽ¯•Ê,ÆŠÇ—□,É–ð—§,Â□à–¾,ð“ü—Í,·,é,±,Æ,ª,Å,«,Ü,·□B

[IID]

ƒCƒ“ƒ^□[ƒtƒFƒCƒX ID ”Ô□†,ª•\Ž¦,³,ê,Ü,·□B

ŠÖ˜A□€–Ú

[ƒCƒ“ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_□AƒCƒ“ƒ^□[ƒtƒFƒCƒX,ìƒvƒ□ƒpƒeƒB

[ƒvƒ□ƒLƒV] ƒ^ƒu (ƒCƒ“ƒ^□[ƒtƒFƒCƒX)

‘I‘ð‚³‚ê‚Ä‚¢‚éƒvƒ□ƒLƒV/ƒXƒ^ƒu‚ÉŠÖ‚·‚éŽ¯•Ê□î•ñ‚ª•\Ž¦‚³‚ê‚Ü‚·□B

[ƒvƒ□ƒLƒV/ƒXƒ^ƒu]

ƒvƒ□ƒLƒV/ƒXƒ^ƒu DLL ‚ÌƒNƒ‰ƒX ID (CLSID) ‚ÆƒtƒHƒ‹ƒ\–¼‚ª•\Ž¦‚³‚ê‚Ü‚·□B

[ƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ]

ƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ‚ÌƒNƒ‰ƒX ID ‚ÆƒtƒHƒ‹ƒ‚□ê□Š‚ª•\Ž¦‚³‚ê‚Ü‚·□B

ŠÖ˜A□€–Ú

[ƒCƒ“ƒ^□[ƒtƒFƒCƒX] ƒtƒHƒ‹ƒ_□A‚ƒCƒ“ƒ^□[ƒtƒFƒCƒX‚Ìƒvƒ□ƒpƒefB

ƒ□ƒ\ƒbƒh,Ìƒvƒ□ƒpƒeƒB

ƒ□ƒ\ƒbƒh,Ìƒvƒ□ƒpƒeƒB,Í□AƒCƒ“ƒ^□[ƒtƒFƒCƒX,ªŒöŠJ,·,éƒ□ƒ\ƒbƒh,ÉŠÖ,·,é□î•ñ,ð•\Ž¦,·,é,Æ,«,ÉŽg,¢,Ü,·□B

[‘S”Ê] ƒ^ƒu

ŠÖ˜A□€–Ú

[ƒ□ƒ\ƒbƒh] ƒtƒHƒ‹ƒ_

[‘S”Ê] ƒ^ƒu (ƒ□ƒ\ƒbƒh)

[‘S”Ê] ƒ^ƒu,É,Í□A‘I‘ð,³,ê,Ä,¢,éƒ□ƒ\ƒbƒh,ÉŠÖ,·,éŽ¯•Ê□î•ñ,ª•\Ž¦,³,ê,Ü,·□B

–¼‘O

ƒ□ƒ\ƒbƒh,Ì–¼‘O,ª•\Ž¦,³,ê,Ü,·□B

[□à–¾]

ƒ□ƒ\ƒbƒh,Ì□à–¾,ª•\Ž¦,³,ê,Ü,·□Bƒ□ƒ\ƒbƒh,ÌŽ¯•Ê,ÆŠÇ—□,É–ð—§,Â□à–¾,ð“ü—Í,·,é,±,Æ,ª,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

[ƒ□ƒ\ƒbƒh] ƒtƒHƒ‹ƒ_□Aƒ□ƒ\ƒbƒh,Ìƒvƒ□ƒpƒeƒB

MTS ƒpƒbƒP□[ƒW,Ì□ì□¬

ƒpƒbƒP□[ƒW,Ì□ì□¬,Í MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŠJ”-
‰ß’ö,Ì□Å□I’iŠK,Å□s,í,ê,Ü,·□BƒpƒbƒP□[ƒW,ð,Ç,¤□ÝŒv,·,é,©,É,æ,Á,Ä□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□\
□¬,Æƒvƒ□ƒpƒefB,ª‹K’è,³,ê,Ü,·□BMTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é,É,Í□Aƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO,Ì’mŽ¯,Í•K—
v, ,è,Ü,¹,ñ,ª□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì□ÝŒvŽd—l,ÆŽÀ‘•Žd—l,É,Â,¢,Ä,Í□n’m,µ,Ä,¨,¢,Ä,,¾,³,¢□B

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,É‘g,Ý□ž,Þ,±
,Æ,ª□ÝŒvŽž,É,Ç,Ì,æ,¤,È^Ó–¡,ðŽ□,Â,©,É,Â,¢,Ä—□‰ð,·,é,½,ß,É□A□wŠÇ—□ŽÒƒKƒCƒh□x,Ì,±
,ÌƒZƒNƒVƒ‡ƒ“,ð□wProgrammer's Guide□x,Æ•¹,¹,Ä“Ç,Þ,±,Æ,ð‹,,¨Š©,ß,µ,Ü,·□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒpƒbƒP□[ƒW,Ö,Ì‘g,Ý□ž,Ý,Í□AƒŠƒ\□[ƒX ƒv□[ƒŠƒ“ƒO□AƒAƒNƒeƒBƒu‰»,Ì□Ý’è□Aƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌƒTƒ|□[ƒg,È,Ç,ÌŠJ”,ÉŠÖ,·,éŒˆ’èŽ–□€
,ð‹K’è,µ,Ü,·□B,½,Æ,¦,Î□AƒpƒbƒP□[ƒW,ð□ì□¬,·,é,Æ,«,Í□AƒŠƒ\□[ƒX,ð‹¤—L,·,é•¡□”,ÌƒRƒ“ƒ|
□[ƒlƒ“ƒg,ð“¯,¶ƒpƒbƒP□[ƒW,É,Ü,Æ,ß,Ä,,¾,³,¢□B,Ü,½□AƒpƒbƒP□[ƒW“à,ÅƒRƒ“ƒ|□[ƒlƒ“ƒg,ª‹¤—L,·,éƒŠƒ\□[ƒX,ÌŽí—
Þ,ð□l—¶,µ□A“Á’è,Ìƒfƒ□[ƒ^ƒx□[ƒX,Ö,Ì□Ú‘±,Ì,æ,¤,É "□,‰¿,È" ƒŠƒ\□[ƒX,ð‹¤—L,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,Í 1
,Â,ÌƒpƒbƒP□[ƒW,É,Ü,Æ,ß,Ä,,¾,³,¢□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,É‘g,Ý□ž,Þ,Æ,«,ÉƒŠƒ\□[ƒX ƒv□[ƒŠƒ“ƒO,ð—˜—p,·,é,Æ□A,æ,èŒø—¦“I,È MTS
ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŠJ”,·,é,±,Æ,ª,Å,«,Ü,·□B□wProgrammer's Guide□x,ð□wŠÇ—
□ŽÒƒKƒCƒh□x,Æ•¹,¹,Ä“Ç,Þ,Æ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒpƒbƒP□[ƒW,ð□ì□¬,µ□A”z’u,·,é,½,ß,É□I—
¶,·,é•K—v,ª, ,éŠJ”□ã,Ì—¯^Ó“_,É,Â,¢,Ä□A,æ,è□[,—□‰ð,·,é,±,Æ,ª,Å,«,Ü,·□B□wProgrammer's
Guide□x,Ì“üŽè•û–@,É,Â,¢,Ä,Í□A□uŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒ□ƒ“ƒg,ÌƒCƒ“ƒXƒg□[ƒ‹□v,ðŽQ□Æ,µ,Ä,-
,¾,³,¢□B

,±,±,Å,Í□AŽŸ,ÌƒgƒsƒbƒN,É,Â,¢,Ä□à–¾,µ,Ü,·□B

‹ó,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

MTS ƒpƒbƒP□[ƒW,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð’Ç‰Á,·,é

MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,ÉƒCƒ“ƒ|□[ƒg,·,é

MTS ƒpƒbƒP□[ƒW,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é

ƒGƒNƒXƒ|□[ƒg—p,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

MTS ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒefB,ð□Ý’è,·,é

MTS ƒAƒNƒeƒBƒu‰»,Ìƒvƒ□ƒpƒefB,ð□Ý’è,·,é

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìƒvƒ□ƒpƒefB,ð□Ý’è,·,é

MTS ”F□ØƒCŒƒxƒ‹,ð□Ý’è,·,é

MTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é

‹ó,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg—p,·,é,Æ,«,Ì□Å□‰
,ÌƒXƒeƒbƒv,ÍƒpƒbƒP□[ƒW,Ì□ì□¬,Å,·□BƒpƒbƒP□[ƒW,Í"¯,¶ƒvƒ□ƒZƒX,ÅŽÀ□s,³,ê,éƒRƒ"ƒ|□[ƒlƒ"ƒg,Ì□W,Ü,è,Å□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄŠÈ'P,É□ì□¬,Å,«,Ü,·□B‹ó,ÌƒpƒbƒP□[ƒW,ð□ì□¬,µ□AŒã,©,çƒRƒ"ƒ|□[ƒlƒ"ƒg,ð'Ç ‰Á,·,é,©□A,Ü,½,ÍŠù□¬,ÌƒpƒbƒP□[ƒW,ðƒCƒ"ƒXƒg□[ƒ‹,·,é,±,Æ,ª,Å,«,Ü,·□BƒpƒbƒP□[ƒW,ð'Ç ‰Á,·,é,É,Í□AƒpƒbƒP□[ƒW ƒEƒBƒU□[ƒh,ðŽg,¤,©□A,Ü,½,ÍƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹ (.pak) ,ð Windows ,Ì ƒGƒNƒXƒvƒ□□[ƒ‰,©,çƒhƒ‰ƒbƒO,µ,Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ"ƒhƒE,Ì‰E'¤,Éƒhƒ□ƒbƒv,µ,Ü,·□B

ƒpƒbƒP□[ƒW,ÍƒT□[ƒo□[ƒRƒ"ƒsƒ…□[ƒ^,ÅŽÀ□s,µ,Ä,¢
,éƒT□[ƒo□[ƒvƒ□ƒZƒX,Ì‹«ŠE,ð'è‹`,µ,Ü,·□B,½,Æ,¦,Î□A"Ì"„,ÌƒRƒ"ƒ|□[ƒlƒ"ƒg,Æ□w"ü,ÌƒRƒ"ƒ|□[ƒlƒ"ƒg,ðˆÙ,È,é 2 ,Â,ÌƒpƒbƒP□[ƒW,É•ª,¯,é,Æ□A,±,ê,ç,Ì 2 ,Â,ÌƒRƒ"ƒ|□[ƒlƒ"ƒg,Í□Aƒvƒ□ƒZƒX•ª—£□«,ð•Û,¿,È,ª,ç•Ê□X,Ìƒvƒ□ƒZƒX,ÅŽÀ□s,³,ê,Ü,·□B,µ,½,ª,Á,Ä□Aˆê•û,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,ª (ƒAƒvƒŠƒP□[ƒVƒ‡ƒ"ƒ,Ì'v–½"I,ÈƒGƒ‰□[,È,Ç,É,æ,Á,Ä) "Ë'R□I—
¹,µ,Ä,à□A,à,¤ˆê•û,ÌƒpƒbƒP□[ƒW,Í•Ê,Ìƒvƒ□ƒZƒX,ÅŽÀ□s,ðŒp'±,Å,«,Ü,·□B

‹ó,ÌƒpƒbƒP□[ƒW,ð□ì□¬,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ"ƒhƒE,Ì□¶'¤,Å□AƒpƒbƒP□[ƒW,ð□ì□¬,·,éƒRƒ"ƒsƒ…□[ƒ^,ð'I'ð,µ,Ü,·□B
2. 'I'ð,µ,½ƒRƒ"ƒsƒ…□[ƒ^,Ì [ƒCƒ"ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
3. ŽŸ,Ì,¢,¸,ê,©,Ì•û–@,ÅƒpƒbƒP□[ƒW ƒEƒBƒU□[ƒh,ðŠJ,«,Ü,·□B
   - ["®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ"ƒg,µ□A[ƒpƒbƒP□[ƒW] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - [ƒCƒ"ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ",ÅƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ| ƒCƒ"ƒg,µ,Ü,·□BŽŸ,É□A[ƒpƒbƒP□[ƒW] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒc□[ƒ‹ ƒo□[,Ì [□V,µ,¢ƒIƒuƒWƒFƒNƒg,Ì□ì□¬] ƒ{ƒ^ƒ",ðƒNƒŠƒbƒN,µ,Ü,·□B
4. ƒpƒbƒP□[ƒW ƒEƒBƒU□[ƒh,ðŽg,Á,Ä□AŠù□¬,ÌƒpƒbƒP□[ƒW,ðƒCƒ"ƒXƒg□[ƒ‹,·,é,©□A,Ü,½,Í‹ó,ÌƒpƒbƒP□[ƒW,ð□ì□¬,µ,Ü,·□B‹ó,Ì ƒpƒbƒP□[ƒW,ð□ì□¬,·,é□ê‡□AƒpƒbƒP□[ƒW,ð‹@"\,³,¹,é,É,Í□AƒRƒ"ƒ|□[ƒlƒ"ƒg,Æƒ□□[ƒ‹,ð'Ç ‰Á,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B
5. [‹ó,ÌƒpƒbƒP□[ƒW,ð□ì□¬,·,é] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
6. □V,µ,¢ƒpƒbƒP□[ƒW,Ì–¼'O,ð"ü—Í,µ□A[ŽŸ,Ö] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
7. [ƒpƒbƒP□[ƒW ID ,Ì□Ý'è] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ÅƒpƒbƒP□[ƒW ID ,ðŽw'è,µ□A[Š®—¹] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

Šù'è,Ì□Ý'è,Å,Í□AƒpƒbƒP□[ƒW ID ,Æ,µ,Ä ['Î˜bƒt□[ƒU□[- Œ»□Ýƒ□ƒOƒlƒ",µ,Ä,¢,éƒt□[ƒU□[] ,ª'I'ð,³,ê,Ü,·□B'Î˜bƒt□[ƒU□[,Í□AƒpƒbƒP□[ƒW,ªŽÀ□s,³,ê,Ä,¢,éƒT□[ƒo□[ƒRƒ"ƒsƒ… □[ƒ^,Éƒ□ƒOƒlƒ",µ,½ƒt□[ƒU□[,Å,·□B•Ê,Ìƒt□[ƒU□[,ð'I'ð,·,é,É,Í□A[,±,Ìƒt□[ƒU□[] ,ð'I'ð,µ□A"Á'è,Ì Windows NT ƒt□[ƒU□[,ð"ü—Í,µ,Ü,·□B

ŠÖ˜A□€–Ú

MTS ƒpƒbƒP□[ƒW,ÉƒRƒ"ƒ|□[ƒlƒ"ƒg,ð'Ç‰Á,·,é□AMTS ƒRƒ"ƒ|□[ƒlƒ"ƒg,ðƒpƒbƒP□[ƒW,ÉƒCƒ"ƒ|□[ƒg,·,é□AƒGƒNƒXƒ|□[ƒg—p,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é, MTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é

MTS ƒpƒbƒP□[ƒW,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,·,é

MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg ƒlƒuƒWƒFƒNƒg ƒ‚ƒƒ‹ (COM) ,ÌŽd—l,É□€‹',µ,Ä□ì□¬,³,ê,é□Ä—˜—p‰Â"\ ,ÈƒR□[ƒh,Æƒf□[ƒ^,Å,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ÍƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìƒrƒ‰ƒ‚ƒ‹ƒX ƒ□ƒWƒbƒN,ð‹K'è,µ,Ü,·□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,É'Ç‰Á,·,é,É,Í□AŽŸ,Ì,¢,¸,ê,©,Ì'€□ì,ðŽÀ□s,µ,Ü,·□B

- ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒEƒBƒU□[ƒh,ðŽg,¤□B
- Šù'¶,ÌƒpƒbƒP□[ƒW,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ðˆÚ"®,·,é□B

ƒpƒbƒP□[ƒW,Ì [ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,·,é,É,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÅƒRƒ“ƒ|□[ƒlƒ“ƒg ƒEƒBƒU□[ƒh,ðŽg,¤,©□A,Ü,½,Í'Ç‰Á,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŠÜ,Þƒ_ƒCƒif~ƒbƒN ƒŠƒ"ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL) ,ð Windows NT ƒGƒNƒXƒvƒ□□[ƒ‰,©,çƒhƒ‰ƒbƒO,µ,ÄƒpƒbƒP□[ƒW,Éƒhƒfƒbƒv,µ,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg ƒEƒBƒU□[ƒh,ðŽg,¤□ê□‡,Í□A□V,µ,¢ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ"ƒXƒg□[ƒ‹,µ□A,»,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒVƒXƒef€ ƒŒƒWƒXƒgƒŠ,É'Ç‰Á,·,é,©□A,Ü,½,ÍŠù,É"o˜^,³,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ"ƒ|□[ƒg,µ,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg,Í‹ó,ÌƒpƒbƒP□[ƒW,Ü,½,ÍŠù'¶,ÌƒpƒbƒP□[ƒW,É'Ç‰Á,Å,«,Ü,·□B

•¡□",ÌƒpƒbƒP□[ƒW,ÉƒCƒ"ƒXƒg□[ƒ‹,Å,«,é•¡□",ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□A'Pˆê,Ì MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉŠÜ,ß,é,± ,Æ,ª,Å,«,Ü,·□B,Â,Ü,è□A"¯,¶ DLL ,ÉŠÜ,Ü,ê,Ä,¢,éˆÙ,È,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ð,Ü,Á,½,- •Ê□X,ÌƒpƒbƒP□[ƒW,É'g,Ý□ž,Þ,±,Æ,ª,Å,«,Ü,·□B

ƒpƒbƒP□[ƒW,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ"ƒhƒE,Ì□¶'¤,Å□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ"ƒXƒg□[ƒ‹,·,éƒRƒ"ƒsƒ…□[ƒ^,ð'I'ð,µ,Ü,·□B
2. [ƒCƒ"ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ðŠJ,«□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ"ƒXƒg□[ƒ‹,·,éƒpƒbƒP□[ƒW,ð'I'ð,µ,Ü,·□B
3. [ƒRƒ"ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
4. ŽŸ,Ì,¢,¸,ê,©,Ì•û–@,ÅƒRƒ“ƒ|□[ƒlƒ“ƒg ƒEƒBƒU□[ƒh,ðŠJ,«,Ü,·□B
   - ["®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ"ƒg,µ□A[ƒRƒ"ƒ|□[ƒlƒ“ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - [ƒRƒ"ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ",ÅƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ| ƒCƒ"ƒg,µ,Ü,·□BŽŸ,É□A[ƒRƒ"ƒ|□[ƒlƒ“ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒc□[ƒ ƒo□[,Ì [□V,µ,¢ƒlƒuƒWƒFƒNƒg,Ì□ì□¬] ƒ{ƒ^ƒ",ðƒNƒŠƒbƒN,µ,Ü,·□B
5. [□V,µ,¢ƒRƒ"ƒ|□[ƒlƒ“ƒg,ðƒCƒ"ƒXƒg□[ƒ‹,·,é] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
6. •\Ž¦,³,ê,½ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å [ƒtƒ@ƒCƒ‹,Ì'Ç‰Á] ,ðƒNƒŠƒbƒN,µ□AƒCƒ"ƒXƒg□[ƒ‹,·,éƒtƒ@ƒCƒ‹,ð'I'ð,µ,Ü,·□BƒCƒ"ƒXƒg□[ƒ‹,·,éƒRƒ"ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,é DLL ,ð'I'ð,µ,Ä,,¾,³,¢□BƒRƒ"ƒ|□[ƒlƒ“ƒg,ÉŠÖ•",Ìƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,âƒvƒ□ƒLƒV/ƒXƒ^ƒu DLL ,ª, ,é□ê□‡,Í□A,»,ê,ç,Ìƒtƒ@ƒCƒ‹,à'Ç‰Á,µ,Ü,·□BWindows NT ƒGƒNƒXƒvƒ□□[ƒ‰,Å□A[ƒtƒ@ƒCƒ‹,Ì•\Ž¦] ,Ì□Ý'è,ª [ŽŸ,ÌŽí—Þ,Ìƒtƒ@ƒCƒ‹,í‰B,·] ,Å,Í,È, [,·,×,Ä,Ìƒtƒ@ƒCƒ‹,ð•\Ž¦] ,É□Ý'è,³,ê,Ä,¢,é,±,Æ,ðŠm"F,µ,Ä,- ,¾,³,¢□B,±,Ìƒlƒvƒfƒ‡ƒ“,ª .dll Šg'£Žq,Ì•t,¢,½ƒtƒ@ƒCƒ‹,ð•\Ž¦,µ,È,¢□Ý'è,É,È,Á,Ä,¢ ,é,Æ□A[ƒCƒ"ƒXƒg□[ƒ‹,·,éƒtƒ@ƒCƒ‹,Ì'I'ð] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,É,ÍƒRƒ"ƒ|□[ƒlƒ“ƒg,ðŠÜ,Þ DLL ,ª•\ Ž¦,³,ê,Ü,¹,ñ□B,±,Ì□Ý'è,ð•Ï□X,µ,½□ê□‡,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ð□Ä‹N"®,·,é•K—v,ª, ,è,Ü,·□B
7. •\Ž¦,³,ê,½ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□A'Ç‰Á,·,éƒtƒ@ƒCƒ‹,ð'I'ð,µ□A[ŠJ,] ,ðƒNƒŠƒbƒN,µ,Ü,·□B[ƒtƒ@ƒCƒ‹,ÌŽí—Þ] ƒ{ƒbƒNƒX,Å–Ú"I,Ìƒtƒ@ƒCƒ‹,ÌŽí—Þ,ðƒNƒŠƒbƒN,·,é,Æ□A—˜—p‰Â"\,È,·,×,Ä,Ìƒtƒ@ƒCƒ‹□ADLL ,¾,¯□A,Ü,½,Íƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,¾,¯,ð•\Ž¦,·,é,±,Æ,ª,Å,«,Ü,·□B
8. ƒtƒ@ƒCƒ‹,ð'Ç‰Á,·,é,Æ□A[ƒRƒ"ƒ|□[ƒlƒ“ƒg,ÌƒCƒ"ƒXƒg□[ƒ‹] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,É□A'Ç ‰Á,µ,½ƒtƒ@ƒCƒ‹,ÆŠÖ˜A,·,éƒRƒ"ƒ|□[ƒlƒ“ƒg,ª•\Ž¦,³,ê,Ü,·□B[□Ú□×] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒ",É,·,é,Æ□AŒŸ□o,³,ê,½ƒtƒ@ƒCƒ‹,Ì"à—e,âƒRƒ"ƒ|□[ƒlƒ“ƒg,É,Â,¢,Ä,Ì□Ú□×,È□î•ñ,ª•\ Ž¦,³,ê,Ü,·□BMicrosoft Transaction Server ,ÌƒRƒ"ƒ|□[ƒlƒ“ƒg,É,Íƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,ª•K— v,Å,·□BƒRƒ"ƒ|□[ƒlƒ“ƒg,Ìƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,ªŒ©,Â,©,ç,È,¢□ê□‡□A^ê——,É,ÍƒRƒ"ƒ|□[ƒlƒ“ƒg,ª•\ Ž¦,³,ê,Ü,¹,ñ□B,È,¨□A[ƒCƒ"ƒXƒg□[ƒ‹,·,éƒtƒ@ƒCƒ‹] ƒ{ƒbƒNƒX,Ìˆê—— ,©,çƒtƒ@ƒCƒ‹,ð□í□œ,·,é,É,Í□A□í□œ,·,éƒtƒ@ƒCƒ‹,ð'I'ð,µ□A[ƒtƒ@ƒCƒ‹,Ì□í□œ] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
9. [Š®—¹] ,ðƒNƒŠƒbƒN,µ□AƒRƒ"ƒ|□[ƒlƒ“ƒg,ðƒCƒ"ƒXƒg□[ƒ‹,µ,Ü,·□BƒRƒ"ƒ| □[ƒlƒ“ƒg,ðƒCƒ"ƒXƒg□[ƒ‹,·,é,Æ□A,»,ÌƒRƒ"ƒ|□[ƒlƒ“ƒg,ÌƒCƒ"ƒ^ˆê,ÌƒfƒCƒX,Æƒ□ƒ\ƒbƒh,ðŒ©,é,±

,Æ,ª,Å,«,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ“ƒ|□[ƒg,µ,½□ê□‡,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÉƒCƒ“ƒ^□[ƒtƒFƒCƒX,Æƒ□ƒ\ƒbƒh,Í•\Ž¦,³,ê,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

‹ó,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é□AMTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,ÉƒCƒ“ƒ|□[ƒg,·,é□AMTS ƒpƒbƒP□[ƒW,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é□AƒGƒNƒXƒ|□[ƒg—p,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,ÉƒCƒ“ƒ|□[ƒg,·,é

COM (ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒlƒuƒWƒFƒNƒg ƒ‚ƒƒ‹) ƒRƒ“ƒ|□[ƒlƒ“ƒg,Æ,µ,ÄŠù,ÉƒRƒ“ƒsƒ…□[ƒ^,É“o˜^,³,ê,Ä,¢,é“Á'è,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□AMicrosoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒpƒbƒP□[ƒW,ÉƒCƒ“ƒ|□[ƒg,·,é,± ,Æ,ª,Å,«,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ“ƒ|□[ƒg,µ,½□ê□‡,Í□AƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ìƒvƒ□ƒpƒeƒB,ð□Ý'è,µ,½,è□AƒŠƒ‚□[ƒg ƒNƒ‰ƒCƒAƒ“ƒg,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,Ö,ÌƒAƒNƒZƒX,ð□\□¬,µ,½,è,·,é,Ì,É•K—v,ÈƒCƒ“ƒ^□[ƒtƒFƒCƒX,âƒ□ƒ\ ƒbƒh,Ì□î•ñ,ÍƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ü,¹,ñ□BƒRƒ“ƒ|□[ƒlƒ“ƒg,Í,Å,«,é,¾,¯ƒCƒ“ƒ|□[ƒg,µ,È,¢,ÅƒCƒ“ƒXƒg□[ƒ‹,µ,Ä,,¾,³,¢□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,ÉƒCƒ“ƒ|□[ƒg,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶'¤,Å□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ“ƒ|□[ƒg,·,éƒRƒ“ƒsƒ…□[ƒ^,ð'I'ð,µ,Ü,·□B
2. [ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ðŠJ,«□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ“ƒ|□[ƒg,·,éƒpƒbƒP□[ƒW,ð'I'ð,µ,Ü,·□B
3. [ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
4. ŽŸ,Ì,¢,¸,ê,©,Ì•û–@,ÅƒRƒ“ƒ|□[ƒlƒ“ƒg ƒEƒBƒU□[ƒh,ðŠJ,«,Ü,·□B
   - [“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ□A[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - [ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ| ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒRƒ“ƒ|□[ƒlƒ“ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒc□[ƒ‹ ƒo□[,Ì [□V,µ,¢ƒlƒuƒWƒFƒNƒg,Ì□ì□¬] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B
5. [Šù,É“o˜^,³,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ“ƒ|□[ƒg,·,é] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
6. ƒCƒ“ƒ|□[ƒg,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'I'ð,µ,Ü,·□B
7. [Š®—¹] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ŠÖ˜A□î•ñ

‹ó,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é□AMTS ƒpƒbƒP□[ƒW,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,·,é

MTS ƒpƒbƒP□[ƒW,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AƒpƒbƒP□[ƒW,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é,±,Æ,ª,Å,«,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,µ,½Œã,Ì□ó‘Ô,í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ð MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,É’Ç‰Á,µ,½•û–@,É,æ,Á,Äˆ Ù,È,è,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ“ƒXƒg□[ƒ‹,µ,½□ê□‡,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é,Æ MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÆƒRƒ“ƒsƒ…□[ƒ^,Ì—¼•û,©,çƒŒƒWƒXƒgƒŠ□î•ñ,ªŠ®‘S,É□í□œ,³,ê,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒCƒ“ƒ|□[ƒg,µ,½□ê□‡□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Í MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,©,ç□í□œ,³,ê,Ü,·,ªA”z’u□æ,ÌƒRƒ“ƒsƒ…□[ƒ^,É,Í COM (ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒlƒuƒWƒFƒNƒg ƒ‚ƒƒ‹) ƒRƒ“ƒ|□[ƒlƒ“ƒg,Æ,µ,Ä“o˜^,³,ê,½,Ü,ÜŽc,è,Ü,·□B,±,Ì□ê□‡,Í□A”z’u□æ,ÌƒRƒ“ƒsƒ…□[ƒ^,©,ç COM ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒŒƒWƒXƒgƒŠ ƒGƒ“ƒgƒŠ,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹,ðŽè“®,Å□í□œ,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,©,ç□í□œ,·,é,É,Í

1. ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì‰E‘¤,Å□A□í□œ,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŠÜ,Þƒpƒbƒ P□[ƒW,ð‘I‘ð,µ,Ü,·□B
2. [ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
3. □í□œ,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ð‘I‘ð,µ,Ü,·□B
4. ŽŸ,Ì,¢,¸,ê,©,Ì‘€□ì,ðŽÀ□s,µ,Ü,·□B
   - [“®□ì] ƒ□ƒjƒ…□[,Ì [□í□œ] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - □í□œ,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[□í□œ] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒc□[ƒ‹ ƒo□[,Ì [□í□œ] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B
5. •\Ž¦,³,ê,½ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å [,Í,¢] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ŠÖ˜A□î•ñ

‹ó,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

ƒGƒNƒXƒ|□[ƒg—p,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

ƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,é,É,Í□AMTS ,ªƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,Ü,½,ÍƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ… □[ƒ^,Ö,ÌƒpƒbƒP□[ƒW,Ì“o˜^,ðŽ©“®‰»,Å,«,é,æ,¤,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð“K□Ø,É□\ □¬,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B,µ,½,ª,Á,Ä□AƒpƒbƒP□[ƒW,ðì□¬,·,é,Æ,«,Í□AƒpƒbƒP□[ƒW,ð”z•z,·,é•û–@,ð□l—¶,µ,Ä,,¾,³,¢□B

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒpƒbƒP□[ƒW ƒGƒNƒXƒ|□[ƒg‹@”\,ðŽg,Á,Ä□AMTS ,ª“®□ì,µ,Ä,¢,é,Ù,©,ÌƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,ÉƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,é,± ,Æ,ª,Å,«,Ü,·□B,Ü,½□AWindows NT ,Ü,½,Í Windows 95/98 ,ª“®□ì,µ,Ä,¢,éƒŠƒ,□[ƒg,ÌƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ªƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,·,é,½,ß,Ì□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŽÀ□s‰Â”\ ƒtƒ@ƒCƒ‹,ð□¶□¬,·,é,±,Æ,à,Å,«,Ü,·□BƒpƒbƒP□[ƒW,ð□ì□¬,µ□A□\□¬,·,é,Æ,«,Í□AƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg,Ì•K —v□ðŒ□,ð□l—¶,µ,Ä,,¾,³,¢□B

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,Í□AŠù’è,Ì□Ý’è,Å,Í□AŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ª□¶□¬,³,ê,½ƒŠƒ,□[ƒg MTS ƒT□[ƒo□[,ÉƒAƒNƒZƒX,·,é,æ,¤,ÉƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ð□\□¬,µ,Ü,·□BƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ƒlƒvƒVƒ‡ƒ“] ƒ^ƒu,ð□\□¬,·,é,± ,Æ,É,æ,Á,Ä□AƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì□ê□Š,ð•Ï□X,Å,«,Ü,·□BŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□¶□¬,·,é‘O,É□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì [ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ƒAƒCƒRƒ“,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒpƒeƒB] ,ðƒNƒŠƒbƒN,µ,Ü,·□BŽŸ,É□A[ƒlƒvƒVƒ‡ƒ“] ƒ^ƒu,ðƒNƒŠƒbƒN,µ□AƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ… □[ƒ^,ªƒAƒNƒZƒX,·,éƒT□[ƒo□[,ÌƒRƒ“ƒsƒ…□[ƒ^–¼,ð [Šƒƒ,□[ƒg ƒT□[ƒo□[–¼] ƒ{ƒbƒNƒX,É“ü—Í,µ,Ü,·□B“ü—Í,·,éƒRƒ“ƒsƒ…□[ƒ^–¼,Í□AƒpƒbƒP□[ƒW,ðŽÀ□s,·,é MTS ƒT□[ƒo□[,Ì–¼‘O,É,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BŽŸ,É□A[OK] ,ðƒNƒŠƒbƒN,µ,Ü,·□BƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□¶□¬,·,é,Æ□A□¶□¬,³,ê,½ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,Í [ƒŠƒ,□[ƒg ƒT□[ƒo□[–¼] ƒ{ƒbƒNƒX,ÅŽw’è,µ,½ƒT□[ƒo□[,ÉƒAƒNƒZƒX,·,é,æ,¤,ÉƒNƒ‰ƒCƒAƒ“ƒg,ð□\□¬,µ,Ü,·□B,± ,ê,É,æ,è□A•¡□”,ÌƒNƒ‰ƒCƒAƒ“ƒg,ª“¯¶ƒpƒbƒP□[ƒW,ðŽÀ□s,·,é•¡□”,ÌƒRƒ“ƒsƒ…□[ƒ^,ðƒAƒNƒZƒX,·,é,æ,¤ ,É,µ,Ä□A□Ã“I,ÉƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì•‰‰×,Ìƒof‰ƒ“ƒX,ðŽæ,é,±,Æ,ª,Å,«,Ü,·□B

ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg,Ì•K—v□ðŒ□

ƒpƒbƒP□[ƒWŠ”ŽÒ□A,Ü,½,ÍƒpƒbƒP□[ƒW,ð”z’u,·,é□ã‹‰ƒVƒXƒef€ŠÇ—□ŽÒ,â□ã‹‰ Web ŠÇ—□ŽÒ,Í□AMTS ƒpƒbƒP□[ƒW,Ì□ì□¬,Æ”z’u,ð□s,¤,Æ,«,É□AŽŸ,Ì•K—v□ðŒ□,ð□‡Žç,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

- ƒNƒ‰ƒCƒAƒ“ƒg ƒCƒ“ƒXƒg□[ƒ‹ ƒt□[ƒefBƒŠƒefB,Í□AŒ¾Œê,ÉˆË‘¶,µ,È,¢ƒŠƒ\□[ƒX,ðŽg,Á,ÄƒRƒ“ƒ| □[ƒlƒ“ƒg,Ìƒo□[ƒWƒ‡ƒ“”Ô□†,ð”»’f,µ,Ü,·□BVisual C++ ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□ê□‡□AŒ¾Œê,ÉˆË‘¶,µ,éƒ\□[ƒX,Æ“¯—l,É□AŒ¾Œê,ÉˆË‘¶,µ,È,¢ƒŠƒ\□[ƒX,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒo□[ƒWƒ‡ƒ“”Ô□†,ð•Ï□X,·,é•K—v,ª, ,è,Ü,·□BŒ¾Œê,ÉˆË‘¶,µ,È,¢ƒŠƒ\□[ƒX,ð•Ï□X,µ,È,¢□ê□‡□AƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ ƒtƒ@ƒCƒ‹,Í□A,æ,è□V,µ,¢ƒo□[ƒWƒ‡ƒ“,ðƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŠúŒÀ,Ì□Ø,ê,½ƒo□[ƒWƒ‡ƒ“,Å□ã‘‘,«,·,é‰Â”\□«,ª, ,è, Ü,·□B
- ƒNƒ‰ƒCƒAƒ“ƒg□ê—pƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìƒf^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,©,ç•W□€ COM ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ì‹L□q,ð□í□œ,µ,Ü,·□B,½,Æ,¦,Î□AƒpƒbƒP□[ƒWŠ”ŽÒ,ÍƒCƒ“ƒ^□[ƒtƒFƒCƒX,ð Visual Basic ,ÅŽg—p,·,é,½,ß,É **IObjectSafety** ,È,Ç,ÌƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,Å’è‹`,µ,Ä,¢,é,± ,Æ,ª, ,è,Ü,·□BƒGƒNƒXƒ|□[ƒg,·,é‘O,ÉƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ì‹L□q,ð□í□œ,·,é,Æ□AƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ… □[ƒ^,Å,»,ÌƒCƒ“ƒ^□[ƒtƒFƒCƒX,ªs“K□Ø,É“o˜^,³,ê,½,è□A“o˜^,ð□í□œ,³,ê,½,è,·,é,±,Æ,ð– hŽ~,Å,«,Ü,·□BƒNƒ‰ƒCƒAƒ“ƒg□ê—pƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,©,ç•W□€ COM ƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ì‹L□q,ð□í□œ,µ,È,¢,Æ□A,±,ê,ç,Ì•W□€ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŽg— p,·,é,Ù,©,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,É□áŠQ,ª”□¶,·,é□ê□‡,ª, ,è,Ü,·□B
- ƒT□[ƒo□[ƒpƒbƒP□[ƒW“à,É, ,è□AƒNƒ‰ƒCƒAƒ“ƒg,ªŽg—p,µ,Ä,¢,é (ƒNƒ‰ƒX ID□AƒCƒ“ƒ^□[ƒtƒFƒCƒX ID□Aƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ ID ,ðŠÜ,Þ) ƒOƒ□□[ƒoƒ‹,ÉˆêˆÓŽ¯•ÊŽq (GUID) ,Ì,¢ ,¸,ê,©,ª•Ï□X,³,ê,½□ê□‡□AƒNƒ‰ƒCƒAƒ“ƒg ƒCƒ“ƒXƒg□[ƒ‹ŽÀ□s‰Â”\ ƒtƒ@ƒCƒ‹,ð□X□V,·,é,É,Í□AƒpƒbƒP□[ƒW,ð□ÄƒGƒNƒXƒ|□[ƒg,·,é•K—v,ª, ,è,Ü,·□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒNƒ‰ƒCƒAƒ“ƒg,Í□A□V,µ,¢ƒNƒ‰ƒCƒAƒ“ƒg ƒCƒ“ƒXƒg□[ƒ‹ŽÀ□s‰Â”\ ƒtƒ@ƒCƒ‹,ðŽÀ□s,·,é,Ü,ÅƒAƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,Å,«,Ü,¹,ñ□Bˆê•”,ÌŠJ”ƒc□[ƒ‹ □iMicrosoft® Visual Basic® ,È,Ç□j ,Å,Í□AŠJ”ŽÒ,É’Ê’m,¹,¸,É,±,ê,ç,Ì GUID ,ð•Ï□X,·,é‰Â”\□«,ª, ,é,±,Æ,É’ˆÓ,µ,Ä,-

,¾,³,¢□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä MTS ƒpƒbƒP□[ƒW,ð”z•z,·,é•û–@,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒpƒbƒP□[ƒW,Ì”z•z□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜A□€–Ú

‹ó,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é□AMTS ƒpƒbƒP□[ƒW,ÉƒfƒRƒ“ƒ|□[ƒlƒ“ƒg,ð’Ç‰Á,·,é□AMTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒpƒbƒP□[ƒW,ÉƒfƒCƒ“ƒ|□[ƒg,·,é

MTS ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é

ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,Í□AƒpƒbƒP□[ƒW,Ì□\□¬,ð‹K’è,µ,Ü,·□BƒpƒbƒP□[ƒW,Ì[ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ð•\Ž¦,·,é,É,Í□AMicrosoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,Å□AƒpƒbƒP□[ƒW,ðƒ}ƒEƒX,Ì ‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒpƒeƒB] ,ðƒNƒŠƒbƒN,·,é,©□A,Ü,½,ÍƒpƒbƒP□[ƒW,ð‘I‘ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ì [ƒvƒ□ƒpƒeƒB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ƒpƒbƒP□[ƒW,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,É,Í□AŽŸ,Ì 5 ,Â,Ìƒ^ƒu,ª, ,è,Ü,·□B

- **[‘S”Ê] ƒ^ƒu**

  ƒpƒbƒP□[ƒW,Ì–¼‘O,Æ□à–¾,ª•\Ž¦,³,ê,Ü,·□B

- **[ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu**

  ”F□Ø,ÌŠm”F,ð—LŒø,É,µ,Ü,·□BƒZƒLƒ…ƒŠƒeƒB,ÌŠù’è,Ì□Ý’è,Å,Í□A”F□Ø,ÌŠm”F,Í—LŒø,É,È,Á,Ä,¢ ,Ü,¹,ñ□B□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

- **[□Ú□×□Ý’è] ƒ^ƒu**

  ƒpƒbƒP□[ƒW,ÉŠÖ˜A•t,¯,ç,ê,Ä,¢,éƒT□[ƒo□[ƒvƒ□ƒZƒX,ð□í,ÉŽÀ□s,·,é,©□A,Ü,½,Íˆê’è,ÌŽžŠÔ,ªŒo ‰ß,µ,½,çƒVƒƒfbƒgƒ_ƒEƒ“,·,é,©,ðŽw’è,µ,Ü,·□B

- **[ID] ƒ^ƒu**

  ƒpƒbƒP□[ƒW,ÉƒAƒNƒZƒX,Å,«,éƒt□[ƒU□[,ð□Ý’è,µ,Ü,·□BŠù’è’l,Í [Î˜bƒt□[ƒU□[- Œ»□Ýƒ□ƒOƒlƒ“,µ,Ä,¢ ,éƒt□[ƒU□[] ,Å,·□B,±,Ì□Ý’è,Í□AŒ»□Ý Windows NT Server ,ÌƒAƒJƒEƒ“ƒg,Éƒ□ƒOƒlƒ“,µ,Ä,¢ ,éƒt□[ƒU□[,Å,·□B,Ù,©,Ìƒt□[ƒU□[,ð‘I‘ð,·,é□ê□‡,Í□A[,±,Ìƒt□[ƒU□[] ,ð‘I‘ð,µ□AƒAƒJƒEƒ“ƒg–¼,ÆƒpƒXƒ□□[ƒh,ðŽw’è,µ,Ü,·□B

- **[ƒAƒNƒeƒBƒu‰»] ƒ^ƒu**

  ƒpƒbƒP□[ƒW,ÆƒpƒbƒP□[ƒW,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒAƒNƒeƒBƒu‰»ƒŒƒxƒ‹,ð□Ý’è,µ,Ü,·□BƒAƒNƒeƒBƒu ‰»ƒŒƒxƒ‹,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ª□Ì□¬Œ³,Ìƒvƒ□ƒZƒX,ÅƒAƒNƒeƒBƒu‰»,³,ê,é [**ƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW**]□A,Ü,½,ÍƒpƒbƒP□[ƒW,ª□ê—p,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,ÅŽÀ□s,³,ê,é [**ƒT□[ƒo□[ƒpƒbƒP□[ƒW**],Ì,¢ ,¸,ê,©,É□Ý’è,Å,«,Ü,·□B

  □Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒAƒNƒeƒBƒu‰»,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒpƒbƒP□[ƒW,Ü,½,ÍƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒpƒeƒB,ð•Ï□X,·,é,É,Í□A[ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,í,È,¯,ê,Î,È,è,Ü,¹,ñ□B

[ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ð•\Ž¦,·,é,É,Í

1. □\□¬,·,éƒpƒbƒP□[ƒW,Ü,½,ÍƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒpƒeƒB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B,Ü,½,Í□A–Ú“I,ÌƒAƒCƒeƒ€,ð‘I‘ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ì [ƒvƒ□ƒpƒeƒB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
2. –Ú“I,Ìƒ^ƒu,ðƒNƒŠƒbƒN,µ,Ü,·□B
3. ƒvƒ□ƒpƒeƒB,Ì□Ý’è,ð□X□V,µ,Ü,·□B
4. [OK] ,ðƒNƒŠƒbƒN,µ,Ä□Ý’è,ð•Û‘¶,µ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,É–ß,è,Ü,·□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□Ý’è,ð□X□V,·,é

ƒvƒ□ƒWƒFƒNƒg,ð□ÄƒRƒ“ƒpƒCƒ‹,·,é,½,Ñ,É□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□Ý’è,ð□X□V,·,é,±,Æ,ª□d—v,Å,·□BƒRƒ“ƒ| □[ƒlƒ“ƒg,Ì□Ý’è,ð□X□V,·,é,Æ□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒŒƒWƒXƒgƒŠ□Ý’è,Ì□‘,«‘Ö,¦,ð–hŽ~,Å,«,Ü,·□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□Ý’è,ð□X□V,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Å□A□X□V,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠÜ,Ü,ê,éƒRƒ“ƒsƒ… □[ƒ^,ð‘I‘ð,µ,Ü,·□B
2. [“®□ì] ƒ□ƒjƒ…□[,Ì [,·,×,Ä,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□X□V] ,ðƒNƒŠƒbƒN,µ,Ü,·□BTransaction Server ,ª□X□V,³,ê□AƒVƒXƒefŒ ƒŒƒWƒXƒgƒŠ□AƒRƒ“ƒ|□[ƒlƒ“ƒg ,Ì CLSID□A,Ü,½,ÍƒCƒ“ƒ^□[ƒtƒFƒCƒX ID (IID) ,Ì•Ï□X,ª”½‰f,³,ê,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□X□V,·,é,É,Í□AƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,ÅƒRƒ“ƒsƒ… □[ƒ^,ð‘I‘ð,µ□AMTS ƒc□[ƒ‹ ƒo□[,Ì [□Å□V,Ì□î•ñ,É□X□V] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,·,é•û–@,à, ,è,Ü,·□B

   [,·,×,Ä,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□X□V] ƒRƒ}ƒ“ƒh,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠK‘w,ÅƒRƒ“ƒsƒ…

□[ƒ^,ð‘I‘ð,µ,½□ê□‡,É,Ì,ÝŽg—p,Å,«,Ü,·□B,±,ÌƒRƒ}ƒ“ƒh,Í□A‘I‘ð,³,ê,½ƒRƒ“ƒsƒ…□[ƒ^,É“K—p,³,ê,Ü,·□B

ŠÖ˜A□€–Ú

MTS ƒAƒNƒeƒBƒu‰»,Ìƒvƒ□ƒpƒefB,ð□Ý’è,·,é□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìƒvƒ□ƒpƒefB,ð□Ý’è,·,é□AMTS ”F□ØƒCƒxƒ‹,ð□Ý’è,·,é□AMTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é

MTS ƒAƒNƒeƒBƒu‰»‚Ìƒvƒ□ƒpƒeƒB‚ð□Ý’è‚·‚é

ƒAƒNƒeƒBƒu‰»‚Ìƒvƒ□ƒpƒeƒB‚Í□Aƒpƒbƒ□[ƒW‚ð□ÝŒv‚·‚é‚Æ‚«‚ÉŽw’è‚µ‚Ä‚-
‚¾‚³‚¢□Bƒpƒbƒ□[ƒW‚Í□AŒÄ‚Ñ□o‚µ‚½ƒNƒ‰ƒCƒAƒ“ƒg‚Ìƒvƒ□ƒZƒX“à‚ÅŽÀ□s‚³‚ê‚é‚æ‚¤‚É (ƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒ□[ƒW‚Æ‚µ‚Ä) □Ý’è‚·‚é‚©□A‚Ü‚½‚Í□ê—p‚ÌƒT□[ƒo□[ƒvƒ□ƒZƒX“à‚ÅŽÀ□s‚³‚ê‚é‚æ‚¤‚É (ƒT□[ƒo□[ƒpƒbƒ□[ƒW‚Æ‚µ‚Ä) □Ý’è‚·‚é‚±‚Æ‚ª‚Å‚«‚Ü‚·□B

[ƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒ□[ƒW]

ƒpƒbƒ□[ƒW,ðƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒ□[ƒW,Æ‚µ‚ÄŽ□s‚·‚é,É,Í□A‚±‚ÌƒvƒVƒ‡ƒ“‚ð‘I‘ð‚µ‚Ü‚·□Bƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒ□[ƒW,Í□Aƒpƒbƒ□[ƒW,Ì□ì□¬Œ³ƒNƒ‰ƒCƒAƒ“ƒg‚Ìƒvƒ□ƒZƒX“à‚ÅŽÀ□s‚³‚ê‚Ü‚·□B‚± ‚ÌƒVƒVƒ‡ƒ“‚Í□Aƒpƒbƒ□[ƒW,ªƒCƒ“ƒXƒg□[ƒ‹‚³‚ê□A□\□¬‚³‚ê‚Ä‚¢‚éƒRƒ“ƒsƒ…□[ƒ^□ã‚ÌƒNƒ ‰ƒCƒAƒ“ƒg‚Å‚¾‚¯‚Ü‚·□Bƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒ□[ƒW,Í□AƒRƒ“ƒ| □[ƒlƒ“ƒg‚Ì‚Ç□Õ□Aƒ□□[ƒ‹‚ƒ`ƒFƒbƒN□Aƒvƒ□ƒZƒX‚Ì•ª—£‚ð□s‚¢,Ü,¹,ñ□B

[ƒT□[ƒo□[ƒpƒbƒ□[ƒW]

ƒpƒbƒ□[ƒW,ðƒT□[ƒo□[ƒpƒbƒ□[ƒW,Æ‚µ‚ÄŽ□s‚·‚é,É,Í□A‚± ‚ÌƒVƒVƒ‡ƒ“‚Í□Aƒpƒbƒ□[ƒW,Í□Aƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ… □[ƒ^□ã‚ÌƒT□[ƒo□[ƒpƒbƒ□[ƒWŽ©‘Ì‚Ìƒvƒ□ƒZƒX“à‚ÅŽÀ□s‚³‚ê‚Ü‚·□BƒT□[ƒo□[ƒpƒbƒ□[ƒW,Í□Aƒ□□[ƒ‹‚ÉŠî‚Ã‚-ƒZƒLƒ…ƒŠƒeƒB□AƒŠƒ\□[ƒX‚Ì‹¤—L□Aƒvƒ□ƒZƒX‚Ì•ª—£□A(ƒpƒbƒ□[ƒW‚Ì□Õ□È□Ç‚Ì) ƒvƒ□ƒZƒXŠÇ—□,ðƒTƒ| □[ƒg‚µ‚Ü‚·□B

ƒpƒbƒ□[ƒW‚ÌƒAƒNƒeƒBƒu‰»‚Ìƒvƒ□ƒpƒeƒB‚ð□Ý’è‚·‚é‚Í

1. □\□¬‚·‚éƒpƒbƒ□[ƒW‚ð‘I‘ð‚µ‚Ü‚·□B
2. ŽŸ‚Ì‚¢‚¢‚¿‚¸‚ê‚©‚Å [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX‚Ì [ƒAƒNƒeƒBƒu‰»] ƒ^ƒu‚ð•\Ž¦‚µ‚Ü‚·□B
   - [“®□ì] ƒ□ƒjƒ…□[‚Ì [ƒvƒ□ƒpƒeƒB] ‚ðƒNƒŠƒbƒN‚µ□A[ƒAƒNƒeƒBƒu‰»] ƒ^ƒu‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□B
   - □\□¬‚·‚éƒpƒbƒ□[ƒW‚ðƒ}ƒEƒX‚Ì‰Eƒ{ƒ^ƒ“‚ÅƒNƒŠƒbƒN‚µ□A[ƒvƒ□ƒpƒeƒB] ‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□BŽŸ‚É [ƒAƒNƒeƒBƒu‰»] ƒ^ƒu‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□B
   - MTS ƒc□[ƒ‹ ƒo□[‚Ì [ƒvƒ□ƒpƒeƒB] ƒ{ƒ^ƒ“‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□BŽŸ‚É [ƒAƒNƒeƒBƒu‰»] ƒ^ƒu‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□B
3. “K□Ø‚ÈƒAƒNƒeƒBƒu‰»‚Ìƒvƒ□ƒpƒeƒB‚ðŽw’è‚µ‚Ü‚·□B
4. [OK] ‚ðƒNƒŠƒbƒN‚µ‚Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰‚É–ß‚è‚Ü‚·□B

ŠÖ˜A□€–Ú

MTS ƒpƒbƒ□[ƒW‚Ìƒvƒ□ƒpƒeƒB‚ð□Ý’è‚·‚é□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒvƒ□ƒpƒeƒB‚ð□Ý’è‚·‚é□AMTS ”F□ØƒŒƒxƒ‹‚ð□Ý’è‚·‚é□AMTS ƒpƒbƒ□[ƒW‚ð□ƒbƒN‚·‚é

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒvƒ□ƒpƒeƒB‚ð□Ý’è‚·‚é

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ªƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒTƒ|□[ƒg‚·‚é‚©‚Ç‚¤‚©‚Í□A□ÝŒvŽž‚Ì‚µ‚Ä‚¾‚¾‚³‚¢□BƒTƒ|□[ƒg‚·‚é□ê□‡‚Í□A"z'u‚ð□s‚¤Æ‚«‚É Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰□ðŽg‚Á‚ÄƒRƒ"ƒ□[ƒlƒ"ƒg‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒvƒ□ƒpƒeƒB‚ð□Ý’è‚µ‚Ä‚¾‚¾‚³‚¢□B[ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX‚Ì [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“] ƒ^ƒu‚Å‚Í□AƒRƒ"ƒ□[ƒlƒ"ƒg‚ªƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒTƒ|□[ƒg‚·‚é•û–@‚ðŽw'è‚µ‚Ü‚·□BƒRƒ"ƒ□[ƒlƒ"ƒg‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒvƒ□ƒpƒeƒB‚Æ‚µ‚Ä□AŽŸ‚Ì□Ý'è‚Ì‚¢‚¸‚ê‚©‚Ç‚¤‚©‚ð'I'ð‚Å‚«‚Ü‚·□B

- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª•K—v] ƒRƒ"ƒ□[ƒlƒ"ƒg‚Ìƒ†ƒUƒWƒFƒNƒg‚ðƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒ†ƒXƒR□[ƒv"à‚ÅŽÀ□s‚·‚é•K—v‚ª‚ ‚é□ê□‡‚É□A±‚Ìƒ†ƒvƒVƒ‡ƒ“‚ð'I'ð‚µ‚Ü‚·□B□V‚µ‚¢ƒ†ƒUƒWƒFƒNƒg‚ª□ì□¬‚³‚ê‚é‚Æ□A»‚Ìƒ†ƒUƒWƒFƒNƒg ƒRƒ"ƒefƒLƒXƒg‚Íƒ‰ƒ“ƒ^ƒCƒAƒ"ƒg‚Ìƒ†ƒUƒNƒVƒ‡ƒ“‚©‚ç‚©‚Ç‚¤‚©‚ð□Œp□³‚µ‚Ü‚·□BƒNƒ‰ƒCƒAƒ"ƒg‚Éƒgƒgƒƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□ê□‡□AMTS ‚Íž©"®"I‚ÉƒIƒuƒWƒFƒNƒg‚Ì□V‚µ‚¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð□ì□¬‚µ‚Ü‚·□B
- [□V‚µ‚¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª•K—v] ƒRƒ"ƒ□[ƒlƒ"ƒg‚Ìƒ†ƒUƒWƒFƒNƒg‚ðƒIƒuƒWƒFƒNƒgŽ©'Ì‚Íƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“"à‚ÅŽÀ□s‚·‚é•K—v‚ª‚ ‚é□ê□‡‚É□A±‚Ìƒ†ƒvƒVƒ‡ƒ“‚ð'I'ð‚µ‚Ü‚·□B□V‚µ‚¢ƒ†ƒUƒWƒFƒNƒg‚ª□ì□¬‚³‚ê‚é‚Æ□AƒNƒ‰ƒCƒAƒ"ƒg‚Éƒgƒgƒƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª‚ ‚é‚©‚Ç‚¤‚©‚ÉŠÖŒW‚È□AMTS ‚Íž©"®"I‚ÉƒIƒuƒWƒFƒNƒg‚Ì□V‚µ‚¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð□ì□¬‚µ‚Ü‚·□B
- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒTƒ|□[ƒg‚·‚é] ƒRƒ"ƒ□[ƒlƒ"ƒg‚Ìƒ†ƒUƒWƒFƒNƒg‚ðƒNƒ‰ƒCƒAƒ"ƒg‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒ†ƒXƒR□[ƒv"à‚ÅŽÀ□s‚Å‚«‚é□ê□‡‚É□A±‚Ìƒ†ƒvƒVƒ‡ƒ“‚ð'I'ð‚µ‚Ü‚·□B□V‚µ‚¢ƒ†ƒUƒWƒFƒNƒg‚ª□ì□¬‚³‚ê‚é‚Æ□A»‚Ìƒ†ƒUƒWƒFƒNƒg ƒRƒ"ƒefƒLƒXƒg‚Íƒ‰ƒ“ƒ^ƒCƒAƒ"ƒg‚Ìƒ†ƒUƒNƒVƒ‡ƒ“‚©‚ç‚©‚Ç‚¤‚©‚ÉŠÖŒW‚È□AMTS ‚Íž©"®"I‚ÉƒIƒuƒWƒFƒNƒg‚Ì□V‚µ‚¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð□ì□¬‚µ‚Ü‚·□B
- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒTƒ|□[ƒg‚µ‚È‚¢] ƒRƒ"ƒ□[ƒlƒ"ƒg‚Ìƒ†ƒUƒWƒFƒNƒg‚ðƒgƒgƒƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“"à‚ÅŽÀ□s‚µ‚È‚¢‚Æ□A±‚Ìƒ†ƒvƒVƒ‡ƒ“‚ð'I'ð‚µ‚Ü‚·□B□V‚µ‚¢ƒ†ƒUƒWƒFƒNƒg‚ª□ì□¬‚³‚ê‚é‚Æ□AƒNƒ‰ƒCƒAƒ"ƒg‚Éƒgƒgƒƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª‚ ‚é‚©‚Ç‚¤‚©‚ÉŠÖŒW‚È□Aƒgƒgƒƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□ì□¬‚³‚ê‚é‚Æ□AMTS ‚Íž©"®"I‚ÉƒIƒuƒWƒFƒNƒg‚Ì□V‚µ‚¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð□ì□¬‚µ‚Ü‚·□B

ƒRƒ"ƒ□[ƒlƒ"ƒg‚ÌƒCƒ^ƒ^ƒfƒX‚ª□ì□¬‚³‚ê‚é‚Æ‚Ñ‚É□AMTS ‚ÍƒRƒ"ƒ□[ƒlƒ"ƒg‚Ìƒgƒgƒƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“"'®□«‚ðƒ`ƒFƒbƒN‚µ□AƒCƒ"ƒfƒbƒNƒX‚ðƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“"à‚ÅŽÀ□s‚·‚é‚©‚Ç‚¤‚©‚ðŒˆ'è‚µ‚Ü‚·□B

‚·‚×‚Ä‚ÌƒRƒ"ƒ□[ƒlƒ"ƒg‚ªƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□ˆ—□,ðƒTƒ|□[ƒg‚·‚é‚©‚Ç‚¤‚©‚É□ÝŒv³‚ê‚Ä¢‚é‚¢‚È‚¢‚Å‚Í‚È‚¢‚Ü‚¹‚ñ□BƒRƒ"ƒ□[ƒlƒ"ƒg‚ªƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□ˆ—□,ðŽg—p‚·‚é‚æ‚¤‚É□ÝŒv³‚ê‚Ä¢‚é‚¢‚È‚¢‚Å‚Í‚È‚¢‚Ü‚¹‚ñ□A□AƒRƒ"ƒ□[ƒlƒ"ƒg‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“"'®□«‚ð•K,¸ [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒTƒ|□[ƒg,µ,È,¢] ‚É□Ý'è‚µ,Ä,¾,³,¢□B

ƒpƒbƒP□[ƒW"à‚ÌƒRƒ"ƒ□[ƒlƒ"ƒg‚ª "f□fbƒN" ‚³‚ê‚Ä¢‚é‚é□ê□‡‚Í□Aƒgƒgƒƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“"'®□«‚ð•Ï□X‚Å‚«‚Ü‚¹‚ñ□B□Ú□×‚É‚Â¢‚Ä□A□uMTS ƒpƒbƒP□[ƒW‚ðƒ□ƒbƒN‚·‚é□v‚ðŽQ□Æ‚µ‚Ä‚-‚¾‚³‚¢□B

ƒRƒ"ƒ□[ƒlƒ"ƒg‚Ìƒgƒgƒƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒvƒ□ƒpƒeƒB‚ð□Ý'è‚·‚é‚É‚Í

1. □\□¬‚·‚éƒRƒ"ƒ□[ƒlƒ"ƒg‚ð'I'ð‚µ‚Ü‚·□B
2. ŽŸ‚Ì‚¢‚¸‚ê‚©‚Ç‚¤‚©‚Å [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX‚Ì [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“] ƒ^ƒu‚ð•\Ž¦‚µ‚Ü‚·□B
   - ["®□ì] ƒ□ƒjƒ…□[‚Ì [ƒvƒ□ƒpƒeƒB] ‚ðƒNƒŠƒbƒN‚µ□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“] ƒ^ƒu‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□B
   - □\□¬‚·‚éƒRƒ"ƒ□[ƒlƒ"ƒg‚ðƒ}ƒEƒX‚ì‰Eƒ{ƒ^ƒ"‚ÅƒNƒŠƒbƒN‚µ□A[ƒvƒ□ƒpƒeƒB] ‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□BŽŸ‚É [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“] ƒ^ƒu‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□B
   - MTS ƒc□[ƒ‹ ƒo□[‚Ì [ƒvƒ□ƒpƒeƒB] ƒ{ƒ^ƒ"‚ðƒNƒŠƒbƒN‚µ□BŽŸ‚É [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“] ƒ^ƒu‚ðƒNƒŠƒbƒN‚µ□B
3. "K□Ø‚È‚ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“"'®□«‚ðƒIƒuƒWƒFƒNƒg‚ðƒNƒŠƒbƒN‚µ□B
4. [OK] ‚ðƒNƒŠƒbƒN‚µ□B

•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠT—v,¨,æ,Ñƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠÄž‹‚µ□AŠÇ—□,·,é•û–@‚É‚Â¢‚Ä□uMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÇ—□□v‚ðŽQ□Æ‚µ‚Ä‚¾‚¾‚³‚¢□B

ŠÖ˜A□€–Ú

MTS ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é□AMTS ƒAƒNƒeƒBƒu‰»,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é□AMTS ”F□ØƒŒƒxƒ‹,ð□Ý’è,·,é□AMTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é

MTS ”F□ØƒŒƒxƒ‹,ð□Ý’è,·,é

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì”F□ØƒŒƒxƒ‹,Í□AƒNƒ‰ƒCƒAƒ“ƒg,©,ç,Ì—v‹□,ð”F□Ø,·,é,½,ß,ÌƒZƒLƒ…ƒŠƒefB,ÌƒŒƒxƒ‹,ðŽ¦,µ,Ü,·□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒNƒ‰ƒCƒAƒ“ƒg,É”F□Ø,ð—v‹□,µ,È,¢□ê□‡,Í□AƒpƒbƒP□[ƒW,Ì[ƒvƒ□ƒpƒefB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ƒZƒLƒ…ƒŠƒefB] ƒ^ƒu,Å□A[”F□Ø,ÌŠm”F,ðŽÀ□s,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒIƒ“,É,µ,Ü,·□B

DCOM ”F□ØƒŒƒxƒ‹,É,Â,¢,ÄŠ®‘S,É—□‰ð,µ,Ä,¢,é□ê□‡ˆÈŠO,Í□AƒpƒbƒP□[ƒW,Ì”F□Ø,Ì□Ý’è,ð MTS ,ÌŠù’è,Ì□Ý’è,Å, ,é [ƒpƒPƒbƒg] ƒŒƒxƒ‹,Ì,Ü,Ü,É,µ,Ä,¨,,±,Æ,ð,¨Š©,ß,µ,Ü,·□B

DCOM ”F□Ø,Ì□Ý’è,É,Â,¢,ÄŽŸ,É□à–¾,µ,Ü,·□B

ƒŒƒxƒ‹	**□à–¾**
[,È,µ]	,±,ÌƒpƒbƒP□[ƒW,Æ•Ê,ÌƒpƒbƒP□[ƒW□A,Ü,½,ÍƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Æ,ÌŠÔ,Ì’Ê□M,Å,Í□AƒZƒLƒ…ƒŠƒefB ƒ`ƒFƒbƒN,Í□s,í,ê,Ü,¹,ñ□B
[□Ú‘±]	□Å□‰,Ì□Ú‘±Žž,É,¾,¯ƒZƒLƒ…ƒŠƒefB ƒ`ƒFƒbƒN,ª□s,í,ê,Ü,·□B
[ŒÄ,Ñ□o,µ]	□Ú‘±□ó‘Ô,ª‘±,¢,Ä,¢ ,éŠÔ,Í□A,·,×,Ä,ÌŒÄ,Ñ□o,µ,É‘Î,µ,ÄƒZƒLƒ…ƒŠƒefB ƒ`ƒFƒbƒN,ª□s,í,ê,Ü,·□B
[ƒpƒPƒbƒg]	‘—□MŒ³,Ì ID ,ª^Ã□†‰»,³,ê,Ü,·□B
[ƒpƒPƒbƒg,Ì□®□‡«]	”]‘—’†,ÉƒpƒPƒbƒg,ª•Ï□X,³,ê,Ä,¢,È,¢,± ,Æ,ðŠm”F,·,é,½,ß,É□A‘—□MŒ³,Ì ID ,Æ□□–¼,ª^Ã□† ‰»,³,ê,Ü,·□B
[ƒpƒPƒbƒg,Ìƒvƒ‰ƒCƒoƒV□[]	ƒZƒLƒ… ƒ§ƒefB,ð□Å‘åŒÀ,É,·,é,½,ß,É□Aƒf□[ƒ^□A,¨,æ,Ñ‘— □MŒ³,Ì ID ,Æ□□–¼,ðŠÜ,ÞƒpƒPƒbƒg‘S‘Ì,ª^Ã□† ‰»,³,ê,Ü,·□B

ƒRƒ“ƒsƒ…□[ƒ^,Ì”F□ØƒŒƒxƒ‹,ð□Ý’è,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å□A□\□¬,·,éƒpƒbƒP□[ƒW,ð‘I‘ð,µ,Ü,·□B
2. ŽŸ,Ì,¢,¸,ê,©,Ì•û–@,Å [ƒvƒ□ƒpƒefB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ƒZƒLƒ…ƒŠƒefB] ƒ^ƒu,ð•\Ž¦,µ,Ü,·□B
   - [“®□ì] ƒ□ƒjƒ…□[,Ì [ƒvƒ□ƒpƒefB] ,ðƒNƒŠƒbƒN,µ□A[ƒZƒLƒ…ƒŠƒefB] ƒ^ƒu,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - □\□¬,·,éƒpƒbƒP□[ƒW,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒpƒefB] ,ðƒNƒŠƒbƒN,µ,Ü,·□BŽŸ,É [ƒZƒLƒ…ƒŠƒefB] ƒ^ƒu,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒc□[ƒ‹ ƒo□[,Ì [ƒvƒ□ƒpƒefB] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□BŽŸ,É [ƒZƒLƒ…ƒŠƒefB] ƒ^ƒu,ðƒNƒŠƒbƒN,µ,Ü,·□B
3. [ŒÄ,Ñ□o,µ,Ì”F□ØƒŒƒxƒ‹] ƒ{ƒbƒNƒX,Å□A,±,ÌƒpƒbƒP□[ƒW,É□\□¬,·,é”F□ØƒŒƒxƒ‹,ð‘I‘ð,µ,Ü,·□B
4. [OK] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ŠÖ˜A□€–Ú

MTS ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒefB,ð□Ý’è,·,é□AMTS ƒAƒNƒefBƒu‰»,Ìƒvƒ□ƒpƒefB,ð□Ý’è,·,é□AMTS ƒgƒ‰ƒfƒUƒNƒVƒ‡ƒ“,Ìƒvƒ□ƒpƒefB,ð□Ý’è,·,é□AMTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é

MTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŠJ”□AƒCƒ“ƒXƒg□[ƒ‹□A,Ü,½,Í”z’u,ªŠ®—¹,µ,½,ç□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□\□¬,ª•Ï□X,³,ê,È,¢,æ,¤,É□ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é,±,Æ,ð□l—¶,µ,Ä,,¾,³,¢□BƒVƒXƒeƒ€ŠÇ—□ŽÒ,Ü,½,Í Web ŠÇ—□ŽÒ,ªŠÇ—□,µ,Ä,¢,é,Ù,©,Ì MTS ƒRƒ“ƒsƒ…□[ƒ^,ÉƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒGƒNƒXƒ|□[ƒg,·,é‘O,É□A,Ü,½,ÍƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŒÚ‹q,É”z•z,·,é‘O,É□A□ì□¬,µ,½ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒpƒbƒP□[ƒW,Ìƒ□ƒbƒN,Í□AŽŸ,Ì‘€□ì,É“K—p,Å,«,Ü,·□B

- •Ï□X

  ,±,Ì‘€□ì,ðƒ□ƒbƒN,·,é,Æ□AŠÇ—□ŽÒ,Íƒ□ƒbƒN,ð–³Œø,É,µ,È,¢,ÆƒpƒbƒP□[ƒW,Ì□\□¬,ð•Ï□X,Å,«,Ü,¹,ñ□B
- □í□œ

  ,±,Ì‘€□ì,ðƒ□ƒbƒN,·,é,Æ□AŠÇ—□ŽÒ,Íƒ□ƒbƒN,ð–³Œø,É,µ,È,¢,ÆƒpƒbƒP□[ƒW,ð□í□œ,Å,«,Ü,¹,ñ□B

ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é,É,Í

1. ƒpƒbƒP□[ƒW,Ì□\□¬,ªŠ®—¹,µ,½,ç□AƒpƒbƒP□[ƒW,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [□Ú□×□Ý’è] ƒ^ƒu,ðƒNƒŠƒbƒN,µ,Ü,·□B
2. [□Ú□×□Ý’è] ƒ^ƒu,Ì [ƒAƒNƒZƒXŒ] ,Å□Aƒ†□[ƒU□[,É,æ,éƒpƒbƒP□[ƒW,Ì□í□œ,ð‹ÖŽ~,·,é [□í□œ,ð–³Œø,É,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX□A,¨,æ,Ñƒ†□[ƒU□[,É,æ,éƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,Ì•Ï□X,ð‹ÖŽ~,·,é [•Ï□X,ð–³Œø,É,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,Ì—¼•û□A,Ü,½,Í,¢,¸,ê,©,ðƒlƒ“,É,µ,Ü,·□B

[□Ú□×□Ý’è] ƒ^ƒu,Å□Aƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒt,É,·,é,Æ□AƒpƒbƒP□[ƒW,Ìƒ□ƒbƒN,ð‰ð□œ,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

MTS ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é□AMTS ƒAƒNƒeƒBƒu‰»,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é□AMTS ƒgƒ‰ƒf“ƒUƒNƒVƒ‡ƒ“,Ìƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é□AMTS ”F□ØƒŒƒxƒ‹,ð□Ý’è,·,é

MTS ƒpƒbƒP□[ƒW,Ì”z•z

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AMTS ,ðŽÀ□s,µ,Ä,¢,éƒNƒ‰ƒCƒAƒ“ƒg,É,à□AŽÀ□s,µ,Ä,¢,È,¢ƒNƒ‰ƒCƒAƒ“ƒg,É,àƒpƒbƒP□[ƒW,ð”z•z,Å,«,Ü,·□BƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^‚ÆƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^‚Ì—¼•û‚Å MTS ‚ðŽÀ□s‚µ‚Ä‚¢‚é□ê□‡‚Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ‚ðŽg‚Á‚Ä□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹‚Ì□ì¬‚â [ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_‚ð—˜—p‚µ‚½ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒvƒbƒ…‚Ü‚½‚Íƒvƒ‹‚ð□s‚¤‚±‚Æ‚ª‚Å‚«‚Ü‚·□BƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^‚Å MTS ‚ðŽÀ□s‚µ‚Ä‚¢‚È‚¢□ê□‡‚Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg‚Á‚Ä□AƒAƒNƒ‰ƒCƒAƒ“ƒg,©‚ç DCOM Œo—R‚Åƒ‚□[ƒg MTS ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ÉƒAƒNƒZƒX‚Å‚«‚é‚æ‚¤‚ÉƒNƒ‰ƒCƒAƒ“ƒg,ðŽ©“®“I‚ÉƒCƒ“ƒXƒg□[ƒ‹‚µ□A□\□¬‚·‚éƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹‚ð□¶□¬‚µ‚Ü‚·□B

‚±‚±‚Å‚Í□AŽŸ‚Ìƒgƒsƒbƒ‚É‚Â‚¢‚Ä‚à–¾‚µ‚Ü‚·□B

ƒŠƒ□[ƒg MTS ƒRƒ“ƒsƒ…□[ƒ^‚ðŽg‚¤

MTS ƒpƒbƒP□[ƒW‚ÌƒGƒNƒXƒ□[ƒg

MTS ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹‚ð□¶□¬‚·‚é

ŠÖ˜A□€–Ú

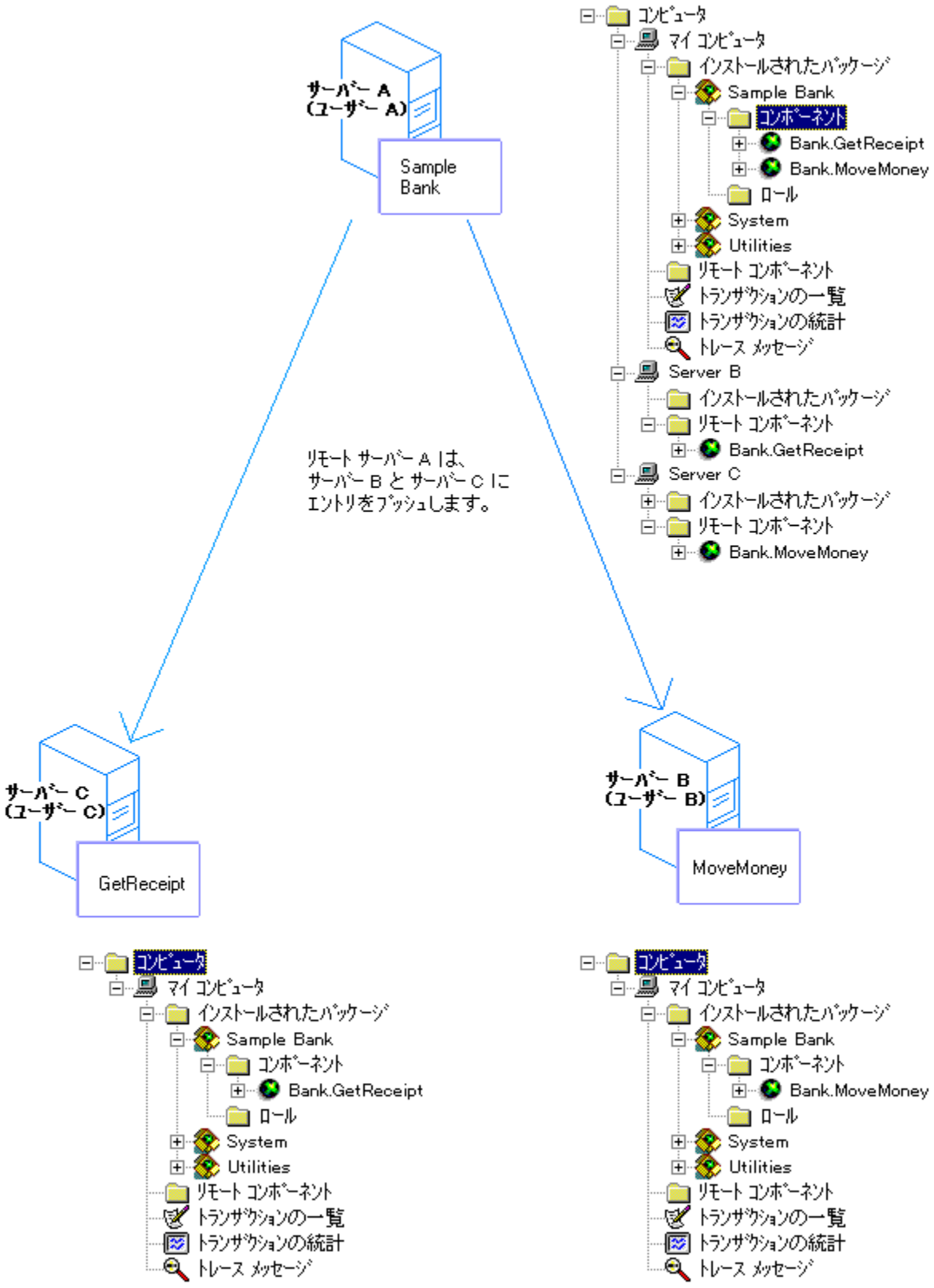
[ƒŠƒ□[ƒg ƒRƒ“ƒ□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_

ƒŠƒ‚□[ƒg MTS ƒRƒ“ƒsƒ…□[ƒ^‚ðŽg‚¤

ƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^‚ÆƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^‚Ì—¼•û‚Å MTS ‚ðŽÀ□s‚µ‚Ä‚¢
‚é□ê□‡‚Í□A•¡□”‚ÌƒRƒ“ƒsƒ…□[ƒ^ŠÔ‚ÅƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ì "ƒvƒ‹" ‚Æ "ƒvƒbƒVƒ…" ‚ð□s‚¤‚±
‚Æ‚É‚æ‚Á‚Ä□AƒpƒbƒP□[ƒW‚ð”z•z‚·‚é‚±‚Æ‚ª‚Å‚«‚Ü‚·□B‚Ü‚½□AMicrosoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰‚ÅƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹ ƒt□[ƒeƒBƒŠƒeƒB‚ðŽg‚Á‚Ä□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹‚ð□¶□¬‚·‚é‚±‚Æ‚à‚Å‚«‚Ü‚·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ðŽg‚Á‚Ä□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹‚ð□¶□¬‚·‚é•û–@‚Ì□Ú□×‚É‚Â‚¢‚Ä□A□uMTS ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹‚ð□¶□¬‚·‚é□v‚ðŽQ□Æ‚µ‚Ä‚-
‚¾‚³‚¢□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒvƒbƒ…‚Æ‚Í□AƒŠƒ‚□[ƒg ƒRƒ“ƒsƒ…□[ƒ^‚ÉƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒGƒ“ƒgƒŠ‚ð□ì□¬‚·‚é‚±‚Æ‚ðˆÓ–¡‚µ‚Ü‚·□BƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒGƒ“ƒgƒŠ‚ª□ì□¬‚³‚ê‚½‚ç□A±‚ê‚ç‚ÌƒRƒ“ƒ|□[ƒlƒ“ƒg ƒGƒ“ƒgƒŠ‚ðƒ□□[ƒ]ƒ‹ƒ‹ ƒRƒ“ƒsƒ…□[ƒ^‚Ì [ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_‚É‚Ç‰Á‚·‚é•—v‚ª ‚è‚Ü‚· □ifRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒvƒ‹)□B

ŽŸ‚Ì□}‚Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒvƒbƒvƒ…‚Æƒvƒ‹‚É‚æ‚Á‚Ä MTS ‚ðŽÀ□s‚µ‚Ä‚¢‚éƒŠƒ‚□[ƒg ƒRƒ“ƒsƒ…□[ƒ^‚ð□\□¬‚·‚éŽd‘g‚Ý‚ðŽ¦‚µ‚Ä‚¢‚Ü‚·□B



□ã,Ì□},Å,Í□AMTS ,ª 3 ‘ä,ÌƒRƒ“ƒsƒ…□[ƒ^,·,×,Ä,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,Ü,·□BGetReceipt ,Æ MoveMoney ,Í,¢ ,¸,ê,à□AŽg—p,µ,Ä,¢,éƒRƒ“ƒsƒ…□[ƒ^ (ƒT□[ƒo□[A) ,ÌƒpƒbƒP□[ƒW,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,Ü,·□BƒT□[ƒo□[B ,ÆƒT□[ƒo□[C ,Ì [ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,É GetReceipt ƒRƒ“ƒ|□[ƒlƒ“ƒg,Æ MoveMoney ƒRƒ“ƒ|

□[ƒlƒ“ƒg,ð’Ç‰Á,·,é,Æ□AŽŸ,Ì 2 ,Â,Ì□ˆ—□,ª□s,í,ê,Ü,·□B,Ü,¸□A“K□Ø,È DLL ƒtƒ@ƒCƒ‹,ªƒlƒbƒgƒ□□[ƒNŒo—R,ÅƒT□[ƒo□[B ,ÆƒT□[ƒo□[C ,ÉƒRƒs□[,³,ê,Ü,·□B,±,ê,ç,Ìƒtƒ@ƒCƒ‹,Í□AƒT□[ƒo□[A ,ÌƒpƒbƒP□[ƒW,Æ“¯,¶–¼‘O,ÌƒTƒuƒffƒBƒŒƒNƒgƒŠ,ÉƒRƒs□[,³,ê,Ü,·□B,±,ÌƒTƒuƒffƒBƒŒƒNƒgƒŠ,Í□A\<Microsoft Transaction Server ,ÌƒCƒ“ƒXƒg□[ƒ‹ ƒffƒBƒŒƒNƒgƒŠ>\Remote ƒffƒBƒŒƒNƒgƒŠ,Ì‰º (,½,Æ,¦,Î□AC:\Program Files\Mts\Remote\Sample Bank) ,É□ì□¬,³,ê,Ü,·□BŽŸ,É□AƒT□[ƒo□[B ,Æ ƒT□[ƒo□[C ,ÌƒVƒXƒeƒ€ ƒŒƒWƒXƒgƒŠ,ªƒT□[ƒo□[A ,ÌƒVƒXƒeƒ€ ƒŒƒWƒXƒgƒŠ,Ì□î•ñ,É,æ,Á,Ä□X□V,³,ê,Ü,·□B

DLL ƒtƒ@ƒCƒ‹,Æƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ ƒtƒ@ƒCƒ‹,Ì•Û‘¶,Æ”z•z,Ì,½,ß,Ì‹¤—Lƒlƒbƒgƒ□□[ƒNƒffƒBƒŒƒNƒgƒŠ,ª□Ý’è,³,ê,Ä,¢,é□ê□‡,É,¾,¯□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒvƒbƒVƒ…,Æƒvƒ‹,ð□s,¤,±,Æ,ª,Å,«,Ü,· (‹¤—LƒffƒBƒŒƒNƒgƒŠ,ÌƒtƒHƒ‹ƒ_,Ü,½,ÍƒTƒuƒtƒHƒ‹ƒ_,Ì,¢,¸,ê,©,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹,ªŠÜ,Ü,ê,Ä,¢,ê,Î□A,Ç,Ì‹¤—LƒffƒBƒŒƒNƒgƒŠ,Å,à'I'ð,Å,«,Ü,·)□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,Í□AƒT□[ƒo□[□ã,Ì—˜—p‰Â”\,È‹¤—Lƒlƒbƒgƒ□□[ƒNƒffƒBƒŒƒNƒgƒŠ,ðŽ©“®“I,É'T,µ,Ü,·□B•¡□”,Ì‹¤—LƒffƒBƒŒƒNƒgƒŠ,ð 1 ,Â,ÌƒT□[ƒo□[□ã,É□ì□¬,µ□A^Ù,È,éƒRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹,ÌƒZƒbƒg,ÉƒAƒNƒZƒX,·,é,±,Æ,ª,Å,«,Ü,·□B

,Ü,½□AƒvƒbƒVƒ…,Æƒvƒ‹,ð□s,¤,É,Í□AŽŸ,Ì□ðŒ□,à–ž,½,³,ê,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□B

- ƒ^□[ƒQƒbƒg ƒRƒ“ƒsƒ…□[ƒ^,ÌƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW,Ì Administrators ƒ□□[ƒ‹,Ìƒ□ƒ“ƒo□[,É,È,Á,Ä,¢,é Windows NT ƒAƒJƒEƒ“ƒg,ðŽg,Á,Äƒ□ƒOƒlƒ“,µ,Ä,¢,é,±,Æ□B
- ƒ^□[ƒQƒbƒg ƒRƒ“ƒsƒ…□[ƒ^,ÌƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW ID ,ª□Aƒ□□[ƒJƒ‹,ÌƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW,Ì Reader ƒ□□[ƒ‹,É□Ý’è,³,ê,Ä,¢,é Windows NT ƒAƒJƒEƒ“ƒg,Éƒ}ƒbƒv,³,ê,Ä,¢,é,±,Æ□B
- —¼•û,ÌƒRƒ“ƒsƒ…□[ƒ^,ÅƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW,ÌƒZƒLƒ…ƒŠƒefB,ª—LŒø,É,È,Á,Ä,¢,é,±,Æ□B□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒvƒbƒVƒ…,·,é,É,Í□A,Ü,¸“K□Ø,È 1 ‘ä,Ü,½,Í•¡□”,ÌƒRƒ“ƒsƒ…□[ƒ^,ð MTS ƒGƒNƒXƒvƒ□□[ƒ‰,É'Ç‰Á,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÉƒRƒ“ƒsƒ…□[ƒ^,ð'Ç‰Á,·,é•û–@,É,Â,¢,Ä,Í□A□uMTS ƒT□[ƒo□[,Ì□\□¬□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŽŸ,É□AƒSƒ,□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒSƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

▶ **ƒSƒ,□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒSƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,·,é,É,Í**

1. [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ð'I'ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒRƒ“ƒsƒ…□[ƒ^] ,ðƒNƒŠƒbƒN,µ,ÄƒSƒ,□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,ð'Ç‰Á,µ,Ü,·□B
2. 'Ç‰Á,·,éƒRƒ“ƒsƒ…□[ƒ^,Ì–¼‘O,ð“ü—Í,µ□A[OK] ,ðƒNƒŠƒbƒN,µ,Ü,·□BƒRƒ“ƒsƒ…□[ƒ^,Ì–¼‘O,ª,í,©,ç,È,¢□ê□‡,Í□A[ŽQ□Æ] ,ðƒNƒŠƒbƒN,µ,ÄƒRƒ“ƒsƒ…□[ƒ^,ð'I'ð,µ,Ü,·□B
3. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å□AƒSƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,·,éƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒSƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
4. ŽŸ,Ì,¢,¸,ê,©,Ì'€□ì,ðŽÀ□s,µ,Ü,·□B
   - [“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ□A[ƒSƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - ƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,Å [ƒSƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ðƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒSƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒc□[ƒ‹ ƒo□[,Ì [□V,µ,¢ƒIƒuƒWƒFƒNƒg,Ì□ì□¬] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B
5. •\Ž¦,³,ê,½ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□AƒSƒ,□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,ÆƒSƒ,□[ƒg,©,çŒÄ,Ñ□o,·ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŠÜ,Þƒpƒbƒ‹P□[ƒW,ð'I'ð,µ,Ü,·□B
6. [—˜—p,Å,«,éƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒ{ƒbƒNƒX,Ìˆê——,Å□AƒSƒ,□[ƒg,©,çŒÄ,Ñ□o,·ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'I'ð,µ□A‰ºŒü,«–îˆó ([’Ç‰Á]) ,ðƒNƒŠƒbƒN,µ,Ü,·□B,±,ê,É,æ,è□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Æ□A,»,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ª•Û‘¶,³,ê,Ä,¢,éƒRƒ“ƒsƒ…□[ƒ^,ª [ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□\□¬□ê□Š] ƒ{ƒbƒNƒX,É'Ç‰Á,³,ê,Ü,·□B[□Ú□×] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒ“,É,·,é,Æ□AƒSƒ,□[ƒg ƒRƒ“ƒsƒ…□[ƒ^□AƒpƒbƒP□[ƒW□A,¨,æ,Ñ DLL ,ÌƒpƒX,ª [ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□\□¬□ê□Š] ƒ{ƒbƒNƒX,É•\Ž¦,³,ê,Ü,·□B
7. [OK] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

'□ ƒSƒ,□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,É•Û‘¶,³,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'Ç‰Á,·,é,Æ□A•K—v,Èƒtƒ@ƒCƒ‹,ª \<MTS

ƒCƒ“ƒXƒg□[ƒ‹ ƒfƒBƒŒƒNƒgƒŠ>\remote ƒfƒBƒŒƒNƒgƒŠ,É•Û‘¶,³,ê,Ü,· (Šù’è,Ì MTS ƒCƒ“ƒXƒg□[ƒ‹ ƒfƒBƒŒƒNƒgƒŠ,Í \Program Files\MTS ,Å,·)□B

ŠÖ˜A□€–Ú

[ƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_

MTS ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg

ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg,É,æ,Á,Ä□A, ,é MTS ƒRƒ“ƒsƒ…□[ƒ^,©,ç•Ê,Ì MTS ƒRƒ“ƒsƒ… □[ƒ^,ÉƒpƒbƒP□[ƒW,ðƒRƒs□[,Å,«,Ü,·□B,½,Æ,¦,Î□AƒpƒbƒP□[ƒW,ðƒeƒXƒg,·,é,½,ß,É□AMicrosoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AƒpƒbƒP□[ƒW,ªŠJ”,³,ê,½ƒT□[ƒo□[,©,ç•Ê,Ì MTS ƒT□[ƒo□[,ÉƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Å□AƒGƒNƒXƒ|□[ƒg,·,éƒpƒbƒP□[ƒW,ð‘I‘ð,µ,Ü,·□B
2. [“®□ì] ƒ□ƒjƒ…□[,Ì [ƒGƒNƒXƒ|□[ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B,Ü,½,Í□AƒpƒbƒP□[ƒW,ðƒ}ƒEƒX,Ì ‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒGƒNƒXƒ|□[ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
3. [ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□A□ì□¬,·,éƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,ÌƒpƒX,Æ–¼‘O,ð“ü—Í,·,é,©□A,Ü,½,Í [ŽQ□Æ] ,ðƒNƒŠƒbƒN,µ,Äƒtƒ@ƒCƒ‹,ðŽw’è,µ,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹,Í□AƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,Æ“¯,¶ƒfƒBƒŒƒNƒgƒŠ,ÉƒRƒs□[,³,ê,Ü,·□B
4. ƒGƒNƒXƒ|□[ƒg,·,éƒpƒbƒP□[ƒW,É□AŠù,Éƒt□[ƒU□[,ªŠ„,è“–,Ä,ç,ê,Ä,¢ ,éƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,ðŠÜ,ß,é□ê□‡,Í□A[ƒ□□[ƒ‹,ÉŠÖ˜A•t,¯,ç,ê,½ NT ,Ìƒt□[ƒU□[ID ,ð•Û‘¶,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒ“,É,µ,Ü,·□B
5. [ƒGƒNƒXƒ|□[ƒg] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,é,Æ□AMicrosoft Transaction Server ,Í□AŒ³,ÌƒpƒbƒP□[ƒW,ÉŠÜ,Ü,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,Æ ([ƒ□□[ƒ‹,ÉŠÖ˜A•t,¯,ç,ê,½ NT ,Ìƒt□[ƒU□[ID ,ð•Û‘¶,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒ“,É,µ,½□ê□‡,Í) ƒ□□[ƒ‹,ÉŠÖ,·,é□î•ñ,ðŠÜ,ÞƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹ (Šg’£Žq .pak) ,ð□ì□¬,µ□AƒpƒbƒP□[ƒW,ÉŠÖ˜A,Ì, ,éƒRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹ (ƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL)□Aƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ□A,¨,æ,Ñƒvƒ□ƒLƒV/ƒXƒ^ƒu DLL) ,ðƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,ª□ì□¬,³,ê,½ƒffƒBƒŒƒNƒgƒŠ,Æ“¯,¶ƒfƒBƒŒƒNƒgƒŠ,ÉƒRƒs□[,µ,Ü,·□BƒpƒbƒP□[ƒW,Å□Ý’è,³,ê,Ä,¢ ,é•Ï□X,¨,æ,Ñ□í□œ,ìƒ□ƒbƒN,Í□AƒpƒbƒP□[ƒW,Æ‹¤,ÉƒGƒNƒXƒ|□[ƒg,³,ê,Ü,·□B

□d—v ƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,µ,½Œã,Å□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒNƒ‰ƒX ID (CLSID)□Aƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ ID (TypeLibId)□A,Ü,½,ÍƒCƒ“ƒ^□[ƒtƒFƒCƒX ID (IID) ,ª•Ï□X,³,ê,½□ê□‡,Í□AƒpƒbƒP□[ƒW,ð,à,¤ˆê“xƒGƒNƒXƒ|□[ƒg,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

ƒGƒNƒXƒ|□[ƒg—p,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é□AMTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é□AMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

MTS ŽÀ□s‰Â”\ftf@ƒCƒ‹,ð□¶□¬,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□AƒŠƒ,□[ƒg ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,Å,«,é,æ,¤,ÉƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ìƒCƒ“ƒXƒg□[ƒ‹,Æ□\□¬,ð□s,¤ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ìŽÀ□s‰Â”\ftf@ƒCƒ‹,ð□¶□¬,Å,«,Ü,·□BŽÀ□s‰Â”\ftf@ƒCƒ‹,ðƒCƒ“ƒXƒg□[ƒ‹,·,éƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ÍADCOM ,ðƒTƒ|□[ƒg,µ,Ä,¢ ,È,¯,ê,Î,È,è,Ü,¹,ñ□B,½,¾,µ□AƒŠƒ,□[ƒg MTS ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,·,é,É,Í□AŽÀ□s‰Â”\ ftf@ƒCƒ‹^ÈŠO,Ì MTS ƒT□[ƒo□[ftf@ƒCƒ‹,Í•K—v, ,è,Ü,¹,ñ□B

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ftf@ƒCƒ‹ ƒt□[ƒefBƒŠƒefB,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ìƒpƒbƒP□[ƒW ƒGƒNƒXƒf| □[ƒg‹@”\,Ì^ê•”,Å,·□B,±,ìƒt□[ƒefBƒŠƒefB,ðŽg,¤,Æ□AƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒCƒ“ƒXƒg□[ƒ‹,µ□AƒŠƒ,□[ƒg MTS ƒT□[ƒo□[□ã,ìƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,Å,«,é,æ,¤,ÉƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ð□\ □¬,·,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ìŽÀ□s‰Â”\ ftf@ƒCƒ‹,ðŽ©“®“I,É□¶□¬,Å,«,Ü,·□BŠù'è,Ì□Ý'è,Å,Í□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ftf@ƒCƒ‹ ƒt□[ƒefBƒŠƒefB,É,æ,Á,Ä□¶□¬,³,ê,éŽÀ□s‰Â”\ftf@ƒCƒ‹,Í□AŽÀ□s‰Â”\ ftf@ƒCƒ‹,ª□¶□¬,³,ê,½ƒT□[ƒo□[,ÉƒAƒNƒZƒX,·,é,æ,¤,ÉƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ð□\□¬,µ,Ü,·□B

ƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒvƒ□ƒpƒefB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ƒIƒvƒVƒ‡ƒ“] ƒ^ƒu,Ì [ƒŠƒ,□[ƒg ƒT□[ƒo□[–¼] ,Å□AƒNƒ ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ªƒAƒNƒZƒX,·,éƒT□[ƒo□[,ðŽw'è,·,é,±,Æ,à,Å,«,Ü,·□BƒpƒbƒP□[ƒW,ðƒGƒNƒXƒf| □[ƒg,µ,ÄŽÀ□s‰Â”\ftf@ƒCƒ‹,ð□¶□¬,·,é'O,É□A,Ù,©,ìƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,Ì–¼'O,ð“ü—Í,µ,È,¯,ê,Î□AƒI□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^,Å□ì□¬,³,ê,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ìŽÀ□s‰Â”\ftf@ƒCƒ‹,Í□AŽ©“®“I,É□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^□ã,ìƒT□[ƒo□[ƒpƒbƒP□[ƒW,ÉƒAƒNƒZƒX,·,é,æ,¤,ÉƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ð□\□¬,µ,Ü,·□B

▶ f□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^ÈŠO,ÌƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,ÉƒAƒNƒZƒX,·,é,æ,¤,ÉƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ìŽÀ□s‰Â”\ftf@ƒCƒ‹,ð□\□¬,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ìƒEƒBƒ“ƒhƒE,Ì□¶'¤,Å□A[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ƒAƒCƒRƒ“,ð'I'ð,µ,Ü,·□B
2. [ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ƒAƒCƒRƒ“,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒpƒefB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
3. [ƒIƒvƒVƒ‡ƒ“] ƒ^ƒu,ð'I'ð,µ□A[ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“] ,Ì [ƒŠƒ,□[ƒg ƒT□[ƒo□[–¼] ƒ{ƒbƒNƒX,ÉƒŠƒ,□[ƒg ƒT□[ƒo□[,Ì–¼'O,ð“ü—Í,µ,Ü,·□B
4. [OK] ,ðƒNƒŠƒbƒN,µ□AƒpƒbƒP□[ƒW,ðƒGƒNƒXƒf|□[ƒg,µ,ÄƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŽÀ□s‰Â”\ftf@ƒCƒ‹,ð□ì□¬,µ,Ü,·□B

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ftf@ƒCƒ‹ ƒt□[ƒefBƒŠƒefB,ªƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŽÀ□s‰Â”\ ftf@ƒCƒ‹,ðŽ©“®“I,É□ì□¬,µ,Ü,·□B

ƒNƒ‰ƒCƒAƒ“ƒg,Å,Í□AŽÀ□s‰Â”\ftf@ƒCƒ‹,ªŽŸ,Ì□ì‹Æ,ðŽ©“®“I,É□s,¢,Ü,·□B

1 ƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,ÆƒfƒXƒ^ƒ€ ƒvƒ□ƒfLƒV/ƒXƒ^ƒu DLL ,ðŠÜ,Þ•K—v,ÈƒNƒ‰ƒCƒAƒ“ƒg'¤,ìftf@ƒCƒ‹,ðƒNƒ ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,Ü,½,ÍƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,Ì^êŽžƒffƒBƒŒƒNƒgƒŠ,ÉƒRƒs□[,µ□AŽæ,è□o,µ,Ü,·□B

2 ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“—p,ìƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,ÆƒvƒfLƒV/ƒXƒ^ƒu DLL ,ð \Program Files\Remote Applications ƒffƒBƒŒƒNƒgƒŠ,É“]'—,µ,Ü,·□BƒŠƒ,□[ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,É,Í□AƒpƒbƒP□[ƒW,ìƒOƒ□[ƒoƒ‹,Éˆê‹Ó,Ì ID (GUID) ,Æ“¯,¶–¼'O,ìƒffƒBƒŒƒNƒgƒŠ,ªŒÂ•Ê,É□ì□¬,³,ê□A,»,ìƒffƒBƒŒƒNƒgƒŠ,Éftf@ƒCƒ‹,ª•Ü'¶,³,ê,Ü,·□B

3 ƒNƒ‰ƒCƒAƒ“ƒg,ªƒŠƒ,□[ƒg,©,çƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽg—p,Å,«,é,æ,¤,É,·,é (ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“□AƒNƒ ‰ƒX□AƒvƒfOƒ‰ƒ€□AƒCƒ“ƒ^□[ƒtƒFƒCƒX□A¨,æ,Ñƒ‰ƒCƒuƒ‰ƒŠ,Ì ID ,ÉŠÖ˜A,·,é□ì•ñ,ðŠÜ,Þ) ƒGƒ“ƒgƒŠ□A,Ü,½,ÍƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^□ã,ÅŽÀ□s,Å,«,é,æ,¤ ,É,·,éƒGƒ“ƒgƒŠ,É,æ,Á,ÄƒVƒXƒef€ ƒŒƒWƒXƒgƒŠ,ð□X□V,µ,Ü,·□B

4 ƒt□[ƒU□[,ªƒRƒ“ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Ì [ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ì'Ç‰Á,Æ□í□œ] ƒAƒCƒRƒ“,ðŽg,Á,Ä□AŒã,©,çƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□í□œ,Å,«,é,æ,¤ ,É□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð“o˜^,µ,Ü,·□BƒCƒ“ƒXƒg□[ƒ‹,ê,½ƒRƒ“ƒ|[ƒlƒ“ƒg,Ì^ê——,ÅŠÈ'P,ÉŒ©•ª,¯,ª,Â,,æ,¤ ,É□A,·,×,Ä,ìƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì'O,É "Remote Application" ,ª•t,¯,ç,ê,Ü,·□B

5 ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ìƒCƒ“ƒXƒg□[ƒ‹'†,É□¶□¬,³,ê,½^êŽžƒffƒBƒŒƒNƒgƒŠ,Ìftf@ƒCƒ‹,ð,·,×,Ä□í□œ,µ,Ü,·□B

ƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ÅƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ftf@ƒCƒ‹,ªŽÀ□s,³,ê,é,Æ□A•K—v,ÈƒvƒfLƒV/ƒXƒ^ƒu

DLL ,Æƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,ªƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ÉƒRƒs□[,³,ê□AƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^–¼,ðŠÜ,Þ
DCOM ,É•K—v,È□î•ñ,É,æ,Á,ÄƒNƒ‰ƒCƒAƒ“ƒg,ÌƒVƒXƒeƒ€ ƒŒƒWƒXƒgŠ,ª□X□V,³,ê,Ü,·□B,±,ê,É,æ,è□AƒNƒ
‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÍƒŠƒ‚□[ƒg ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,Å,«,é,æ,¤,É,È,è,Ü,·□B

ƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,µ,ÄŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□ì□¬,·,é‘O,É□AƒNƒ‰ƒCƒAƒ“ƒg ƒCƒ“ƒXƒg□[ƒ‹ ƒtƒ@ƒCƒ‹
(clients.ini) ,ð□ì□¬,·,é,±,Æ,É,æ,è□AƒNƒ‰ƒCƒAƒ“ƒg
ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒCƒ“ƒXƒg□[ƒ‹,ðƒJƒXƒ^ƒ}ƒCƒY,Å,«,Ü,·□Bclients.ini ƒtƒ@ƒCƒ‹,É,æ,Á,Ä□A□”Ží—
Þ,Ì^Ù,È,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“—p,ÌƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\
ƒtƒ@ƒCƒ‹,ÌƒCƒ“ƒXƒg□[ƒ‹,ð‘g,Ý□‡,í,¹,½,è□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒhƒLƒ…ƒ□ƒ“ƒg,ðŠÜ,ß,é,±,Æ,à,Å,«,Ü,·□B

ƒCƒ“ƒXƒg□[ƒ‹,ðƒJƒXƒ^ƒ}ƒCƒY,·,é,É,Í

1. clients.ini ,Æ,¢,¤–¼‘O,Ìƒtƒ@ƒCƒ‹,ð□ì□¬,µ□AƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,éƒtƒHƒ‹ƒ_,É•Û‘¶,µ,Ü,·□B
2. ŽŸ,ÌƒefƒLƒXƒg,ð‹ó,Ì clients.ini ƒtƒ@ƒCƒ‹,ÉƒRƒs□[,µ,Ü,·□B

```
[ClientApplicationFiles]
SourcePath=c:\temp\custom vb bank

[ClientApplicationInstallCommands]
1=notepad {{{readme.txt}}}
2={{{vbbank.exe}}}

[ClientApplicationSetup]
ExploreApplication=Y
```

3. Œ©□o,µ ClientApplicationFiles ,Ì‰º,ÌƒpƒXŽw’è,ð□AƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,ÉƒCƒ“ƒXƒg□[ƒ‹,·,éƒ\
□[ƒX ƒR□[ƒh,ª, ,éƒfƒBƒŒƒNƒgƒŠ,É•Ï□X,µ,Ü,·□B,½,Æ,¦,Î□AŽŸ,Ì,æ,¤,É•Ï□X,µ,Ü,·□B

```
SourcePath=c:\program files\mts\test\vb bank
```

4. Œ©□o,µ ClientApplicationInstallCommands ,Ì‰º,É,Í□AƒCƒ“ƒXƒg□[ƒ‹,·,éƒtƒ@ƒCƒ‹,Ì–
¼‘O,ðŽw’è,µ,Ü,·□Bƒtƒ@ƒCƒ‹–¼,Í□A3 □d,©,Á,±,Å^Í,Ý,Ü,·□B,½,Æ,¦,Î□AŽŸ,Ì,æ,¤,ÉŽw’è,µ,Ü,·□B

```
1=notepad {{{readme.txt}}}
2={{{vbbank.ex}}}}
```

ƒCƒ“ƒXƒg□[ƒ‹,·,éƒtƒ@ƒCƒ‹,Í□ASourcePath= ,ÅŽw’è,µ,½ƒfƒBƒŒƒNƒgƒŠ,É‘¶□Ý,·,é•K—v,ª, ,è,Ü,·□B

5. Œ©□o,µ ClientApplicationSetup ,Ì‰º,Å,Í□AExploreApplication ƒZƒbƒgƒAƒbƒv ƒIƒvƒVƒ‡ƒ“,ð—
LŒø,É,·,é,©,Ç,¤,©,ðŽ¦,µ,Ü,·□B ExploreApplication=Y ,ðŽw’è,·,é,Æ□AƒZƒbƒgƒAƒbƒv,Ì□I—
¹’¼Œã,ÉƒGƒNƒXƒvƒ□□[ƒ‰,ªŽ©“®“I,É‹N“®,µ,ÄƒCƒ“ƒXƒg□[ƒ‹,µ,½ƒtƒ@ƒCƒ‹,ð•\Ž¦,µ,Ü,·□B,±
,ÌƒIƒvƒVƒ‡ƒ“,ðŽg,¤,Æ□AƒNƒ‰ƒCƒAƒ“ƒg,ªƒffƒXƒNƒgƒbƒv□ã,ÉƒVƒ‡□[ƒgƒJƒbƒg,ð□ì□¬,·,é,Ì,É•Ö—
˜,Å,·□BExploreApplication=N ,ðŽw’è,·,é,©□A**[ClientApplicationSetup]**
ƒZƒNƒVƒ‡ƒ“,ðŽw’è,µ,È,©,Á,½□ê□‡,Í□AƒGƒNƒXƒvƒ□□[ƒ‰,Í‹N“®,³,ê,Ü,¹,ñ□B

clients.ini ƒtƒ@ƒCƒ‹,ð•Ï□X,µ,Ä•Û‘¶,µ,½,ç□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄƒT□[ƒo□[ƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|
□[ƒg,µ□AƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□ì□¬,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□ì□¬,·,é,É,Í

1. ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ä,¢,È,¢□ê□‡,Í□AƒpƒbƒP□[ƒW
ƒEƒBƒU□[ƒh,ðŽg,Á,ÄƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B

   ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□ì□¬,·,é”z’uƒT□[ƒo□[ˆÈŠO,ÌƒT□[ƒo□[,ðƒNƒ
   ‰ƒCƒAƒ“ƒg,ªƒAƒNƒZƒX,·,é□ê□‡,Í□A[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ƒAƒCƒRƒ“,ðƒ}ƒEƒX,Ì
   ‰Eƒ{ƒ^ƒ“,ÅƒNŠƒbƒN,µ□A[ƒvƒ□ƒpƒefB] ,ðƒNƒŠƒbƒN,µ,Ü,·□BŽŸ,É [ƒIƒvƒVƒ‡ƒ“] ƒ^ƒu,ðƒNƒŠƒbƒN,µ,Ä
   [ƒŠƒ,□[ƒg ƒT□[ƒo□[–¼] ƒ{ƒbƒNƒX,É□AƒNƒ‰ƒCƒAƒ“ƒg,ªƒAƒNƒZƒX,·,éƒT□[ƒo□[,Ì–¼‘O,ð“ü—Í,µ,Ü,·□B

3. ƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,µ,Ü,·□BƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,éƒtƒ@ƒCƒ‹–¼,Æ,µ,Ä□V,µ,¢ƒtƒ@ƒCƒ‹–¼
(YourNewFileName) ,ðŽw’è,µ,Ü,·□B
4. ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg□æƒtƒHƒ‹ƒ_,ð’T,µ,Ü,·□B"YourNewFileName.exe" ,Æ,¢,¤–¼‘O,Ìƒtƒ@ƒCƒ‹,ª 1

,ÂŠÜ,Ü,ê,Ä,¢,é Clients ƒfƒBƒŒƒNƒgƒŠ,ª, ,é,±,Æ,ðŠm”F,µ,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰
,ÅŠù‘¶,lƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,é,Æ□AƒpƒbƒP□[ƒW,lƒGƒNƒXƒ|□[ƒg□æƒfƒBƒŒƒNƒgƒŠ,Ì‰º,É Clients ƒfƒBƒŒƒNƒgƒŠ,ª□ì□¬,³,ê,Ü,·□BClients ƒfƒBƒŒƒNƒgƒŠ,É,Í□AƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|
□[ƒg,·,é,Æ,«,ÉŽw’è,µ,½–¼‘O,lŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ª 1 ,ÂŠÜ,Ü,ê,Ä,¢,Ü,·□BDCOM ,ðƒTƒ|□[ƒg,µ,Ä,¢
,é”Cˆ Ó,lƒNƒ‰ƒCƒAƒ“ƒg,Å,±,lŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ªŽÀ□s,³,ê,é,Æ□AƒŠƒ,□[ƒg ƒNƒ
‰ƒCƒAƒ“ƒg,ªƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,·,é,½,ß,É•K—v,È□ì•ñ,ª,·,×,ÄƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ü,·□B

□d—v ƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,Å,±,lƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ðŽÀ□s,µ,È,¢,Å,,¾,³,¢□BƒNƒ
‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ðƒT□[ƒo□[ƒRƒ“ƒsƒ…□[ƒ^,ÅŽÀ□s,·,é,Æ□AƒT□[ƒo□[ƒpƒbƒP□[ƒW,lŽÀ□s,É•K—v,ÈƒCƒWƒXƒgƒŠ ƒGƒ“ƒgƒŠ,ª□í□œ,³,ê,Ü,·□BŒë,Á,ÄŽÀ□s,µ,½□ê□‡,Í□AƒRƒ“ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Ì
[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì’Ç‰Á,Æ□í□œ]
ƒAƒCƒRƒ“,ðŽg,Á,ÄƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□í□œ,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B,»,lŒã□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰
,ðŽg,Á,ÄƒpƒbƒP□[ƒW,ð□í□œ,µ□A□ÄƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B

ƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ªƒCƒ“ƒXƒg□[ƒ‹,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì,Ù,©,lƒo□[ƒWƒ‡ƒ“,ªƒNƒ‰ƒCƒAƒ“ƒg
ƒRƒ“ƒsƒ…□[ƒ^,ÉŠù,É‘¶□Ý,·,é□ê□‡□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ÍŽŸ,Ì□ê□‡,É,Ì,Ý□X□V,³,ê,Ü,·□B

- ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì—¼•û,lƒo□[ƒWƒ‡ƒ“,ªƒo□[ƒWƒ‡ƒ“”Ô□†,ðŽ□,Á,Ä,¢,é□B
- ƒCƒ“ƒXƒg□[ƒ‹,³,ê,éƒRƒ“ƒ|□[ƒlƒ“ƒg,lƒo□[ƒWƒ‡ƒ“,ª□AŠù,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒRƒ“ƒ|
□[ƒlƒ“ƒg,lƒo□[ƒWƒ‡ƒ“,æ,è,àŒã,lƒo□[ƒWƒ‡ƒ“,Å, ,é□B

ƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð”z•z,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Í□A”z•z—p,lŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,Ö,lƒNƒ‰ƒCƒAƒ“ƒg
ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,lŽÀ‘•,Æ‘g,Ý□ž,Ý,ðŽ©“®‰»,µ,Ü,·□B,±,ê,ç,lŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,Í□AŽŸ,Ì,¢,¸,ê,©,Ì•û–@,Å”z•z,µ,Ü,·□B

- ƒfƒBƒŒƒNƒgƒŠ,ð‹¤—L,µ□AƒNƒ‰ƒCƒAƒ“ƒg,ªŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ðƒRƒs□[,µ,Ä□AƒNƒ‰ƒCƒAƒ“ƒg,lƒRƒ“ƒsƒ…□[ƒ^,ÅŽÀ□s,Å,«,é,æ,¤,É,·,é•û–@□B
- ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð“dŽqƒ□□[ƒ‹,Ì“Y•tƒtƒ@ƒCƒ‹,Æ,µ,Ä‘—□M,µ□AƒNƒ‰ƒCƒAƒ“ƒg,ªŽÀ□s‰Â”\
ƒtƒ@ƒCƒ‹,ð•Ù‘¶,µ,Ä□AƒNƒ‰ƒCƒAƒ“ƒg,lƒRƒ“ƒsƒ…□[ƒ^,ÅŽÀ□s,Å,«,é,æ,¤,É,·,é•û–@□B
- <OBJECT> ƒ^ƒO,ðŽg,Á,ÄŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð HTML ƒXƒNƒŠƒvƒg,É‘g,Ý□ž,Þ•û–@□B<OBJECT> ƒ^ƒO,ðŽg,¤,Æ□AƒNƒ‰ƒCƒAƒ“ƒg,ª HTML ƒy□[ƒW,Å (ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,·,é,È,Ç,Ì) ƒCƒxƒ“ƒg,ðŠJŽn,·,ê,Î□Aƒuƒ‰ƒEƒU,ðŽg,Á,Ä□AŽw’è,³,ê,½ƒlƒuƒWƒFƒNƒgŠi”[□ê□Š,©,çƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒNƒ‰ƒCƒAƒ“ƒg ƒRƒ“ƒsƒ…□[ƒ^,Éƒ_ƒEƒ“ƒ□□[ƒh,·,é,±,Æ,ª,Å,«,Ü,·□B<OBJECT> ƒ^ƒO,ðŽg—p,µ,ÄŽÀ□s‰Â”\
ƒtƒ@ƒCƒ‹,ð”z•z,·,é,Æ□Aƒuƒ‰ƒEƒU,Íƒ©“®“I,ÉƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŒ»□Ý,lƒo□[ƒWƒ‡ƒ“,ÆƒNƒ‰
‰ƒCƒAƒ“ƒg,lƒCƒWƒXƒgƒŠ,ð□Æ□‡,·,é,Ì,Å□AƒAƒbƒvƒOƒCƒ□[ƒh,ª—eˆÕ,É,È,è,Ü,·□BŠù‘¶,ÌŽÀ□s‰Â”\
ƒtƒ@ƒCƒ‹,ªˆÈ‘O,lƒo□[ƒWƒ‡ƒ“,Ì□ê□‡□Afuƒ
‰ƒEƒU,ÍƒlƒuƒWƒFƒNƒgŠi”[□ê□Š,©,ç□Å□V,lƒo□[ƒWƒ‡ƒ“,ðƒ_ƒEƒ“ƒ□□[ƒh,µ,Ü,·□B
- Microsoft System Management Server (SMS) ,ðŽg,Á,Ä□AƒZƒ“ƒgƒ‰ƒ‹ ƒTƒCƒg,©,çˆê“x,É‰½□\‘ä□A‰½•S‘ä,à,lƒRƒ“ƒsƒ…□[ƒ^,ÉƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì”z•z,ð "ƒvƒbƒVƒ…" ,·,é•û–@□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒCƒ“ƒXƒg□[ƒ‹,µ,½Œã□AƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“Ž©‘Ì,ðƒŠƒ,□[ƒg ƒRƒ“ƒsƒ…□[ƒ^,ÉƒCƒ“ƒXƒg□[ƒ‹,µ,È,¯,ê,Î,È,ç,È,¢,±,Æ,É’□ˆÓ,µ,Ä,,¾,³,¢□B

ƒNƒ‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□í□œ,·,é

ƒNƒ‰ƒCƒAƒ“ƒg,Í□AƒRƒ“ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Ì [ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì’Ç‰Á,Æ□í□œ] ƒAƒCƒRƒ“,ðŽg,Á,ÄƒNƒ‰
‰ƒCƒAƒ“ƒgŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□í□œ,·,é,±,Æ,ª,Å,«,Ü,·□BMTS ŽÀ□s‰Â”\
ƒtƒ@ƒCƒ‹,É,æ,Á,ÄƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í□A[ƒCƒ“ƒXƒg□[ƒ‹,Æ□í□œ] ƒ^ƒu (Windows 95/98 ,Å,Í [ƒZƒbƒgƒAƒbƒv,Æ□í□œ] ƒ^ƒu) ,Ìˆê——,É•\Ž¦,³,ê,é,Æ,«□A□æ“ª,É "Remote Application" ,ª•t,«,Ü,·□BŽÀ□s
‰Â”\ƒtƒ@ƒCƒ‹,ð□í□œ,·,é,É,Í□A–Ú“I,lƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð‘I‘ð,µ□A[’Ç‰Á,Æ□í□œ] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ŠÖ˜A□€–Ú

MTS ƒpƒbƒP□[ƒW,ìƒGƒNƒXƒ|□[ƒg□AƒGƒNƒXƒ|□[ƒg—p,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é

MTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹

MTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä MTS ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÖƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,µ□A"z'u,·,é,±,Æ,Å,·□BƒCƒ“ƒXƒg□[ƒ‹,Æ"z'u,ÌŽè□‡,Í□A"K□Ø,ÈƒpƒbƒP□[ƒW ID ,Ì□Ý'è,È,Ç,Ì□ÝŒv,ÉŠÖ,·,é□I—¶"_,Æ–§□Ú,ÉŠÖ˜A,µ,Ä,¢,Ü,·□B

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ",ð"z'u,·,é'O,É□AƒpƒbƒP□[ƒW,ÆƒpƒbƒP□[ƒW,ÌƒRƒ"ƒ|□[ƒlƒ"ƒg,Ì□\'¢,ð□\•ª,É—□‰ð,µ,Ä,-,¾,³,¢□BMTS ƒRƒ"ƒ|□[ƒlƒ"ƒg,Ì□ÝŒv,Æ□ì¬,Ì□Ú□×,É,Â,¢,Ä,Í□A□wProgrammer's Guide□x,ðŽQ□Æ,µ,Ä,-,¾,³,¢□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,ÄŠù□¬,ÌƒpƒbƒP□[ƒW,ðŠÈ'P,É"z'u,Å,«,Ü,·□B,±,±,Å,Í□AŽŸ,ÌƒgƒsƒbƒN,É,Â,¢,Ä□à–¾,µ,Ü,·□B

Šù□¬,Ì MTS ƒpƒbƒP□[ƒW,ÌƒCƒ"ƒXƒg□[ƒ‹

MTS ƒpƒbƒP□[ƒW,ÌƒAƒbƒvƒOƒŒ□[ƒh

MTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é

MTS ƒpƒbƒP□[ƒW ID ,Ì□Ý'è

□V,µ,¢ MTS ƒ□□[ƒ‹,Ì'Ç‰Á

MTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

ŠÖ˜A□€–Ú

[ID] ƒ^ƒu (ƒpƒbƒP□[ƒW)□A[ƒCƒ"ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_□A[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_□A[ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_□A[ƒ□□[ƒ‹ ƒ□ƒ"ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_□AMTS ƒ□□[ƒ‹,Ìƒt□[ƒU□[,ðŠÇ—□,·,é

Šù□¬,Ì MTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹

Šù□¬,ÌƒpƒbƒP□[ƒW,Í□AƒpƒbƒP□[ƒW,ÉŠÖ˜A,·,éƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹ (ƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL) ,Æƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ) ,©,ç□\□¬,³,ê,Ä,¢,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ðŽg,Á,Ä□AƒT□[ƒh ƒp□[ƒefB,Ü,½,ÍŠé‹Æ,ÌŠJ”•”–
å,É,æ,Á,Ä□ì□¬,³,ê,½Šù□¬,ÌƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,Æ”z'u,ð□s,¤,±,Æ,ª,Å,«,Ü,·□B

ƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,µ,½,ç□AƒpƒbƒP□[ƒW,ð□\□¬,·,é,½,ß,É□A□,È,,Æ,à□AMTS ,É,æ,Á,Ä•Û'¶,³,ê,Ä,¢ ,éƒIƒuƒWƒFƒNƒg,Ö,ÌƒAƒNƒZƒX,Ì□§Œä□Aƒ□□[ƒ‹,Ö,Ìƒt□[ƒU□[,ÌŠ„,è“–,Ä□A,¨,æ,ÑƒpƒbƒP□[ƒW,ÌƒZƒLƒ…ƒŠƒefB ƒCƒxƒ‹,Ì□Ý'è,ð□s,¤•K—v,ª, ,è,Ü,·□BƒpƒbƒP□[ƒW,ÌƒZƒLƒ…ƒŠƒefB,ð□\□¬,·,é•û–@,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

Šù□¬,ÌƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶'¤,Å□AƒpƒbƒP□[ƒW,ð□ì□¬,·,éƒRƒ“ƒsƒ…□[ƒ^,ð'I'ð,µ,Ü,·□B
2. 'I'ð,µ,½ƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
3. ŽŸ,Ì,¢,¸,ê,©,Ì•û–@,Å□AƒpƒbƒP□[ƒW ƒEƒBƒU□[ƒh,ðŠJ,«,Ü,·□B
   - [“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ□A[ƒpƒbƒP□[ƒW] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒc□[ƒ‹ ƒo□[,Ì [□V,µ,¢ƒIƒuƒWƒFƒNƒg,Ì□ì□¬] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - [ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒpƒbƒP□[ƒW] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
4. [Šù□¬,ÌƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,·,é] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
5. [ƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,Ì'I'ð] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□A['Ç‰Á] ,ðƒNƒŠƒbƒN,µ,ÄƒIƒbƒgƒ□□[ƒN□ã,Ì—˜—p,Å,«,éƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,ð'T,µ,Ü,·□BƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹ (.pak) ,ð'I'ð,µ□A[ŠJ,] ,ðƒNƒŠƒbƒN,µ,Ü,·□BŽŸ,É□A[ŽŸ,Ö] ,ðƒNƒŠƒbƒN,µ,Ü,·□B"¯Žž,É•¡□",ÌƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,Å,«,Ü,·□BƒpƒbƒP□[ƒW,ÉŠÜ,Ü,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹,Í□AƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,Æ"¯,¶ƒfƒBƒŒƒNƒgƒŠ,É'u,©,ê,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□B
6. [ƒpƒbƒP□[ƒW ID ,Ì□Ý'è] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□AƒpƒbƒP□[ƒW ID ,ðŽw'è,µ□A[ŽŸ,Ö] ,ðƒNƒŠƒbƒN,µ,Ü,·□BŠù'è,Ì□Ý'è,Å,Í ['Î˜bƒt□[ƒU□[- Œ»□Ýƒ□ƒOƒIƒ“,µ,Ä,¢,éƒt□[ƒU□[] ,ª'I'ð,³,ê,Ü,·□B'Î˜bƒt□[ƒU□[,Í□AƒpƒbƒP□[ƒW,ªŽÀ□s,³,ê,éƒRƒ“ƒsƒ…□[ƒ^,Ì Windows NT ƒAƒJƒEƒ“ƒg,Éƒ□ƒOƒIƒ“,µ,½ƒt□[ƒU□[,Å,·□B•Ê,Ìƒt□[ƒU□[,ð'I'ð,·,é,É,Í□A[,±,Ìƒt□[ƒU□[] ,ð'I'ð,µ□A"Á'è,Ì Windows NT ƒt□[ƒU□[,Ì□Ú□×□î•ñ,ð"ü—Í,µ,Ü,·□B
7. [ƒCƒ“ƒXƒg□[ƒ‹ ƒIƒvƒVƒ‡ƒ“] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□AƒCƒ“ƒXƒg□[ƒ‹□æƒffBƒŒƒNƒgƒŠ,ðŽw'è,µ,Ü,·□BƒfRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹,ªƒ□[ƒU□[ƒ‹□AƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,ÌƒffBƒŒƒNƒgƒŠ,©,çƒCƒ“ƒXƒg□[ƒ‹□æƒffBƒŒƒNƒgƒŠ,ÉƒRƒs□[,³,ê,Ü,·□BŠù'è,ÌƒffBƒŒƒNƒgƒŠ,ðŽó,¯"ü,ê,é,©□A,Ü,½,Í [ŽQ□Æ] ,ðƒNƒŠƒbƒN,µ,Ä•Ê,Ì□ê□Š,ð'T,µ,Ü,·□B
8. ƒCƒ“ƒXƒg□[ƒ‹,·,éƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,É'è‹`□Ï,Ý,Ì Windows NT ƒt□[ƒU□[,ªŠÜ,Ü,ê,Ä,¢,é□ê□‡,Í□A[ƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,É•Û'¶,³,ê,½ NT ƒt□[ƒU□[,ð'Ç‰Á,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ð—˜—p,Å,«,Ü,·□B,±,ìƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒIƒ“,É,·,é,Æ□A,±,ê,ç,Ìƒt□[ƒU□[,ª□V,µ,¢ƒpƒbƒP□[ƒW,É'Ç‰Á,³,ê,Ü,·□B
9. [Š®—¹] ,ðƒNƒŠƒbƒN,µ,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì‰E'¤,É□V,µ,¢ƒpƒbƒP□[ƒW,ª•\Ž¦,³,ê,Ü,·□B•¡□",ÌƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,·,é□ê□‡,Í□AŽè□‡ 6 ,ÆŽè□‡ 7 ,Å'I'ð,·,éƒIƒvƒVƒ‡ƒ“,ª,·,×,Ä,ÌƒpƒbƒP□[ƒW,É"K—p,³,ê,Ü,·□B

Windows NT ,ÅŠù□¬,ÌƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,É,Í□AƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ðŠJ,«□AƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,ð Windows ,Ì ƒGƒNƒXƒvƒ□□[ƒ‰,©,ç MTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ÌƒEƒBƒ“ƒhƒE,Ì‰E'¤,Éƒhƒ‰ƒbƒO,·,é•û–@,à, ,è,Ü,·□B

ŠÖ˜A□€–Ú

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_

MTS ƒpƒbƒP□[ƒW,ÌƒAƒbƒvƒOƒŒ□[ƒh

MTS ƒpƒbƒP□[ƒW,ðƒAƒbƒvƒOƒŒ□[ƒh,·,é□ê□‡,Í□A,Ü,¸ˆÈ‘O,Ìƒo□[ƒWƒ‡ƒ“,ð□í□œ,µ□AŽŸ,É□X□V,³,ê,½ƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ƒpƒbƒP□[ƒW,ð□í□œ,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Å□A□í□œ,·,éƒpƒbƒP□[ƒW,ª•Û‘¶,³,ê,Ä,¢,éƒRƒ“ƒsƒ…□[ƒ^,ð‘I‘ð,µ,Ü,·□B
2. ‘I‘ð,µ,½ƒRƒ“ƒsƒ…□[ƒ^,Ì [ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ðŠJ,¢,Ä,·,×,Ä,ÌƒpƒbƒP□[ƒW,ð•\Ž¦,µ,Ü,·□B
3. □í□œ,·,éƒpƒbƒP□[ƒW,ð‘I‘ð,µ,Ü,·□B
4. ŽŸ,Ì,¢,¸,ê,©,Ì‘€□ì,ðŽÀ□s,µ,Ü,·□B
   - [“®□ì] ƒ□ƒjƒ…□[,Ì [□í□œ] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - □í□œ,·,éƒpƒbƒP□[ƒW,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[□í□œ] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - **Del** ƒL□[,ð‰Ÿ,µ,Ü,·□B
5. [,Í,¢] ,ðƒNƒŠƒbƒN,µ,ÄƒpƒbƒP□[ƒW,ð□í□œ,µ,Ü,·□B

ƒpƒbƒP□[ƒW,ð□í□œ,·,é,Æ□AƒpƒbƒP□[ƒW,ÉŠÜ,Ü,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,à,·,×,Ä□í□œ,³,ê,Ü,·□B

ƒAƒbƒvƒOƒŒ□[ƒh,³,ê,½ƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,·,é•û–@,É,Â,¢,Ä,Í□A□uŠù□¬,Ì MTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜A□€–Ú

[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_

MTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é

MTS ,Í 2 Ží—Þ,Ìƒpƒbƒ P□[ƒW ƒZƒLƒ…ƒŠƒefB,ð'ñ‹Ÿ,µ,Ü,·□B

- ƒvƒ□ƒOƒ‰ƒ€,É,æ,éƒZƒLƒ…ƒŠƒefB

  ƒJƒXƒ^ƒ}ƒCƒY,³,ê,½ƒZƒLƒ…
  ƒŠƒefB,ðƒAƒvƒŠƒP□[ƒVƒ‡ƒ",ÅŽÀŒ»,·,é,½,ß,Ìƒ fCƒ"ƒ^□[ƒtƒFƒCƒX,ð'ñ‹Ÿ,µ,Ü,·□Bƒvƒ□ƒOƒ‰ƒ€,É,æ,éƒZƒLƒ…
  ƒŠƒefB,ÌŽg,¢•û,Ì□Ú□×,É,Â,¢,Ä,Í□A□wProgrammer's Guide□x,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

- □éŒ¾,É,æ,éƒZƒLƒBƒŠƒefB

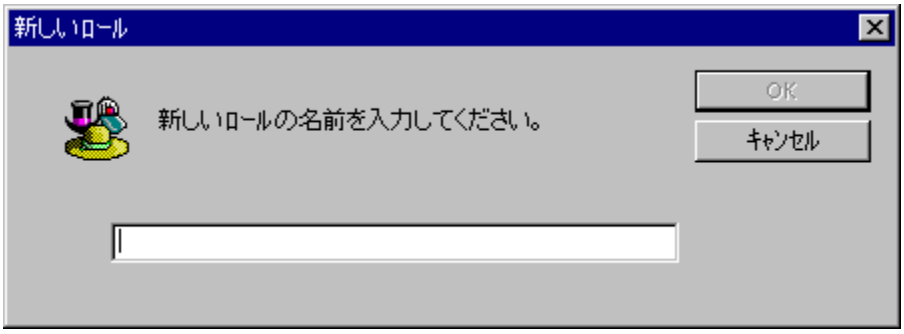
  ƒ□□[ƒ‹,ð'è‹`,µ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä Windows NT
  ƒt□[ƒU□[,Ü,½,Íƒt□[ƒU□[,ÌƒOƒ‹□[ƒv,ðƒ□□[ƒ‹,ÉŠ„,è"–,Ä,é,±,Æ,ª,Å,«,Ü,·□B

□d—v ƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,Íƒ□□[ƒ‹,Ìƒ`ƒFƒbƒN,ðƒTƒ|□[ƒg,µ,Ü,¹,ñ□BƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é,É,Í□AƒAƒNƒefBƒu‰»,Ì□Ý'è,ðƒT□[ƒo□[ƒpƒbƒP□[ƒW,É•Ï□X,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□Bƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,ÆƒT□[ƒo□[ƒpƒbƒP□[ƒW,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒAƒNƒefBƒu‰»,Ìƒvƒ□ƒpƒefB,ð□Ý'è,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÇ—□ŽÒ,Í□A□éŒ¾,É,æ,éƒZƒLƒ…ƒŠƒefB,ðŽg,Á,Ä□AƒAƒNƒZƒXŒ ,ðŽ□,ÂƒNƒ‰ƒCƒAƒ"ƒg,¾,¯,ªƒpƒbƒP□[ƒW,ðŽÀ□s,Å,«,é,æ,¤,É,µ□AƒpƒbƒP□[ƒW,ÌƒZƒLƒ…ƒŠƒefB,ðŠm•Û,µ,Ü,·□BƒAƒNƒZƒX,ð‹–‰Â,·,é,É,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å MTS ƒ□□[ƒ‹,Æ Windows NT ,Ìƒt□[ƒU□[ƒAƒJƒEƒ"ƒg,¨,æ,ÑƒOƒ‹□[ƒv ƒAƒJƒEƒ"ƒg,ðŽg,¢,Ü,·□B□éŒ¾,É,æ,éƒZƒLƒ…ƒŠƒefB,Í Windows NT ƒAƒJƒEƒ"ƒg,Ì"F□Ø,ð—˜—p,·,é,Ì,Å□AWindows 95/98 ƒRƒ"ƒsƒ…□[ƒ^,ÅŽÀ□s,µ,Ä,¢,éƒpƒbƒP□[ƒW,É,Í□éŒ¾,É,æ,éƒZƒLƒ…ƒŠƒefB,ðŽg,¤,±,Æ,ª,Å,«,Ü,¹,ñ□B

□éŒ¾,É,æ,éƒZƒLƒ…ƒŠƒefB,ðƒpƒbƒP□[ƒW,É□Ý'è,·,é,É,Í□AŽŸ,ÌŽè□‡,É□],Á,Ä'€□ì,µ,Ü,·□B

1 [□V,µ,¢ƒ□□[ƒ‹] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,Á,Ä□Aƒ□□[ƒ‹,ðƒpƒbƒP□[ƒW ƒŒƒxƒ‹,Å'è‹`,µ,Ü,·□B



□V,µ,¢ƒ□□[ƒ‹,ð'Ç‰Á,·,é•û–@,É,Â,¢,Ä,Í□A□u□V,µ,¢ MTS ƒ□□[ƒ‹,Ì'Ç‰Á□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

2 [ƒ□□[ƒ‹,Éƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,ð'Ç‰Á] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,Á,Ä□Aƒt□[ƒU□[,ðƒ□□[ƒ‹,Éƒ}ƒbƒv,µ,Ü,·□Bƒ□□[ƒ‹,É"K□Ø,Èƒt□[ƒU□[,ªƒ}ƒbƒv,³,ê,Ä,¢,È,¢ƒpƒbƒP□[ƒW,Í□AŒÄ,Ñ□o,·,±,Æ,ª,Å,«,Ü,¹,ñ□B

ƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,ðƒ□□[ƒ‹,É'Ç‰Á,·,é•û–@,É,Â,¢,Ä,Í□A□uMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

3 "Á'è,ìƒRƒ"ƒ|□[ƒlƒ"ƒg,Ü,½,ÍƒCƒ"ƒ^□[ƒtƒFƒCƒX,Ö,ÌƒAƒNƒZƒX,ð□§ŒÀ,·,é□ê□‡,Í□A'è‹`,µ,½ƒ□□[ƒ‹,ðƒRƒ"ƒ|□[ƒlƒ"ƒg,Ü,½,ÍƒCƒ"ƒ^□[ƒtƒFƒCƒX,Ì [ƒ□□[ƒ‹ ƒ□ƒ"ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_,ÉŠ„,è"–,Ä,Ü,·□B

4 ƒpƒbƒP□[ƒW,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ƒZƒLƒ…ƒŠƒeƒB] ƒ^ƒu,Å□AƒpƒbƒP□[ƒW,ìƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,µ,Ü,·□B"F□Ø,ÌŠm"F,ð—LŒø,É,·,é•û–@,É,Â,¢,Ä,Í□A,±,ÌŒã,Ì□à–¾,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒVƒXƒef€ ƒpƒbƒP□[ƒW,ìƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,·,é'O,É□AŒ»□ÝŽg—p,µ,Ä,¢,éƒt□[ƒU□[ƒAƒJƒEƒ"ƒg,ð Administrator ƒ□□[ƒ‹,Éƒ}ƒbƒv,µ,È,¢,Æ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì (ƒ□□[ƒ‹,Ö,ìƒt□[ƒU□[,Ì'Ç‰Á,È,Ç) □\□¬•Ï□X‹@"\,ðŽg—p,Å,«,Ü,¹,ñ□B,»,Ì□ê□‡,Í□AAdministrator ƒ□□[ƒ‹,Éƒ}ƒbƒv,³,ê,Ä,¢,éƒt□[ƒU□[,Æ,µ,Äƒ□ƒOƒlƒ",·,é•K—v,ª, ,è,Ü,·□BŠÇ—□ŽÒ,ªƒVƒXƒef€ ƒpƒbƒP□[ƒW,ÖƒAƒNƒZƒX,Å,«,È,,È,é,±,Æ,ð–hŽ~,·,é,½,ß,É□AŽŸ,Ì□ê□‡□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒGƒ‰□[ƒ□ƒbƒZ□[ƒW,ª•\Ž¦,³,ê,Ü,·□B

- Administrator ƒ□□[ƒ‹,É,Ç,Ìƒt□[ƒU□[,àƒ}ƒbƒv,³,ê,Ä,¢,È,¢,Æ,«,É□AƒVƒXƒef€ ƒpƒbƒP□[ƒW,ìƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,µ,æ,¤,Æ,µ,½□ê□‡□B
- ƒVƒXƒef€ ƒpƒbƒP□[ƒW,ìƒZƒLƒ…ƒŠƒeƒB,ª—LŒø,É,È,Á,Ä,¢,é,Æ,«,É□AAdministrator ƒ□□[ƒ‹,©,ç□ÅŒã,ÉŽc,Á,½ƒt□[ƒU□[ƒAƒJƒEƒ"ƒg,ð□í□œ,µ,æ,¤,Æ,µ,½□ê□‡□B

'□ ƒvƒ‰ƒCƒ}ƒŠ ƒhƒ□ƒCƒ" ƒRƒ"ƒgƒ□□[ƒ‰,Ü,½,ÍƒoƒbƒNƒAƒbƒv ƒhƒ□ƒCƒ" ƒRƒ"ƒgƒ□□[ƒ‰,Æ,È,Á,Ä,¢,éƒT□[ƒo□[,É MTS ,ªƒCƒ"ƒXƒg□[ƒ‹,³,ê,Ä,¢,é□ê□‡□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÅƒpƒbƒP□[ƒW,ðŠÇ—□,·,é,É,Í□Aƒt□[ƒU□[,Íƒhƒ□ƒCƒ"ŠÇ—□ŽÒ,Å,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ƒpƒbƒP□[ƒW,ìƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,µ,È,¢,Æ□AƒRƒ"ƒ|□[ƒlƒ"ƒg,Ü,½,ÍƒCƒ"ƒ^□[ƒtƒFƒCƒX,ìƒ□□[ƒ‹,Í MTS ,É,æ,Á,Äƒ`ƒFƒbƒN,³,ê,Ü,¹,ñ□B,Ü,½□AƒRƒ"ƒ|□[ƒlƒ"ƒg,ìƒZƒLƒ…ƒŠƒeƒB,ð—LŒø,É,µ,È,¢,Æ□AƒRƒ"ƒ|□[ƒlƒ"ƒg,ìƒCƒ"ƒ^□[ƒtƒFƒCƒX,ìƒ□□[ƒ‹,Í MTS ,É,æ,Á,Äƒ`ƒFƒbƒN,³,ê,Ü,¹,ñ□B

[ƒ□□[ƒ‹ ƒ□ƒ"ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_,Éƒ□□[ƒ‹,ðŠ„,è"–,Ä,é•û–@,É,Â,¢,Ä,Í□A□u□V,µ,¢ MTS ƒ□□[ƒ‹,ì'Ç‰Á□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

’□ ƒpƒbƒP□[ƒW,Ìfffƒoƒbƒ O’†,Í□AŒÂ□X,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ü,½,ÍƒpƒbƒP□[ƒW,Ì□é Œ¾,É,æ,éƒZƒLƒ… ƒŠƒefB,ðƒlƒt,É,µ,Ä,¨,,Æ•Ö—˜,Å,·□B

,½,Æ,¦,Î□A□ÝŒÉŠÇ—□,ð□s,¤ Inventory ƒT□[ƒo□[ƒpƒbƒP□[ƒW,Ö,ÌƒAƒNƒZƒX,ð□§ŒÀ,·,é— á,ð□l,¦,Ä,Ý,Ü,·□BƒVƒXƒef€ŠÇ—□ŽÒ,Í□AInventory ƒpƒbƒP□[ƒW,Ö,ÌƒAƒNƒZƒXŒ ,ð”Ì”„•”–å,Ìƒ□ƒ“ƒo□[,¾,¯,É‹– ‰Â,µ,½,¢,Æ□l,¦,Ä,¢,Ü,·□B,±,Ì□ê□‡,Í□A,Ü,¸ Inventory ƒpƒbƒP□[ƒW,Ì [ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,ð‘I‘ð,µ,Ü,·□B[“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ□A[ƒ□□[ƒ‹] ,ðƒNƒŠƒbƒN,µ,Ü,·□BŽŸ,É□A□V,µ,¢ƒ□□[ƒ‹,Ì– ¼‘O,Æ,µ,Ä□uSales□v,Æ“ü—Í,µ,Ü,·□BŽŸ,É□A[ƒ†□[ƒU□[] ƒtƒHƒ‹ƒ_,ð‘I‘ð,µ,Ü,·□B[“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ□A[ƒ†□[ƒU□[] ,ðƒNƒŠƒbƒN,µ,Ü,·□BŽŸ,É□A”Ì”„•”–å,Ì Windows NT ƒOƒ‹□[ƒv ƒAƒJƒEƒ“ƒg–¼,ð“ü—Í,µ,Ü,·□BŠeƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì [ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_,É Sales ƒ□□[ƒ‹,ð’Ç ‰Á,µ,Ü,·□B,±,ÌŽž“_,Å□A”Ì”„•”–å,ÌƒƒƒbƒZ□[ƒW,¾,¯,ª Inventory ƒpƒbƒP□[ƒW,Ö,ÌƒAƒNƒZƒX,ð‹– ‰Â,³,ê,Ü,·□B□ÅŒã,É□A,»,ÌƒpƒbƒP□[ƒW,ð‘I‘ð,µ□A[ƒvƒ□ƒpƒefB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ƒZƒLƒ…ƒŠƒefB] ƒ^ƒu,Å□A[”F□Ø,ÌŠm”F,ðŽÀ□s,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒ“,É,µ,ÄƒpƒbƒP□[ƒW,Ì□V,µ,¢ƒZƒLƒ…ƒŠƒefB,Ì□Ý’è,ð— LŒø,É,µ,Ü,·□B

ƒpƒbƒP□[ƒW“à,Ì“Á’è,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ö,ÌƒAƒNƒZƒX,ð□§ŒÀ,·,é□ê□‡,Í□AƒpƒbƒP□[ƒW,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ç,¤ ,µ,ª,Ç,Ì,æ,¤,É,¨ŒÝ,¢,ðŒÄ,Ñ□o,µ,Ä,¢,é,©,ð—□‰ð,µ,Ä,¨,©,È,¯,ê,Î,È,è,Ü,¹,ñ□BƒRƒ“ƒ|□[ƒlƒ“ƒg,ªƒx□[ƒX ƒNƒ ‰ƒCƒAƒ“ƒg,©,ç’¼□ÚŒÄ,Ñ□o,³,ê,é□ê□‡□AMTS ,ÍƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒ□□[ƒ‹,ðƒ`ƒFƒbƒN,µ,Ü,·□BƒRƒ“ƒ| □[ƒlƒ“ƒg,ª“¯,¶ƒpƒbƒP□[ƒW“à,Ì•Ê,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŒÄ,Ñ□o,·□ê□‡□A“¯,¶ƒpƒbƒP□[ƒW“à,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Í‘ŠŒÝ,É "□M—Š,µ,Ä,¢,é" ,à,Ì,ÆŒ©,È,³,ê,é,½,ß□AMTS ,Íƒ□□[ƒ‹,ðƒ`ƒFƒbƒN,µ,Ü,¹,ñ□B

,½,Æ,¦,Î□AƒNƒ‰ƒCƒAƒ“ƒg,É (□ÝŒÉ,ðŠm”F,·,é) CheckInventory ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŒÄ,Ñ□o,µ,ð‹– ‰Â,·,éƒ□□[ƒ‹,ð□\□¬,µ□AƒNƒ‰ƒCƒAƒ“ƒg,ª’¼□Ú (Žó’□Žc□,,ðŠm”F,·,é) Backorder ƒRƒ“ƒ| □[ƒlƒ“ƒg,ðŒÄ,Ñ□o,·,±,Æ,ð‹ÖŽ~,·,é—á,ð□l,¦,Ä,Ý,Ü,·□BCheckInventory ƒRƒ“ƒ|□[ƒlƒ“ƒg,Æ Backorder ƒRƒ“ƒ| □[ƒlƒ“ƒg,Í□A—¼•û,Æ,à Inventory ƒpƒbƒP□[ƒW“à,É, ,è,Ü,·□B,±,Ì□ê□‡,Í□A,Ü,¸ CheckInventory ƒRƒ“ƒ| □[ƒlƒ“ƒg,ÉƒNƒ‰ƒCƒAƒ“ƒg,Ì“K□Ø,Èƒ□□[ƒ‹,ð□Ý’è,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BŽŸ,É□AƒNƒ‰ƒCƒAƒ“ƒg ID ,Éƒ}ƒbƒv,Å,«,éƒ□□[ƒ‹,ª Backorder ƒRƒ“ƒ|□[ƒlƒ“ƒg,É□Ý’è,³,ê,Ä,¢,È,¢,±,Æ,ðŠm”F,µ,Ü,·□BCheckInventory ƒRƒ“ƒ|□[ƒlƒ“ƒg,Æ Backorder ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÍƒpƒbƒP□[ƒW,ð‹¤—L,µ,Ä,¢,é,Ì□Å□ACheckInventory ƒRƒ“ƒ| □[ƒlƒ“ƒg,ª Backorder ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŒÄ,Ñ□o,·,Æ,«,É□Aƒ□□[ƒ‹,ðƒ`ƒFƒbƒN,³,ê,Ü,¹,ñ□B

ŽŸ,Ì□ðŒ□,ð–ž,½,µ,Ä,¢,é□ê□‡□ACheckInventory ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÍƒNƒ‰ƒCƒAƒ“ƒg,Ì‘ã,í,è,É Backorder ƒRƒ“ƒ| □[ƒlƒ“ƒg,ðŒÄ,Ñ□o,·,±,Æ,ª,Å,«,Ü,·□B

- ƒNƒ‰ƒCƒAƒ“ƒg ID ,ª CheckInventory ,Ì“K□Ø,Èƒ□□[ƒ‹,Éƒ}ƒbƒv,³,ê,Ä,¢,é□B
- ƒvƒ□ƒOƒ‰ƒ€,É,æ,éƒZƒLƒ…ƒŠƒefB,Ì, ,ç,ä,é•K—v□ðŒ□,ª–ž,½,³,ê,Ä,¢,é□B

,±,Ì,æ,¤,É,µ,Ä□A‘I‘ð,³,ê,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ö,ÌƒAƒNƒZƒX,ð□§ŒÀ,µ,È,ª,ç□A‘ŠŒÝ,É□M—Š,³,ê,é•¡□”,ÌƒRƒ“ƒ| □[ƒlƒ“ƒg,ðŠÜ,Þƒpƒbƒ P□[ƒW,ð□ì□¬,·,é,±,Æ,ª,Å,«,Ü,·□B

Backorder ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð’¼□ÚŒÄ,Ñ□o,·□Å□‰,ÌŒÄ,Ñ□o,µŒ³,Ìƒ□□[ƒ‹,ðƒ`ƒFƒbƒN,·,é,æ,¤ ,É□Ý’è,·,é,É,Í□ABackorder ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì [ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_,ð‘I‘ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒ□□[ƒ‹] ,ðƒNƒŠƒbƒN,µ□ASales ƒ□□[ƒ‹,ð‘I‚Ñ,ê,Å (ƒt□[ƒU□[,ðƒ}ƒbƒv,³,ê,½) Sales ƒ□□[ƒ‹,ª Backorder ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉŠ„,è“–,Ä,ç,ê,½,Ì,Å□A”Ì”„•”– å,Ìƒ□ƒ“ƒo□[,¾,¯,ª Backorder ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŽÀ□s,µ,Ä□ÝŒÉ□Ø,Ì□¤•i,ðŠm”F,·,é,± ,Æ,ª,Å,«,Ü,·□B□V,µ,¢ƒZƒLƒ…ƒŠƒefB,Ì□Ý’è,ð—LŒø,É,·,é,É,Í□AInventory ƒpƒbƒP□[ƒW,Æ Backorder ƒRƒ“ƒ| □[ƒlƒ“ƒg,Ì [”F□Ø,ÌŠm”F,ðŽÀ□s,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒlƒ“,É,µ,Ü,·□B

ƒ□□[ƒ‹,Ìƒ`ƒFƒbƒN,Ì□Ú□×,É,Â,¢,Ä,Í□A□wProgrammer's Guide□x,Ì□uProgrammatic Security□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒZƒLƒ…ƒŠƒefB”F□Ø,ð—LŒø,É,·,é,É,Í

1. ƒVƒXƒef€ ƒpƒbƒP□[ƒW,Ì Administrator ƒ□□[ƒ‹,É,Ü,¾Ž©•ª,Ìƒ†□[ƒU□[ƒAƒJƒEƒ“ƒg,ðƒ}ƒbƒv,µ,Ä,¢ ,È,¢□ê□‡,Í□Aƒ}ƒbƒv,µ,Ü,·□B
2. ƒVƒXƒef€ ƒpƒbƒP□[ƒW,ð‘I‘ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ü,½,Íƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,½,Æ,«,É•\ Ž¦,³,ê,éƒ□ƒjƒ…□[,Ì [ƒvƒ□ƒpƒefB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

3 [ƒZƒLƒ…ƒŠƒefB] ƒ^ƒu,ðƒNƒŠƒbƒN,µ□A[”F□Ø,ÌŠm”F,ðŽÀ□s,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒIƒ“,É,µ,Ü,·□B

4 ƒVƒXƒef€ ƒpƒbƒP□[ƒW,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒVƒƒƒbƒgƒ_ƒEƒ“] ,ðƒNƒŠƒbƒN,µ,ÄƒVƒXƒef€ ƒpƒbƒP□[ƒW,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,ð’âŽ~,µ,Ü,·□B

Žè□‡ 4 ,ÆŽè□‡ 7 ,ðˆê“x,É□s,Á,Ä□A,·,×,Ä,ÌƒT□[ƒo□[ƒpƒbƒP□[ƒW,ðƒVƒƒƒbƒgƒ_ƒEƒ“,·,é,± ,Æ,à,Å,«,Ü,·□B,·,×,Ä,ÌƒT□[ƒo□[ƒpƒbƒP□[ƒW,ðƒVƒƒƒbƒgƒ_ƒEƒ“,·,é,É,Í□A[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ƒAƒCƒRƒ“,ð‘I‘ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ì [ƒT□[ƒo□[ƒvƒ□ƒZƒX,ÌƒVƒƒƒbƒgƒ_ƒEƒ“] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

5 ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,éƒpƒbƒP□[ƒW,ð‘I‘ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ü,½,Íƒ}ƒEƒX,Ì ‰Eƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,½,Æ,«,É•\Ž¦,³,ê,éƒ□ƒjƒ…□[,Ì [ƒvƒ□ƒpƒefB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

6 [ƒZƒLƒ…ƒŠƒefB] ƒ^ƒu,ðƒNƒŠƒbƒN,µ□A[”F□Ø,ÌŠm”F,ðŽÀ□s,·,é] ƒ`ƒFƒbƒN ƒ{ƒbƒNƒX,ðƒIƒ“,É,µ,Ü,·□B

7 ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,éƒpƒbƒP□[ƒW,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒVƒƒƒbƒgƒ_ƒEƒ“] ,ðƒNƒŠƒbƒN,µ,ÄƒVƒXƒef€ ƒpƒbƒP□[ƒW,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,ð’âŽ~,µ,Ü,·□B

”z’u□æ,ÌƒT□[ƒo□[,ÉƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,µ□A□\□¬,µ,½,ç□AƒRƒ“ƒ|□[ƒIƒ“ƒg,Ì□\□¬,ª•Ï□X,³,ê,È,¢,æ,¤ ,ÉƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é,±,Æ,ª,Å,«,Ü,·□BƒpƒbƒP□[ƒW,Ì□\□¬,ðƒ□ƒbƒN,·,é•û–@,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜A□€–Ú

System ƒpƒbƒP□[ƒW□A[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_□A[ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_□A[ƒ□□[ƒ‹ ƒ□ƒ“ƒo□[ƒVƒbƒv] ƒtƒHƒ‹ƒ_□AMTS ƒ□□[ƒ‹,Ìƒt□[ƒU□[,ðŠÇ—□,·,é□AMTS Overview and Concepts (ŠT—v,Æƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO,ÌŠT”O)

MTS ƒpƒbƒP□[ƒW ID ,Ì□Ý'è

ƒpƒbƒP□[ƒW ('Pˆê,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX) ,ðŽŸ,Ì,¢,¸,ê,©,ÌƒpƒbƒP□[ƒW ID ,Æ,µ,ÄŽÀ□s,·,é,æ,¤,É□\□¬,Å,«,Ü,·□B

- 'Î˜bƒt□[ƒU□[
- "Á'è,Ì Windows NT ƒt□[ƒU□[ƒAƒJƒEƒ"ƒg

Šù'è,Ì□Ý'è,Å,Í□AMTS ƒpƒbƒP□[ƒW,Í'Î˜bƒt□[ƒU□[,Æ,µ,ÄŽÀ□s,³,ê,Ü,·□B

'□ ƒpƒbƒP□[ƒW ID ,ÉƒOƒ‹□[ƒv,ðŽw'è,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

'½,,Ì"z'uŠÂ‹«,Å,Í□AƒpƒbƒP□[ƒW,ð Windows NT ƒt□[ƒU□[ƒAƒJƒEƒ"ƒg,Æ,µ,ÄŽÀ□s,·,é,±
,Æ,ð,¨Š©,ß,µ,Ü,·□BƒpƒbƒP□[ƒW,ª'Pˆê,Ì Windows NT
ƒt□[ƒU□[ƒAƒJƒEƒ"ƒg,Æ,µ,ÄŽÀ□s,³,ê,é□ê□‡,Í□AƒpƒbƒP□[ƒW,ðŽg—p,·,é,·,×,Ä,ÌƒNƒ
‰ƒCƒAƒ"ƒg,Éƒf□[ƒ^ƒx□[ƒX,Ö,ÌƒAƒNƒZƒX,ð□\
□¬,·,é'ã,í,è,É□A,»,ÌƒAƒJƒEƒ"ƒg,¾,¯,Éƒf□[ƒ^ƒx□[ƒX,Ö,ÌƒAƒNƒZƒX,ð□\□¬,·,é,±,Æ,ª,Å,«,Ü,·□BŒÂ□X,ÌƒNƒ
‰ƒCƒAƒ"ƒg,ÉƒAƒNƒZƒX,ð‹–‰Â,·,é,Ì,Å,Í,È,□AƒAƒJƒEƒ"ƒg,ÉƒAƒNƒZƒX,ð‹–‰Â,·,é,Æ□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ",ÌƒXƒP□[ƒ
‰ƒrƒŠƒefB,ªŒü□ã,µ,Ü,·□B

,½,Æ,¦,Î□ASQL Sever ƒf□[ƒ^ƒx□[ƒX,Ì□¿‹□□î•ñ□A"Ì"„□î•ñ,ð□X□V,·,é Accounting ƒpƒbƒP□[ƒW,Ì—
á,ð□I,¦,Ä,Ý,Ü,·□Bƒf□[ƒ^ƒx□[ƒX,Ì Accounting ƒe□[ƒuƒ‹,ð□\□¬,µ□AWindows NT
ƒt□[ƒU□[,É□u"Ç,ÝŽæ,è□vƒAƒNƒZƒXŒ ,ð—^,¦,é,±,Æ,ª,Å,«,Ü,·□B

ƒpƒbƒP□[ƒW ID ,ð□Ý'è,µ,½,Æ,«□AMTS ,Í"ü—
Í,³,ê,½ƒpƒXƒ□□[ƒh,ðŠm"F,µ,Ü,·□B,½,¾,µ□AƒAƒJƒEƒ"ƒg,ÌƒpƒXƒ□□[ƒh,ð•Ï□X,µ,½,Æ,«,É MTS ƒGƒNƒXƒvƒ□□[ƒ‰
,ÅƒpƒXƒ□□[ƒh,ð□X□V,µ,È,©,Á,½□ê□‡,Í□AƒpƒbƒP□[ƒW,ÍŽÀ□s,Å,«,Ü,¹,ñ□B

ƒpƒbƒP□[ƒW ID ,ð"Á'è,Ìƒt□[ƒU□[ƒAƒJƒEƒ"ƒg,É□Ý'è,·,é,É,Í

1. ID ,ð•Ï□X,·,éƒpƒbƒP□[ƒW,ð'I'ð,µ,Ü,·□B
2. ["®□ì] ƒ□ƒjƒ…□[,Ì [ƒvƒ□ƒpƒefB] ,ðƒNƒŠƒbƒN,µ□A[ID] ƒ^ƒu,ðƒNƒŠƒbƒN,µ,Ü,·□B
3. [,±,Ìƒt□[ƒU□[] ,ð'I'ð,µ□AWindows NT ƒt□[ƒU□[ƒAƒJƒEƒ"ƒg,Ìƒt□[ƒU□[–¼,ÆƒpƒXƒ□□[ƒh,ð"ü—
   Í,µ,Ü,·□Bƒt□[ƒU□[–¼,Í□Aƒt□[ƒU□[,Ìƒhƒ□ƒCƒ"–¼,Æ‰~‹L□† (\) ,É'±,¯,Ä"ü—Í,µ,Ü,·□B
   ƒf□[ƒ^ƒx□[ƒX,Ö,ÌƒAƒNƒZƒX,ð□§ŒÀ,·,é,½,ß,ÉƒpƒbƒP□[ƒW ID ,ðŽg—
   p,·,é□ê□‡,Í□Aƒt□[ƒU□[ƒAƒJƒEƒ"ƒg,Éƒf□[ƒ^ƒx□[ƒX,Ö,ÌƒAƒNƒZƒXŒ ŒÀ,ð□Ý'è,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

MTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é□A□V,µ,¢ MTS ƒ□□[ƒ‹,Ì'Ç‰Á□A[ID] ƒ^ƒu (ƒpƒbƒP□[ƒW)

□V,µ,¢ MTS ƒ□□[ƒ‹,Ì’Ç‰Á

’Ê□í□AƒpƒbƒP□[ƒW,ìŠJ”Žž,É□V,µ,¢ƒ□□[ƒ‹,ð’Ç‰Á,µ,Ü,·,ª□AŠù‘¶,ÌƒpƒbƒP□[ƒW,É□V,µ,¢ƒ□□[ƒ‹,ð’Ç‰Á,µ,È,¯,ê,Î,È,ç,È,¢,±,Æ,à, ,è,Ü,·□Bƒ□□[ƒ‹,Æ,Í□A“Á’è,Ì□E–±,É•K—v,È 1 ‘g,ÌƒVƒXƒeƒ€ƒŒƒxƒ‹,ÌŒ ŒÀ,Å,·□Bƒ□□[ƒ‹,ÍƒpƒbƒP□[ƒW ƒŒƒxƒ‹,Å□Ý’è,³,ê,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ðŽg,Á,Ä□A□ì□¬,·,éƒ□□[ƒ‹,É Windows NT ƒt□[ƒU□[,Ü,½,Íƒt□[ƒU□[,ÌƒOƒ‹□[ƒv,ðƒ}ƒbƒv,·,é,±,Æ,ª,Å,«,Ü,·□B

□V,µ,¢ƒ□□[ƒ‹,ð□ì□¬,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Å□Aƒ□□[ƒ‹,ð‘g,Ý□ž,ÞƒpƒbƒP□[ƒW,ð‘I‘ð,µ,Ü,·□B
2. [ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
3. ŽŸ,Ì,¢,¸,ê,©,Ì‘€□ì,ðŽÀ□s,µ,Ü,·□B
   - [“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ□A[ƒ□□[ƒ‹] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒc□[ƒ‹ ƒo□[,Ì [□V,µ,¢ƒIƒuƒWƒFƒNƒg,Ì□ì□¬] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - [ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,ðƒ}ƒƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒ□□[ƒ‹] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
4. •\Ž¦,³,ê,éƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□A□V,µ,¢ƒ□□[ƒ‹,Ì–¼‘O,ð“ü—Í,µ,Ü,·□B
5. [OK] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

Œx□□ —LŒø,Èƒt□[ƒU□[,ðƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,Éƒ}ƒbƒv,µ,È,¢,Æ□AƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

MTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é□AMTS ƒpƒbƒP□[ƒW ID ,Ì□Ý’è□A[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_□AMTS ƒ□□[ƒ‹,Ìƒt□[ƒU□[,ðŠÇ—□,·,é

MTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒCƒ“ƒXƒg□[ƒ‹,µ□A”z’u,·,é,Æ,«,Í□A,·,×,Ä,ÌŠù‘¶,Ìƒ□□[ƒ‹,É Windows NT ƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,ðŠ„,è“–,Ä,È,¯,ê,Î,È,è,Ü,¹,ñ□Bƒ□□[ƒ‹,Í□AƒRƒ“ƒ| □[ƒlƒ“ƒg,ÆƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ö,Ìƒt□[ƒU□[,ÌƒAƒNƒZƒX,ð’è‹`,µ,Ü,·□B

ƒt□[ƒU□[,ðƒ□□[ƒ‹,ÉŠ„,è“–,Ä,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Å□Aƒ□□[ƒ‹,ðŠ„,è“–,Ä,éƒRƒ“ƒ| □[ƒlƒ“ƒg,ªŠÜ,Ü,ê,éƒpƒbƒP□[ƒW,ð‘I‘ð,µ,Ü,·□B
2. [ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
3. ƒt□[ƒU□[,ðŠ„,è“–,Ä,éƒ□□[ƒ‹,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ,Ü,·□B
4. [ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
5. ŽŸ,Ì,¢,¸,ê,©,Ì‘€□ì,ðŽÀ□s,µ,Ü,·□B
   - [“®□ì] ƒ□ƒjƒ…□[,Ì [□V‹K□ì□¬] ,ðƒ|ƒCƒ“ƒg,µ□A[ƒt□[ƒU□[] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒc□[ƒ‹ ƒo□[,Ì [□V,µ,¢ƒIƒuƒWƒFƒNƒg,Ì□ì□¬] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - [ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[□V‹K□ì□¬] ,ðƒ| ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[ƒt□[ƒU□[] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
6. •\Ž¦,³,ê,éƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□Aƒt□[ƒU□[–¼,Ü,½,ÍƒOƒ‹□[ƒv,ðƒ□□[ƒ‹,É’Ç‰Á,µ,Ü,·□B[ƒt□[ƒU□[,Ì•\Ž¦] ƒ{ƒ^ƒ“,¨,æ,Ñ [ŒŸ□õ] ƒ{ƒ^ƒ“,ðŽg,Á,Ä□Aƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,ð’T,·,±,Æ,ª,Å,«,Ü,·□B
7. [OK] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

’□ Windows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,Åƒt□[ƒU□[,ðƒ□□[ƒ‹,ÉŠ„,è“–,Ä,é,É,Í□AWindows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,Éƒt□[ƒU□[ƒŒƒxƒ‹,ÌƒAƒNƒZƒXŠÇ—□,ð□Ý’è,·,é•K—v,ª, ,è,Ü,·□BWindows 95/98 ƒRƒ“ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,ðŠJ,¢,Ä [ƒlƒbƒgƒ□□[ƒN] ,ðƒNƒŠƒbƒN,µ,Ü,·□B[ƒAƒNƒZƒXŒ ,ÌŠÇ—□] ƒ^ƒu,ðƒNƒŠƒbƒN,µ,Ä [ƒt□[ƒU□[ƒŒƒxƒ‹,ÌƒAƒNƒZƒXŠÇ—□] ,ð‘I‘ð,µ,Ü,·□B,±,Ì□Ý’è,Ì□Ú□×,É,Â,¢,Ä,Í□AWindows 95/98 ,Ìƒ}ƒjƒ… ƒAƒ‹,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜A□€–Ú

□V,µ,¢ MTS ƒ□□[ƒ‹,Ì’Ç‰Á□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é□AMTS ƒpƒbƒP□[ƒW ID ,Ì□Ý’è□A[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_□A[ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_□AMTS ƒ□□[ƒ‹,Ìƒt□[ƒU□[,ðŠÇ—□,·,é

MTS ƒpƒbƒP□[ƒW,Ì•ÛŽç

ƒpƒbƒP□[ƒW,Ì□ì¬□A”z•z□A”z’u,É□¬Œ÷,µ,½,ç□AŠÇ—□ŽÒ,Í MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä MTS ƒpƒbƒP□[ƒW,ð•ÛŽç,·,é,±,Æ,ª,Å,«,Ü,·□BMTS ƒpƒbƒP□[ƒW,Ì•ÛŽç,É,Í□AƒpƒbƒP□[ƒW,ÆƒpƒbƒP□[ƒW,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□ó‘Ô,¨,æ,Ñƒvƒ□ƒpƒeƒB,ðŠÄŽ‹,·,é,±,Æ□A•K—v,É‰ž,¶,ÄƒpƒbƒP□[ƒW,ÆƒpƒbƒP□[ƒW,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒpƒeƒB,ð□Ä□\□¬,·,é,±,Æ,ªŠÜ,Ü,ê,Ü,·□B

,±,±,Å,Í□AŽŸ,Ìƒgƒsƒbƒ‰N,É,Â,¢,Ä□à–¾,µ,Ü,·□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□ó‘Ô,Æƒvƒ□ƒpƒeƒB,ðŠÄŽ‹,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,¤

MTS ƒ□□[ƒ‹,Ìƒt□[ƒU□[,ðŠÇ—□,·,é

MTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽg,¤

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□ó‘Ô,Æƒvƒ□ƒpƒeƒB,ðŠÄŽ‹,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì□ó‘Ô•\Ž¦,Æƒvƒ□ƒpƒeƒB•\Ž¦,ðŽg,Á,Ä□AƒAƒNƒeƒBƒu,È□ƒpƒbƒP□[ƒW,Æ,»,Ìƒfƒ‚□[ƒlƒ“ƒg,Ì□ó‘Ô,¨,æ,Ñƒvƒ□ƒpƒeƒB,ðŠÄŽ‹,Å,«,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÅƒAƒCƒeƒ€,ð‘I‘ð,µ□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ƒc□[ƒ‹ ƒo□[,Ì [□ó‘Ô•\Ž¦] ƒ{ƒ^ƒ“,Ü,½,Í [ƒvƒ□ƒpƒeƒB•\Ž¦] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,·,é,Æ□AƒAƒCƒeƒ€,Ì□ó‘Ô,âƒvƒ□ƒpƒeƒB,ð•\Ž¦,Å,«,Ü,·□B

□ó‘Ô•\Ž¦,ðŽg,¤,Æ□AƒRƒ“ƒsƒ…□[ƒ^□AƒpƒbƒP□[ƒW□A,¨,æ,ÑƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□ó‘Ô,ð MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å•\Ž¦,·,é,±,Æ,ª,Å,«,Ü,·□B,½,Æ,¦,Î□A[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,Ì□ó‘Ô•\Ž¦,Å,Í□AMicrosoft •ªŽÙƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒl□[ƒ^ (MS DTC) ,ª‹N“®,³,ê,Ä,¢,é,©,Ç,¤,©,ðŽ¦,µ□A[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,Ì□ó‘Ô•\Ž¦,Å,Í□A,Ç,ÌƒpƒbƒP□[ƒW,ªŽÀ□s,³,ê,Ä,¢,é,©,ðŽ¦,µ,Ü,·□BƒpƒbƒP□[ƒW“à,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□ó‘Ô•\Ž¦,Å,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒOƒ‰ƒ€ ID (progID)□AƒAƒNƒeƒBƒu‰»,³,ê,Ä,¢,éƒIƒuƒWƒFƒNƒg□A,¨,æ,ÑŒÄ,Ñ□o,³,ê,Ä,¢,éƒIƒuƒWƒFƒNƒg,ðŽ¦,µ,Ü,·□B□ó‘Ô•\Ž¦,ðŽg,Á,Ä□AŠÇ—□,·,éƒRƒ“ƒsƒ…□[ƒ^,ÆƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒvƒ□ƒZƒX,Ì□ó‘Ô,ð,·,Î,â,•\Ž¦,·,é,±,Æ,ª,Å,«,Ü,·□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å□ó‘Ô•\Ž¦,ðŽg,¤,É,Í

1. [ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_□A[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_□A,Ü,½,Í [ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,ð‘I‘ð,µ,Ü,·□B
2. MTS ƒc□[ƒ‹ ƒo□[,Ì [□ó‘Ô•\Ž¦] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B

’□ □ó‘Ô•\Ž¦,Å,Í□Aƒ†□[ƒeƒBƒŠƒeƒB,¨,æ,ÑƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW,Ì□ó‘Ô,Í•\Ž¦,³,ê,Ü,¹,ñ□B

ƒpƒbƒP□[ƒW,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒpƒeƒB,ÉŠÖ,·,é□î•ñ,ð,·,Î,â,Žû□W,·,é,É,Í□Aƒvƒ□ƒpƒeƒB•\Ž¦,ðŽg,¢,Ü,·□B,½,Æ,¦,Î□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒpƒeƒB•\Ž¦,Å,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒOƒ‰ƒ€ ID (ProgID)□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì□Ý’è□Aƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL) ,Ì□ê□Š□AƒNƒ‰ƒX ID (CLSID)□AƒXƒŒƒbƒhƒ‚ƒfƒ‹□A,¨,æ,ÑƒZƒLƒ…ƒŠƒeƒB”F□Ø,Ì□Ý’è,ª•\Ž¦,³,ê,Ü,·□B,Ü,½□Aƒvƒ□ƒpƒeƒB•\Ž¦,ðŽg,Á,Ä□AƒpƒbƒP□[ƒW“à,Ì,·,×,Ä,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒvƒ□ƒpƒeƒB,ð,·,Î,â,•\Ž¦,·,é,±,Æ,ª,Å,«,Ü,·□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Åƒvƒ□ƒpƒeƒB•\Ž¦,ðŽg,¤,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Å□A[ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,â [ƒRƒ“ƒ|□[ƒlƒ“ƒg] ƒtƒHƒ‹ƒ_,È,Ç□A□ó‘Ô,ð•\Ž¦,·,éƒtƒHƒ‹ƒ_,ð‘I‘ð,µ,Ü,·□B
2. MTS ƒGƒNƒXƒvƒ□□[ƒ‰ ƒc□[ƒ‹ ƒo□[,Ì [ƒvƒ□ƒpƒeƒB•\Ž¦] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,Ü,·□B‘I‘ð,µ,½ƒtƒHƒ‹ƒ_“à,ÌƒAƒCƒeƒ€,Ì□ó‘Ô,ª□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì‰E‘¤,Ì—ñ,É•\Ž¦,³,ê,Ü,·□B

ŠÖ˜A□€–Ú

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,¤

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,¤

ƒpƒbƒP□[ƒW,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì[ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å,Í□AMTS ƒpƒbƒP□[ƒW,Ì□\□¬,ð•Ï□X,Å,«,Ü,·□B,½,Æ,¦,Î□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒvƒ□ƒpƒeƒB,ð•Ï□X,·,é□ê□‡,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ,Ì□Ý’è,ð•Ï□X,·,é,± ,Æ,ª,Å,«,Ü,·□B“Á’è,Ìƒvƒ□ƒpƒeƒB,Ì□Ý’è,ð□Ä□\□¬,·,é‘O,É□A□uMTS ƒpƒbƒP□[ƒW,Ì□ì□¬□v,Æ □uMTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹□v,ðŽQ□Æ,µ,Ä□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,Å,³,Ü,´,Ü,Èƒvƒ□ƒpƒeƒB,ð□Ý’è,·,é,± ,Æ,ÌˆÓ–¡,ÆŽè□‡,ðŠm”F,µ,Ä,,¾,³,¢□B

ƒpƒbƒP□[ƒW,Æ,»,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì•Ï□X,â□í□œ,ð–hŽ~,·,é,½,ß,É□AƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é,± ,Æ,ª,Å,«,Ü,·□BƒCƒ“ƒXƒg□[ƒ‹Žž,Ü,½,Í”z’uŽž,ÉƒpƒbƒP□[ƒW,ªƒ□ƒbƒN,³,ê,Ä,¢,é□ê□‡,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒvƒ□ƒpƒeƒB,ð•Ï□X,·,é‘O,É□AƒpƒbƒP□[ƒW,Ìƒ□ƒbƒN,ð‰ð□œ,·,é•K—v,ª, ,è,Ü,·□Bƒ□ƒbƒN,Ì□Ú□×,É,Â,¢ ,Ä,Í□A□uMTS ƒpƒbƒP□[ƒW,ðƒ□ƒbƒN,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

[ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ðŽg,Á,Ä□AƒpƒbƒP□[ƒW,ÆƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìƒvƒ□ƒpƒeƒB,ð□\□¬,·,é,É,Í

1. ŽŸ,Ì,¢,¸,ê,©,Ì•û–@,Å□A[ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,ð•\Ž¦,µ,Ü,·□B
   - ƒvƒ□ƒpƒeƒB,ð□Ý’è,·,éƒAƒCƒef€,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒpƒeƒB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì□¶‘¤,Å–Ú“I,ÌƒAƒCƒeƒ€,ð‘I‘ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ì [ƒvƒ□ƒpƒeƒB] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
2. [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å□A“K□Ø,Èƒvƒ□ƒpƒeƒB,ð•Ï□X,µ□A[OK] ,Ü,½,Í [“K—p] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
3. [ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ƒAƒCƒRƒ“,ð‘I‘ð,µ□A[“®□ì] ƒ□ƒjƒ…□[,Ì [,·,×,Ä,ìƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□X□V] ,ðƒNƒŠƒbƒN,µ,Ä [ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^] ,Ì,·,×,Ä,ìƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□X□V,µ,Ü,·□B,Ü,½,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì ‰E‘¤,Å□A[ƒCƒ“ƒXƒg□[ƒ‹,³,è,½ƒpƒbƒP□[ƒW] ƒtƒHƒ‹ƒ_,ìƒpƒbƒP□[ƒW,ð‘I‘ð,µ□AMTS ƒc□[ƒ‹ ƒo□[,Ì [□Å□V,Ì□î•ñ,É□X□V] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,ÄŒÂ□X,ìƒpƒbƒP□[ƒW,ð□X□V,µ,Ü,·□B

   ’□ ƒpƒbƒP□[ƒW,ÌƒAƒNƒefBƒu‰»ƒŒƒxƒ‹,ð•Ï□X,µ,½□ê□‡□A•Ï□X,ð— LŒø,É,·,é,É,Í□A,»,ìƒpƒbƒP□[ƒW,ìƒvƒ□ƒZƒX,ðƒVƒƒfbƒgƒ_ƒEƒ“,·,é•K—v,ª, ,è,Ü,·□BƒpƒbƒP□[ƒW ƒvƒ□ƒZƒX,ðƒVƒƒfbƒgƒ_ƒEƒ“,·,é,É,Í□A•Ï□X,µ,½ƒpƒbƒP□[ƒW,ðƒ}ƒEƒX,Ì ‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒVƒƒfbƒgƒ_ƒEƒ“] ,ðƒNƒŠƒbƒN,µ,Ü,·□BMTS 1.x ƒRƒ“ƒsƒ…□[ƒ^□ã,ìƒŠƒ‚□[ƒg,ÅŠÇ— □,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,Í□AŒÂ•Ê,ÉƒVƒƒfbƒgƒ_ƒEƒ“,·,é,±,Æ,Í,Å,«,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□ó‘Ô,Æƒvƒ□ƒpƒeƒB,ðŠÄŽ‹,·,é

MTS ƒ□□[ƒ‹,Ìƒt□[ƒU□[,ðŠÇ—□,·,é

ŠÇ—□ŽÒ,Í□AŽŸ,Ì‘€□ì,ðŽÀ□s,µ,Äƒ□□[ƒ‹,Éƒ}ƒbƒv,µ,½ Windows NT ƒt□[ƒU□[,Æƒt□[ƒU□[,ÌƒOƒ‹□[ƒv,ðŠÇ—□,·,é•K—v,ª, ,è,Ü,·□B

- ƒ□□[ƒ‹,É□V,µ,¢ƒt□[ƒU□[,âƒOƒ‹□[ƒv,ð'Ç‰Á,·,é□B
- ƒ□□[ƒ‹,©,çƒt□[ƒU□[,âƒOƒ‹□[ƒv,ð□í□œ,·,é□B
- , ,éƒ□□[ƒ‹,©,ç•Ê,Ìƒ□□[ƒ‹,Öƒt□[ƒU□[,âƒOƒ‹□[ƒv,ðˆÚ“®,·,é□B

ƒT□[ƒo□[ƒpƒbƒP□[ƒW,ÌƒNƒ‰ƒCƒAƒ“ƒg,Ì'□‰Á,É□‡,í,¹,Ä□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ðŽg,Á,Ä□V,µ,¢ƒt□[ƒU□[,ðƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,Éƒ}ƒbƒv,µ,Ü,·□B□V,µ,¢ƒt□[ƒU□[,ðƒ□□[ƒ‹,Éƒ}ƒbƒv,·,é•û–@,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒ□□[ƒ‹,©,çƒt□[ƒU□[,ð□í□œ,µ,½,è□AŠù'¶,Ìƒ□□[ƒ‹,©,ç□V,µ,-□ì□¬,µ,½ƒ□□[ƒ‹,Öƒt□[ƒU□[,Ü,½,ÍƒOƒ‹□[ƒv,ðˆÚ“®,·,é•K—v,ª, ,é□ê□‡,ª, ,è,Ü,·□B,½,Æ,¦,Î□A□]‹Æˆõ,ª‰ïŽÐ,ðŽ«,ß,½□ê□‡,Í□AŽ«,ß,½□]‹Æˆõ,ÉŠÖ˜A•t,¯,ç,ê,Ä,¢,½ƒ□□[ƒ‹,©,ç,»,Ìƒt□[ƒU□[,ð□í□œ,µ,Ü,·□B

Šù'¶,ÌƒpƒbƒP□[ƒW,Ì□V,µ,¢ƒ□□[ƒ‹,ð□ì□¬,·,é,Æ,«,É□A, ,éƒ□□[ƒ‹,©,ç•Ê,Ìƒ□□[ƒ‹,Öƒt□[ƒU□[,Ü,½,ÍƒOƒ‹□[ƒv,ðˆÚ“®,·,é•K—v,ª, ,é□ê□‡,ª, ,è,Ü,·□B,½,Æ,¦,Î□AAccounting ƒpƒbƒP□[ƒW,Ì Clerk ƒ□□[ƒ‹,ð□ì□¬,·,é□ê□‡,Í□AAccounting ƒf□[ƒ^,Ì“Ç,ÝŽæ,èŒ ŒÀ,Æ□',«□ž,ÝŒ ŒÀ,ðŽ□,ÂŠù'¶,Ì Manager ƒ□□[ƒ‹,©,ç□A“Ç,ÝŽæ,èŒ ŒÀ,¾,¯,ðŽ□,Â Clerk ƒ□□[ƒ‹,Éˆê•”,Ìƒt□[ƒU□[,Ü,½,Íƒt□[ƒU□[,ÌƒOƒ‹□[ƒv,ðˆÚ“®,·,é•K—v,ª, ,é,©,à,µ,ê,Ü,¹,ñ□Bƒt□[ƒU□[,Ü,½,Íƒt□[ƒU□[,ÌƒOƒ‹□[ƒv,ðˆÚ“®,·,é,É,Í□Aƒ□□[ƒ‹,©,çƒt□[ƒU□[,Ü,½,Íƒt□[ƒU□[,ÌƒOƒ‹□[ƒv,ð□í□œ,µ□A□í□œ,µ,½ƒt□[ƒU□[,ð□V,µ,¢ƒ□□[ƒ‹,Éƒ}ƒbƒv,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ƒt□[ƒU□[,ðƒ□□[ƒ‹,©,ç□í□œ,·,é,É,Í

1. ƒOƒ‹□[ƒv,Ü,½,Íƒt□[ƒU□[,ð□í□œ,·,éƒ□□[ƒ‹,ªŠÜ,Ü,ê,Ä,¢,éƒpƒbƒP□[ƒW,ð'T,µ,Ü,·□BMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ÌƒEƒBƒ“ƒhƒE,Ì□¶'¤,Å□A,»,ÌƒpƒbƒP□[ƒW,ð'I'ð,µ,Ü,·□B
2. [ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
3. □í□œ,·,éƒt□[ƒU□[,ªˆê‹`,³,ê,Ä,¢,éƒ□□[ƒ‹,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,µ,Ü,·□B
4. [ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_,ðŠJ,«,Ü,·□B
5. □í□œ,·,éƒt□[ƒU□[,ð'I'ð,µ,Ü,·□B
6. ŽŸ,Ì,¢,¸,ê,©,Ì'€□ì,ðŽÀ□s,µ,Ü,·□B
   - ["®□ì] ƒ□ƒjƒ…□[,Ì [□í□œ] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - □í□œ,·,éƒt□[ƒU□[,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[□í□œ] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
   - **Del** ƒL□[,ð‰Ÿ,µ,Ü,·□B
7. •\Ž¦,³,ê,éƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Å [,Í,¢] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ŠÖ˜A□€–Ú

[ƒ□□[ƒ‹] ƒtƒHƒ‹ƒ_□A[ƒt□[ƒU□[] ƒtƒHƒ‹ƒ_□AMTS ƒpƒbƒP□[ƒW ƒZƒLƒ…ƒŠƒefB,ð—LŒø,É,·,é□AMTS ƒpƒbƒP□[ƒW ID ,Ì□Ý'è□A□V,µ,¢ MTS ƒ□□[ƒ‹,Ì'Ç‰Á□AMTS ƒ□□[ƒ‹,ðƒt□[ƒU□[,ÆƒOƒ‹□[ƒv,Éƒ}ƒbƒv,·,é

MTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽg,¤

MTS ,É,Í□AMicrosoft Cluster Server (MSCS) ,Æ‘g,Ý□‡,í,¹,ÄŽg—p,Å,«,éƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“‹@”\,ª—p^Ó,³,ê,Ä,¢,Ü,·□BƒNƒ‰ƒXƒ^“à,Ì MTS ƒT□[ƒo□[,ªŒÌ□á,µ,½,è□Aƒlƒtƒ‰ƒCƒ“,É,È,Á,½,è,µ,½□ê□‡,Í□A,à,¤^ê•û,ÌƒT□[ƒo□[,ªŒÌ□á,µ,½ƒT□[ƒo□[,Ì“®□ì,ðˆø,«Œp,¬,Ü,·□B

MTS ƒT□[ƒo□[,ðƒŒƒvƒŠƒP□[ƒg,·,é•û–@

MTS ƒT□[ƒo□[,ðƒŒƒvƒŠƒP□[ƒg,·,é,É,Í

1. MTS ,ðƒNƒ‰ƒXƒ^,Ì—¼•û,ÌƒRƒ“ƒsƒ…□[ƒ^,ÉƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B□Ú□×,É,Â,¢,Ä,Í□A□uMTS ,ð Microsoft Cluster Server ,Æ‘g,Ý□‡,í,¹,ÄŽg,¤,æ,¤,É□\□¬,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B
2. ƒ}ƒXƒ^ ƒRƒ“ƒsƒ…□[ƒ^,ðŽw’è,µ□A,»,ÌƒRƒ“ƒsƒ…□[ƒ^,ÉƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B
3. ƒCƒ“ƒ|□[ƒg,³,ê,½,·,×,Ä,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ð—¼•û,ÌƒRƒ“ƒsƒ…□[ƒ^,É“o˜^,µ,Ü,·□B
4. ƒ‰ƒ“ƒ^ƒCƒ€ ƒ‰ƒCƒuƒ‰ƒŠ,È,Ç,Ì^Ë‘¶ƒtƒ@ƒCƒ‹,ª—¼•û,ÌƒRƒ“ƒsƒ…□[ƒ^,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,é,±,Æ,ðŠm”F,µ,Ü,· (,±,ÌŒã,Ì□u§ŒÀŽ–□€□v,ðŽQ□Æ,µ,Ä,,¾,³,¢)□B
5. [ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^,ÌƒvƒOƒpƒefB] ƒ_ƒCƒAƒOƒO ƒ{ƒbƒNƒX,Ì [ƒIƒvƒVƒ‡ƒ“] ƒ^ƒu,Å□A[ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì‹¤—LƒffBƒŒƒNƒgƒŠ] ƒ{ƒbƒNƒX,Éƒ}ƒXƒ^,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì‹¤—LƒffBƒŒƒNƒgƒŠ–¼,ðŽw’è,µ,Ü,· (,±,ÌŒã,Ì□uƒpƒbƒP□[ƒW,Ì“à—e,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“□v,ðŽQ□Æ,µ,Ä,,¾,³,¢)□B
6. MTXREPL.EXE ƒRƒ}ƒ“ƒhƒ‰ƒCƒ“ ƒt□[ƒefBƒŠƒefB,ðŽÀ□s,µ,Ü,·□B

MTS ƒT□[ƒo□[,ðƒŒƒvƒŠƒP□[ƒg,·,é,É,Í□AMTXREPL.EXE ƒRƒ}ƒ“ƒhƒ‰ƒCƒ“ ƒt□[ƒefBƒŠƒefB,ðŽg,¢,Ü,·□Bƒ}ƒXƒ^,ÆƒŒƒvƒŠƒP□[ƒg□æ,Ì—¼•û,ÌƒRƒ“ƒsƒ…□[ƒ^,ðŽÀ□s,µ,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□BMTXREPL.EXE ,Å,ÍŽŸ,Ìˆø□”,ðŽw’è,µ,Ü,·□B

```
MTXREPL.EXE <ƒ}ƒXƒ^> <ƒŒƒvƒŠƒP□[ƒg□æ>
```

MTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Å,Í□A’Pˆêƒ}ƒXƒ^•ûŽ®,ðŽg—p,µ□AMTS ƒJƒ^ƒ□ƒO□î•ñ,ð ‘ä,Ìƒ}ƒXƒ^ ƒRƒ“ƒsƒ…□[ƒ^,©,ç 1 ‘ä,ÌƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,ÖƒRƒs□[,µ,Ü,·□B•Ê,ÌƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,ÉƒŒƒvƒŠƒP□[ƒg,·,é,É,Í□AMTXREPL.EXE ƒt□[ƒefBƒŠƒefB,ð,à,¤^ê“xŽÀ□s,µ,Ü,·□B’Pˆêƒ}ƒXƒ^•ûŽ®,ÍŒµ–§,É□s,í,ê,é,í,¯,Å,Í,È,¢,Ì,Ä□AƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,ð•Ê,Ì MTS ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ìƒ}ƒXƒ^,Æ,µ,ÄŽg,¤,±,Æ,à,Å,«,Ü,·□B

,±,ÌŒã,Ì□uMTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Å,Í□s,í,ê,È,¢,±,Æ□v,É‹L□q,³,ê,Ä,¢,é□ê□‡,ð□œ,¢,Ä□AƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□s,¤,ÆAMTS ,ÍƒŒƒvƒŠƒP□[ƒg□æ,ÌƒJƒ^ƒ□ƒO,ðƒŒƒvƒŠƒP□[ƒgŒ³,ÌƒJƒ^ƒ□ƒO,É"Š®‘S,É" ’u,«Š·,¦,Ü,·□B•”•ª“I,ÈƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Í,Å,«,Ü,¹,ñ□BMTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ªŽ¸”s,µ,½□ê□‡,Í□AMTXREPL.EXE ƒt□[ƒefBƒŠƒefB,ð,à,¤^ê“xŽÀ□s,µ,Ü,·□B

ŠÇ—□ŽÒ,É,Í□AMTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□s,¤□Ó”C,ª, ,è□Aƒ}ƒXƒ^ ƒT□[ƒo□[,ÌƒpƒbƒP□[ƒW,É‰Á,¦,ç,ê,½•Ï□X,ªƒNƒ‰ƒXƒ^“à,Ì,·,×,Ä,ÌƒRƒ“ƒsƒ…□[ƒ^,ÉƒCƒvƒVƒ‡ƒ“,³,ê,Ä,¢,é,±,Æ,ðŠm”F,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,Å MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŽÀ□s,³,ê,Ä,¢,È,¢,Æ,«,ÉƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□s,¤,±,Æ,ð,¨Š©,ß,µ,Ü,·□BMTS ,Ì‹N“®Žž,É,Í□AƒJƒ^ƒ□ƒO□î•ñ,ª“Ç,Ý□ž,Ü,ê,Ä,¢,éŠÔ,ÉŽ¸”s,ð’m,ç,¹,éƒEƒBƒ“ƒhƒE,ª•\Ž¦,³,ê,é,±,Æ,ª, ,è,Ü,·□B,±,ÌŽžŠÔ‘Ñ,ÉƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ª□s,í,ê,é,Æ□AƒpƒbƒP□[ƒW,Ì‹N“®,ÍŽ¸”s,µ□AƒNƒ‰ƒCƒAƒ“ƒg,Í□ÄŽŽ□s,ð—v‹□,³,ê,Ü,·□B

IIS ƒpƒbƒP□[ƒW,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“

IIS ƒpƒbƒP□[ƒW,ðƒŒƒvƒŠƒP□[ƒg,·,é,É,Í□AMicrosoft Internet Information Server (IIS) Version 4.0 ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒt□[ƒefBƒŠƒefB,ðŽg—p,µ,Ü,·□BIIS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Å,Í□A,·,×,Ä,Ì MTS ƒJƒ^ƒ□ƒO□î•ñ,ªŽ©“®“I,ÉƒRƒs□[,³,ê,Ü,·□B,±,Ì,½,ß□AMTS ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒt□[ƒefBƒŠƒefB,ðŽÀ□s,·,é•K—v,Í, ,è,Ü,¹,ñ□BŽÀ□Û,É□AIIS 4.0 ,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,é□ê□‡□AMTS ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒt□[ƒefBƒŠƒefB,Í“®□ì,µ,Ü,¹,ñ□B

ƒpƒbƒP□[ƒW,Ì“à—e,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“

MTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Å,Í□A,·,×,Ä,Ì•K—v,ÈƒRƒ“ƒ|□[ƒlƒ“ƒg (ƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL)□Aƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ□A,¨,æ,Ñƒvƒ□ƒLƒV/ƒXƒ^ƒu DLL) ,ªƒRƒs□[,³,ê,Ü,·□BƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,ª,±,ê,ç,Ìƒtƒ@ƒCƒ‹,ÉƒAƒNƒZƒX,Å,«,é,æ,¤,É,·,é,É,Í□A[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^,Ìƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ƒlƒvƒVƒ‡ƒ“] ƒ^ƒu,Å□A[ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì‹¤—LƒffƒBƒŒƒNƒgƒŠ] ƒ{ƒbƒNƒX,Éƒ}ƒXƒ^,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ð‹¤—L,·,é,½,ß,ÌƒffƒBƒŒƒNƒgƒŠ–¼,ðŽw’è,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B,±,Ì‹¤—L,ðƒ}ƒXƒ^□ã,É□ì□¬,µ□AƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,É“Ç,ÝŽæ,èŒ ŒÀ,ð‹–‰Â,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B[ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì‹¤—LƒffƒBƒŒƒNƒgƒŠ] ƒ{ƒbƒNƒX,ÉŽw’è,·,é–¼‘O,Í‹¤—L–¼,Å, ,è□A‹¤—L–¼,ªƒ}ƒbƒv,³,ê,éƒffƒBƒŒƒNƒgƒŠ,Å,Í, ,è,Ü,¹,ñ□B,½,Æ,¦,Î□Ad:\mtx\repl ,ð MyReplPoint ,Æ,µ,Ä‹¤—L,·,é□ê□‡,Í□A[ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì‹¤—LƒffƒBƒŒƒNƒgƒŠ] ƒ{ƒbƒNƒX,É d:\mtx\repl ,Å,Í,È, MyReplPoint ,ðŽw’è,µ,Ü,·□B

ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì‹¤—LƒffƒBƒŒƒNƒgƒŠ,Í□AƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,ÌƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW ID ,É“Ç,ÝŽæ,èŒ ŒÀ,ð‹–‰Â,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BƒŒƒvƒŠƒP□[ƒg□æ,ÌƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW,ª□[ƒ‹,ÌƒtƒbƒN,ð□s,¤□ê□‡□Aƒ}ƒXƒ^,ÌƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW ID ,ÍƒŒƒvƒŠƒP□[ƒg□æ,ÌƒVƒXƒeƒ€ ƒpƒbƒP□[ƒW,Ì Administrator ƒ□□[ƒ‹,É‘®,µ,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□B

ƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,Åƒtƒ@ƒCƒ‹,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,é□ê□Š,Í□AƒpƒbƒP□[ƒW“à,Ì□Å□‰,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg DLL ,Ì□ê□Š,É,æ,Á,ÄŒˆ,Ü,è,Ü,·□B,±,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ªŠù’è,ÌƒpƒbƒP□[ƒW ƒffƒBƒŒƒNƒgƒŠ,Ì‰‰º,ÌƒTƒuƒffƒBƒŒƒNƒgƒŠ,É, ,é□ê□‡,Í□A,»,ÌƒTƒuƒffƒBƒŒƒNƒgƒŠ,ªƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,É,»,Ì,Ü,ÜƒRƒs□[,³,ê,Ü,·□BŠù’è,ÌƒpƒbƒP□[ƒW ƒffƒBƒŒƒNƒgƒŠ,Í□AMTS ,ÌƒZƒbƒgƒAƒbƒvŽž,É□Ý’è,³,ê,Ü,·□B,½,Æ,¦,Î□AMTS ,ð c:\Program Files\Mts ,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é,Æ□AŠù’è,ÌƒpƒbƒP□[ƒW ƒCƒ“ƒXƒg□[ƒ‹□æƒffƒBƒŒƒNƒgƒŠ,Í c:\Program Files\Mts\Packages ,É,È,è,Ü,·□B

ƒpƒbƒP□[ƒW,ª,±,ÌƒffƒBƒŒƒNƒgƒŠ,Ì‰º,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢ ,é□ê□‡□A,»,ê,ç,ÌƒpƒbƒP□[ƒW,Í□AƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,ÌŠù’è,ÌƒpƒbƒP□[ƒW ƒCƒ“ƒXƒg□[ƒ‹□æƒffƒBƒŒƒNƒgƒŠ,ðŠî□€,É,µ,Ä“¯,¶□ê□Š,ÉƒRƒs□[,³,ê,Ü,·□B,½,Æ,¦,Î□AŽŸ,Ì•\,Ì,æ,¤,É,È,è,Ü,·□B

ƒRƒ“ƒsƒ…□[ƒ^	**Šù’è,ÌƒpƒbƒP□[ƒW ƒffƒBƒŒƒNƒgƒŠ**	**ƒpƒbƒP□[ƒW ƒCƒ“ƒXƒg□[ƒ‹□æƒffƒBƒŒƒNƒgƒŠ**
ƒ}ƒXƒ^	c:\Program Files\Mts\Packages	c:\Program Files\Mts\Packages\MyPak
ƒŒƒvƒŠƒP□[ƒg□æ	d:\Mts\Packages	d:\Mts\Packages\MyPak

Šù’è,ÌƒpƒbƒP□[ƒW ƒCƒ“ƒXƒg□[ƒ‹□æƒffƒBƒŒƒNƒgƒŠ^ÈŠO,Ì□ê□Š,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW,Í□AƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,Ì "ŠO•",Ì" ƒTƒuƒffƒBƒŒƒNƒgƒŠ,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ü,·□B,½,Æ,¦,Î□AŽŸ,Ì•\,Ì,æ,¤,É,È,è,Ü,·□B

ƒRƒ“ƒsƒ…□[ƒ^	**ƒpƒbƒP□[ƒW ƒCƒ“ƒXƒg□[ƒ‹□æƒffƒBƒŒƒNƒgƒŠ**
ƒ}ƒXƒ^	ƒpƒbƒP□[ƒW A ,Í c:\Program Files\Mts\Packages\MyPak ƒpƒbƒP□[ƒW B ,Í d:\misc
ƒŒƒvƒŠƒP□[ƒg□æ	ƒpƒbƒP□[ƒW A□AB ‹¤,É d:\Mts\Packages\MyPak

Šù’è,ÌƒpƒbƒP□[ƒW ƒffƒBƒŒƒNƒgƒŠ,Ì‰º,ÉƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,± ,Æ,ð,¨Š©,ß,µ,Ü,·□B,Ü,½□A“¯,¶ƒpƒbƒP□[ƒW,ÌƒtƒbƒgƒCƒ‹,ð,·,×,Ä“¯,¶ƒffƒBƒŒƒNƒgƒŠ,É’u,,±,Æ,ð,¨Š©,ß,µ,Ü,·□B

MTS ƒT□[ƒo□[ƒNƒ‰ƒXƒ^,ÌŠÇ—□

ŠÇ—□ŽÒ,Í□AMSCS ƒNƒ‰ƒXƒ^,ðŠÇ—□,·,é,Æ,«,Í□A□í,É•¨—□ƒRƒ“ƒsƒ…□[ƒ^–¼,ðŽg,í,È,¯,ê,Î,È,è,Ü,¹,ñ□BMSCS ƒNƒ‰ƒXƒ^,Ìƒ}ƒXƒ^ ƒRƒ“ƒsƒ…□[ƒ^□ã,ÅƒŒƒvƒŠƒP□[ƒg□æ,ð•\Ž¦,·,é□ê□‡,Í□AƒŒƒvƒŠƒP□[ƒg□æ,Ì•¨—□ƒRƒ“ƒsƒ…□[ƒ^–¼,ðŽg,¢ ,Ü,·□Bƒ}ƒXƒ^,ÆƒŒƒvƒŠƒP□[ƒg□æ,ÆìŠÔ,ÅƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□s,¤,Æ,«,Í□A•¨—□ƒRƒ“ƒsƒ…□[ƒ^–¼,ðŽg,¢ ,Ü,·□B

MSCS ƒNƒ‰ƒXƒ^,É MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð”z'u,·,é,Æ,«,Í□A[ƒ}ƒC ƒRƒ“ƒsƒ…□[ƒ^,Ìƒvƒ□[ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [ƒIƒvƒVƒ‡ƒ“] ƒ^ƒu,Ì [ƒŠƒ,□[ƒg ƒT□[ƒo□[–¼] ƒ{ƒbƒNƒX,É□A‰¼'zƒT□[ƒo□[–¼,ðŽw'è,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â"\ƒtƒ@ƒCƒ‹ ƒt□[ƒeƒBƒŠƒefB,ðŽg,Á,ÄƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,é,Æ□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðƒCƒ"ƒXƒg□[ƒ‹,·,éƒNƒ ‰ƒCƒAƒ"ƒg,Í□A‰¼'zƒT□[ƒo□[,ÉƒAƒNƒZƒX,·,é,æ,¤,É□Ý'è,³,ê,Ü,·□B[ƒŠƒ,□[ƒg ƒT□[ƒo□[–¼] ƒ{ƒbƒNƒX,É ‰¼'zƒT□[ƒo□[–¼,ðŽw'è,µ,È,¢,Æ□AƒNƒ‰ƒCƒAƒ"ƒg,Í•¨—□ƒRƒ"ƒsƒ…□[ƒ^,ÉƒAƒNƒZƒX,·,é,æ,¤ ,É□Ý'è,³,ê,Ü,·,ª□A,±,ê,ÍƒtƒF□[ƒ‹ƒI□[ƒo□[,É,æ,é•ÛŒì,ð□s,¤,½,ß,Ì□\□¬,Æ,µ,Ä•s"K□Ø,Å,·□B

ˆÙ,È,é‰¼'zƒT□[ƒo□[–¼,ðŽw'è,·,é,Æ□AƒtƒF□[ƒ‹ƒI□[ƒo□[,É,æ,é•ÛŒì,Æ‹¤,É□Ã"I,È•‰‰×,Ìƒof ‰ƒ"ƒX,ð,Æ,é,±,Æ,ª,Å,«,Ü,·□BŠe•¨—□ƒRƒ"ƒsƒ…□[ƒ^,É 1 ‚Â‚¸‚Â□A□‡Œv 2 ‚Â,Ì‰¼'zƒT□[ƒo□[–¼,ðŽw'è,·,é,±,Æ,É,æ,è□A—¼•û,ÌƒRƒ"ƒsƒ…□[ƒ^,ÉƒCƒ"ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW,ÌƒNƒ‰ƒCƒAƒ"ƒg fCƒ"ƒXƒg□[ƒ‹ŽÀ□s‰Â"\ƒtƒ@ƒCƒ‹,ð□ì□¬,·,é,±,Æ,ª,Å,«,Ü,·□BƒpƒbƒP□[ƒW,Ì"z•z,ðƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒNƒ ‰ƒCƒAƒ"ƒgŠÔ,Å•ª'S,·,é,±,Æ,É,æ,è□A□Ã"I,È•‰‰×,Ìƒof‰ƒ"ƒX,ð,Æ,é,±,Æ,ª,Å,«,Ü,·□B

‰¼'zƒT□[ƒo□[,ðŽg,Á,Ä MTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì•‰‰×,Ìƒof‰ƒ"ƒX,ð,Æ,é,É,Í

1. •‰‰×,Ìƒof‰ƒ"ƒX,ð,Æ,é•û–@,ðŒˆ,ß,Ü,·□B,½,Æ,¦,Î□AŽŸ,Ì,æ,¤,È•û–@,ª, ,è,Ü,·□B
   - ,·,×,Ä,Ìƒpƒbƒ P□[ƒW,ðˆê"x,É 1 'ä,Ìƒ Rƒ"ƒsƒ…□[ƒ^,¾,¯,ÅŽÀ□s,·,é•û–@□B
   - ,·,×,Ä,Ìƒpƒbƒ P□[ƒW,ð—¼•û,ÌƒRƒ"ƒsƒ…□[ƒ^,ÅŽÀ□s,·,é•û–@□B
   - ˆê•",Ìƒpƒbƒ P□[ƒW,ðˆê•û,ÌƒRƒ"ƒsƒ…□[ƒ^,Å□A,»,Ì,Ù,©,Ìƒpƒbƒ P□[ƒW,ð,à,¤ˆê•û,ÌƒRƒ"ƒsƒ… □[ƒ^,ÅŽÀ□s,·,é•û–@□B
2. MSCS ƒNƒ‰ƒXƒ^ ƒAƒhƒ~ƒjƒXƒgƒŒ□[ƒ^,ðŽg,Á,Ä□A—¼•û,ÌƒRƒ"ƒsƒ…□[ƒ^,Ì‰¼'zƒT□[ƒo□[– ¼,ð□ì□¬,µ,Ü,·□B
3. Žè□‡ 2 ,Å□ì□¬,µ,½‰¼'zƒT□[ƒo□[–¼,ðŽg,Á,Ä□AƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,µ,Ü,·□B
   - ,·,×,Ä,Ìƒpƒbƒ P□[ƒW,ðˆê"x,É 1 'ä,ÌƒRƒ"ƒsƒ…□[ƒ^,¾,¯,ÅŽÀ□s,·,é□ê□‡,Í□A1 ‚Â,Ì‰¼'zƒT□[ƒo□[– ¼,ðŽg,Á,Ä,·,×,Ä,Ìƒpƒbƒ P□[ƒW,ðƒGƒNƒXƒ|□[ƒg,µ,Ü,·□B
   - ,·,×,Ä,Ìƒpƒbƒ P□[ƒW,ð"¯Žž,É—¼•û,ÌƒRƒ"ƒsƒ…□[ƒ^,ÅŽÀ□s,·,é□ê□‡,Í□A,·,×,Ä,Ìƒpƒbƒ P□[ƒW,ð,»,ê,¼,ê,Ì ‰¼'zƒT□[ƒo□[–¼,ðŽg,Á,Ä ‰ñ‚¸‚Â□A□‡Œv 2 ‰ñƒGƒNƒXƒ|□[ƒg,µ,Ü,·□BŽŸ,É□A2 ‚Â,Ìˆ Ù,È,éƒNƒ ‰ƒCƒAƒ"ƒg ƒCƒ"ƒXƒg□[ƒ‹ŽÀ□s‰Â"\ƒtƒ@ƒCƒ‹,ÌƒZƒbƒg,ðƒNƒ‰ƒCƒAƒ"ƒg,É"z•z,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B
   - ƒpƒbƒP□[ƒW,Ì"¼•ª,ðˆê•û,ÌƒRƒ"ƒsƒ…□[ƒ^,Å□AŽc,è,Ì"¼•ª,ð,à,¤ˆê•û,ÌƒRƒ"ƒsƒ… □[ƒ^,ÅŽÀ□s,·,é□ê□‡,Í□A,»,ê,¼,ê,Ì‰¼'zƒT□[ƒo□[–¼,ðŽg,Á,ÄƒpƒbƒP□[ƒW,ð"¼•ª,‚,Âƒ Gƒ Nƒ Xƒ| □[ƒg,µ,Ü,·□B
4. ƒNƒ‰ƒCƒAƒ"ƒg ƒRƒ"ƒsƒ…□[ƒ^,ÅƒNƒ‰ƒCƒAƒ"ƒgŽÀ□s‰Â"\ƒtƒ@ƒCƒ‹,ðŽÀ□s,µ,Ü,·□BŽè□‡ 1 ,ÅŽw'è,µ,½• ‰‰×,Ìƒof‰ƒ"ƒX,ð,Æ,é•û–@,É‰ž,¶,Ä□AŠeƒNƒ‰ƒCƒAƒ"ƒg,Å□"ŒÂ,ÌŽÀ□s‰Â"\ ƒtƒ@ƒCƒ‹,ªŽÀ□s,³,ê,Ü,·□B

MTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Å,Í□s,í,ê,È,¢,±,Æ

MTS ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Å,ÍŽŸ,Ì□ì‹Æ,Í□s,í,ê,Ü,¹,ñ□B

- MTS ,ÌƒCƒ"ƒXƒg□[ƒ‹□BMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŽÀ□s,·,é'O,É□AŠeƒŒƒvƒŠƒP□[ƒg□æƒRƒ"ƒsƒ…□[ƒ^,É MTS ,ðƒZƒbƒgƒAƒbƒv,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B
- MTS ƒVƒXƒef€ ƒpƒbƒP□[ƒW□AMTS ƒt□[ƒeƒBƒŠƒefB ƒpƒbƒP□[ƒW□A,¨,æ,Ñ IIS ƒt□[ƒeƒBƒŠƒefB ƒpƒbƒP□[ƒW,Ì'u,«Š·,¦□B,±,ê,ç,ÌƒpƒbƒP□[ƒW,Í□Aƒ}ƒXƒ^,ÆƒŒƒvƒŠƒP□[ƒg□æ,Å,Ü,Á,½,"¯,¶,æ,¤,É□\ □¬,µ,È,¯,ê,Î,È,è,Ü,¹,ñ (ƒtƒ@ƒCƒ‹ ƒpƒX,Íˆ Ù,È,é,±,Æ,ª, ,è,Ü,·)□BMTS ,Í,±,Ì□ì‹Æ,ð□s,í,È,¢,Ì,Å□AŠÇ— □ŽÒ,ª•K—v,É‰ž,¶,ÄƒpƒbƒP□[ƒW,Ì—¼•û,ÌƒZƒbƒg,ð"¯Šú‰»,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B
- ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì‹¤—LƒfƒBƒŒƒNƒgƒŠ,âƒŠƒ,□[ƒg ƒT□[ƒo□[–¼,Ìƒvƒ□[ƒpƒeƒB,È,Ç□AƒRƒ"ƒsƒ…□[ƒ^ŒÅ— L,Ì□î•ñ,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“□B
- ƒ‰ƒ"ƒ^ƒCƒ€ ƒ‰ƒCƒuƒ‰ƒŠ,È,Ç□AMTS ,ªŒŸ□o,Å,«,È,¢ MTS ƒRƒ"ƒ| □[ƒlƒ"ƒg,Ì^Ë'¶ƒtƒ@ƒCƒ‹,ÌƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□³,µ,ŽÀ□s,·,é,É,Í□A,± ,ê,ç,ÌƒfƒCƒ‹,ðŠeƒŒƒvƒŠƒP□[ƒg□æƒRƒ"ƒsƒ…□[ƒ^,ÉƒRƒs□[,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

- ƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^,ìƒZƒLƒ…ƒŠƒeƒB ƒAƒJƒEƒ“ƒg,ìƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“□B,±,Ì,æ,¤,ÈƒAƒJƒEƒ“ƒg,ðƒNƒ‰‰ƒXƒ^“à,Åƒ□□[ƒ‹,âƒpƒbƒP□[ƒW ID ,ÉŽg—p,·,é,±,Æ,Í,¨Š©,ß,Å,«,Ü,¹,ñ□B
- ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ªˆË‘¶,·,éŽ□‘±“I,Èƒf□[ƒ^,ìƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“□B,½,Æ,¦,Î□ASQL Server ,Í SQL Server ƒf□[ƒ^ƒx□[ƒX,ðŽ©“®“I,ÉƒtƒF□[ƒ‹ƒl□[ƒo□[,µ,Ü,·,ª□AŠÇ—□ŽÒ,Í□Aƒf□[ƒ^ƒx□[ƒX,ìƒtƒF□[ƒ‹ƒl□[ƒo□[,ªƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,É“K—p,³,ê,é,æ,¤,É□\□¬,³,ê,Ä,¢,é,± ,Æ,ðŠm”F,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

□§ŒÀŽ–□€

- Windows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,©,ç,Ì MTS ,ìƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“□A,Ü,½,Í Windows 95/98 ƒRƒ“ƒsƒ…□[ƒ^,Ö,Ì MTS ,ìƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ÍƒTƒ|□[ƒg,µ,Ä,¢,Ü,¹,ñ□B
- ƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,ÅŽÀ□s,·,é,æ,¤,É□Ý’è,³,ê,Ä,¢,éƒŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ªƒ}ƒXƒ^□ã,É, ,é,Æ□AMTS ,ìƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ÍŽ¸”s,µ,Ü,·□B
- ƒCƒ“ƒ|□[ƒg,³,ê,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,Í□AƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,ÉŠù,É“o˜^,³,ê,Ä,¢,é□ê□‡,¾,¯ƒŒƒvƒŠƒP□[ƒg,·,é,±,Æ,ª,Å,«,Ü,·□BƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,Í□AƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒ†□[ƒeƒBƒŠƒeƒB,É,æ,Á,ÄƒŒƒvƒŠƒP□[ƒg□æ,ÅŽ©“®“I,É“o˜^,³,ê□A□\□¬,³,ê,Ü,·□Bƒ}ƒXƒ^,ÉƒCƒ“ƒ|□[ƒg,³,ê,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒŒƒvƒŠƒP□[ƒg,·,é□ê□‡□AƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒ†□[ƒeƒBƒŠƒeƒB,Í□AƒŒƒvƒŠƒP□[ƒg□æ,ÉŠù,É“o˜^,³,ê,Ä,¢,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŒŸ□o,µ,æ,¤ ,Æ,µ,Ü,·□BƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒ†□[ƒeƒBƒŠƒeƒB,ðŽÀ□s,·,é‘O,É□AƒCƒ“ƒ|□[ƒg,³,ê,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŠeƒŒƒvƒŠƒP□[ƒg□æƒRƒ“ƒsƒ…□[ƒ^,ÉŽè“®,Å“o˜^,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B
- MTS 2.0 ,æ,è,à‘O,Ìƒo□[ƒWƒ‡ƒ“,Ì MTS ,©,ç,ìƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“□A,Ü,½,Í MTS 2.0 ,æ,è,à‘O,Ìƒo□[ƒWƒ‡ƒ“,Ì MTS ,Ö,ìƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,ÍƒTƒ|□[ƒg,µ,Ä,¢,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

MTS ,ð Microsoft Cluster Server ,Æ‘g,Ý□‡,í,¹,ÄŽg,¤,æ,¤,É□\□¬,·,é□A[ƒIƒvƒVƒ‡ƒ“] ƒ^ƒu (ƒRƒ“ƒsƒ…□[ƒ^)

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÇ—

•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“®ì‚ð—‰,é,Æ AMicrosoft Transaction Server (MTS) ƒpƒbƒP[ƒW‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŒø—¦‚æ‚ŠÇ—‚·‚é‚½‚ß‚É–§‚¿‚Ü‚B±‚±‚Å‚Íƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“®ì‚A¨‚æ‚Ñ MTS ƒGƒNƒXƒvƒ[ƒ‰‚Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠÄŽ‹‚µAŠÇ—‚·‚é•û–@‚É‚Â‚¢‚Ä‚à–¾‚µ‚Ü‚·B

‚±‚±‚Å‚Íƒ‚ŽŸ‚Ìƒgƒsƒbƒ‰N‚É‚Â‚¢‚Ä‚à–¾‚µ‚Ü‚·B

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚É‚Â‚¢‚Ä

MS DTC ‚ÌŠÇ—

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÄŽ‹

Windows 95/98 ‚Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÄŽ‹

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìó‘Ô‚É‚Â‚¢‚Ä

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì‰ðŒˆ

ŠÖ˜A€–Ú

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“Œv] ƒEƒBƒ“ƒhƒE Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ƒAƒCƒRƒ“ A[ƒgƒŒ[ƒX ƒ[ƒbƒZ[ƒW] ƒEƒBƒ“ƒhƒE

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,É,Â,¢,Ä

Microsoft Transaction Server (MTS) ,ðŽg,¤,Æ□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‘“à,Å•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,ðŠÈ’P,ÉŽg—p□AŠÄŽ‹□A,¨,æ,ÑŠÇ—□,·,é,±,Æ,ª,Å,«,Ü,·□B•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,Æ,Í□A•¡□”,ÌƒVƒXƒeƒ€□ã,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,É•ÛŒì,³,ê,½ƒŠƒ\□[ƒX,ð□X□V,·,é•K—v,ª, ,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,Å,·□BMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘ƒ}ƒl□[ƒWƒƒ,Å, ,é Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘ ƒR□[ƒfƒBƒl□[ƒ^ (MS DTC) ,Í□AWindows NT® ƒVƒXƒeƒ€,Æ Windows® 95/98 ƒVƒXƒeƒ€,Å•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,ðˆµ,¤‹@”\,ð’ñ‹Ÿ,µ,Ä,¢,Ü,·□B

MS DTC ,ðŽg,¤,Æ□A1 ,Â,ÌƒVƒXƒeƒ€□ã,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,É•ÛŒì,³,ê,½•¡□”,ÌƒŠƒ\□[ƒX,ð□X□V,·,é,± ,Æ,à,Å,«,Ü,·□B,½,Æ,¦,Î□AMS DTC ,ðŽg,¤,Æ□AMicrosoft SQL Server ƒf□[ƒ^ƒx□[ƒX□AMicrosoft Message Queue Server ƒ□ƒbƒZ□[ƒW ƒLƒ…□[□A,¨,æ,Ñ Oracle ƒf□[ƒ^ƒx□[ƒX,ð ,P ,Â,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,ÌŒ³,Å□X□V,·,é,±,Æ,ª,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

MS DTC ,ÌŠÇ—□□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,ÌŠÄŽ‹□AWindows 95/98 ,Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,ÌŠÄŽ‹□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,Ì□ó‘Ô,É,Â,¢,Ä□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘,Ì‰ðŒˆ

MS DTC ,ÌŠÇ—□

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŠÇ—□,·,é,É,Í□A,Ü,¸□AMicrosoft® •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒl□[ƒ^ (MS DTC) ,ðŠJŽn,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BMS DTC ,ðŠJŽn,·,é,Æ□AŽŸ,Ì,±,Æ,ª‰Â"\,É,È,è,Ü,·□B

- MTS ,Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì'ã,í,è,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð□‰Šú‰»,Å,«,Ü,·□B
- MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒ|□[ƒlƒ“ƒg,Í□A,Ù,©,Ìƒlƒbƒgƒ□□[ƒN ƒVƒXƒeƒ€,Å□‰Šú‰»,³,ê,½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÉŽQ‰Á,Å,«,Ü,·□B
- , ,éƒT□[ƒo□[,Ì Microsoft SQL Server ,¨,æ,Ñ Microsoft Message Queue Server (MSMQ) ,Í□A,Ù,©,ÌƒT□[ƒo□[,Ì MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,Æ‹¤,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÉŽQ‰Á,Å,«,Ü,·□B

ŠÇ—□ŽÒ,ª MTS ƒT□[ƒo□[□ã,Å MS DTC ,ðŠJŽn,µ,È,©,Á,½□ê□‡□AƒNƒ‰ƒCƒAƒ“ƒg,ÍŠÇ—□ŽÒ,ªŠÇ—□,µ,Ä,¢,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒpƒbƒP□[ƒW,ÉƒAƒNƒZƒX,·,é,±,Æ,Í,Å,«,Ü,·,ª□AƒT□[ƒo□[ƒRƒ“ƒ|□[ƒlƒ“ƒg,Íƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÉŽQ‰Á,Å,«,Ü,¹,ñ□B

'□: MS DTC ,ðƒNƒ‰ƒCƒAƒ“ƒg,Æ,µ,ÄƒCƒ“ƒXƒg□[ƒ‹,µ,Ä,¢,éƒt□[ƒU□[,ª□AWindows NT ƒRƒ“ƒsƒ…□[ƒ^□ã,Å MS DTC ƒNƒ‰ƒCƒAƒ“ƒg,ð□Ý'è,µ□AŽÀ□s,·,é,É,Í□Aƒ□□[ƒlƒ‹ ƒŒƒWƒXƒgƒŠ,Ö,Ì□',«□ž,ÝƒAƒNƒZƒX,ÆƒT□[ƒo□[,Ì Software\Classes ƒL□[,Ö,Ìƒ§ƒŠ,□[ƒg"Ç,ÝŽæ,èƒAƒNƒZƒX,ª‹–‰Â,³,ê,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□B

MS DTC ,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ü,½,Í (MTS ,ðƒCƒ"ƒXƒg□[ƒ‹,µ,Ä,¢,È,¢□ê□‡,Í) Windows NT ,ÌƒRƒ"ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,ðŽg—p,µ,ÄŠJŽn,Å,«,Ü,·□B

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä MS DTC ,ðŠJŽn,Ü,½,Í'âŽ~,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ"ƒhƒE,Ì□¶'¤,Å□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒpƒbƒP□[ƒW,ðŠÇ—□,µ,Ä,¢,éƒRƒ"ƒsƒ…□[ƒ^,ð'I'ð,µ,Ü,·□B
2. ["®□ì] ƒ□ƒjƒ…□[,Ì [MS DTC ŠJŽn] ,Ü,½,Í [MS DTC 'âŽ~] ,ðƒNƒŠƒbƒN,µ,Ü,·□B,Ü,½,Í□A–Ú"I,ÌƒRƒ"ƒsƒ…□[ƒ^,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ",ÅƒNƒŠƒbƒN,µ□A[MS DTC ŠJŽn] ,Ü,½,Í [MS DTC 'âŽ~] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ƒRƒ"ƒsƒ…□[ƒ^,Ì [ƒvƒ□ƒpƒeƒB] ƒ_ƒCƒAƒ□ƒO ƒ{ƒbƒNƒX,Ì [□Ú□×□Ý'è] ƒ^ƒu,ðŽg,Á,Ä□ADTC ƒ□ƒO ƒtƒ@ƒCƒ‹,Ì□ê□Š,ÆƒTƒCƒY,È,Ç□ADTC ,Ì□Ý'è,ð□\□¬,·,é,±,Æ,à,Å,«,Ü,·□B

Windows NT ,ÌƒRƒ"ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,ðŽg,Á,Ä MS DTC ,ðŠJŽn,Ü,½,Í'âŽ~,·,é,É,Í

1. [ƒXƒ^□[ƒg] ƒ{ƒ^ƒ",ðƒNƒŠƒbƒN,µ□A[□Ý'è] ,ðƒlƒCƒ"ƒg,µ,Ü,·□BŽŸ,É□A[ƒRƒ"ƒgƒ□□[ƒ‹ ƒpƒlƒ‹] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
2. [ƒT□[ƒrƒX] ƒAƒCƒRƒ",ðƒNƒŠƒbƒN,µ,Ü,·□B
3. [ƒT□[ƒrƒX] ,Ìˆê——,Å [MSDTC] ,ð'I'ð,µ□A[ŠJŽn] ,ðƒNƒŠƒbƒN,µ,Ü,·□B
4. [•Â,¶,é] ,ðƒNƒŠƒbƒN,µ,Ä [ƒT□[ƒrƒX] ƒEƒBƒ"ƒhƒE,ð•Â,¶,Ü,·□B

Šù'è,Ì□Ý'è,Å,Í□AMicrosoft •ªŽUƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ" ƒR□[ƒfƒBƒl□[ƒ^ (MS DTC) ,Í□AWindows NT ƒVƒXƒeƒ€ ,Ü,½,Í Windows 95/98 ƒVƒXƒeƒ€,Ì‹N"®Žž,ÉŽ©"®"I,ÉŠJŽn,·,é,æ,¤,É□\□¬,³,ê,Ü,·□BWindows 95/98 ƒRƒ"ƒsƒ…□[ƒ^,Ì□Ä‹N"®Œã,É MS DTC ,ªŽ©"®"I,ÉŠJŽn,³,ê,È,¢,æ,¤,É,·,é,É,Í□AƒCŒƒWƒXƒgƒŠ ƒGƒfƒBƒ^,ðŽg—p,µ,Ä "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices" ƒŒƒWƒXƒgƒŠ ƒL□[,ð'T,µ□AMSDTC ,Æ,¢,¤–¼'O,Ì'lƒGƒ"ƒgƒŠ,ð□í□œ,µ,Ü,·□BWindows NT ƒRƒ"ƒsƒ…□[ƒ^,Ì□Ä‹N"®Œã,É MS DTC ,ªŽ©"®"I,ÉŠJŽn,³,ê,È,¢,æ,¤,É,·,é,É,Í□AƒRƒ"ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Ì [ƒT□[ƒrƒX] ƒAƒCƒRƒ",ðƒNƒŠƒbƒN,µ,Ü,·□BŽŸ,É□Aˆê——,Ì "MSDTC" ,ð'I'ð,µ,Ä [ƒXƒ^□[ƒtƒgƒAƒbƒv] ,ðƒNƒŠƒbƒN,µ,Ü,·□B[ƒXƒ^□[ƒtƒgƒAƒbƒv,ÌŽí—Þ] ,Ì [Žè"®] ,ð'I'ð,µ,Ä [OK] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

Windows 95/98 ƒRƒ"ƒsƒ…□[ƒ^,Ì□Ä‹N"®Œã,É MS DTC ,ªŽ©"®"I,ÉŠJŽn,³,ê,é,æ,¤,É,·,é,É,Í□AƒCŒƒWƒXƒgƒŠ ƒGƒfƒBƒ^,ðŽg—p,µ,Ä "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices" ƒŒƒWƒXƒgƒŠ ƒL□[,Ì‰ºÉ□A•¶Žš—ñ'l "msdtcw -start" ,ðŽ□,Â MSDTC ,Æ,¢,¤–¼'O,ÌƒGƒ"ƒgƒŠ,ð□ì□¬,µ,Ü,·□BWindows NT ƒRƒ"ƒsƒ…□[ƒ^,Å,Í□AƒRƒ"ƒgƒ□□[ƒ‹ ƒpƒlƒ‹,Ì [ƒT□[ƒrƒX] ƒlƒvƒVƒ‡ƒ",ðŽg,Á,Ä□AMS DTC ,ðŽ©"®"I,ÉŠJŽn,·,é,æ,¤,É□Ý'è,Å,«,Ü,·□B

□d—v: MTS ƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ" ƒRƒ"ƒ|□[ƒlƒ"ƒg□ASQL Server□A,Ü,½,Í MSMQ ,ðŽÀ□s,·,éƒRƒ"ƒsƒ…□[ƒ^,Å,Í□AMS DTC ,ðŽ©"®"I,ÉŠJŽn,·,é,æ,¤,É□Ý'è,·,é,±,Æ,ð‹,□„□§,µ,Ü,·□B

ƒNƒ‰ƒXƒ^ ƒT[ƒo[,Å,Ì MS DTC ,ÌŠJŽn,Æ’âŽ~

ƒNƒ‰ƒXƒ^ ƒT[ƒo[,Å,ÍAMTS ƒGƒNƒXƒvƒ[ƒ‰,Ü,½,Í MSCS ŠÇ—ƒ†[ƒeƒBƒŠƒeƒB,ðŽg,Á,Ä MS DTC ,ðŠJŽn,Ü,½,Í’âŽ~,·,é,±,Æ,ª,Å,«,Ü,·B"net start msdtc" ,¨,æ,Ñ "net stop msdtc" ƒRƒ}ƒ“ƒh,ÍAŽg—p,Å,«,Ü,¹,ñBMSCS ƒNƒ‰ƒXƒ^ã,ÅƒRƒ}ƒ“ƒh ƒvƒƒ“ƒvƒg,©,ç MS DTC ,ðŠJŽn,Ü,½,Í’âŽ~,·,é,É,ÍA"msdtc -start" ,¨,æ,Ñ "msdtc -stop",ðŽg—p,µ,Ü,·B

MS DTC ,ÌÝ’è,ð\¬,·,é,É,Í

1. MTS ƒGƒNƒXƒvƒ[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì‰E‘¤,ÅAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒpƒbƒP[ƒW,ðŠÇ—,µ,Ä,¢,éƒRƒ“ƒsƒ…[ƒ^,ð'I'ð,µ,Ü,·B
2. ["®ì] ƒƒjƒ…[,Ì [ƒvƒƒpƒeƒB] ,ðƒNƒŠƒbƒN,µ,Ü,·B,Ü,½,ÍA–Ú"I,ÌƒRƒ“ƒsƒ…[ƒ^,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µA[ƒvƒƒpƒeƒB] ,ðƒNƒŠƒbƒN,µ,Ü,·B
3. [Ú×Ý’è] ƒ^ƒu,ð'I'ð,µ,Ü,·B[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——] ƒEƒBƒ“ƒhƒE,Æ [ƒgƒŒ[ƒX ƒƒbƒZ[ƒW]n ƒEƒBƒ“ƒhƒE,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª,Ç,Ì,æ,¤,É•\Ž¦,³,ê,é,©,ð’²®,·,é,±,Æ,ª,Å,«,Ü,·BMS DTC ƒƒO,ÌêŠ,ÆƒTƒCƒY,ð•ÏX,·,é,±,Æ,à,Å,«,Ü,·BMS DTC ƒƒO,ÌƒTƒCƒY,ð'å,«,,·,é,ÆA"¯Žž,É,æ,è'½,,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŽÀs,·,é,±,Æ,ª,Å,«,Ü,·B

d—v MS DTC ,ÌƒƒO ƒtƒ@ƒCƒ‹,ÌƒTƒCƒY,É,ÍãŒÀ,ª, ,è,Ü,·B

- Windows NT ,Å,ÍAƒƒO ƒTƒCƒY,ÌãŒÀ,Í 512MB ,Å,·B
- Windows 95/98 ,Å,ÍAƒƒO ƒTƒCƒY,ÌãŒÀ,Í 64MB ,Å,·B

MS DTC ƒƒO ƒtƒ@ƒCƒ‹

MS DTC ƒƒO ƒtƒ@ƒCƒ‹,ÍAˆ³k,³,ê,½ƒfƒBƒŒƒNƒgƒŠ,É'u,,± ,Æ,ª,Å,«,Ü,·B,½,¾,µAÅ"K,ÈƒpƒtƒH[ƒ}ƒ"ƒX,ð"¾,é,É,ÍAMS DTC ƒƒO ƒtƒ@ƒCƒ‹,ðˆ³k,µ,½,èAˆ³k,³,ê,½ƒfƒBƒŒƒNƒgƒŠ,É'u,©,È,¢,Å,,¾,³,¢BMS DTC ,ÍAƒƒO ƒtƒ@ƒCƒ‹,ðŽg—p,·,é'O,Éˆ³k,ð‰ðœ,·,é•K—v,ª, ,è,Ü,·B

DTCXATM.LOG ƒtƒ@ƒCƒ‹,Ìíœ

ƒCƒ“ƒXƒg[ƒ‹,³,ê,½ MTS 2.0 ,ðXV,·,é,Æ,«,ÍAXV,·,é'O,É DTCXATM.LOG ƒtƒ@ƒCƒ‹,ðíœ,µ,Ü,·BDTCXATM.LOG ,ðíœ,·,é'O,ÉAMS DTC ƒT[ƒrƒX,ð’âŽ~,µ,Ä,,¾,³,¢B

ƒŠƒ,[ƒg,Ì MS DTC ƒRƒ“ƒsƒ…[ƒ^ŠÔ,ÌƒZƒLƒ…ƒŠƒeƒB,Æ'Ê M

ƒAƒvƒŠƒP[ƒVƒ‡ƒ“,ª 2 ,Âˆ È ã,ÌƒRƒ“ƒsƒ…[ƒ^,É,í,½,Á,Äƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðˆ—,·,éê‡A,±,ê,ç,ÌƒRƒ“ƒsƒ…[ƒ^ã,Ì MS DTC ,ð'ŠŒÝ,É'ÊM,Å,«,é,æ,¤,É\¬,·,é•K—v,ª, ,è,Ü,·BŠù’è,ÌÝ’è,Å,ÍAMS DTC ,ÍƒVƒXƒeƒ€ ƒAƒJƒEƒ"ƒg ID ,ÌŒ³,ÅŽÀs,³,ê,Ü,·B2 ,Â,ÌƒRƒ“ƒsƒ…[ƒ^,Ì Guest ƒAƒJƒEƒ"ƒg,ª—˜—p,Å,«,éê‡,Ì,ÝAˆê•û,ÌƒRƒ“ƒsƒ…[ƒ^ã,Ì MS DTC ,ÍA,à,¤ˆê•û,ÌƒRƒ“ƒsƒ…[ƒ^ã,Ì MS DTC ,Æ'ÊM,Å,«,Ü,·B,¢,¸,ê,©,ÌƒRƒ“ƒsƒ…[ƒ^,Ü,½,Í,»,ê,ç,ÌƒRƒ“ƒsƒ…[ƒ^,ª'®,· Windows NT ƒhƒƒCƒ"'Å Guest ƒAƒJƒEƒ"ƒg,ª—˜—p,Å,«,È,¢ê‡,ÍAƒŠƒ,[ƒg ƒRƒ“ƒsƒ…[ƒ^,Æ'ÊM,Å,«,é,Ù,©,Ìƒ†[ƒU[–¼,ÌŒ³,Å MS DTC ,ð\¬,·,é•K—v,ª, ,è,Ü,·B

’ Microsoft Cluster Server (MSCS) ŠÂ‹«,Å MS DTC ,ðŽg,¤ê‡,ÍAWindows NT QFE ,ðƒCƒ"ƒXƒg[ƒ‹,µ,È,¯,ê,Î,È,è,Ü,¹,ñB,±,Ì QFE ,ÍAƒNƒ‰ƒXƒ^ ƒT[ƒo[ã,Ì MS DTC ,ÖÚ'± ,ðŽŽ,Ý,½,Æ,«,ÉAMS DTC ƒNƒ‰ƒCƒAƒ"ƒg,ðƒnƒ"ƒO,³,¹,é‰Â"\«,ª, ,éƒŠƒ,[ƒg ƒvƒƒV[ƒWƒƒ ƒR[ƒ‹ (RPC) ,Ì–â'è,ð‰ðŒˆ,µ,Ü,·Bhttp://www.microsoft.com/japan/products/ntupdate/nt4sp3/j041091.htm ,©,çƒCƒ"ƒXƒg[ƒ‹,·,é,±,Æ,ª,Å,«,Ü,·B

ŠÖ˜A€–Ú

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,É,Â,¢,ÄAMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹AWindows 95/98 ,Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹A[Ú×Ý’è] ƒ^ƒu (ƒRƒ“ƒsƒ…[ƒ^)A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——] ƒEƒBƒ“ƒhƒEA[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì"Œv] ƒEƒBƒ“ƒhƒEAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒAƒCƒRƒ“A[ƒgƒŒ[ƒX ƒƒbƒZ[ƒW] ƒEƒBƒ“ƒhƒE

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠÄŽ‹

MTS ƒGƒNƒXƒvƒ□□[ƒ‰‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ŠÖ˜A‚ÌƒEƒBƒ“ƒhƒE‚ðŽg‚Á‚Ä□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ª‚à‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠÇ—□‚·‚é‚±‚Æ‚ª‚Å‚«‚Ü‚·□B[ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ƒEƒBƒ“ƒhƒE□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv] ƒEƒBƒ“ƒhƒE□A‚¨‚æ‚Ñ [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE‚Å‚Í□AMicrosoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒffBƒl□[ƒ^ (MS DTC) ‚ªŠÇ—□‚·‚éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó‘Ô‚ÉŠÖ‚·‚é□d—v‚È□î•ñ‚ð‘ñ‹Ÿ‚µ‚Ü‚·□B

’□ ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ªŠù’è‚Ìƒ^ƒCƒ€ƒAƒEƒg’l‚Ì 60•b‚È“à‚É□I—¹‚µ‚È‚¢□ê□‡‚Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÍŽ©“®“I‚ÉƒAƒ{□[ƒg‚ª‚Ü‚·□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ^ƒCƒ€ƒAƒEƒg‚Í□A[ƒ}ƒC ƒRƒ“ƒfƒsƒ…□[ƒ^] ‚Ì [ƒvƒ□ƒpƒefB] ‚Ì [ƒlƒvƒVƒ‡ƒ“] ƒ^ƒu‚Å•Ï□X‚Å‚«‚Ü‚·□B

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE‚ðŽg‚Á‚Ä□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó‘Ô‚ð‰ðŒˆ□v‚ðŽQ□Æ‚µ‚Ä‚¾‚³‚¢□B

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE‚ðŽg‚Á‚Ä□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠÄŽ‹‚·‚é‚É

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ÌƒEƒBƒ“ƒhƒE‚Ì□¶‘¤‚Å□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒzƒ‚ƒXƒg‚É‚È‚Á‚Ä‚¢‚éƒRƒ“ƒfƒsƒ…□[ƒ^‚ð‘I‘ð‚µ‚Ü‚·□B
2. [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒAƒCƒRƒ“‚ðƒ_ƒuƒ‹ƒNƒŠƒbƒN‚µ‚Ü‚·□B
3. ƒEƒBƒ“ƒhƒE‚Ì‰¤‚Ì”C‚Ó‚Ì□ê□Š‚ðƒ}ƒEƒX‚Ì‰¤Eƒ{ƒ^ƒ“‚ÅƒNƒŠƒbƒN‚µ□A[•\Ž¦] ‚ðƒ|ƒCƒ“ƒg‚µ‚Ü‚·□B
4. ŽŸ‚Ì‚¢‚¸‚ê‚©‚Ìƒfƒ‚ƒXƒbƒN‚µ‚Ü‚·□B
   - **[‘å‚«‚¢ƒAƒCƒRƒ“]**

     ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª‘å‚«‚¢ƒAƒCƒRƒ“‚Å•\Ž¦‚³‚ê‚Ü‚·□B
   - **[□¬‚³‚¢ƒAƒCƒRƒ“]**

     ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□¬‚³‚¢ƒAƒCƒRƒ“‚Å•\Ž¦‚³‚ê‚Ü‚·□B
   - **[□Ú□×]**

     1 —ñ‚ÉƒgƒŒ[ƒUƒ‚ƒfƒ‹‚Ì 1 —ñ‚ÉŠeƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÉŠÖ˜A•t‚¯‚ç‚ê‚Ä‚¢‚éƒ‚ƒfƒ‹ ID ƒ†ƒjƒbƒg‚ª•\Ž¦‚³‚ê‚Ü‚·□B‚±‚Ì ID ‚Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŠJnŽž‚É MS DTC ‚ª□¶□¬‚·‚é□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌƒOƒ□[ƒoƒ‹‚ÉˆêˆÓ‚Ó‚Ì ID ‚Å‚·□B□eƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Æ□]‘®ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——‚à•\Ž¦‚³‚ê‚Ü‚·□B
   - **[ˆê——]**

     1 —ñ‚ÉƒgƒŒ[ƒUƒNƒVƒ‡ƒ“‚ª□‡”Ô‚É•\Ž¦‚³‚ê‚Ü‚·□B

’□ [□¬‚³‚¢ƒAƒCƒRƒ“] ƒRƒ}ƒ“ƒh‚Æ [ˆê——] ƒRƒ}ƒ“ƒh‚ðŽg‚Á‚Ä□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª•\Ž¦‚³‚ê‚Ü‚·□B[□Ú□×] ƒRƒ}ƒ“ƒh‚ðŽg‚Á‚Ä□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□î•ñ‚ª□Å‚à•\Ž¦‚³‚ê‚Ü‚·□B[‘å‚«‚¢ƒAƒCƒRƒ“] ƒRƒ}ƒ“ƒh‚ðŽg‚Á‚Ä□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□Å‚à‘©‚¢‚â‚·‚¢•\Ž¦‚³‚ê‚Ü‚·□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv□î•ñ‚ð•\Ž¦‚·‚é‚É

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ÌƒEƒBƒ“ƒhƒE‚Ì□¶‘¤‚Å□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv□î•ñ‚ð•\Ž¦‚·‚é‚É□[ƒ^‚ð‘I‘ð‚µ‚Ü‚·□B
2. [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv] ƒAƒCƒRƒ“‚ðƒ_ƒuƒ‹ƒNƒŠƒbƒN‚µ‚Ü‚·□B

ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW‚ð•\Ž¦‚·‚é‚É

1. MTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ÌƒEƒBƒ“ƒhƒE‚Ì□¶‘¤‚Å□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——‚Ìƒzƒ‚ƒXƒg‚É‚È‚Á‚Ä‚¢‚éƒRƒ“ƒfƒsƒ…□[ƒ^‚ð‘I‘ð‚µ‚Ü‚·□B
2. [ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW] ƒAƒCƒRƒ“‚ðƒ_ƒuƒ‹ƒNƒŠƒbƒN‚µ‚Ü‚·□B

[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì“□Œv] ƒEƒBƒ“ƒhƒE‚ÍAMTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ÌƒEƒBƒ“ƒhƒE‚Ì‰¤Eƒ{ƒ^ƒ“‚ðƒNƒŠƒbƒN‚µ‚Ü‚·□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒvƒ□ƒpƒefB

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìƒvƒ□ƒpƒefB‚ð•\Ž¦‚·‚é‚É‚Í□A‘I‘ð‚³‚ê‚½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒ}ƒEƒX‚Ì‰¤Eƒ{ƒ^ƒ“‚ÅƒNƒŠƒbƒN‚µ□A[ƒvƒ□ƒpƒefB] ‚ðƒNƒŠƒbƒN‚µ□B[ƒvƒ□ƒpƒefB] ƒRƒ}ƒ“ƒh‚ðŽg‚Á‚Ä□Aƒgƒ

‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÉŠÖŒW,µ,Ä,¢,é,·,×,Ä,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ}ƒ‹[ƒWƒƒ,Ìˆê——,ª•\Ž¦,³,ê,Ü,·B Ž‰ƒgƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒ}ƒEƒX,Ì‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µA[ƒvƒ[ƒpƒeƒB] ,ðƒNƒŠƒbƒN,·,é,Æ A,»,Ìƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì’¼Ú,Ì eƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——,ª•\Ž¦,³,ê,Ü,·B‘I‘ð,µ,½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª eƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì ê ‡,Í A,»,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì’¼Ú,ÌŽqƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——,ª•\Ž¦,³,ê,Ü,·B

ŠÖ˜A€–Ú

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,É,Â,¢,ÄAWindows 95/98 ,Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹ AMTS ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì ó‘Ô,É,Â,¢,ÄAMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì‰ðŒˆ A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——]
ƒEƒBƒ“ƒhƒE A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“Œv] ƒEƒBƒ“ƒhƒE Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒAƒCƒRƒ“ A[ƒgƒŒ [ƒX
ƒ fbƒZ [ƒW] ƒEƒBƒ“ƒhƒE

Windows 95/98 ,Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹

MTS ƒGƒNƒXƒvƒ[ƒ‰,ÌŽŸ,ÌƒEƒBƒ“ƒhƒE,ðŽg,Á,Ä\AWindows 95®\AWindows 98® ƒRƒ“ƒsƒ…[ƒ^\ã,Ü,½,Í Windows NT ƒRƒ“ƒsƒ…[ƒ^\ã,Ì MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŠÇ—,·,é,±,Æ,ª,Å,«,Ü,·\B

- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“Œv]
- [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——]
- [ƒgƒŒ[ƒX ƒ\ƒbƒZ[ƒW]

Šù'è,Ì\Ý'è,Å,Í\AMicrosoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR[ƒfƒBƒl[ƒ^ (MS DTC)asdefDTC ,Í\AWindows NT ƒVƒXƒeƒ€,Ü,½,Í Windows 95/98 ƒVƒXƒeƒ€,Ì‹N“®Žž,ÉŽ©“®“I,ÉŠJŽn,·,é,æ,¤,É\\¬,³,ê,Ü,·\BWindows 95/98 ƒRƒ“ƒsƒ…[ƒ^,Ì\Ä‹N“®Žž,É MS DTC ,ªŽ©“®“I,ÉŠJŽn,µ,È,¢,æ,¤,É,·,é,É,Í\AƒŒƒWƒXƒgƒŠ ƒGƒfƒBƒ^,ðŽg,Á,Ä\AHKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion¥RunServices ƒŒƒWƒXƒgƒŠ ƒL[,ð'T,µ\AMSDTC ,Æ,¢,¤–¼'O,Ì'lƒGƒ“ƒgƒŠ,ð\í\œ,µ,Ü,·\BMS DTC ,ÌŽ©“®ƒX^[ƒgƒAƒbƒv,ð,à,¤ˆê“x—LŒø,É,·,é,É,Í\AƒŒƒWƒXƒgƒŠ ƒGƒfƒBƒ^,ðŽg,Á,Ä\A•¶Žš—ñ'l msdtcw -start ,ðŽ\,Â MSDTC ,Æ,¢,¤–¼'O,Ì'lƒGƒ“ƒgƒŠ,ð HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion¥RunServices ƒŒƒWƒXƒgƒŠ ƒL[,Ì‰º,É\ì\¬,µ,Ü,·\B

' Windows NT ƒRƒ“ƒsƒ…[ƒ^\ã,Å MS DTC ƒNƒ‰ƒCƒAƒ“ƒg,ð\\¬,µ\AŽÀ\s,·,é,É,Í\Aƒ†\[ƒU\[,Íƒ\[ƒJƒ‹ ƒŒƒWƒXƒgƒŠ,Ö,Ì\'‚«\ž,ÝƒAƒNƒZƒX,ÆƒT\[ƒo\[,Ì Software\Classes ƒL\[,Ö,ÌƒŠƒ,\[ƒg“Ç,ÝŽæ,èƒAƒNƒZƒX,ð‹–‰Â,³,ê,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ\B

Windows 95/98 ƒRƒ“ƒsƒ…[ƒ^,©,ç Windows NT ƒRƒ“ƒsƒ…[ƒ^,ðƒŠƒ,\[ƒg,ÅŠÇ—,·,é\ê\‡,Í\AƒŠƒ,\[ƒg ƒŒƒWƒXƒgƒŠ ƒT\[ƒrƒX,ðƒCƒ“ƒXƒg\[ƒ‹,µ,È,¯,ê,Î,È,è,Ü,¹,ñ\BƒŠƒ,\[ƒg ƒŒƒWƒXƒgƒŠ ƒT\[ƒrƒX,ðŽg,¤ ,Æ\A("K\Ø,ÈƒAƒNƒZƒXŒ ,ª, ,ê,Î) ƒŠƒ,\[ƒg,Ì Windows NT ƒRƒ“ƒsƒ…[ƒ^,ÌƒŒƒWƒXƒgƒŠ ƒGƒ“ƒgƒŠ,ð•Ï\X,Å,«,Ü,·\BƒŠƒ,\[ƒg ƒŒƒWƒXƒgƒŠ ƒT\[ƒrƒX,Í\AWindows 95/98 ,Ì CD-ROM ,Ì \Admin\Nettols\Remotreg ƒfƒBƒŒƒNƒgƒŠ,É ,è,Ü,·\BƒŠƒ,\[ƒg ƒŒƒWƒXƒgƒŠ ƒT\[ƒrƒX,ÌƒCƒ“ƒXƒg\[ƒ‹Žè\‡,É,Â,¢ ,Ä,Í\ARegserv.txt ƒtƒ@ƒCƒ‹,ðŽQ\Æ,µ,Ä,,¾,³,¢\B

Windows 95/98 ,É,ÍƒCƒxƒ“ƒg ƒ\ƒO,ª,È,¢,Ì,Å\AMS DTC ,ÌƒCƒxƒ“ƒg,Í msdtc.txt ,Æ,¢,¤ƒeƒLƒXƒg ƒtƒ@ƒCƒ‹,É•Û'¶,³,ê,Ü,·\Bmsdtc.txt ƒtƒ@ƒCƒ‹,Í\AWindows ƒfƒBƒŒƒNƒgƒŠ,Ì \MTSLogs ƒTƒuƒfƒBƒŒƒNƒgƒŠ,É ,è\AMS DTC ,ÌƒCƒxƒ“ƒg,ÉŠÖ,·,é\î•ñ,ð•Û'¶,µ,Ü,·\B

Windows 95/98 ,Å MS DTC ƒ\ƒO ƒtƒ@ƒCƒ‹,ð\Ä\Ý'è,·,é\ê\‡\A"MTxOCI" ,Æ•\Ž¦,·,éƒ_ƒCƒAƒ\ƒO ƒ{ƒbƒNƒX,ÌŒã,É\A"MS DTC ƒ\ƒO ƒtƒ@ƒCƒ‹,Ì\ì\¬,ÉŽ¸"s,µ,Ü,µ,½\B" ,Æ•\Ž¦,·,éƒ_ƒCƒAƒ\ƒO ƒ{ƒbƒNƒX,ªŒ»,ê,é\ê\‡,ª, ,è,Ü,·\B,±,Ì–â'è,ª"\¶,µ,½\ê\‡\AƒRƒ“ƒsƒ…[ƒ^,Ì\Ä‹N“®Žž,É MS DTC ,ðŽ©“®“I,ÉŠJŽn,µ,È,¢,æ,¤,É\Ý'è,·,é•K—v,ª, ,è,Ü,·\B\Ý'è,ª\I,í,Á,½,ç\AƒRƒ“ƒsƒ…[ƒ^,ð\Ä‹N“®,µ,Ä\Aƒ\ƒO ƒtƒ@ƒCƒ‹,ðƒŠƒZƒbƒg,µ,Ü,·\B,±,ê,É,æ,Á,Ä\AMS DTC ,ðŽ©“®“I,ÉŠJŽn,·,é,æ,¤ ,É\Ä\Ý'è,Å,«,é,æ,¤,É,È,è,Ü,·\B

MS DTC ƒ\ƒO ƒtƒ@ƒCƒ‹,Ì\Ä\Ý'èŽž,É‹N,±,é,»,Ì,Ù,©,Ì–â'è,Í\AMTS ƒGƒNƒXƒvƒ[ƒ‰,ð•Â,¶,Ä,©,ç MTS ƒGƒNƒXƒvƒ[ƒ‰,ð\ÄŠJ,µ\A\Ä"x\Ý'è,ð\s,¤,±,Æ,Å‰ðŒˆ,Å,«,Ü,·\B\ê\‡,É,æ,Á,Ä,Í\AMS DTC ƒ\ƒO ƒtƒ@ƒCƒ‹,Ì–â'è,ª‰ðŒˆ,·,é,Ü,Å\AMTS ƒGƒNƒXƒvƒ[ƒ‰,ðƒVƒƒƒbƒgƒ_ƒEƒ“,µ,Ä\ÄŠJ,·,é,±,Æ,ð ‰½"x,©ŒJ,è•Ô,·•K—v,ª, ,è,Ü,·\B

ŠÖ˜A\€–Ú

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,É,Â,¢,Ä\AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹\A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——] ƒEƒBƒ“ƒhƒE\A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì“Œv] ƒEƒBƒ“ƒhƒE\Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒAƒCƒRƒ“\A[ƒgƒŒ\[ƒX ƒ\ƒbƒZ\[ƒW] ƒEƒBƒ“ƒhƒE

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó‘Ô‚É‚Â‚¢‚Ä

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠÇ—□‚·‚é‚Í□A‚³‚Ü‚´‚Ü‚Èƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó‘Ô‚Æ□AŠÇ—□‚µ‚Ä‚¢‚é MTS ƒpƒbƒP□[ƒW‚Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó‘Ô‚ª‹y‚Ú‚·‰e‹¿‚É‚Â‚¢‚Ä—□‰ð‚µ‚È‚¯‚ê‚Î‚È‚è‚Ü‚¹‚ñ□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì□ó‘Ô‚Í□AMicrosoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒffBƒl□[ƒ^ (MS DTC) ‚Ì [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì^ê——] ƒEƒBƒ“ƒhƒE‚Ì [‘å‚«‚¢ƒAƒCƒRƒ“] Œ`Ž®‚Å•\Ž¦‚³‚ê‚éŽŸƒAƒCƒRƒ“‚É‚æ‚Á‚Ä•\‚³‚ê‚Ü‚·□B

ƒAƒCƒRƒ“	□à–¾
	ƒAƒNƒeƒBƒu
	ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠJŽn‚³‚ê‚Ü‚µ‚½□B
	ƒAƒ{□[ƒg’†
	ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒAƒ{□[ƒg‚µ‚Ä‚¢‚Ü‚·□BMS DTC ‚Í□A‚·‚×‚Ä‚ÌŠÖ˜A•”•ª‚É□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒAƒ{□[ƒg‚µ‚È‚¯‚ê‚Î‚È‚ç‚È‚¢‚Æ‚ð’Ê’m‚µ‚Ä‚¢‚Ü‚·□B
	‚±‚ÌŽž“_‚Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŒ‹‰Ê‚ð•Ï□X‚·‚é‚±‚Æ‚Í‚Å‚«‚Ü‚¹‚ñ□B
	ƒAƒ{□[ƒg
	ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ªƒAƒ{□[ƒg‚³‚ê‚Ü‚µ‚½□B‚·‚×‚Ä‚ÌŠÖ˜A•”•ª‚Í□A’Ê’m‚ðŽó¯Ü‚µ‚½□BƒAƒ{□[ƒg‚³‚ê‚½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì^ê——] ƒEƒBƒ“ƒhƒE‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì^ê—— ‚©‚ç‚·‚é‚±‚Æ‚ª□í□œ‚³‚ê‚Ü‚·□B
	‚±‚ÌŽž“_‚Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŒ‹‰Ê‚ð•Ï□X‚·‚é‚±‚Æ‚Í‚Å‚«‚Ü‚¹‚ñ□B
	□€”õ’†
	ƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ªƒRƒ~ƒbƒg—v‹□‚ð”-□s‚µ‚Ü‚µ‚½□BMS DTC ‚Í□A‚·‚×‚Ä‚ÌŠÖ˜A•”•ª‚©‚ç□€”õŠ®—¹‚Ì‰ž“š‚ð□W‚ß‚Ä‚¢‚Ü‚·□B
	□€”õŠ®—¹
	‚·‚×‚Ä‚ÌŠÖ˜A•”•ª‚□€”õŠ®—¹‚Ì‰ž“š‚ð•Ô‚µ‚Ü‚µ‚½□B
	ƒCƒ“ƒ‰ƒEƒ“ƒg
	ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª•Ê‚Ì MS DTC ‚É‚æ‚Á‚Ä□€”õ’†‚É□A’²□®‚³‚ê‚Ä‚¨‚è□A’²□®–ð‚Ì MS DTC ‚ªƒAƒNƒZƒX‚Å‚«‚È‚¢□BƒVƒXƒeƒ€ŠÇ—□ŽÒ‚Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì^ê——] ƒEƒBƒ“ƒhƒE“à‚Åƒ}ƒEƒX‚Ì‰Eƒ{ƒ^ƒ“‚ðNƒŠƒbƒN‚µ□A[‰ðŒˆ] ‚ðƒ|ƒCƒ“ƒg‚µ‚Ü‚·□BŽŸ‚É□A[ƒRƒ~ƒbƒg] ‚Ü‚½‚Í [ƒAƒ{□[ƒg] ‚ðƒNƒŠƒbƒN‚·‚é‚Æ□A§“I‚Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðƒRƒ~ƒbƒg‚µ‚È‚¯‚ê‚Î‚È‚ç‚È‚¢‚Æ‚¨‚è□B§“I‚É□ˆ—□,ð□s‚¤Æ□A»‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Í [§□ƒRƒ~ƒbƒg] ‚Ü‚½‚Í [§□ƒAƒ{□[ƒg] ‚Æ‚µ‚Ä•\Ž¦‚³‚ê‚Ü‚·□B
	Œx□□ ƒCƒ“ƒ‰ƒEƒg‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚Å‹-□§“I‚É□ˆ—□,·‚é‘O‚É□A•K,¸□uMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì

‰ðŒˆ□v,ð,¨“Ç,Ý,,¾,³,¢□B

- **‹□§ƒRƒ~ƒbƒg**

  ŠÇ—□ŽÒ,ª□AƒCƒ“ ƒ_ƒEƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‹-
  □§“I,ÉƒRƒ~ƒbƒg,µ,Ü,µ,½ (□uMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì
  ‰ðŒˆ□v,ðŽQ□Æ,µ,Ä,,¾,³,¢)□B

- **‹□§ƒAƒ{□[ƒg**

  ŠÇ—□ŽÒ,ª□AƒCƒ“ ƒ_ƒEƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‹-
  □§“I,ÉƒAƒ{□[ƒg,µ,Ü,µ,½ (□uMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì
  ‰ðŒˆ□v,ðŽQ□Æ,µ,Ä,,¾,³,¢)□B

- **ƒRƒ~ƒbƒg’†**

  ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª□³,µ,□€”õ,³,ê□AMS DTC
  ,ªŠÖ˜A•”•ª,É□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒRƒ~ƒbƒg,³,ê,½,±
  ,Æ,ð’Ê’m,µ,Ä,¢,Ü,·□BMS DTC
  ,Í□A,·,×,Ä,ÌŠÖ˜A•”•ª,ªƒRƒ~ƒbƒg—v‹□,ÌŽó□M
  (,¨,æ,Ñƒ□ƒO,Ì‹L˜^) ,ðŠm”F,·,é,Ü,Å□A,»,Ìƒgƒ
  ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð□I—¹,µ,Ü,¹,ñ□B

  ,±,ÌŽž“_,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŒ‹‰Ê,ð•Ï□X,·,é,±
  ,Æ,Í,Å,«,Ü,¹,ñ□B

- **ƒAƒ{□[ƒg,ð’Ê’m,Å,«,Ü,¹,ñ**

  MS DTC ,Í□A,·,×,Ä,Ì□Ú‘±,³,ê,Ä,¢,éŠÖ˜A•”•ª,É□Aƒgƒ
  ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒAƒ{□[ƒg,³,ê,½,±
  ,Æ,ð’Ê’m,µ,Ü,µ,½□B’Ê’m,ðŽó,¯,Ä,¢,È,¢
  ,Ì,Í□AŒ»□ÝƒAƒNƒZƒX,Å,«,È,¢ŠÖ˜A•”•ª,¾,¯,Å,·□B

  ,±,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□ó‘Ô,Í□AMS DTC ,ª,¢,¸,ê,©,ÌƒŠƒ\
  □[ƒX ƒ}ƒl□[ƒWƒƒ (IBM LU 6.2 ƒVƒXƒeƒ€,È,Ç) ,Éƒgƒ
  ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒAƒ{□[ƒg,³,ê,½,±
  ,Æ,ð’Ê’m,µ,È,¯,ê,Î,È,ç,È,¢,Ì,É□A,»,ÌƒVƒXƒeƒ€,Æ,Ì□Ú‘±
  ,ª□Ø’f,³,ê,Ä,¢,é,½,ß□A’Ê’m,Å,«,È,¢,Æ,«,É”□¶,µ,Ü,·□B

  ƒVƒXƒeƒ€ŠÇ—□ŽÒ,Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——]
  ƒEƒBƒ“ƒhƒE“à,Åƒ}ƒEƒX,Ì
  ‰Eƒ{ƒ^ƒ“,ðƒNŠƒbƒN,µ□A[‰ðŒˆ] ,ðƒ|
  ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[”jŠü] ,ðƒNŠƒbƒN,·,é,Æ□AMS
  DTC ,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‹□§“I,É”jŠü,³,¹,é,±
  ,Æ,ª,Å,«,Ü,·□B

  Œx□□ ƒgƒ
  ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŽè“®,Å”jŠü,·,é‘O,É□A•K,¸□uMTS ƒgƒ
  ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì‰ðŒˆ□v,ð,¨“Ç,Ý,,¾,³,¢□B

- **ƒRƒ~ƒbƒg,ð’Ê’m,Å,«,Ü,¹,ñ**

  MS DTC ,Í□A,·,×,Ä,Ì□Ú‘±,³,ê,Ä,¢,éŠÖ˜A•”•ª,É□Aƒgƒ
  ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒRƒ~ƒbƒg,³,ê,½,±
  ,Æ,ð’Ê’m,µ,Ü,µ,½□B’Ê’m,ðŽó,¯,Ä,¢,È,¢
  ,Ì,Í□AŒ»□ÝƒAƒNƒZƒX,Å,«,È,¢ŠÖ˜A•”•ª,¾,¯,Å,·□B

  ƒVƒXƒeƒ€ŠÇ—□ŽÒ,Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——]
  ƒEƒBƒ“ƒhƒE“à,Åƒ}ƒEƒX,Ì
  ‰Eƒ{ƒ^ƒ“,ðƒNŠƒbƒN,µ□A[‰ðŒˆ] ,ðƒ|
  ƒCƒ“ƒg,µ,Ü,·□BŽŸ,É□A[”jŠü] ,ðƒNŠƒbƒN,·,é,Æ□AMS
  DTC ,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‹□§“I,É”jŠü,³,¹,é,±

,Æ,ª,Å,«,Ü,·□B

Œx□□ ƒgƒ

‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŽè“®,Å”jŠü,·,é‘O,É□A•K,¸□uMTS ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì‰ðŒˆ□v,ð,¨“Ç,Ý,,¾,³,¢□B

ƒRƒ~ƒbƒg

ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒRƒ~ƒbƒg,³,ê□A,·,×,Ä,ÌŠÖ˜A•”•ª,ª'Ê'
m,ðŽó,¯,Ü,µ,½□BƒRƒ~ƒbƒg,³,ê,½ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,Í□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——]
ƒEƒBƒ“ƒhƒE,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——
,©,ç,·,®,É□í□œ,³,ê,Ü,·□B

,±,ÌŽž“_,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŒ‹‰Ê,ð•Ï□X,·,é,±
,Æ,Í,Å,«,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,É,Â,¢,Ä□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹□AWindows 95/98 ,Å MTS ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì‰ðŒˆ□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì^ê——] ƒEƒBƒ“ƒhƒE

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ì‰ðŒˆ

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠÇ—□,µ,Ä,¢,é,Æ□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚Å‰ðŒˆ‚·‚é•K—v‚ª ‚é□ê□‡‚ª ‚è‚Ü‚·□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð‰ðŒˆ‚·‚é‚É□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰Ì [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE‚ÅAŽŸ‚Ì¢‚¸‚ê‚©‚Ìƒ}ƒ“ƒh‚ðŽg‚¢‚Ü‚·□B

- **[ƒRƒ~ƒbƒg]** ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð‹§“I‚ÉƒRƒ~ƒbƒg‚µ‚Ü‚·□B
- **[ƒAƒ{□[ƒg]** ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð‹§“I‚ÉƒAƒ{□[ƒg‚µ□AŒ³‚Ì□ó‘Ô‚ÉƒÍ□[ƒ‹ ƒoƒbƒN‚µ‚Ü‚·□B
- **[”jŠü]** ƒRƒ~ƒbƒg‚Ü‚½ƒIƒAƒ{□[ƒg‚³‚ê‚½ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒffƒBƒl□[ƒ^ (MS DTC) ‚Ìƒ□ƒO‚©‚ç□í□œ‚µ‚Ü‚·□B‚±‚Ìƒ}ƒ“ƒh‚ðŽg‚¤‘O‚É□A•K‚¸ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð‹§“I‚É□ˆ—□‚µ‚Ä‚,,¾‚³‚¢□B

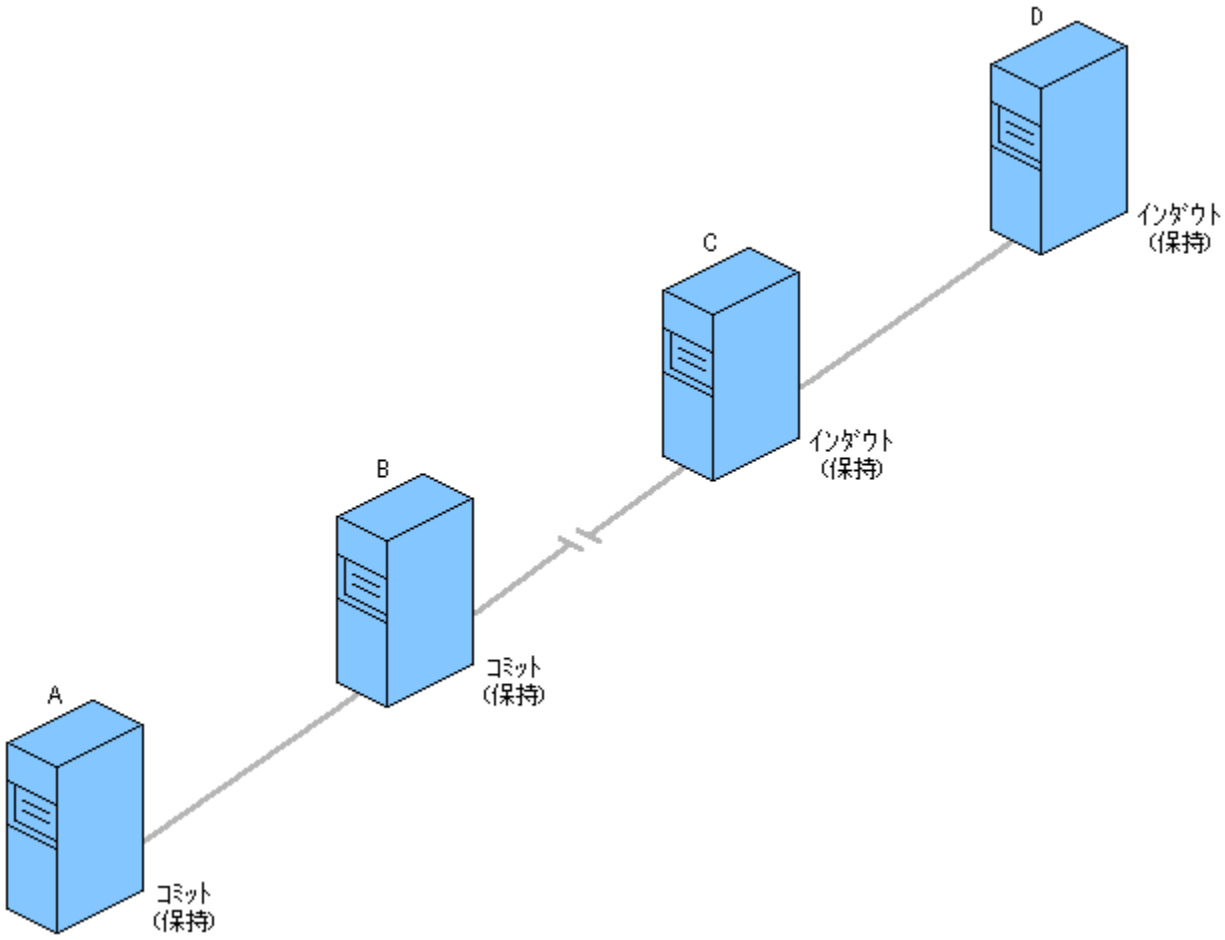
□ê□‡‚É‚æ‚Á‚Ä‚Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð‹§“I‚ÉƒRƒ~ƒbƒg□A‚Ü‚½‚ÍƒIƒAƒ{□[ƒg‚µ‚Ä‚Äƒ□ƒbƒN‚µ‚Ü‚ð□œ‚µ‚Äƒf□[ƒ^ƒx□[ƒX‚ÌƒSƒ\□[ƒX‚ð‚Ù‚©‚Ìƒlƒbƒgƒ□□[ƒN ƒt□[ƒU□[‚âƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ª—˜—p‚Å‚«‚é‚æ‚¤‚É‚·‚é•K—v‚ª ‚è‚Ü‚·□B

‚½‚Æ‚¦‚Î□Aƒlƒbƒgƒ□□[ƒN□ã‚Ì 2 ‘ä‚ÌƒRƒ“ƒsƒ…□[ƒ^ŠÔ‚Ì’Ê□M‰ñ□ü‚É□áŠQ‚ª”□¶‚·‚é‚Æ□A±‚Ì‘€□ì‚ª•K—v‚É‚È‚è‚Ü‚·□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒRƒ~ƒbƒg‚µ‚½‚çAŽè“®‚Å‰ðŒˆ‚·‚é•K—v‚ª‚ ‚è‚Ü‚·□B”jŠü" ‚³‚¹‚é•K—v‚ª ‚é□ê□‡‚ª ‚è‚Ü‚·□B”jŠü‚·‚é‚Æ□Aƒ□□[ƒJƒ‹‚Ì MS DTC ƒ□ƒO ƒtƒ@ƒCƒ‹‚©‚çƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□í□œ‚³‚ê‚Ü‚·□B

ŽŸ‚Ì□}‚Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒRƒ~ƒbƒg‚·‚é□ê□‡‚ðŽ¦‚µ‚Ä‚Ü‚·□B‚±‚Ì—á‚Å‚Í□AŽŸ‚Ì□ðŒ□‚ð‘z’è‚µ‚Ä‚¢‚Ü‚·□B

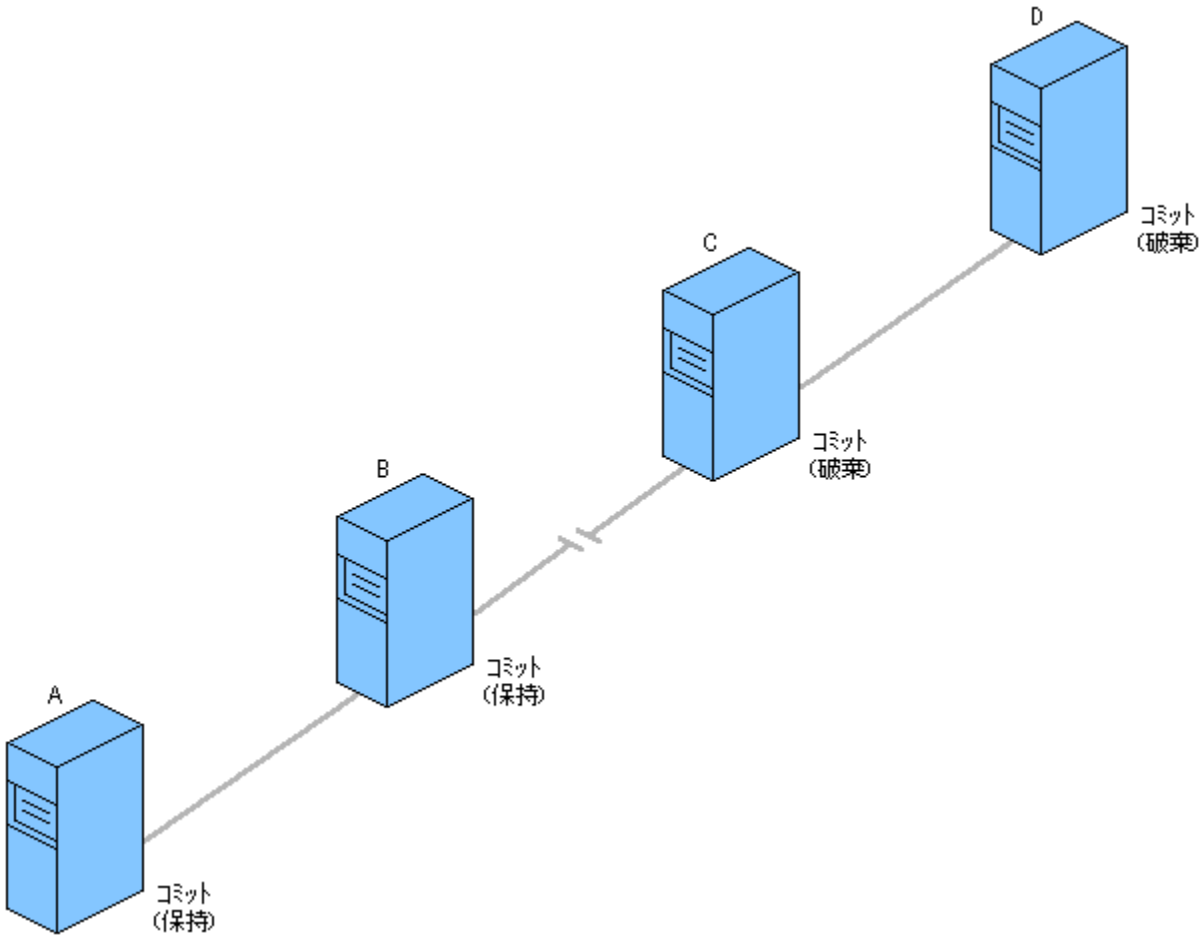
- ƒRƒ“ƒsƒ…□[ƒ^ A ‚Ì MS DTC ‚ªƒRƒ~ƒbƒg ƒR□[ƒffƒBƒl□[ƒ^‚Å‚·□B
- 2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg ƒvƒ□ƒgƒRƒ‹‚ªŽÀs‚³‚ê‚é’’Ê□M‰ñ□ü‚Å‚Í□AƒRƒ“ƒsƒ…□[ƒ^ A ‚©‚ç ƒRƒ“ƒsƒ…□[ƒ^ D ‚Ö□‡”Ô‚É’Ê□M‚ª□i□s‚µ‚Ü‚·□B
- 2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg ƒvƒ□ƒgƒRƒ‹‚ì‘æ 1 ƒtƒF□[ƒY‚ªŠ®—¹‚µ□AMS DTC ‚Í COMMITTED ƒŒƒR□[ƒh‚ðƒ□ƒO‚É□‘‚«□ž‚Ý‚Ü‚µ‚½□B
- 2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg ƒvƒ□ƒgƒRƒ‹‚ì‘æ 2 ƒtƒF□[ƒY‚Ì“r’†‚Å□AƒRƒ“ƒsƒ…□[ƒ^ B ‚ÆƒRƒ“ƒsƒ…□[ƒ^ C ‚ÌŠÔ‚Ì’Ê□M‚É□áŠQ‚ª”□¶‚µ‚Ü‚µ‚½□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Í□AŽŸ‚Ì‚æ‚¤‚È–¢‰¢□ó‘Ô‚Ì□Ü‚Ü‚É‚È‚Á‚Ä‚¢‚Ü‚·□B

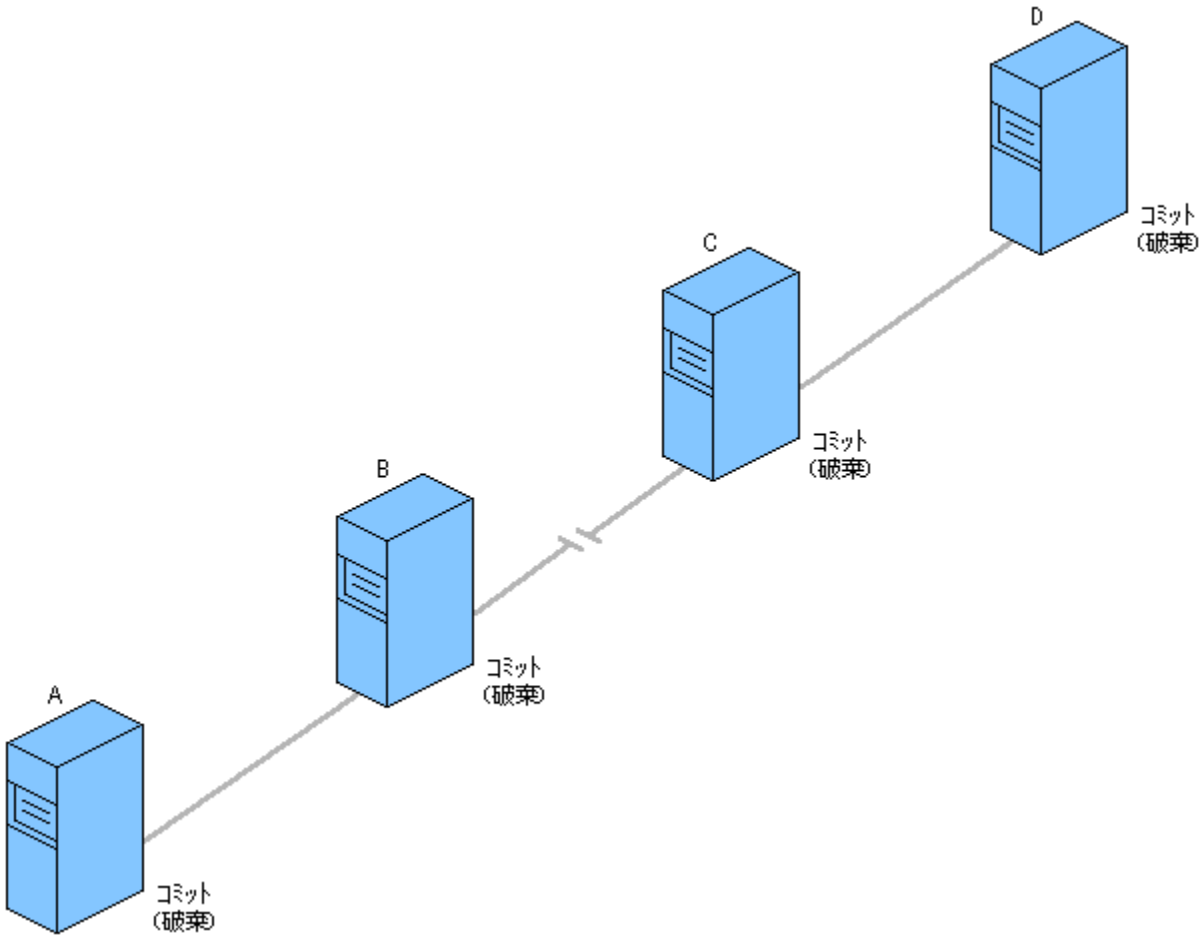


ƒRƒ“ƒsƒ…□[ƒ^ A ,ÆƒRƒ“ƒsƒ…□[ƒ^ B ,ÌŠÔ,Ì’Ê□M‰ñ□ü,ª,Ü,¾‹@”\,µ,Ä,¢,é,½,ß□AB ,àƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒRƒ~ƒbƒg,µ,Ü,µ,½□B,µ,©,µ□AƒRƒ“ƒsƒ…□[ƒ^ C ,ÆƒRƒ“ƒsƒ…□[ƒ^ D ,ªƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒRƒ~ƒbƒg,µ,½,±,Æ,ðŠm”F,·,é,Ü,Å□AA ,Æ B ,Ì,¢,¸,ê,ÌƒRƒ“ƒsƒ…□[ƒ^,àŠeŽ©,ÌƒtƒOƒtƒ@ƒCƒ‹,É COMMITTED ƒŒƒR□[ƒh,ð•ÛŽ□,µ,Ä,¨,©,È,¯,ê,Î,È,è,Ü,¹,ñ□BƒgƒŠƒ“ƒUƒNƒVƒ‡ƒ“,ð‰ðŒˆ,µ□AƒRƒ“ƒsƒ…□[ƒ^ C ,ÆƒRƒ“ƒsƒ…□[ƒ^ D ,ÌƒtƒŠ□[ƒ^ƒx□[ƒX,ÌƒtƒbƒN,ð‰ð□œ,·,é,½,ß,É□AƒVƒXƒeƒ€ŠÇ—□ŽÒ,Í□AƒRƒ“ƒsƒ…□[ƒ^ C ,É‹§“I,ÉƒgƒŠƒ“ƒUƒNƒVƒ‡ƒ“,ðƒRƒ~ƒbƒg,³,¹,Ü,· (ŽŸ,Ì□},ðŽQ□Æ,µ,Ä,,¾,³,¢)□B

ƒRƒ“ƒsƒ…□[ƒ^ C ,ÆƒRƒ“ƒsƒ…□[ƒ^ D ,ÌŠÔ,Ì’Ê□M‰ñ□ü,ª,Ü,¾‹@”\,µ,Ä,¢,é,½,ß□AƒRƒ“ƒsƒ…□[ƒ^ C ,Ì‹-□§ƒRƒ~ƒbƒg,É,æ,Á,Ä□AƒRƒ“ƒsƒ…□[ƒ^ D ,ÍƒgƒŠƒ“ƒUƒNƒVƒ‡ƒ“,ðƒRƒ~ƒbƒg,Å,«,é,æ,¤,É,È,è,Ü,·□BƒRƒ“ƒsƒ…□[ƒ^ D ,Í□AƒfƒŠ□[ƒ^ƒx□[ƒX,ÌƒtƒbƒN,ð‰ð□œ,µ□AƒgƒŠƒ“ƒUƒNƒVƒ‡ƒ“,ð”jŠü,·,é,±,Æ,ª,Å,«,é,æ,¤,É,È,è,Ü,·□BƒRƒ“ƒsƒ…□[ƒ^ D ,ªƒgƒŠƒ“ƒUƒNƒVƒ‡ƒ“,ðƒRƒ~ƒbƒg,µ□A”jŠü,µ,½,±,Æ,ðƒRƒ“ƒsƒ…□[ƒ^ C ,ªŠm”F,·,é,Æ□AƒRƒ“ƒsƒ…□[ƒ^ C ,àƒ□ƒbƒN,ð‰ð□œ,µ□AƒgƒŠƒ“ƒUƒNƒVƒ‡ƒ“,ð”jŠü,·,é,±,Æ,ª,Å,«,Ü,·□B



,±,ê,Å□A,·,×,Ä,ÌƒRƒ“ƒsƒ…□[ƒ^,Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒRƒ~ƒbƒg,³,ê,Ü,µ,½□B,½,¾,µ□AƒRƒ“ƒsƒ…□[ƒ^ C ,ªƒRƒ“ƒsƒ…□[ƒ^ B ,ÉƒRƒ~ƒbƒg,ð'Ê'm,Å,«,È,¢,Ì,Å□AƒRƒ“ƒsƒ…□[ƒ^ B ,Íƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðˆø,«'±,«•ÛŽ□,µ,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□BƒRƒ“ƒsƒ…□[ƒ^ B ,ªƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð”jŠü,µ,Ä,¢,È,¢,Ì,Å□AƒRƒ“ƒsƒ…□[ƒ^ A ,àƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð•ÛŽ□,µ,Ä,¢,È,¯,ê,Î,È,è,Ü,¹,ñ□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŠ®—¹,·,é,½,ß,É□AƒVƒXƒeƒ€ŠÇ—□ŽÒ,Í□AƒRƒ“ƒsƒ…□[ƒ^ B ,É‹□§"I,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð”jŠü,³,¹,Ü,· (ŽŸ,Ì□},ðŽQ□Æ,µ,Ä,-,¾,³,¢)□BƒRƒ“ƒsƒ…□[ƒ^ B ,Ì‹□§"jŠü,É,æ,Á,Ä□AƒRƒ“ƒsƒ…□[ƒ^ A ,àƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð"jŠü,Å,«,é,æ,¤,É,È,è,Ü,·□B,±,ê,Å□A2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg ƒvƒ□ƒgƒRƒ‹,ªŽè"®,Å□I—¹,µ□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªŠ®—¹,µ,Ü,·□B



□d—v 2 ƫƒF□[ƒY ƒRƒ~ƒbƒg ƒvƒ□ƒgƒRƒ‹,Ìˆø,«“n,µ□AŽó,¯Žæ,è’Ê□Mƒpƒ^□[ƒ“,Ì□«Ž¿□ã□AŽè“®,Åƒgƒ%ƒ“ƒUƒNƒVƒ‡ƒ“,ð‰ðŒˆ,·,é□ê□‡,Í□A’Ê□M,ªŽÕ’f,³,ê,½ˆÊ’u,É□Å,à‹ß,¢ƒRƒ“ƒsƒ…□[ƒ^,Å‰ðŒˆ,·,é,± ,Æ,ð,¨Š©,ß,µ,Ü,·□B,µ,½,ª,Á,Ä□A□ã,Ì—á,Å,Í□AƒRƒ“ƒsƒ…□[ƒ^ D ,Å,Í,È,ƒRƒ“ƒsƒ…□[ƒ^ C ,Å‹- □§ƒRƒ~ƒbƒg,ð□s,¢□AƒRƒ“ƒsƒ…□[ƒ^ A ,Å,Í,È,ƒRƒ“ƒsƒ…□[ƒ^ B ,Å‹□§”jŠü,ð□s,Á,Ä,¢,Ü,·□B

’Ê□í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÉŠÖŒW,·,éƒVƒXƒeƒ€,â‰ñ□ü,É□áŠQ,ª”□¶,µ,½□ê□‡,Í□AƒVƒXƒeƒ€,ª□Ä‹N“®,³,ê□A ‰ñ□ü,ª•œ‹Œ,µ,½,Æ,«,É MS DTC ,ªŽ©“®“I,Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‰ðŒˆ,µ,Ü,·□BƒVƒXƒeƒ€ ,ª□Ä‹N“®,³,ê,È,©,Á,½,è□A‰ñ□ü,ª•œ‹Œ,µ,È,©,Á,½,è,·,é,Æ□AMS DTC ,Íƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð ‰ðŒˆ,Å,«,Ü,¹,ñ□B,±,Ì□ê□‡□A[ƒCƒ“ ƒ_ƒEƒg]□A[ƒAƒ{□[ƒg,ð’Ê’m,Å,«,Ü,¹,ñ]□A,Ü,½,Í [ƒRƒ~ƒbƒg,ð’Ê’m,Å,«,Ü,¹,ñ] ,Ì□ó‘Ô,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðŽè“®,Å‰ðŒˆ,·,é,±,Æ,ª,Å,«,Ü,·□B

[ƒCƒ“ ƒ_ƒEƒg] ,Ì□ó‘Ô

[ƒCƒ“ ƒ_ƒEƒg] ,Ì□ó‘Ô,Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªŽq MS DTC ,É‘¶□Ý,µ□AMS DTC ,ª□€”õŠ®—¹,Ì□ó‘Ô,Å□A□e MS DTC ,ÉƒAƒNƒZƒX,Å,«,È,¢,±,Æ,ðŽ¦,µ,Ü,·□BƒCƒ“ ƒ_ƒEƒg,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð‰ðŒˆ,·,é,É,Í□AŽŸ,ÌŽè□‡,É□] ,Á,Ä‘€□ì,µ,Ü,·□B

1. [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——] ƒEƒBƒ“ƒhƒE,Å□AƒCƒ“ ƒ_ƒEƒg,Ìƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì’¼□Ú,Ì□e,ð’T,µ,Ü,·□B’¼□Ú,Ì□e,ð’T,·,É,Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒ}ƒEƒX,Ì ‰Eƒ{ƒ^ƒ“,ÅƒNƒŠƒbƒN,µ□A[ƒvƒ□ƒpƒefB] ,ðƒNƒŠƒbƒN,µ,Ü,·□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì□e MS DTC ƒRƒ“ƒsƒ… □[ƒ^,ÆŽq MS DTC ƒRƒ“ƒsƒ…□[ƒ^,ª•\Ž¦,³,ê,Ü,·□B
2. □e MS DTC ,ð’T,µ□AefRƒ“ƒsƒ…□[ƒ^,Ì [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——] ƒEƒBƒ“ƒhƒE,Å□AƒCƒ“ ƒ_ƒEƒg,Ìƒgƒ ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŒ‹‰Ê,ð’²,×,Ü,·□B
   - ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——] ƒEƒBƒ“ƒhƒE,É•\Ž¦,³,ê,Ä,¢,È,¢□ê□‡□A,»,Ìƒgƒ

‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÍƒAƒ{□[ƒg‚³‚ê‚Ä‚¨‚è□AŽqƒRƒ“ƒsƒ…□[ƒ^□ã‚Åƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒAƒ{□[ƒg‚Å‚«‚Ü‚·□B

- ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□eƒRƒ“ƒsƒ…□[ƒ^□ã‚Å [ƒRƒ~ƒbƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ] ‚Æ•\Ž¦‚³‚ê‚Ä‚¢‚é□ê□‡□A‚»‚Ìƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÍƒRƒ~ƒbƒg‚³‚ê‚Ä‚¨‚è□AŽqƒRƒ“ƒsƒ…□[ƒ^□ã‚Åƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒRƒ~ƒbƒg‚Å‚«‚Ü‚·□B
- ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□eƒRƒ“ƒsƒ…□[ƒ^□ã‚Å [ƒAƒ{□[ƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ] ‚Æ•\Ž¦‚³‚ê‚Ä‚¢‚é□ê□‡□A‚»‚Ìƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÍƒAƒ{□[ƒg‚³‚ê‚Ä‚¨‚è□AŽqƒRƒ“ƒsƒ…□[ƒ^□ã‚Åƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒAƒ{□[ƒg‚Å‚«‚Ü‚·□B
- ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□eƒRƒ“ƒsƒ…□[ƒ^‚Å [ƒCƒ“ ƒ_ƒEƒg] ‚Æ•\Ž¦‚³‚ê‚Ä‚¢‚é□ê□‡‚Í□A□eƒRƒ“ƒsƒ…□[ƒ^‚Ì [ƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒEƒBƒ“ƒhƒE‚Å□A‚»‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŽŸ‚É‹ß‚¢□e‚ð‘T‚µ‚Ü‚·□Bƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª•\Ž¦‚³‚ê‚È‚¢ (ƒAƒ{□[ƒg‚³‚ê‚Ä‚¢‚é) ‚©□A[ƒAƒ{□[ƒg‚ð‘Ê’m‚Å‚«‚Ü‚¹‚ñ] ‚Ì□ó‘Ô‚É‚È‚Á‚Ä‚¢‚é (ƒAƒ{□[ƒg‚³‚ê‚Ä‚¢‚é) ‚©□A‚Ü‚½‚Í [ƒRƒ~ƒbƒg‚ð‘Ê’m‚Å‚«‚Ü‚¹‚ñ] ‚Ì□ó‘Ô‚É‚È‚Á‚Ä‚¢‚é (ƒRƒ~ƒbƒg‚³‚ê‚Ä‚¢‚é) ‚©‚Ì□eƒRƒ“ƒsƒ…□[ƒ^‚ªŒ©‚Â‚©‚é‚Ü‚Å‚ÅAƒRƒ~ƒbƒg ƒcƒŠ□[‚ð‚³‚©‚Ì‚Ú‚Á‚Äƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð‚½‚Ç‚è‚Ü‚·□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□eƒRƒ“ƒsƒ…□[ƒ^‚ÅƒAƒ{□[ƒg‚³‚ê‚Ä‚¢‚é□ê□‡‚Í□A‚»‚Ìƒgƒ‰ƒ“ƒsƒ…□[ƒ^‚Ì’¼□Ú‚ÌŽqƒRƒ“ƒsƒ…□[ƒ^‚ÅƒAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚Å‹-□§“I‚ÉƒAƒ{□[ƒg‚µ‚Ü‚·□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª□eƒRƒ“ƒsƒ…□[ƒ^‚ÅƒAƒRƒ~ƒbƒg‚³‚ê‚Ä‚¢‚é□ê□‡‚Í□A□eƒRƒ“ƒsƒ…□[ƒ^‚Ì’¼□Ú‚ÌŽqƒRƒ“ƒsƒ…□[ƒ^‚ÅƒAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚Å‹-□§“I‚ÉƒRƒ~ƒbƒg‚µ‚Ü‚·□B

3 ŽqƒRƒ“ƒsƒ…□[ƒ^‚ÅƒAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒRƒ~ƒbƒg‚µ‚½ƒAƒ{□[ƒg‚Ì□ó‘Ô‚ðŠm”F‚µ‚Ü‚·□B

[ƒRƒ~ƒbƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ]

[ƒRƒ~ƒbƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ] ‚Ì□ó‘Ô‚ÍAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ªƒRƒ~ƒbƒg‚³‚ê‚½‚É‚à‚©‚©‚í‚炸A□eƒRƒ“ƒsƒ…□[ƒ^‚ê•””Ì□]‘® MS DTC ‚É‘Ê’m‚³‚ê‚È‚©‚Á‚½‚±‚Æ‚ðŽ¦‚µ‚Ü‚·□BŽŸ‚ÌŽè□‡‚É‚Ä□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚Å‰ðŒˆ‚Å‚«‚Ü‚·□B[ƒRƒ~ƒbƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ] Ì□ó‘Ô‚ÌƒgƒUƒNƒVƒ‡ƒ“‚ðƒEƒBƒ“ƒhƒE‚Ì‰E‰º‚É‚ ‚é MS DTC ‚Æ□]‘® MS DTC ‚ª•\Ž¦‚³‚ê‚Ü‚·□B□]‘® MS DTC ‚ŒÆ©‚Â‚©‚Á‚½‚çAŠe MS DTC ‚Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚Å‹-□§“I‚ÉƒRƒ~ƒbƒg‚µ‚Ü‚·□B·‚×‚Ä□]‘® MS DTC ‚Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒRƒ~ƒbƒg‚µ‚½ƒAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª [ƒRƒ~ƒbƒg‚ð‘Ê’m‚Å‚«‚Ü‚¹‚ñ] ‚Æ•\Ž¦‚³‚ê‚Ä‚¢‚é MS DTC ‚É–ß‚è□A‚»‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Í□eƒRƒ“ƒsƒ…□[ƒ^‚ÅƒAƒRƒ~ƒbƒg‚³‚ê‚Ä‚¢‚é MS DTC ‚É‘Ê’m‚³‚ê‚Ü‚·□B

Œx□□ ‚·‚×‚Ä□]‘® MS DTC ‚Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŒ‹‰Ê‚ª‘Ê’m‚³‚ê‚é‚Ü‚Å‚ÍAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒRƒ~ƒbƒg‚µ‚È‚¢‚Å‚¾‚³‚¾‚¢□B

[ƒAƒ{□[ƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ]

[ƒAƒ{□[ƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ] Ì□ó‘Ô‚ÍAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ªƒAƒ{□[ƒg‚³‚ê‚½‚É‚à‚©‚©‚í‚炸A□eƒRƒ“ƒsƒ…□[ƒ^‚ê•””Ì□]‘® MS DTC ‚É‘Ê’m‚³‚ê‚È‚©‚Á‚½‚±‚Æ‚ðŽ¦‚µ‚Ü‚·□B±‚Ì□ó‘Ô‚ÍA□[ƒAƒ{□[ƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ] Ì□ó‘Ô‚ÌƒgƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚Å‰ðŒˆ‚Å‚«‚Ü‚·□B[ƒAƒ{□[ƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ] Ì□ó‘Ô‚ÌƒgƒUƒNƒVƒ‡ƒ“‚ðƒEƒBƒ“ƒhƒE‚Ì‰E‰º‚É‚ ‚é MS DTC ‚Æ□]‘® MS DTC ‚ª•\Ž¦‚³‚ê‚Ü‚·□B□]‘® MS DTC ‚ŒÆ©‚Â‚©‚Á‚½‚çAŠe MS DTC ‚Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚Å‹-□§“I‚ÉƒRƒ~ƒbƒg‚µ‚Ü‚·□B·‚×‚Ä□]‘® MS DTC ‚Åƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒRƒ~ƒbƒg‚µ‚½ƒAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ª [ƒAƒ{□[ƒg‚Ê’m‚Å‚«‚Ü‚¹‚ñ] ‚Æ•\Ž¦‚³‚ê‚Ä‚¢‚é MS DTC ‚É–ß‚è□A‚»‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Í□eƒRƒ“ƒsƒ…□[ƒ^‚ÅƒAƒRƒ~ƒbƒg‚³‚ê‚Ä‚¢‚é MS DTC ‚É‘Ê’m‚³‚ê‚Ü‚·□B

Œx□□ ‚·‚×‚Ä□]‘® MS DTC ‚Éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÌŒ‹‰Ê‚ª‘Ê’m‚³‚ê‚é‚Ü‚Å‚ÍAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒRƒ~ƒbƒg‚µ‚È‚¢‚Å‚¾‚³‚¾‚¢□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠ‰ðŒˆ‚·‚é•û–@

1 MTS ƒGƒfƒXƒvƒ□□[ƒ‰Ìƒ‰ƒEƒBƒ“ƒhƒE‚Ì¶‘¤‚Å□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŠ‰ðŒˆ‚·‚é•û–@‚ðŽ¦‚µ‚Ü‚·□B

2 [ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚Ìˆê——] ƒAƒCƒRƒ“‚ðƒ_ƒuƒ‹ƒNƒŠƒbƒN‚µ‚Ü‚·□B

3 ƒEƒBƒ“ƒhƒE‚Ì‰E‘¤‚Å□A‰ðŒˆ‚·‚é‚Ü‚Å‚ÍƒgƒUƒNƒVƒ‡ƒ“‚ðƒEƒBƒ“ƒhƒE‚Ì‰E‰º‚É‚ ‚é‚Ü‚Å‚ÅAƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ðŽè“®‚ÅƒRƒ~ƒbƒg‚µ‚Ü‚·□B

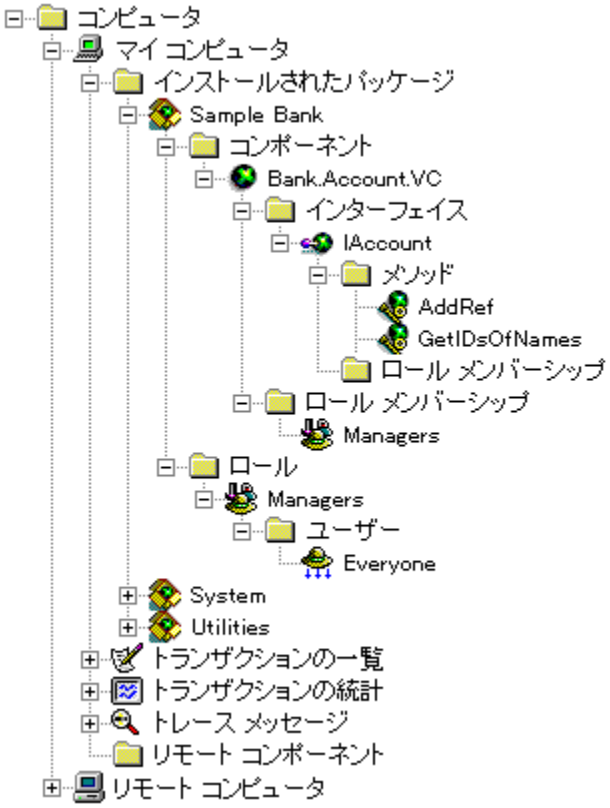
4 [‰ðŒˆ] ,ðƒ|ƒCƒ“ƒg,µ□A[ƒRƒ~ƒbƒg]□A[ƒAƒ{□[ƒg] ,Ü,½,Í [”jŠü] ,ðƒNƒŠƒbƒN,µ,Ü,·□B

ŠÖ˜A□€–Ú

MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,É,Â,¢,Ä□AMS DTC ,ÌŠÇ—□□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹□AWindows 95/98 ,Å MTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠÄŽ‹□AMTS ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì□ó‘Ô,É,Â,¢,Ä□A[ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ìˆê——] fEƒBƒ“ƒhƒE

MTS ŠÇ—□,ÌŽ©“®‰»

Microsoft Transaction Server (MTS) ŠJ”ŽÒ,Æ□ã‹‰ Web ŠÇ—□ŽÒ,Í□AŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä□AMTS ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Ì”z’u,ÆŠÇ—□,ðŽ©“®‰»,Å,«,Ü,·□BƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,Í□AMicrosoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒRƒŒƒNƒVƒ‡ƒ“,ÌŠK‘w,É‘Î‰ž,µ,Ä,¢ ,Ü,·□B“K□Ø,È "ŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg" ,ÅƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŒÄ,Ñ□o,·,±,Æ,É,æ,Á,Ä□AŠÇ— □□ì‹Æ,ðŽ©“®‰»»,·,é,±,Æ,ª,Å,«,Ü,·□BŽŸ,Ì□},Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰,É,æ,Á,ÄŠÇ— □,¨,æ,Ñ”z’u,³,ê,éƒRƒŒƒNƒVƒ‡ƒ“,ÌŽí—Þ,ðŽ¦,µ,Ä,¢,Ü,·□B



ŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŒø—¦“I,ÉŽg,¤,½,ß,É□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,É,æ,Á,ÄŽ©“®‰»»,³,ê,é MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì□ì‹Æ,ð□•ª,É—□‰ð,µ,Ä,,¾,³,¢□BŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,Í□A**IDispatch** ƒCƒ“ƒ^□[ƒtƒFƒCƒX,©,ç”h□¶,µ,Ä,¢ ,é,Ì,Å□AƒI□[ƒgƒ□□[ƒVƒ‡ƒ“ŒÝŠ·,Ì”C^Ó,ÌŒ¾Œê,ðŽg,Á,Ä□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ðŠJ”,Å,«,Ü,·□BActiveX® ƒefNƒmƒ□ƒW,¨,æ,Ñ COM ,ðƒTƒ|□[ƒg,µ,Ä,¢,é Microsoft® Visual Basic® Version 5.0 ^È□~,¨,æ,Ñ Microsoft® Visual C++® Version 5.0 ^È□~,ðŠJ”ƒc□[ƒ‹,Æ,µ,ÄŽg,¤,±,Æ,ð,¨Š©,ß,µ,Ü,·□BŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ÌŽg,¢•û,ð Visual Basic ,ìƒTƒ“ƒvƒ‹,É,Â,¢,Ä,Í□A□uMTS Administrative Reference (ŠÇ— □ƒŠƒtƒ@ƒŒƒ“ƒX)□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

,±,±,Å,Í□AŽŸ,ÌƒgƒsƒbƒN,É,Â,¢,Ä□à–¾,µ,Ü,·□B

MTS ŠÇ—□ƒIƒuƒWƒFƒNƒg

MTS ŠÇ—□Ž©“®‰» Visual Basic Script ƒTƒ“ƒvƒ‹

MTS ŠÇ—□Ž©“®‰» Visual Basic ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“

Visual Basic ,É,æ,é MTS ŠÇ—□,ÌŽ©“®‰»

Visual Basic ,É,æ,é□,“x,È MTS ŠÇ—□,ÌŽ©“®‰»

ŠÖ˜A□€–Ú

MTS Administrative Reference (ŠÇ—□fŠƒtƒ@ƒŒƒ“ƒX)

MTS ŠÇ—□ƒIƒuƒWƒFƒNƒg

MTS ,ÌŠÇ—□,É,Í□AŽŸ,ÌƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŽg,¢,Ü,·□B

Catalog ƒIƒuƒWƒFƒNƒg

CatalogObject ƒIƒuƒWƒFƒNƒg

CatalogCollection ƒIƒuƒWƒFƒNƒg

PackageUtil ƒIƒuƒWƒFƒNƒg

ComponentUtil ƒIƒuƒWƒFƒNƒg

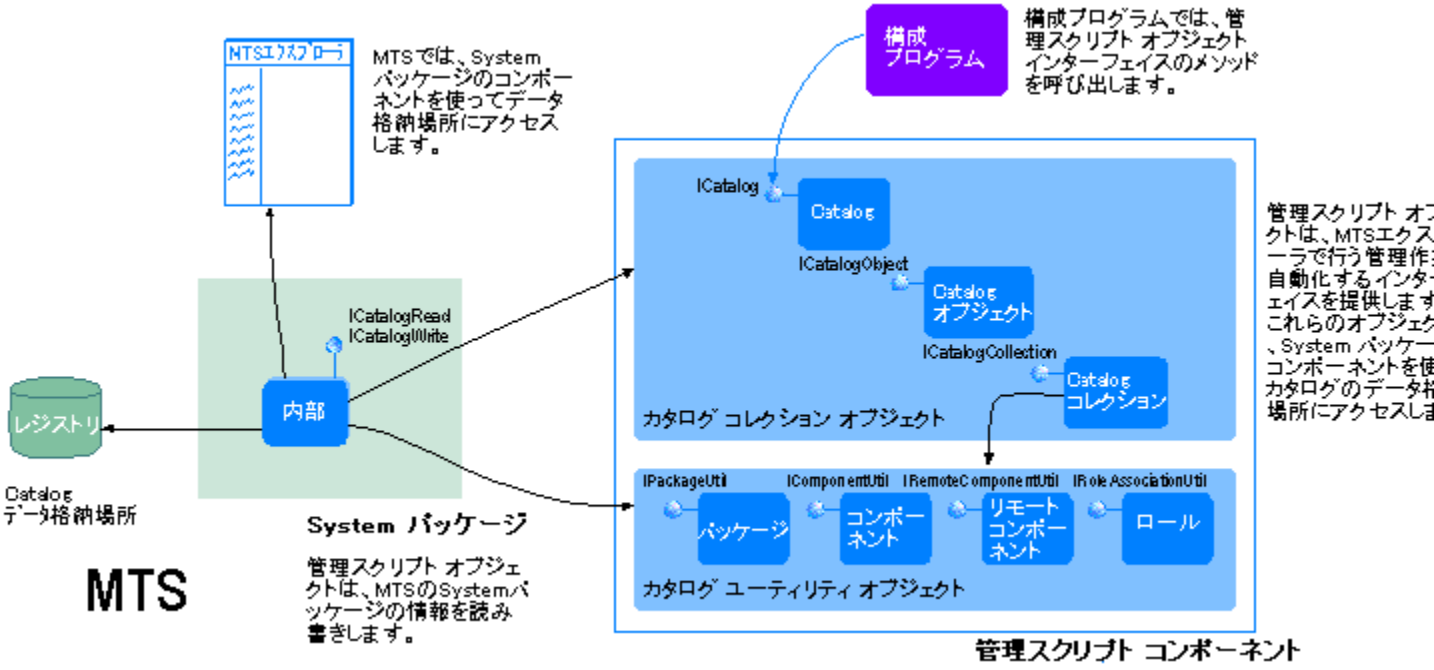
RemoteComponentUtil ƒIƒuƒWƒFƒNƒg

RoleAssociationUtil ƒIƒuƒWƒFƒNƒg

,±,ê,ç,ÌƒIƒuƒWƒFƒNƒg,Í□A,³,Ü,´,Ü,ÈƒXƒNƒŠƒvƒg,Ì—v‹□,É‰ž,¶,Ä□A"Ä—pƒCƒ"ƒ^□[ƒtƒFƒCƒX,Æƒt□[ƒeƒBƒŠƒeƒB ƒCƒ"ƒ^□[ƒtƒFƒCƒX,ð'ñ‹Ÿ,µ,Ü,·□B**Catalog** ƒIƒuƒWƒFƒNƒg□A**CatalogObject** ƒIƒuƒWƒFƒNƒg□A,¨,æ,Ñ **CatalogCollection** ƒIƒuƒWƒFƒNƒg,Í□AƒJƒ^□fO ƒRƒŒƒNƒVƒ‡ƒ" ƒIƒuƒWƒFƒNƒg'w,ð□\ □¬,µ□AƒIƒuƒWƒFƒNƒg,Ì□ì□¬,â•Ï□X,È,Ç□A□Å□ãˆÊƒŒƒxƒ‹,Ì‹@"\,ð'ñ‹Ÿ,µ,Ü,·□B**Catalog** ƒIƒuƒWƒFƒNƒg,ðŽg,¤ ,Æ□A"Á'è,ÌƒT□[ƒo□[,É□Ú'±,µ□AƒRƒŒƒNƒVƒ‡ƒ",ÉƒAƒNƒZƒX,·,é,±,Æ,ª,Å,«,Ü,·□B**CatalogCollection** ƒIƒuƒWƒFƒNƒg,ðŒÄ,Ñ□o,·,±,Æ,É,æ,Á,Ä□AƒIƒuƒWƒFƒNƒg,Ì— ñ‹"□A□ì□¬□A□í□œ□A,¨,æ,Ñ•Ï□X,ð□s,Á,½,è□AŠÖ˜A,·,éƒRƒŒƒNƒVƒ‡ƒ",ÉƒAƒNƒZƒX,µ,½,è,Å,«,Ü,·□B**Catalo gObject** ƒIƒuƒWƒFƒNƒg,Í□AƒIƒuƒWƒFƒNƒg,Ìƒvƒ□fpƒefB,ÌŽæ"¾,Æ□Ý'è,ð□s,¤,Æ,«,ÉŽg,¢,Ü,·□B

ƒJƒ^ƒ□fO ƒt□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg'w,Å,Í□A**PackageUtil** ƒIƒuƒWƒFƒNƒg□A**ComponentUtil** ƒIƒuƒWƒFƒNƒg□A**RemoteComponentUtil** ƒIƒuƒWƒFƒNƒg□A,¨,æ,Ñ **RoleAssociationUtil** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä□AƒRƒ"ƒ□[ƒIƒ"ƒg,Ì'g,Ý□ž,Ý,âƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg,È,Ç□A"Á'è,Ì□ì‹Æ,ðŽ©"® ‰»,·,é,±,Æ,ª,Å,«,Ü,·□B,±,Ìƒt□[ƒeƒBƒŠƒeƒB'w,Å,Í□Aƒt□[ƒU□[,Ü,½,Íƒt□[ƒU□[,ÌƒNƒ ‰ƒX,Æƒ□□[ƒ‹,Æ,ÌŠÖ˜A•t,¯,È,Ç□AƒRƒŒƒNƒVƒ‡ƒ",É'Î,·,é"Á'è,Ì□ì‹Æ,ð□ˆ—□,·,éƒvƒ□fOƒ‰ƒ€,ð□ì□¬,Å,«,Ü,·□B

ŽŸ,Ì□},Í□A□\□¬—p,Ìƒvƒ□fOƒ‰ƒ€,ªŠÇ—□ƒIƒuƒWƒFƒNƒg,Ìƒ□ƒ\ ƒbƒh,ðŽg,Á,Ä□AƒIƒuƒWƒFƒNƒg,Ìƒf□[ƒ^,Ì"Ç,Ý□ž,Ý□A,¨,æ,Ñ□',«□ž,Ý,ð□s,¤•û–@,ðŽ¦,µ,Ä,¢,Ü,·□B



ŠÇ—□ƒIƒuƒWƒFƒNƒg,ÌƒTƒ“ƒvƒ‹,ÆƒŠƒtƒ@ƒŒƒ“ƒX ƒhƒLƒ…ƒ□ƒ“ƒg,Ì“üŽè•û–@,É,Â,¢,Ä,Í□A□uŠJ”ŽÒ—pƒTƒ“ƒvƒ‹,ÆƒhƒLƒ…ƒ□ƒ“ƒg,ÌƒCƒ“ƒXƒg□[ƒ‹□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜A□€–Ú

MTS Administrative Reference (ŠÇ—□ƒŠƒtƒ@ƒŒƒ“ƒX)

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods

MTS ƒJƒ^ƒ□ƒO ƒRƒŒƒNƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg,ðŽg,¤

MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÌŠeƒtƒHƒ‹ƒ_,Í□AƒJƒ^ƒ□ƒO,É•Û‘¶,³,ê,Ä,¢,éŠeƒRƒŒƒNƒVƒ‡ƒ“,É‘Î‰ž,µ,Ü,·□B**Catalog** ƒIƒuƒWƒFƒNƒg□A**CatalogObject** ƒIƒuƒWƒFƒNƒg□A,¨,æ,Ñ **CatalogCollection** ƒIƒuƒWƒFƒNƒg,Í□AƒJƒ^ƒ□ƒO ƒRƒŒƒNƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg‘w,ð□\□¬,µ,Ü,·□B,±,ê,ç,ÌƒIƒuƒWƒFƒNƒg,ðŽg,¤ ,Æ□AƒRƒŒƒNƒVƒ‡ƒ“,Ì□ì□¬□A□í□œ□A,Ü,½,Í•Ï□X,È,Ç,Ì^ê”Ê“I,ÈŠÇ—□□ì‹Æ,ðŽ©“® ‰»,µ□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,É‘g,Ý□ž,Þ,±,Æ,ª,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods

MTS CatalogCollection ƒIƒufWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,·,é

MTS ƒGƒNƒXƒvƒ□□[ƒ‰
,ÌƒtƒHƒ‹ƒ_“à,Ì”CˆÓ,ÌƒRƒŒƒNƒVƒ‡ƒ“,Ü,½,Íƒf□[ƒ^,ÉƒAƒNƒZƒX,·,é,½,ß,É□A**CatalogCollection** ƒIƒufWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,·,é,±,Æ,ª,Å,«,Ü,·□B**CatalogCollection** ƒIƒufWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,·,é,É,Í□A**GetCollection** ƒ□ƒ\ƒbƒh,ðŽg—p,µ□AƒRƒŒƒNƒVƒ‡ƒ“–¼,ð“n,µ,Ü,·□B**CatalogCollection** ƒIƒufWƒFƒNƒg,Í□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰
,ÌƒtƒHƒ‹ƒ_“à,Ìƒf□[ƒ^,È,Ç□A”CˆÓ,ÌƒRƒŒƒNƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,·,é,½,ß,ÉŽg,í,ê,é”Ä—pƒIƒufWƒFƒNƒg,Å,·□B**Catalog** ƒIƒufWƒFƒNƒg,©,ç **GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ□A□Å□ãˆÊƒŒƒxƒ‹,ÌƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,·,é,±,Æ,ª,Å,«,Ü,·□B,Ü,½□A**CatalogCollection** ƒIƒufWƒFƒNƒg,©,ç **GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,ÄŠÖ˜A,·,éƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,·,é,±,Æ,à,Å,«,Ü,·□B

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods

MTS CatalogCollection ƒIƒuƒWƒFƒNƒg,Éƒf□[ƒ^,ð‘}“ü,·,é

CatalogCollection ƒIƒuƒWƒFƒNƒg,ÉƒJƒ^ƒ□ƒO,Ìƒf□[ƒ^,ð‘}“ü,·,é,É,Í□AŠÇ—□ƒIƒuƒWƒFƒNƒg,ðŽg,¢
,Ü,·□BƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,½,¾,¯,Å,Í□AƒJƒ^ƒ□ƒO,Ìƒf□[ƒ^,Í“Ç,Ý□ž,Ü,ê,Ü,¹,ñ□BƒRƒŒƒNƒVƒ‡ƒ
“‘à,Åƒf□[ƒ^,ðŽQ□Æ,Ü,½,Í•Ï□X,·,é,É,Í□A,Ü,¸ **Populate** ƒ□ƒ\ƒbƒh,Ü,½,Í **PopulateByKey** ƒ□ƒ\
ƒbƒh,ðŒÄ,Ñ□o,µ,Ü,·□B**Populate** ƒ□ƒ\
ƒbƒh,Í□A,·,×,Ä,Ìƒf□[ƒ^,ðƒRƒŒƒNƒVƒ‡ƒ“,É“Ç,Ý□ž,Ý,Ü,·□B**PopulateByKey** ƒ□ƒ\
ƒbƒh,Í□AŽw’è,µ,½ƒIƒuƒWƒFƒNƒg,¾,¯,ð“Ç,Ý□ž,Ý,Ü,·□BƒRƒŒƒNƒVƒ‡ƒ“,Éƒf□[ƒ^,ð’Ç
‰Á,µ,½,è□Aƒ†□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŽg—
p,µ,½,è,·,é□ê□‡,Í□AƒRƒŒƒNƒVƒ‡ƒ“,Éƒf□[ƒ^,ð‘}“ü,·,é•K—v,Í, ,è,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods

MTS ƒIƒuƒWƒFƒNƒg‚ÌŽæ“¾‚¨‚æ‚Ñƒvƒ□ƒpƒeƒB‚ÌŽæ“¾‚Æ□Ý’è

CatalogCollection ƒIƒuƒWƒFƒNƒg‚Í□AƒRƒŒƒNƒVƒ‡ƒ“‚à‚ÌƒIƒuƒWƒFƒNƒg‚É‘Î‚·‚éŒJ‚è•Ô‚µ□ˆ—□‚ðƒTƒ|□[ƒg‚µ‚Ä‚¢‚·□BŠJ”fc□[ƒ‹‚Æ‚µ‚Ä Visual Basic ‚ðŽg‚Á‚Ä‚¢‚é□ê□‡‚Í□A**For Each** ƒXƒe□[ƒgƒ□ƒ“ƒg‚ðŽg‚Á‚ÄŒJ‚è•Ô‚µ□ˆ—□‚ð□s‚¤‚±‚Æ‚ª‚Å‚«‚Ü‚·□BVisual C++ ‚ðŽg‚Á‚Ä‚¢‚é□ê□‡‚Í□A**Item** ƒ□ƒ\ƒbƒh‚Æ **Count** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚ÄƒRƒŒƒNƒVƒ‡ƒ“‚à‚ÌƒIƒuƒWƒFƒNƒg‚ð—ñ‹“‚µ‚Ü‚·□BƒIƒuƒWƒFƒNƒg‚Ìƒvƒ□ƒpƒeƒB‚ÉƒAƒNƒZƒX‚·‚é‚É‚Í□AƒIƒuƒWƒFƒNƒg‚Ì **Value** ƒvƒ□ƒpƒeƒB‚ðŽg‚Á‚Ä□Aƒvƒ□ƒpƒeƒB–¼‚ðƒpƒ‰ƒ□□[ƒ^‚Æ‚µ‚Ä“n‚µ‚Ü‚·□B

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods

□V,µ,¢ MTS ƒIƒuƒWƒFƒNƒg,Ì□ì□¬

ˆê•”,ÌƒRƒCƒNƒVƒ‡ƒ“,Å,Í□A**Add** ƒ□ƒ\ƒbƒh,É,æ,é□V,µ,¢ƒIƒuƒWƒFƒNƒg,Ì□ì□¬,ðƒTƒ|□[ƒg,µ,Ä,¢ ,Ü,·□B,»,Ì,Ù,©,ÌƒRƒCƒNƒVƒ‡ƒ“,Å,Í□Aƒt□[ƒeƒBƒŠƒeƒB ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŽg,Á,Ä□AƒIƒuƒWƒFƒNƒg (ƒRƒ“ƒ|□[ƒlƒ“ƒg,È,Ç) ,ðƒCƒ“ƒXƒg□[ƒ‹,·,é•K—v,ª, ,è,Ü,·□B□Ú□×,É,Â,¢,Ä,Í□A□uMTS ƒIƒ^□fO ƒt□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg,ðŽg,¤□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods

MTS ƒIƒuƒWƒFƒNƒg‚Ö‚Ì•Ï□X‚ð•Û‘¶‚·‚é

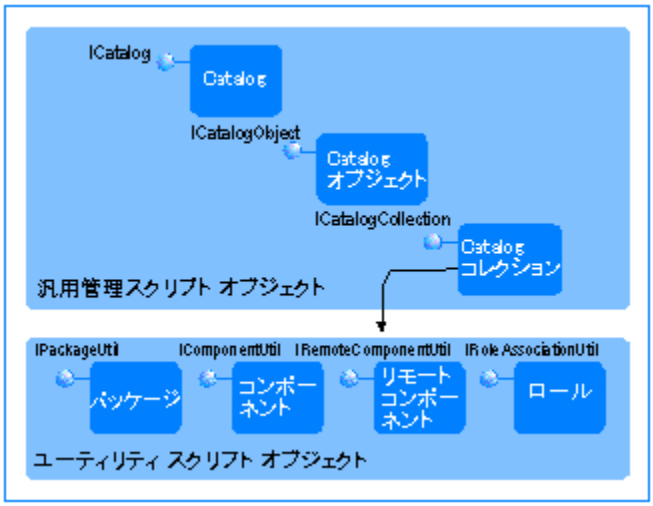
ƒvƒ□ƒpƒeƒB‚Ì•Ï□X□A(**Add** ƒ□ƒ\ƒbƒh‚É‚æ‚Á‚Ä□ì□¬‚³‚ê‚½) □V‚µ‚¢ƒIƒuƒWƒFƒNƒg□AƒIƒuƒWƒFƒNƒg‚Ì□í□œ(‚Ù‚Æ‚ñ‚Ç‚ÌƒRƒŒƒNƒVƒ‡ƒ“‚ªƒTƒ|□[ƒg‚µ‚Ä‚¢‚é **Remove** ƒ□ƒ\ƒbƒh‚É‚æ‚é□í□œ) ‚Í□A**SaveChanges** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚·‚Ü‚Å□Aƒ□ƒ‚ƒŠ‚É•Û‘¶‚³‚ê‚Ä‚¢‚Ü‚·□B**SaveChanges** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚·‚Æ□A**CatalogCollection** ƒIƒuƒWƒFƒNƒg‚É‰Á‚¦‚ç‚ê‚½‚Ü‚Ü‚Ì•Ï□X‚ª□ƒf□fO‚É“K—p‚³‚ê‚Ü‚·□B**SaveChanges** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚·‘O‚É□A**CatalogCollection** ƒIƒuƒWƒFƒNƒg‚ð‰ð•ú‚µ‚½‚è□A**Populate** ƒ□ƒ\ƒbƒh‚Ü‚½‚Í **PopulateByKey** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚½‚è‚·‚é‚Æ□Aƒ□ƒ‚ƒŠ‚É•Û‘¶‚³‚ê‚Ä‚¢‚é•Ï□X‚Í□A‚·‚×‚Ä‚Ž¸‚í‚ê‚Ü‚·□B

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods

MTS ƒJƒ^ƒ□ƒO ƒt□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg,ðŽg,¤

ƒJƒ^ƒ□ƒO ƒt□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg‘w,Í□A**PackageUtil** ƒIƒuƒWƒFƒNƒg□A**ComponentUtil** ƒIƒuƒWƒFƒNƒg□A**RemoteComponentUtil** ƒIƒuƒWƒFƒNƒg□A,¨,æ,Ñ **RoleAssociationUtil** ƒIƒuƒWƒFƒNƒg,©,ç□\□¬,³,ê,Ü,·□B,±,ê,ç,ÌƒIƒuƒWƒFƒNƒg,Í□AƒJƒ^ƒ□ƒO,É‘Î,µ,Ä“Á’è,Ì□ì‹Æ (ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì‘g,Ý□ž,Ý,âŠù□¬,ÌƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,È,Ç) ,ð’¼□Ú□s,¤,Æ,«,ÉŽg,¢,Ü,·□Bƒt□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg,Í□AƒRƒŒƒNƒVƒ‡ƒ“,ÌŽí—Þ,É‘Î‰ž,µ,Ä,¢,Ü,·□Bƒt□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg,ðŽæ“¾,·,é,É,Í□A**CatalogCollection** ƒIƒuƒWƒFƒNƒg□ã,Å **GetUtilInterface** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ü,·□B



ƒt□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg,ðŽg,Á,½□ê□‡□AƒJƒ^ƒ□ƒO,É‰Á,¦,ç,ê,½•Ï□X,Í□Aƒt□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg,ÌŽæ“¾Œ³,ì **CatalogCollection** ƒIƒuƒWƒFƒNƒg,É,Í”½‰f,³,ê,Ü,¹,ñ□B•Ï□X,ðŠm”F,·,é,É,Í□A**CatalogCollection** ƒIƒuƒWƒFƒNƒg□ã,Å **Populate** ƒ□ƒ\ƒbƒh,Ü,½,Í **PopulateByKey** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,³,È,¯,ê,Î,È,è,Ü,¹,ñ□B•Ï□X,Í,·,®,ÉƒJƒ^ƒ□ƒO,É‘,«□ž,Ü,ê,é,Ì,Å□Aƒt□[ƒeƒBƒŠƒeƒB ƒ□ƒ\ƒbƒh,ðŽg,Á,½Œã,Å **SaveChanges** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,·•K—v,Í, ,è,Ü,¹,ñ□B

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods

MTS ƒJƒ^ƒ□ƒO,ÌƒGƒ‰□[□^—□

ƒJƒ^ƒ□ƒO ƒRƒŒƒNƒVƒ‡ƒ“ ƒ□ƒ\ƒbƒh,ÆƒJƒ^ƒ□ƒO ƒ†□[ƒeƒBƒŠƒeƒB ƒ□ƒ\ƒbƒh,Í□A□¬Œ÷,Ü,½,ÍŽ¸”s,ðŽ¦,· HRESULT ,ð•Ô,µ,Ü,·□BVisual Basic Version 5.0 ,Å,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,Æ **Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä,± ,ê,ç,ÌƒGƒ‰□[,ðƒgƒ‰ƒbƒv,µ□AƒGƒ‰□[ƒR□[ƒh,ÉƒAƒNƒZƒX,µ,Ü,·□B‘½,,ÌƒIƒuƒWƒFƒNƒg,ðˆµ,¤ƒ□ƒ\ƒbƒh (**SaveChanges** ƒ□ƒ\ƒbƒh,â **InstallPackage** ƒ□ƒ\ƒbƒh,È,Ç) ,Å,Í□A“Á’è,ÌƒIƒuƒWƒFƒNƒg,ÌƒGƒ‰□[,ð□à–¾,·,é•¡□”,ÌƒGƒ‰□[ƒR□[ƒh,ªŽæ,è□ž,Ü,ê,é,±,Æ,ª, ,è,Ü,·□B,± ,ÌƒR□[ƒh,ÌƒZƒbƒg,ÉƒAƒNƒZƒX,·,é,É,Í□A**ErrorInfo** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽg,¢,Ü,·□B**ErrorInfo** ƒRƒŒƒNƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,·,é,É,Í□A**GetCollection** ƒ□ƒ\ƒbƒh,ðŽg,¢,Ü,·□Bƒ†□[ƒU□[,ª□¶□¬,·,é **CatalogCollection** ƒIƒuƒWƒFƒNƒg,ÌŠeƒCƒ“ƒXƒ^ƒ“ƒX,Í□A□ÅŒã,ÉŽ¸”s,µ,½ƒ□ƒ\ƒbƒhŒÄ,Ñ□o,µ,ÌƒGƒ ‰□[ƒR□[ƒh,ªŠi”[,³,ê,½ **ErrorInfo** ƒRƒŒƒNƒVƒ‡ƒ“,ð•ÛŽ□,µ,Ä,¢,Ü,·□BƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ä,¢ ,é,Æ,«,Í□A,»,ÌƒIƒuƒWƒFƒNƒg,Ì **ErrorInfo** ƒRƒŒƒNƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,·,é,±,Æ,É,æ,Á,Ä□A,Ç,ÌƒRƒ“ƒ| □[ƒIƒ“ƒg,ªŠù,É‘g,Ý□ž,Ü,ê,Ä,¢,é,©,ðŠm”F,·,é,±,Æ,ª,Å,«,Ü,·□B

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□ì□¬,·,é,Æ,«,Í□AŠeƒ□ƒ\ ƒbƒhŒÄ,Ñ□o,µ,ª□¬Œ÷,µ,½,©□A,Ü,½,ÍŽ¸”s,µ,½,©,ðƒ`ƒFƒbƒN,·,é,æ,¤,Éƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO,·,é,± ,Æ,ð,¨Š©,ß,µ,Ü,·□B“Á,É□AƒRƒŒƒNƒVƒ‡ƒ“–¼,Ü,½,Íƒvƒ□ƒpƒeƒB–¼,ð“n,·,Æ,«,Í□Aƒvƒ□ƒOƒ‰ƒ€,Å E_INVALIDARG ƒŠƒ^□[ƒ“ ƒR□[ƒh (Visual Basic Version 5.0 ,Å,Íƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰□[5) ,ðƒeƒXƒg,µ,Ä,- ,¾,³,¢□B,±,ÌƒR□[ƒh,Í□A“n,³,ê,½ƒRƒŒƒNƒVƒ‡ƒ“–¼,Ü,½,Íƒvƒ□ƒpƒeƒB–¼,Ì,¤,¿□A1 ,Â,Ü,½,Í•¡□”,ªƒTƒ| □[ƒg,³,ê,Ä,¢,È,¢,±,Æ,ðŽ¦,µ,Ü,·□B

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AMTS ErrorInfo Collection

MTS ŠÇ—□Ž©“®‰» Visual Basic Script ƒTƒ“ƒvƒ‹

Visual Basic Scripting Edition (VBScript) ,È,Ç,ÌƒI□[ƒgƒ□□[ƒVƒ‡ƒ“ŒÝŠ·,ÌŒ¾Œê,ðŽg,Á,Ä□AŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŒÄ,Ñ□o,·,±,Æ,ª,Å,«,Ü,·□BMTS ,É,Í□AŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì□^—□,ðŽ©“®‰»,·,é•û–@,ðŽ¦,·ŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ª—p^Ó,³,ê,Ä,¢,Ü,·□B

HTML ƒy□[ƒW^ÈŠO,ÅŽ©“®‰»ƒXƒNƒŠƒvƒg,ðŽÀ□s,·,é,É,Í□AŽg—p,µ,Ä,¢,éƒRƒ“ƒsƒ…□[ƒ^,É Windows® Scripting Host (WSH) ,ðƒCƒ“ƒXƒg□[ƒ‹,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BWSH ,Í□AƒRƒ}ƒ“ƒhƒ‰ƒCƒ“ ƒXƒNƒŠƒvƒg ƒt□[ƒeƒBƒŠƒefB,Å□AWindows NT 4.0 Option Pack ,ðŽg,Á,Ä Windows NT ƒRƒ“ƒsƒ… □[ƒ^,ÉƒCƒ“ƒXƒg□[ƒ‹,Å,«,Ü,·□BWSH ,ðŽg,¤,Æ□AƒXƒNƒŠƒvƒg,ð’¼□Ú Windows ,ÌƒffƒXƒNƒgƒbƒv□ã,Å□A,Ü,½,ÍƒRƒ}ƒ“ƒh ƒvƒ□ƒ“ƒvƒg,©,çŽÀ□s,Å,«,Ü,·□BƒXƒNƒŠƒvƒg,ð HTML ƒhƒLƒ…ƒ□ƒ“ƒg,É–„,ß□ž,Þ•K—v,Í, ,è,Ü,¹,ñ□BƒXƒNƒŠƒvƒg,Í□AffƒXƒNƒgƒbƒv□ã,ÅƒXƒNƒŠƒvƒg ƒtƒ@ƒCƒ‹,ðƒ_ƒuƒ‹ƒNƒŠƒbƒN,·,é,± ,Æ,É,æ,Á,Ä□A,Ü,½,ÍƒRƒ}ƒ“ƒh ƒvƒ□ƒ“ƒvƒg,©,çŽÀ□s,Å,«,Ü,·□B□Ú□×,É,Â,¢,Ä,Í□AWindows Scripting Host ,ÌƒhƒLƒ…ƒ□ƒ“ƒg,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÇ—□ƒIƒuƒWƒFƒNƒg,ðƒXƒNƒŠƒvƒg,Å,Ç,Ì,æ,¤,ÉŽg—p,·,é,©,ð—□‰ð,·,é,½,ß,É□A\Program Files\Mts\Samples\wsh ffƒBƒŒƒNƒgƒŠ,É, ,éƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ðŽÀ□s,µ,Ü,·□B,±,ÌffƒBƒŒƒNƒgƒŠ,É,Í□AVB Script ,Å‹L□q,³,ê,½ŽŸ,Ì 5 ,Â,ÌƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ªŠÜ,Ü,ê,Ä,¢,Ü,·□B

- InstDLL.vbs
- InstPak.vbs
- Uninst.vbs
- InstDllCLI.vbs
- InstPakCLI.vbs

,±,ê,ç,ÌƒXƒNƒŠƒvƒg,Í□ASample Bank ,ÌŠÇ—□□ì‹Æ,ðŽ©“®‰»,µ,Ü,·□B,½,Æ,¦,Î□AInstDLL.vbs ƒXƒNƒŠƒvƒg,Í□AƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŒÄ,Ñ□o,µ,ÄŽŸ,Ì□ì‹Æ,ð□s,¢,Ü,·□B,Ü,¸ Sample Bank ,ÌŠù‘¶,Ìƒo□[ƒWƒ‡ƒ“,ð□í□œ,µ□ASample Bank ,Æ,¢,¤–¼‘O,Ì□V,µ,¢ƒpƒbƒP□[ƒW,ð□ì□¬,µ,Ü,·□BŽŸ,É□ASample Bank ,Ì Visual Basic DLL□AVisual C++ DLL□A,¨,æ,Ñ Visual J++ DLL ,©,çƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□V,µ,¢ƒpƒbƒP□[ƒW,É‘g,Ý□ž,Ý□Aƒgƒ‰‰ƒ“ƒUƒNƒVƒ‡ƒ“‘®□«,ð•Ï□X,µ,Ü,·□BŽŸ,É□A□V,µ,¢ƒ□□[ƒ‹,ð’Ç‰Á,µ,Ü,·□BInstPak.vbs ƒXƒNƒŠƒvƒg,Í□AƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä Sample Bank ,ðƒCƒ“ƒXƒg□[ƒ‹,µ□AUninst.vbs ,Í□AMTS ,©,ç Sample Bank ƒpƒbƒP□[ƒW,ð□í□œ,·,é□ì‹Æ,ðŽ©“®‰»,µ,Ü,·□BƒXƒNƒŠƒvƒg,ðŽÀ□s,µ,½,ç□Aƒc□[ƒ‹fo□[,Ì [□Å□V,Ì□î•ñ,É□X□V] ƒ{ƒ^ƒ“,ðƒNƒŠƒbƒN,µ,½Œã□AMTS ƒGƒNƒXƒvƒ□□[ƒ‰ ,ÅƒXƒNƒŠƒvƒg,ÌŒ‹‰Ê,ðŠm”F,·,é,±,Æ,ª,Å,«,Ü,·□B

InstDLL.vbs ƒXƒNƒŠƒvƒg□AInstPak.vbs ƒXƒNƒŠƒvƒg□A,¨,æ,Ñ Uninst.vbs ƒXƒNƒŠƒvƒg,ðŽg,¤ ,É,Í□AƒXƒNƒŠƒvƒg“à,Ìƒtƒ@ƒCƒ‹,ÌƒpƒX,ªŽ©•ª,ÌƒRƒ“ƒsƒ…□[ƒ^□ã,Ì•K—v,Èƒtƒ@ƒCƒ‹,Ì□ê□Š,ðŽw,·,æ,¤ ,É•Ï□X,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B,½,Æ,¦,Î□AInstPak.vbs ƒXƒNƒŠƒvƒg“à,Ìƒtƒ@ƒCƒ‹,ÌƒpƒX,Í□AŽŸ,ÉŽ¦,· Sample Bank ƒpƒbƒP□[ƒW,ÌŠù’è,Ì□ê□Š,Å,·□B

```
path="C:¥Program Files¥Mts¥Samples¥Packages¥"
```

Sample Bank ƒpƒbƒP□[ƒW,ª•Ê,Ì□ê□Š,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,é□ê□‡,Í□AƒXƒNƒŠƒvƒg“à,ÌŠY“–,·,é□s,ð•Ï□X,µ,ÄŒ»□Ý,ÌƒpƒbƒP□[ƒW,Ì□ê□Š,ðŽw,·,æ,¤,É,µ,Ü,·□B

InstDllCLI.vbs ƒXƒNƒŠƒvƒg,Æ InstPakCLI.vbs ƒXƒNƒŠƒvƒg,Í□AƒRƒ}ƒ“ƒhƒ‰ƒCƒ“ ƒpƒ‰ƒ□[ƒ^,ðƒTƒ|□[ƒg,µ,Ä,¢,é,Ì,Å□AƒRƒ}ƒ“ƒh ƒvƒ□ƒ“ƒvƒg,ÅƒXƒNƒŠƒvƒg,ðŽÀ□s,·,é,Æ,«,Éƒtƒ@ƒCƒ‹,Ì□ê□Š,ðƒpƒ‰ƒ□[ƒ^,Æ,µ,Ä“ü—Í,·,é,± ,Æ,ª,Å,«,Ü,·□BƒRƒ}ƒ“ƒh ƒvƒ□ƒ“ƒvƒg,Å□AŽg—p,·,éƒXƒNƒŠƒvƒg,ªŠÜ,Ü,ê,Ä,¢ ,éƒffƒBƒŒƒNƒgƒŠ,ÉˆÚ®,µ□AƒXƒNƒŠƒvƒg–¼,ÆƒpƒbƒP□[ƒW,Ü,½,Í DLL ,Ì□ê□Š,ð“ü—Í,µ,Ü,·□B,½,Æ,¦,Î□AŠù’è,ÌffƒBƒŒƒNƒgƒŠ,É, ,é Sample Bank ƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,·,é,É,Í□AƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ªŠÜ,Ü,ê,Ä,¢,éffƒBƒŒƒNƒgƒŠ,ÉˆÚ®,µ□AƒRƒ}ƒ“ƒh ƒvƒ□ƒ“ƒvƒg,ÅŽŸ,ÌƒRƒ}ƒ“ƒh,ðŽg—p,µ,Ü,·□B

```
InstPakCLI.vbs "C:¥Program Files¥Mts¥Samples¥Packages¥"
```

InstDLL.vbs ƒXƒNƒŠƒvƒg,ðŽg,¤,É,Í□A,Ü,¸□AŽŸ,Ì‘€□ì,ðŽÀ□s,µ,Ä Vjacct.dll ƒtƒ@ƒCƒ‹,É Java ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð“o˜^,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□B

- Microsoft® Visual Studio® 97 ,Ì ActiveX ƒEƒBƒU□[ƒh,ðŽg,Á,Ä□AJava ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð“o˜^,·,é□B

ŠÖ˜A□€–Ú

ŠÇ—□ƒIƒuƒWƒFƒNƒg ƒTƒ“ƒvƒ‹ ƒXƒNƒŠƒvƒg,ÌƒZƒbƒgƒAƒbƒv

MTS ŠÇ—□Ž©“®‰» Visual Basic ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“

Visual Basic Version 5.0 ƒTƒ“ƒvƒ‹ ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í□A**Catalog** ƒIƒuƒWƒFƒNƒg□A**CatalogObject** ƒIƒuƒWƒFƒNƒg□A,¨,æ,Ñ **CatalogCollection** ƒIƒuƒWƒFƒNƒg□ã,Åƒ□ƒ\ƒbƒh,ðŽg,¤,±,Æ,É,æ,è□A"Scriptable Admin Demo" ,Æ,¢,¤–¼‘O,ÌƒpƒbƒP□[ƒW,ÌŠî–{“I,ÈŠÇ—□‹@”\,ðŽ©“®‰»,·,é•û–@,ðŽ¦,µ,Ü,·□B

’□ □ì□¬,·,é Visual Basic Version 5.0 ,Ìƒvƒ□ƒWƒFƒNƒg,ª MTS ŠÇ—□ƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ (MTSAdmin ƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ) ,ðŽQ□Æ,·,é,æ,¤,É□\□¬,µ,È,¯,ê,Î,È,è,Ü,¹,ñ□BMTSAdmin ƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,ðŽQ□Æ,·,é,É,Í□AVisual Basic ,Ì [ƒvƒ□ƒWƒFƒNƒg] ƒ□ƒjƒ…□[,Ì [ŽQ□Æ□Ý’è] ,ðƒNƒŠƒbƒN,µ,Ü,·□BŽŸ,É□A—˜—p,Å,«,éŽQ□Æƒtƒ@ƒCƒ‹,Ì’†,Ì "MTS 2.0 Admin Type Library" ,ð’T,µ,Ü,·□BŽÀ□sŽžƒofƒCƒ“ƒfƒBƒ“ƒO•Ï□” (ƒvƒ□ƒOƒ‰ƒ€,ÌŽÀ□sŽž,É□¶,¶,éƒofƒCƒ“ƒfƒBƒ“ƒO) ,É‘Î‰ž,·,éƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,Í□AŽg—p,µ,Ä,¢,éƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^,É MTXADMIN.DLL ƒtƒ@ƒCƒ‹,ªƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,ê,Î□A“Á,É□\□¬□ì‹Æ,ð□s,í,È,,Ä,à Visual Basic ,ª’T,µ,Ü,·□B

□ "Scriptable Admin Demo" ,Æ,¢,¤–¼‘O,ÌŠù¶,ÌƒpƒbƒP□[ƒW,ð,·,×,Ä□í□œ,·,é,É,Í

1 **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,ÄƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð¶□¬,µ,Ü,·□B

```
Dim catalog As Object
Set catalog = CreateObject("MTSAdmin.Catalog.1")
```

2 **GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Packages** ƒRƒŒƒNƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg,ðŽæ“¾,µ,Ü,·□B**Packages** ƒRƒŒƒNƒVƒ‡ƒ“,Í□AƒIƒ^ƒ□ƒO,©,çƒf□[ƒ^,ðŽæ“¾,¹,¸,É–ß,é,Ì,Å□A**GetCollection** ƒ□ƒ\ƒbƒh,©,ç–ß,Á,½,Æ,«□AƒRƒŒƒNƒVƒ‡ƒ“,Í‹ó,É,È,Á,Ä,¢,Ü,·□B

```
Dim packages As Object
Set packages = catalog.GetCollection("Packages")
```

3 ,·,×,Ä,ÌƒpƒbƒP□[ƒW,ð“Ç,Ý□ž,ñ,Å **Packages** ƒRƒŒƒNƒVƒ‡ƒ“,É‘}“ü,µ□A"Scriptable Admin Demo" ,ðŒŸ□õ,·,é,±,Æ,É,æ,Á,Ä□A‘O,Ìƒo□[ƒWƒ‡ƒ“,Ì "Scriptable Admin Demo" ƒpƒbƒP□[ƒW,ð’T,µ,Ü,·□B□Å‘å,ÌƒCƒ“ƒfƒbƒNƒX,©,çŠJŽn,µ,ÄƒRƒŒƒNƒVƒ‡ƒ“‘à,ð—ñ‹“,µ□Aƒ‹□[ƒv,Ì’†,©,ç **Remove** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,·,±,Æ,ª,Å,«,é,æ,¤,É,µ,Ü,·□B**Remove** ƒ□ƒ\ƒbƒh,Í□AƒIƒuƒWƒFƒNƒg,ð‰ð•ú,µ□AƒRƒŒƒNƒVƒ‡ƒ“,©,çƒIƒuƒWƒFƒNƒg,ð□í□œ,µ,Ü,·□B,Ü,½□A□í□œ,³,ê,éƒCƒ“ƒfƒbƒNƒXˆÈ□ã,Ì,·,×,Ä,Ì n ,É‘Î‰ž,·,éƒIƒuƒWƒFƒNƒg (n+1) ,ª ƒIƒuƒWƒFƒNƒg (n) ,É,È,é,æ,¤ ,É□AƒRƒŒƒNƒVƒ‡ƒ“‘à,ÌƒIƒuƒWƒFƒNƒg,ðƒVƒtƒg,µ,Ü,·□B**Remove** ƒ□ƒ\ƒbƒh,ªƒRƒŒƒNƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg,É‹y,Ú,·‰e‹¿,Í,·,®,É”½‰f,³,ê,Ü,·□B**Remove** ƒ□ƒ\ƒbƒh,ªŒÄ,Ñ□o,³,ê,½Œã□A**Item** ƒ□ƒ\ƒbƒh,â **Count** ƒ□ƒ\ƒbƒh,Í□A,Ç,ÌŽž“_,ÅŒÄ,Ñ□o,³,ê,Ä,à□AƒCƒ“ƒfƒbƒNƒX,Ì•Ï□X,ª”½‰f,³,ê,Ü,·□B,½,¾,µ□AƒpƒbƒP□[ƒW,Ì□í□œ,Í□A**SaveChanges** ƒ□ƒ\ƒbƒh,ªŒÄ,Ñ□o,³,ê,é,Ü,Å□AƒIƒ^ƒ□ƒO,É,Í“K—p,³,ê,Ü,¹,ñ (Žè□‡ 4 ,ðŽQ□Æ)□B

```
packages.Populate
n = packages.Count
For i = n - 1 To 0 Step -1
    If packages.Item(i).Value("Name") = "Scriptable Admin Demo" Then
    packages.Remove (i)
    End If
Next
```

4 **SaveChanges** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä•Ï□X,ðƒf□[ƒ^Ši”[□ê□Š,É•Ù‘¶,µ,Ü,·□B

```
packages.SaveChanges
```

□ "Scriptable Admin Demo" ,Æ,¢,¤–¼‘O,Ì□V,µ,¢ƒpƒbƒP□[ƒW,ð□ì□¬,·,é,É,Í

1 **Add** ƒ□ƒ\ƒbƒh,ðŽg,Á,Ä□V,µ,¢ƒpƒbƒP□[ƒW,ð’Ç‰Á,µ□AŠ„,è“–,Ä,ç,ê,½ƒpƒbƒP□[ƒW ID ,ð‹L˜^,µ,Ü,·□B**Add** ƒ□ƒ\ƒbƒh,Í□AƒRƒŒƒNƒVƒ‡ƒ“,ÉƒIƒuƒWƒFƒNƒg,ð’Ç‰Á,µ,Ü,·,ª□A**SaveChanges** ƒ□ƒ\ƒbƒh,ªŒÄ,Ñ□o,³,ê,é,Ü,Å□A•Ï□X,ÍƒIƒ^ƒ□ƒO,É“K—p,³,ê,Ü,¹,ñ (Žè□‡ 3 ,ðŽQ□Æ)□B**Add** ƒ□ƒ\ƒbƒh,Å,Í□A,·,×,Ä,ÌƒvƒOƒfƒeB,ÉŠù’è‘l,ª“K—p,³,ê,é,±,Æ,É’□ˆÓ,µ,Ä,,¾,³,¢□BŠù’è‘l,Ì ID ,Í□A□V,µ,¢□ˆê□Ó,Ì ID ,É,È,è,Ü,·□B

```
Dim newPack As Object
Dim newPackID As Variant
Set newPack = packages.Add
newPackID = newPack.Value("ID")
```

2 **Name** ƒvƒ□ƒpƒeƒB‚Æ **SecurityEnabled** ƒvƒ□ƒpƒeƒB‚ð□X□V‚µ‚Ü‚·□B

```
newPack.Value("Name") = "Scriptable Admin Demo"
newPack.Value("SecurityEnabled") = "N"
```

3 **SaveChanges** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚Ä□V‚µ‚¢ƒpƒbƒP□[ƒW‚ðƒJƒ^ƒ□ƒO‚É•Û‘¶‚µ‚Ü‚·□B±‚ÌŒÄ‚Ñ□o‚µ‚Íß‚è'l‚Í□A•Ï□X□A'Ç‰Á□A‚Ü‚½‚Í□í□œ‚³‚ê‚½ƒIƒuƒWƒFƒNƒg‚Ì□”‚Å‚·□B•Ï□X‚ª‚È‚¢□ê□‡□Aƒ□ƒ\ƒbƒh‚Í 0 ‚ð•Ô‚µ‚Ü‚·□B

```
n = packages.SaveChanges
```

"Scriptable Admin Demo" ‚Æ‚¢‚¤¼‘O‚ÌƒpƒbƒP□[ƒW‚Ìƒvƒ□ƒpƒeƒB‚ð□X□V‚µ□AComponentsInPackage ƒRƒŒƒNƒVƒ‡ƒ“‚ðŽæ“¾‚·‚é‚É‚Í

1 **PopulateByKey** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚ÄƒJƒ^ƒ□ƒO‚©‚çƒpƒbƒP□[ƒW‚ð"Ç‚Ý□ž‚Ý‚Ü‚·□B"Ç‚Ý□ž‚ÞƒpƒbƒP□[ƒW‚ÌƒfƒNƒg‚ÌƒL□[‚ðŠÜ‚Þ"z—ñ‚ð"n‚µ‚Ü‚·□BƒTƒ"ƒvƒ‹ ƒR□[ƒh‚Å‚Í□A1 ‚Â‚Ì—v'f (□ì□¬‚µ‚½‚Î‚©‚è‚ÌƒpƒbƒP□[ƒW‚Ì ID) ‚ðŠÜ‚Þ"z—ñ‚ðŽg‚Á‚Ä‚¢‚Ü‚·□B

```
Dim keys(0) as Variant
keys(0) = newPackId
packages.PopulateByKey keys
```

2 ƒRƒŒƒNƒVƒ‡ƒ“‚©‚çƒpƒbƒP□[ƒW ƒIƒuƒWƒFƒNƒg‚ðŽæ“¾‚µ‚Ü‚·□B

```
Dim package As Object
Set package = packages.Item(0)
```

3 ƒpƒbƒP□[ƒW‚Ì **SecurityEnabled** ƒvƒ□ƒpƒeƒB‚ð□X□V‚µ‚Ü‚·□B

```
package.Value("SecurityEnabled") = "Y"
```

4 **GetCollection** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚Ä **ComponentsInPackage** ƒRƒŒƒNƒVƒ‡ƒ“‚ðŽæ“¾‚µ‚Ü‚·□B"Scriptable Admin Demo" ƒpƒbƒP□[ƒW‚ÌƒL□[‚ðƒpƒ‰‰ƒ□□[ƒ^‚Æ‚µ‚Ä"n‚µ‚Ü‚·□B

```
Dim components As Object
Set components = packages.GetCollection("ComponentsInPackage", package.Key)
```

5 **SaveChanges** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚Ä•Ï□X‚ðƒJƒ^ƒ□ƒO‚É•Û‘¶‚µ‚Ü‚·□B

```
packages.SaveChanges
```

ƒRƒ"ƒ|□[ƒlƒ"ƒg‚ð "Scriptable Admin Demo" ƒpƒbƒP□[ƒW‚É'g‚Ý□ž‚Þ‚É‚Í

1 **GetUtilInterface** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚ÄƒRƒ"ƒ|□[ƒlƒ"ƒg ƒt□[ƒeƒBƒŠƒeƒB ƒIƒuƒWƒFƒNƒg‚ðŽæ“¾‚µ‚Ü‚·□B±‚ÌƒIƒuƒWƒFƒNƒg‚ÍƒRƒ"ƒ|□[ƒlƒ"ƒg‚Ì'g‚Ý□ž‚Ý‚ÉŽg‚¢‚Ü‚·□B

```
Dim util As Object
Set util = components.GetUtilInterface
On Error GoTo installFailed
```

2 '‚g‚Ý□ž‚ÞƒRƒ"ƒ|□[ƒlƒ"ƒg‚Ì‚ƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ"ƒN ƒ‰‰ƒCƒuƒ‰ƒŠ (DLL) ‚Ì–¼'O‚ðŠÜ‚Þ¶Žš—ñ‚ð"n‚µ‚Ä **InstallComponent** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚Ü‚·□BƒRƒ"ƒ|□[ƒlƒ"ƒg‚ÉŠÖ•"ƒ^ƒCƒv ‰ƒCƒuƒ‰ƒŠ‚Ü‚½‚Íƒvƒ□ƒfƒ‹ƒV/ƒXƒ^ƒu DLL ‚ª‚È‚¢□ê□‡‚Í□A‹ó‚Ì•¶Žš—ñ‚ð'æ 2 ˆø□"‚Æ'æ 3 ˆø□"‚Æ‚µ‚Ä"n‚µ‚Ü‚·□B□V‚µ‚¢ƒRƒ"ƒ|□[ƒlƒ"ƒg‚ð'g‚Ý□ž‚ñ‚¾Œã‚É **SaveChanges** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚·•K—v‚Í‚è‚Ü‚¹‚ñ□BDLL ‚ÉŠÜ‚Ü‚ê‚é‚·‚×‚Ä‚ÌƒRƒ"ƒ|□[ƒlƒ"ƒg‚Í□A‚±‚ÌƒIƒuƒWƒFƒNƒg‚Ì **GetCLSIDs** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚Ä'g‚Ý□ž‚Ü‚ê‚é‚ÍƒRƒ"ƒ|□[ƒlƒ"ƒg‚ÌƒX ID (CLSID) ‚ðŽæ“¾‚µ‚Ü‚·□B

```
Form2.Show 1
```

```
    Dim thePath As String
    thePath = Form2.MTSPath + "¥samples¥packages¥vbacct.dll"
    util.InstallComponent thePath, "", ""
    Dim installedCLSIDs() as Variant
    util.GetCLSIDs thePath, "", installedCLSIDs
    On Error GoTo 0
```

3 **PopulateByKey** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ□A‘g,Ý□ž,Ü,ê,½,Î,©,è,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ð“Ç,Ý□ž,Ý,Ü,·□B**InstallComponent** ƒ□ƒ\ƒbƒh,ðŽg,Á,ÄƒpƒbƒP□[ƒW,É‘g,Ý□ž,ñ,¾ƒRƒ“ƒ|□[ƒlƒ“ƒg,Í□A**Populate** ƒ□ƒ\ƒbƒh,Ü,½,Í **PopulateByKey** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,ÄƒJƒ^□ƒO,©,çƒf□[ƒ^,ð“Ç,Ý□ž,Þ,Ü,Å□AƒRƒŒƒNƒVƒ‡ƒ“‘à,Å,Í•\Ž¦,³,ê,Ü,¹,ñ□B

```
    components.PopulateByKey installedCLSIDs
```

"Scriptable Admin Demo" ƒpƒbƒP□[ƒW,©,ç Bank.CreateTable ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð’T,µ,Ä□í□œ,·,é,É,Í

1 **Item** ƒ□ƒ\ƒbƒh,Æ **Count** ƒ□ƒ\ƒbƒh,ðŽg,Á,Ä□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉŒJ,è•Ô,µ□ˆ—□,ð□s,¢□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘®□«,ð•Ï□X,µ,Ü,·□B

```
    Dim component As Object
    n = components.Count
    For i = n - 1 To 0 Step -1
        Set component = components.Item(i)
        component.Value("Transaction") = "Required"
```

2 ƒCƒ“ƒfƒbƒNƒX,ðŽg,Á,Ä `Bank.CreateTable` ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð’T,µ□A□í□œ,µ,Ü,·□Bƒ‹□[ƒv,Ì’†,Å **Remove** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,·,É,Í□AƒRƒŒƒNƒVƒ‡ƒ“‘à,ð‹t•ûŒü,ÉŒJ,è•Ô,µ□ˆ—□,ð□s,í,È,¯,ê,Î,È,ç,È,¢,±,Æ,É’□ˆÓ,µ,Ä,,¾,³,¢□B

```
        If component.Value("ProgID") = "Bank.CreateTable" Then
            components.Remove (i)
        End If
    Next
```

3 □V,µ,¢‘□□”,ðŽæ“¾,µ□AƒRƒŒƒNƒVƒ‡ƒ“‘à,Å,à,¤ˆê“xŒJ,è•Ô,µ□ˆ—□,ð□s,¢,Ü,·□B**SaveChanges** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,·,Ü,Å□A`Bank.CreateTable` ƒRƒ“ƒ|□[ƒlƒ“ƒg,Íƒf□[ƒ^Ši”[□ê□Š,©,ç□í□œ,³,ê,Ü,¹,ñ□B‘g,Ý□ž,Ý,ª¬Œ÷,µ,½□ê□‡,Í□Aƒt□[ƒU□[,É’m,ç,¹,éƒ□ƒbƒZ□[ƒW,ð•\Ž¦,µ,Ü,·□B

```
    n = components.Count
    For i = 0 To n - 1
        Set component = components.Item(i)
        Debug.Print component.Value("ProgID")
        Debug.Print component.Value("DLL")
    Next

    n = components.SaveChanges
    MsgBox "Scriptable Admin Demo package installed and configured."
    Exit Sub

installFailed:
    MsgBox "Error code " + Str$(Err.Number) + " installing " + thePath + "  Make
sure the MTS path you entered is correct and that vbacct.dll is not already
installed."
End Sub
```

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ‚É‚æ‚é MTS ŠÇ—□‚ÌŽ©“®‰»□AVisual Basic ‚É‚æ‚é□‚“x‚È MTS ŠÇ—□‚ÌŽ©“®‰»

Visual Basic ,É,æ,é MTS ŠÇ—□,ÌŽ©“®‰»

ŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,Í□AƒpƒbƒP□[ƒW,ÆƒRƒ“ƒ|□[ƒIƒ“ƒg,ÌƒCƒ“ƒXƒg□[ƒ‹□A□í□œ□A,¨,æ,Ñƒvƒ□ƒpƒeƒB,Ì□X□V,ÉŽg—p,Å,«,Ü,·□Bˆ‰º,Ìƒgƒsƒbƒ‰N,Å,Í□AŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg□ã,Åƒ□ƒ\ƒbƒh,ðŽg,¤,±,Æ,É,æ,Á,Ä□AŽŸ,ÌŠÇ—□□ì‹Æ,ðŽ©“®‰»,·,é,½,ß,ÌŽè□‡,Æ Microsoft® Visual Basic® Version 5.0 ,ÌƒTƒ“ƒvƒ‹ ƒR□[ƒh,É,Â,¢,Ä□à–¾,µ,Ü,·□B

- Šù□¬,Ì MTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ðŽ©“®‰»,·,é
- □V,µ,¢ MTS ƒpƒbƒP□[ƒW,Ì□ì□¬,ÆƒRƒ“ƒ|□[ƒIƒ“ƒg,Ì‘g,Ý□ž,Ý,ðŽ©“®‰»,·,é
- ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ MTS ƒpƒbƒP□[ƒW,ð—ñ‹“,µ□Aƒvƒ□ƒpƒeƒB,ð□X□V,·,é□ì‹Æ,ðŽ©“®‰»,·,é
- ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ MTS ƒpƒbƒP□[ƒW,ð—ñ‹“,µ□AƒpƒbƒP□[ƒW,ð□í□œ,·,é□ì‹Æ,ðŽ©“®‰»,·,é
- ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ MTS ƒRƒ“ƒ|□[ƒIƒ“ƒg,ð—ñ‹“,µ□AƒRƒ“ƒ|□[ƒIƒ“ƒg,ð□í□œ,·,é□ì‹Æ,ðŽ©“®‰»,·,é

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ,É,æ,é□,“x,È MTS ŠÇ—□,ÌŽ©“®‰»

Šù□¬,Ì MTS ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ðŽ©“®‰»,·,é

"Test.pak" ,Æ,¢,¤‐¼'O,ÌŠù□¬,ÌƒpƒbƒP□[ƒW,ð MTS ƒGƒNƒXƒvƒ□□[ƒ‰,ÉƒCƒ“ƒXƒg□[ƒ‹,·,é,É,Í

1 Šù□¬,ÌƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ÉŽg,¤ƒIƒuƒWƒFƒNƒg,ð□éŒ¾,µ,Ü,·□B

```
Private Sub InstallPackage_Click()
    Dim catalog As Object
    Dim packages As Object
    Dim util As Object
```

2 ƒ□ƒ\ƒbƒh,ªŽ¸”s,Ì HRESULT ,ð•Ô,µ,½□ê□‡,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,ðŽg,Á,Äƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰□[,ð□ˆ—□,µ,Ü,·□B**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,Æ **Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ,ªƒgƒ‰□fbƒv,Å,«,éƒGƒ‰□[,ðƒeƒXƒg,µ□AƒGƒ‰□[,É‰ž“š,·,é,±,Æ,ª,Å,«,Ü,·□B

```
    On Error GoTo failed
```

3 **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Catalog** ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä□Å□ãˆÊƒŒƒxƒ‹,Ì **Packages** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□B

```
    Set catalog = CreateObject("MTSAdmin.Catalog.1")
    Set packages = catalog.GetCollection("Packages")
```

4 **PackageUtil** ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ□A**InstallPackage** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä "test.pak" ,Æ,¢,¤‐¼'O,ÌƒpƒbƒP□[ƒW,ðƒCƒ“ƒXƒg□[ƒ‹,µ,Ü,·□B

```
    Set util = packages.GetUtilInterface
    util.InstallPackage "c:¥test.pak", "", 0
    Exit Sub
```

5 ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ªŽ¸”s,µ,½□ê□‡,Í□A**Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,ÄƒGƒ‰□[ƒ□ƒbƒZ□[ƒW,ð•\Ž¦,µ,Ü,·□B

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ,É,æ,é□,“x,È MTS ŠÇ—□,ÌŽ©“®‰»

□V,µ,¢ MTS ƒpƒbƒP□[ƒW,Ì□ì□¬,ÆƒRƒ"ƒ|□[ƒlƒ"ƒg,Ì'g,Ý□ž,Ý,ðŽ©"®‰»,·,é

"My Package" ,Æ,¢,¤-¼'O,Ì□V,µ,¢ƒpƒbƒP□[ƒW,ð□ì□¬,µ□A,»,Ìƒpƒbƒ P□[ƒW,ÉƒRƒ"ƒ|□[ƒlƒ"ƒg,ð'g,Ý□ž,Þ,Ê,Í

1 □V,µ,¢ƒpƒbƒP□[ƒW,ÌƒCƒ"ƒXƒg□[ƒ‹,Æ,»,Ìƒpƒbƒ P□[ƒW,Ö,ÌƒRƒ"ƒ|□[ƒlƒ"ƒg,Ì'g,Ý□ž,Ý,ÉŽg,¤ƒlƒuƒWƒFƒNƒg,ð□éŒ¾,µ,Ü,·□B

```
Dim catalog As Object
Dim packages As Object
Dim newPack As Object
Dim componentsInNewPack As Object
Dim util As Object
```

2 ƒ□ƒ\ƒbƒh,ªŽ¸"s,Ì HRESULT ,ð•Ô,µ,½□ê□‡,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ"ƒg,ðŽg,Á,Äƒ‰ƒ"ƒ^ƒCƒ€ ƒGƒ ‰□[,ð□ˆ—□,µ,Ü,·□B**On Error** ƒXƒe□[ƒgƒ□ƒ"ƒg,Æ **Err** ƒlƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ,ªƒgƒ‰ƒbƒv,Å,«,éƒGƒ ‰□[,ðƒeƒXƒg,µ□AƒGƒ‰□[,É‰ž"š,·,é,±,Æ,ª,Å,«,Ü,·□B

```
On Error GoTo failed
```

3 **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Catalog** ƒlƒuƒWƒFƒNƒg,ÌƒCƒ"ƒXƒ^ƒ"ƒX,ð□¶□¬,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä□Å□ãˆÊƒŒƒxƒ‹,Ì **Packages** ƒRƒŒƒNƒVƒ‡ƒ",ðŽæ"¾,µ,Ü,·□BŽŸ,É□A**Add** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä□V,µ,¢ƒpƒbƒP□[ƒW,ð'Ç ‰□Á,µ,Ü,·□B

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set packages = catalog.GetCollection("Packages")
Set newPack = packages.Add
Dim newPackID As String
```

4 ƒpƒbƒP□[ƒW–¼,ð "My Package" ,É□Ý'è,µ□A•Ï□X,ð **Packages** ƒRƒŒƒNƒVƒ‡ƒ",É•Û'¶,µ,Ü,·□B

```
newPackID = newPack.Key
newPack.Value("Name") = "My Package"
packages.SaveChanges
```

5 **GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **ComponentsInPackage** ƒRƒŒƒNƒVƒ‡ƒ",ÉƒAƒNƒZƒX,µ,Ü,·□BŽŸ,É□A**ComponentUtil** ƒlƒuƒWƒFƒNƒg,ÌƒCƒ"ƒXƒ^ƒ"ƒX,ð□¶□¬,µ□A**InstallComponent** ƒ□ƒ\ ƒbƒh,ðŒÄ,Ñ□o,µ,Ä□V,µ,¢ƒpƒbƒP□[ƒW,ÉƒRƒ"ƒ|□[ƒlƒ"ƒg,ð'g,Ý□ž,Ý,Ü,·□B

```
    Set componentsInNewPack = packages.GetCollection("ComponentsInPackage",
newPackID)
    Set util = componentsInNewPack.GetUtilInterface
    util.InstallComponent"d:¥dllfilepath", "", ""
    Exit Sub
```

6 ƒpƒbƒP□[ƒW,ÌƒCƒ"ƒXƒg□[ƒ‹,ªŽ¸"s,µ,½□ê□‡,Í□A**Err** ƒlƒuƒWƒFƒNƒg,ðŽg,Á,ÄƒGƒ‰□[ƒ□ƒbƒZ□[ƒW,ð•\ Ž¦,µ,Ü,·□B

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ,É,æ,é□,"x,È MTS ŠÇ—□,ÌŽ©"®‰»

ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ MTS ƒpƒbƒP□[ƒW,ð—ñ‹“,µ□Aƒvƒ□ƒpƒeƒB,ð□X□V,·,é□ì‹Æ,ðŽ©“®‰»,·,é

ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW,ð—ñ‹“,µ□A"My Package" ,Æ,¢,¤-¼'O,ÌƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,ð□X□V,·,é,É,Í

1 ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW,ð—ñ‹“,µ□Aƒpƒbƒ P□[ƒW,Ìƒvƒ□ƒpƒeƒB,ð□X□V,·,é,Æ,«,ÉŽg,¤ƒIƒuƒWƒFƒNƒg,ð□éŒ¾,µ,Ü,·□B

```
Private Sub BrowseUpdate_Click()
    Dim catalog As Object
    Dim packages As Object
    Dim pack As Object
```

2 ƒ□ƒ\ƒbƒh,ªŽ¸”s,Ì HRESULT ,ð•Ô,µ,½□ê□‡,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,ðŽg,Á,Äƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰ƒ□[,ð□ˆ—□,µ,Ü,·□B**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,Æ **Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ,ªƒgƒ‰ƒbƒv,Å,«,éƒGƒ‰ƒ□[,ðƒefXƒg,µ□AƒGƒ‰ƒ□[,É‰ž“š,·,é,±,Æ,ª,Å,«,Ü,·□B

```
    On Error GoTo failed
```

3 **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Catalog** ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Packages** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□BŽŸ,É□A**Populate** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,ÄƒIƒ^ƒ□ƒO,ÌƒpƒbƒP□[ƒW,ðƒRƒŒƒNƒVƒ‡ƒ“,É'}“ü,µ,Ü,·□B

```
    Set catalog = CreateObject("MTSAdmin.Catalog.1")
    Set packages = catalog.GetCollection("Packages")
    packages.Populate
```

4 ƒRƒŒƒNƒVƒ‡ƒ““à,ð—ñ‹“,µ□A"My Package" ,Æ,¢,¤-¼'O,ÌƒpƒbƒP□[ƒW,ð'T,µ,Ü,·□B"My Package" ,ªŒ©,Â,©,Á,½,ç□A**SecurityEnabled** ƒvƒ□ƒpƒeƒB,ð "Y" ,É□Ý'è,µ,Ü,·□B**SaveChanges** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,ÄƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒeƒB,Ì□X□V,ð•Û'¶,µ,Ü,·□B

```
    For Each pack In packages
        If pack.Name = "My Package" Then
            pack.Value("SecurityEnabled") = "Y"
            Exit For
        End If
    Next
    packages.SaveChanges
    Exit Sub
```

5 ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ªŽ¸”s,µ,½□ê□‡,Í□A**Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,ÄƒGƒ‰ƒ□[ƒ□ƒbƒZ□[ƒW,ð•\Ž¦,µ,Ü,·□B

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A□€-Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ,É,æ,é□,“x,È MTS ŠÇ—□,ÌŽ©“®‰»

ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ MTS ƒpƒbƒP□[ƒW,ð—ñ‹“,µ□AƒpƒbƒP□[ƒW,ð□í□œ,·,é□ì‹Æ,ðŽ©“®‰»,·,é

ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW,ð—ñ‹“,µ□A"My Package" ,Æ,¢,¤-¼'O,ÌƒpƒbƒP□[ƒW,ð□í□œ,·,é,É,Í

1. ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW,ð—ñ‹“,µ□A‘Á'è,ÌƒpƒbƒP□[ƒW,ð□í□œ,·,é,Æ,«,ÉŽg,¤ƒIƒuƒWƒFƒNƒg,ð□éŒ¾,µ,Ü,·□B

```
Dim catalog As Object
Dim packages As Object
Dim pack As Object
```

2. ƒ□ƒ\ƒbƒh,ªŽ¸”s,Ì HRESULT ,ð•Ô,µ,½□ê□‡,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,ðŽg,Á,Äƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰ƒ□[,ð□ˆ—□,µ,Ü,·□B**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,Æ **Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ,ªƒgƒ‰ƒbƒv,Å,«,éƒGƒ‰ƒ□[,ðƒeƒXƒg,µ□AƒGƒ‰ƒ□[,É‰ž“š,·,é,±,Æ,ª,Å,«,Ü,·□B

```
On Error GoTo failed
```

3. **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Catalog** ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Packages** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□BŽŸ,É□A**Populate** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ□Aƒ‰ƒ^□ƒ□ƒO,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,ðƒRƒŒƒNƒVƒ‡ƒ“,É'}“ü,µ,Ü,·□B

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set packages = catalog.GetCollection("Packages")
packages.Populate
```

4. **Count** ƒ□ƒ\ƒbƒh,Æ **Item** ƒ□ƒ\ƒbƒh,ðŽg,Á,ÄƒpƒbƒP□[ƒW ƒRƒŒƒNƒVƒ‡ƒ“"à,ð—ñ‹“,µ□A"My Package" ,Æ,¢,¤-¼'O,ÌƒpƒbƒP□[ƒW,ð'T,µ,Ü,·□B"My Package" ,ªŒ©,Â,©,Á,½,ç□A**Remove** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,ÄƒpƒbƒP□[ƒW,ð□í□œ,µ,Ü,·□BŽŸ,É□A**SaveChanges** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä•Ï□X,ðƒRƒŒƒNƒVƒ‡ƒ“,É• Û'¶,µ,Ü,·□B

```
For i = 0 To packages.Count-1
    Set pack = packages.Item(i)
    If pack.Name = "My Package" Then
        packages.Remove (i)
        packages.savechanges
        Exit For
    End If
Next
Exit Sub
```

5. ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ªŽ¸”s,µ,½□ê□‡,Í□A**Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,ÄƒGƒ‰ƒ□[ƒ□ƒbƒZ□[ƒW,ð•\Ž¦,µ,Ü,·□B

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A□€-Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ,É,æ,é□,“x,È MTS ŠÇ—□,ÌŽ©“®‰»

ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ MTS ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð—ñ‹“,µ□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é□ì‹Æ,ðŽ©“®‰»,·,é

ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð—ñ‹“,µ□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é,É,Í

1 ƒCƒ“ƒXƒg□[ƒ‹,³,ê,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð—ñ‹“,µ□A“Á’è,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,·,é,Æ,«,ÉŽg,¤ƒIƒuƒWƒFƒNƒg,ð□éŒ¾,µ,Ü,·□B

```
Dim catalog As Object
Dim packages As Object
Dim pack As Object
Dim componentsInPack As Object
Dim comp As Object
```

2 ƒ□ƒ\ƒbƒh,ªŽ¸”s,Ì HRESULT ,ð•Ô,µ,½□ê□‡,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,ðŽg,Á,Äƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰□[,ð□^—□,µ,Ü,·□B**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,Æ **Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ,ªƒgƒ‰ƒbƒv,Å,«,éƒGƒ‰□[,ðƒefXƒg,µ□AƒGƒ‰□[,É‰ž“š,·,é,±,Æ,ª,Å,«,Ü,·□B

```
On Error GoTo failed
```

3 **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Catalog** ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Packages** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□BŽŸ,É□A**Populate** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Äƒ‰ƒ^ƒ□ƒO,ÉƒCƒ“ƒXƒg□[ƒ‹,³,ê,Ä,¢,éƒpƒbƒP□[ƒW,ðƒRƒŒƒNƒVƒ‡ƒ“,É‘}“ü,µ,Ü,·□B

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set packages = catalog.GetCollection("Packages")
packages.Populate
```

4 ƒRƒŒƒNƒVƒ‡ƒ““à,ð—ñ‹“,µ□A"My Package" ,Æ,¢,¤–¼‘O,ÌƒpƒbƒP□[ƒW,ð’T,µ,Ü,·□BŽŸ,É□A**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **ComponentsInPackage** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□B**Populate** ƒ□ƒ\ƒbƒh,ðŽg,Á,Ä **ComponentsInPackage** ƒRƒŒƒNƒVƒ‡ƒ“,ÉƒRƒ“ƒ|□[ƒlƒ“ƒg,ð‘}“ü,µ□AƒRƒŒƒNƒVƒ‡ƒ““à,ð—ñ‹“,µ□A"Bank.Account" ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð’T,µ,Ü,·□B**Remove** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,ÄƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□í□œ,µ□A**SaveChanges** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä•Ï□X,ðƒRƒŒƒNƒVƒ‡ƒ“,É•Û‘¶,µ,Ü,·□B

```
    For Each pack In packages
        If pack.Name = "My Package" Then
            Set componentsInPack = packages.GetCollection("ComponentsInPackage",
pack.Key)
            componentsInPack.Populate
            For i = 0 To componentsInPack.Count
                Set comp = componentsInPack.Item(i)
                If comp.Name = "Bank.Account" Then
                    componentsInPack.Remove (i)
                    componentsInPack.savechanges
                    Exit For
                End If
            Next
            Exit For
        End If
    Next
    Exit Sub
```

5 ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ªŽ¸”s,µ,½□ê□‡,Í□A**Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Äƒ‰ƒGƒ‰□[ƒ□ƒbƒZ□[ƒW,ð•\Ž¦,µ,Ü,·□B

```
failed:
```

```
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A€-Ú

Using MTS Administration Objects A Using MTS Collection Types A MTS Administration Object Methods A Visual Basic ,É,æ,é,“x,È MTS ŠÇ—,ÌŽ©“®‰»

Visual Basic ,É,æ,é□,“x,È MTS ŠÇ—□,ÌŽ©“®‰»

ŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg,Í□AƒNƒ‰ƒXƒ“ƒg,Æƒ□□[ƒ‹,Ì□\□¬□AƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg□A,¨,æ,ÑƒJƒ^□fO,ªƒTƒ|□[ƒg,·,éƒRƒŒƒNƒVƒ‡ƒ“,Æƒvƒ□ƒpƒefB,Ì–¼‘O,Ö,ÌƒAƒNƒZƒX,ÉŽg—p,Å,«,Ü,·□Bˆ‰»º,ÌƒgƒsƒbƒN,Å,Í□AŠÇ—□ƒXƒNƒŠƒvƒg ƒIƒuƒWƒFƒNƒg□ã,Åƒ□ƒ\ƒbƒh,ðŽg,¤,±,Æ,É,æ,Á,Ä□AŽŸ,ÌŠÇ—□□ì‹Æ,ðŽ©“®‰»»,·,é,½,ß,ÌŽè□‡,Æ Visual Basic Version 5.0 ,ÌƒTƒ“ƒvƒ‹ ƒR□[ƒh,É,Â,¢,Ä□à–¾,µ,Ü,·□B

- MTS ŠÖ˜AƒRƒŒƒNƒVƒ‡ƒ“–¼,Ö,ÌƒAƒNƒZƒX,ðŽ©“®‰»»,·,é
- MTS ƒvƒ□ƒpƒefB□î•ñ,Ö,ÌƒAƒNƒZƒX,ðŽ©“®‰»»,·,é
- MTS ƒ□□[ƒ‹,Ì□\□¬,ðŽ©“®‰»»,·,é
- MTS ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg,ðŽ©“®‰»»,·,é
- fŠƒ,□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŽg,¤,æ,¤,É MTS ƒNƒ‰ƒCƒAƒ“ƒg,ð□\□¬,·,é□ì‹Æ,ðŽ©“®‰»»,·,é
- fŠƒ,□[ƒg ƒT□[ƒo□[□ã,Å MTS ƒpƒbƒP□[ƒW,Ìƒvƒ□ƒpƒefB,ð□X□V,·,é□ì‹Æ,ðŽ©“®‰»»,·,é

MTS ŠÖ˜AƒRƒŒƒfƒNƒVƒ‡ƒ“–¼,Ö,ÌƒAƒNƒZƒX,ðŽ©“®‰»,·,é

RelatedCollectionInfo ƒRƒŒƒfƒNƒVƒ‡ƒ“,É,Í□A1
,Â,ÌƒRƒŒƒfƒNƒVƒ‡ƒ“,©,çƒAƒNƒZƒX,Å,«,éŠÖ˜AƒRƒŒƒfƒNƒVƒ‡ƒ“,Ìˆê——,ªŠi”[,³,ê,Ä,¢,Ü,·□B,±,ÌƒRƒŒƒfƒNƒVƒ‡ƒ“,ªƒTƒ|□[ƒg,·,éƒvƒ□ƒpƒeƒB,Ìˆê——,É,Â,¢,Ä,Í□A□uMTS Administrative Reference (ŠÇ—□ƒŠƒtƒ@ƒŒƒ“ƒX)□v,Ì **RelatedCollectionInfo** ƒRƒŒƒfƒNƒVƒ‡ƒ“,ÌƒgƒsƒbƒN,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ŠÖ˜AƒRƒŒƒfƒNƒVƒ‡ƒ“–¼,ÉƒAƒNƒZƒX,µ□AŠÖ˜AƒRƒŒƒfƒNƒVƒ‡ƒ“–¼,ð•\Ž¦,·,é,É,Í

1 ŠÖ˜AƒRƒŒƒfƒNƒVƒ‡ƒ“,Ì–¼‘O,ªŠi”[,³,ê,Ä,¢,éƒIƒuƒWƒFƒNƒg,Ö,ÌƒAƒNƒZƒX,ÉŽg,¤ƒIƒuƒWƒFƒNƒg,ð□éŒ¾,µ,Ü,·□B

```
Dim catalog As Object
Dim packages As Object
Dim RelatedCollectionInfo As Object
Dim collName As Object
```

2 ƒ□ƒ\ƒbƒh,ªŽ¸”s,Ì HRESULT ,ð•Ô,µ,½□ê□‡,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,ðŽg,Á,Äƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰□[,ð□ˆ—□,µ,Ü,·□B**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,Æ **Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ,ªƒgƒ‰ƒbƒv,Å,«,éƒGƒ‰□[,ðƒeƒXƒg,µ□AƒGƒ‰□[,É‰ž“š,·,é,±,Æ,ª,Å,«,Ü,·□B

```
On Error GoTo failed
```

3 **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Catalog** ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Packages** ƒRƒŒƒfƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□BŽŸ,É□A**Packages** ƒRƒŒƒfƒNƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg□ã,Å **GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **RelatedCollectionInfo** ƒRƒŒƒfƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **RelatedCollectionInfo** ƒRƒŒƒfƒNƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,·,é,Æ,«,Í□AƒL□['l,ð‹ó”',Ì,Ü,Ü,É,µ,Ä,,¾,³,¢□B**RelatedCollectionInfo** ƒRƒŒƒfƒNƒVƒ‡ƒ“,Ì□î•ñ,Í□A,Ç,ÌƒpƒbƒP□[ƒW,Å,à“¯,¶,È,Ì,Å□AƒL□['l,ÍŽg,í,ê,Ü,¹,ñ□B**Populate** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,ÄƒJƒ^ƒ□ƒO,Ì□î•ñ,ð **RelatedCollectionInfo** ƒRƒŒƒfƒNƒVƒ‡ƒ“,É'}“ü,µ,Ü,·□B

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set packages = catalog.GetCollection("Packages")
Set RelatedCollectionInfo = packages.GetCollection("RelatedCollectionInfo",
"")
RelatedCollectionInfo.Populate
```

4 **RelatedCollectionInfo** ƒRƒŒƒfƒNƒVƒ‡ƒ“‘à,ð—ñ‹“,µ□AŠeƒRƒŒƒfƒNƒVƒ‡ƒ“,Ì–¼‘O,ð•\Ž¦,µ,Ü,·□B

```
For Each collName In RelatedCollectionInfo
    Debug.Print collName.Name
Next
Exit Sub
```

5 ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ªŽ¸”s,µ,½□ê□‡,Í□A**Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,ÄƒGƒ‰□[ƒ□ƒbƒZ□[ƒW,ð•\Ž¦,µ,Ü,·□B

```
MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ,É,æ,é MTS ŠÇ—□,ÌŽ©“®‰»

MTS ƒvƒ□ƒpƒeƒB□î•ñ,Ö,ÌƒAƒNƒZƒX,ðŽ©“®‰»,·,é

PropertyInfo ƒRƒŒƒNƒVƒ‡ƒ“,É,Í□AƒRƒŒƒNƒVƒ‡ƒ“‘à,ÌŠeƒvƒ□ƒpƒeƒB,ÉŠÖ,·,é□î•ñ,ªŠi”[,³,ê,Ä,¢,Ü,·□B,± ,ÌƒRƒŒƒNƒVƒ‡ƒ“,Ì□Ú□×,É,Â,¢,Ä,Í□A□uMTS Administrative Reference (ŠÇ—□ƒŠƒtƒ@ƒŒƒ“ƒX)□v,Ì **PropertyInfo** ƒRƒŒƒNƒVƒ‡ƒ“,ÌƒgƒsƒbƒN,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒRƒŒƒNƒVƒ‡ƒ“‘à,ÌŠeƒvƒ□ƒpƒeƒB,ÉƒAƒNƒZƒX,µ□Aƒvƒ□ƒpƒeƒB,Ì–¼‘O,Ìˆê——,ð•\Ž¦,·,é,É,Í

1 ƒJƒ^ƒ□ƒO,ÉŠi”[,³,ê,Ä,¢,éƒvƒ□ƒpƒeƒB□î•ñ,Ö,ÌƒAƒNƒZƒX,ÉŽg,¤ƒIƒuƒWƒFƒNƒg,ð□éŒ¾,µ,Ü,·□B

```
Dim catalog As Object
Dim packages As Object
Dim propertyInfo As Object
Dim property As Object
```

2 ƒ□ƒ\ƒbƒh,ªŽ¸”s,Ì HRESULT ,ð•Ô,µ,½□ê□‡,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,ðŽg,Á,Äƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ ‰□[,ð□ˆ—□,µ,Ü,·□B**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,Æ **Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ,ªƒgƒ‰ƒbƒv,Å,«,éƒGƒ ‰□[,ðƒeƒXƒg,µ□AƒGƒ‰□[,É‰ž“š,·,é,±,Æ,ª,Å,«,Ü,·□B

```
On Error GoTo failed
```

3 **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Catalog** ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Packages** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□B**Packages** ƒRƒŒƒNƒVƒ‡ƒ“□ã,Å **GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **PropertyInfo** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **PropertyInfo** ƒRƒŒƒNƒVƒ‡ƒ“,ÉƒAƒNƒZƒX,·,é,Æ,«,Í□AƒL□['I,ð‹ó”',Ì,Ü,Ü,É,µ,Ä,,¾,³,¢□B**PropertyInfo** ƒRƒŒƒNƒVƒ‡ƒ“,Ì□î•ñ,Í□A,Ç,ÌƒpƒbƒP□[ƒW,Å,à“¯,¶,È,Ì,Å□AƒL□['I,ÍŽg,í,ê,Ü,¹,ñ□B**Populate** ƒ□ƒ\ ƒbƒh,ðŒÄ,Ñ□o,µ,ÄƒJƒ^ƒ□ƒO,Ì□î•ñ,ð **PropertyInfo** ƒRƒŒƒNƒVƒ‡ƒ“,É‘}“ü,µ,Ü,·□B

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set packages = catalog.GetCollection("Packages")
Set propertyInfo = packages.GetCollection("PropertyInfo", "")
propertyInfo.Populate
```

4 **PropertyInfo** ƒRƒŒƒNƒVƒ‡ƒ“‘à,ð—ñ‹“,µ□AƒRƒŒƒNƒVƒ‡ƒ“‘à,ÌŠeƒvƒ□ƒpƒeƒB–¼,Ìˆê——,ð•\Ž¦,µ,Ü,·□B

```
For Each property In propertyInfo
    Debug.Print property.Name
Next
Exit Sub
```

5 ƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹,ªŽ¸”s,µ,½□ê□‡,Í□A**Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,ÄƒGƒ‰□[ƒ□ƒbƒZ□[ƒW,ð•\ Ž¦,µ,Ü,·□B

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ,É,æ,é MTS ŠÇ—□,ÌŽ©“®‰»

MTS ƒ□□[ƒ‹,Ì□\□¬,ðŽ©“®‰»,·,é

ƒpƒbƒP□[ƒW,Æƒfƒƒƒ|□[ƒlƒ“ƒg,Ìƒ□□[ƒ‹,ð□\□¬,µ□Aƒt□[ƒU□[,ÉŠÇ—□ŽÒŒ ŒÀ,ðŠ„,è“-,Ä,é,É,Í

1 “Á'è,Ìƒfƒƒƒlƒ“ƒg,Ìƒ□□[ƒ‹,ð□\□¬,·,é,Æ,«,ÉŽg,¤ƒlƒuƒWƒFƒNƒg,ð□éŒ¾,µ,Ü,·□B

```
Dim catalog As Object
Dim packages As Object
Dim pack As Object
Dim comp As Object
Dim newUser As Object
Dim newRole As Object
Dim componentsInPack As Object
Dim RolesInPackage As Object
Dim usersInRole As Object
Dim rolesForComponent As Object
Dim util As Object
```

2 ƒ□ƒ\ƒbƒh,ªŽ¸”s,Ì HRESULT ,ð•Ô,µ,½□ê□‡,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,ðŽg,Á,Äƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰□[,ð□ˆ—□,µ,Ü,·□B**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,Æ **Err** ƒlƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ,ªƒgƒ‰ƒbƒv,Å,«,éƒGƒ‰□[,ðƒefXƒg,µ□AƒGƒ‰□[,É‰ž“š,·,é,±,Æ,ª,Å,«,Ü,·□B

```
On Error GoTo failed
```

3 **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Catalog** ƒlƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Packages** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□BŽŸ,É□Aƒ□ƒ^ƒ□ƒO,Ìƒf□[ƒ^,ð **Packages** ƒRƒŒƒNƒVƒ‡ƒ“,É'}“ü,µ,Ü,·□B

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set packages = catalog.GetCollection("Packages")
packages.Populate
```

4 **Packages** ƒRƒŒƒNƒVƒ‡ƒ““à,ð—ñ‹“,µ□A"My Package" ,Æ,¢,¤–¼'O,Ìƒpƒbƒp□[ƒW,ð'T,µ,Ü,·□B"My Package" ,ªŒ©,Â,©,Á,½,ç□A**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **RolesInPackage** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□B**Add** ƒ□ƒ\ƒbƒh,ðŽg,Á,Ä□A□V,µ,¢ƒ□□[ƒ‹,ðƒpƒbƒP□[ƒW,É'Ç‰Á,µ,Ü,·□B□V,µ,¢ƒ□□[ƒ‹,É "R1" ,Æ,¢,¤–¼'O,ð•t,¯,Ä•Ï□X,ðƒRƒŒƒNƒVƒ‡ƒ“,É•Û'¶,µ,Ü,·□B

```
For Each pack In packages
    If pack.Name = "My Package" Then
        Set rolesInPack = packages.GetCollection("RolesInPackage", pack.Key)
        Set newRole = rolesInPack.Add
        newRole.Value("Name") = "R1"
        rolesInPack.savechanges
```

5 **RolesInPackage** ƒRƒŒƒNƒVƒ‡ƒ“□ã,Å **GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **UsersInRole** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□B**Add** ƒ□ƒ\ƒbƒh,ðŽg,Á,Ä□AŠù'¶,Ì NT ƒt□[ƒU□[,ðƒ□□[ƒ‹,É'Ç‰Á,µ,Ü,·□Bƒt□[ƒU□[–¼,ð "administrator" ,É□Ý'è,µ,Ü,·□B•Ï□X,ð **UsersInRole** ƒRƒŒƒNƒVƒ‡ƒ“,É•Û'¶,µ,Ü,·□B

```
            Set usersInRole = RolesInPackage.GetCollection("UsersInRole",
newRole.Key)
            Set newUser = usersInRole.Add
            newUser.Value("User") = "administrator"
            usersInRole.savechanges
```

6 **GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **ComponentsInPackage** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□B**ComponentsInPackage** ƒRƒŒƒNƒVƒ‡ƒ“,Éƒf□[ƒ^,ð'}“ü,µ□AƒRƒŒƒNƒVƒ‡ƒ““à,ð—ñ‹“,µ□A"Bank Account" ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð'T,µ,Ü,·□B□V,µ,¢ƒ□□[ƒ‹,ðƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉŠÖ˜A•t,¯,é,½,ß,É□A**GetUtilInterface** ƒ□ƒ\

ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚Ä **RoleAssociationUtil**
ƒIƒuƒWƒFƒNƒg‚ìƒCƒ“ƒXƒ^ƒ“ƒX‚ð□¶□¬‚µ‚Ü‚·□BŽŸ‚É□A**AssociateRole** ƒ□ƒ\
ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚Ä□V‚µ‚¢ƒ□□[ƒ‹‚ðƒRƒ“ƒ|[ƒlƒ“ƒg‚ÉŠÖ˜A•t‚¯‚Ü‚·□B

```
            Set componentsInPack = packages.GetCollection("ComponentsInPackage",
pack.Key)
            componentsInPack.Populate
            For Each comp In componentsInPack
                If comp.Name = "Bank.Account" Then
                    Set rolesForComponent =
componentsInPack.GetCollection("RolesForPackageComponent", comp.Key)
                    Set util = rolesForComponent.GetUtilInterface
                    util.associateRole (newRole.Key)
                    Exit For
                End If
            Next
            Exit For
        End If
    Next
    Exit Sub
```

7 ƒRƒ“ƒ|[ƒlƒ“ƒg‚ìƒCƒ“ƒXƒg□[ƒ‹‚ªŽ¸”s‚µ‚½□ê□‡‚Í□A**Err** ƒIƒuƒWƒFƒNƒg‚ðŽg‚Á‚ÄƒG‚ª‰□[ƒ□ƒbƒZ□[ƒW‚ð•\Ž¦‚µ‚Ü‚·□B

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ‚É‚æ‚é MTS ŠÇ—□‚ìŽ©“®‰»

MTS ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg,ðŽ©“®‰»,·,é

"test.pak" ,Æ,¢,¤–¼‘O,ÌƒpƒbƒP□[ƒW,ðƒGƒNƒXƒ|□[ƒg,·,é,É,Í

1 ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg,ÉŽg,¤ƒIƒuƒWƒFƒNƒg,ð□éŒ¾,µ,Ü,·□B

```
Dim catalog As Object
Dim packages As Object
Dim pack As Object
Dim util As Object
```

2 ƒ□ƒ\ƒbƒh,ªŽ¸”s,Ì HRESULT ,ð•Ô,µ,½□ê□‡,Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,ðŽg,Á,Äƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰ƒ□[,ð□ˆ—□,µ,Ü,·□B**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg,Æ **Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,Ä MTS ,ªƒgƒ‰ƒbƒv,Å,«,éƒGƒ‰ƒ□[,ðƒefXƒg,µ□AƒGƒ‰ƒ□[,É‰ž“š,·,é,±,Æ,ª,Å,«,Ü,·□B

```
On Error GoTo failed
```

3 **CreateObject** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Catalog** ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B**GetCollection** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ä **Packages** ƒRƒŒƒNƒVƒ‡ƒ“,ðŽæ“¾,µ,Ü,·□B**Populate** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Äƒ‚ƒ^□ƒ□O,Ìƒf□[ƒ^,ðƒpƒbƒP□[ƒW,É‘}“ü,µ,Ü,·□B

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set packages = catalog.GetCollection("Packages")
packages.Populate
```

4 **Packages** ƒRƒŒƒNƒVƒ‡ƒ“‘à,ð—ñ‹“,µ□A"My Package" ,Æ,¢,¤–¼‘O,ÌƒpƒbƒP□[ƒW,ð’T,µ,Ü,·□BƒpƒbƒP□[ƒW,ªŒ©,Â,©,Á,½,ç□AƒpƒbƒP□[ƒW ƒ†□[ƒefBƒŠƒefB ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ□A**ExportPackage** ƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,µ,Ü,·□B

```
For Each pack In packages
    If pack.Name = "My Package" Then
        Set util = packages.GetUtilInterface
        util.ExportPackage pack.Key, "c:¥test.pak", 0
        Exit For
    End If
Next
Exit Sub
```

5 ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒCƒ“ƒXƒg□[ƒ‹,ªŽ¸”s,µ,½□ê□‡,Í□A**Err** ƒIƒuƒWƒFƒNƒg,ðŽg,Á,ÄƒGƒ‰ƒ□[ƒ□ƒbƒZ□[ƒW,ð•\Ž¦,µ,Ü,·□B

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ,É,æ,é MTS ŠÇ—□,ÌŽ©“®‰»

ƒŠƒ‚[ƒg ƒRƒ“ƒ|[ƒlƒ“ƒg‚ðŽg‚¤‚æ‚¤‚É MTS ƒNƒ‰ƒCƒAƒ“ƒg‚ð\¬‚·‚é‰ì‹Æ‚ðŽ©“®‰»‚·‚é

Bank.CreateTable ƒŠƒ‚[ƒg ƒRƒ“ƒ|[ƒlƒ“ƒg‚ðŽg‚¤‚æ‚¤‚ÉƒNƒ‰ƒCƒAƒ“ƒg‚ð\¬‚·‚é‚É‚Í

1 ƒŠƒ‚[ƒg ƒRƒ“ƒ|[ƒlƒ“ƒg‚ðŽÀs‚·‚é‚æ‚¤‚É (MTS ‚ðŽÀs‚µ‚Ä‚¢‚È‚¢) ƒNƒ‰ƒCƒAƒ“ƒg‚ð\¬‚·‚é‚Æ‚«‚ÉŽg‚¤ƒIƒuƒWƒFƒNƒg‚ðéŒ¾‚µ‚Ü‚·B

```
Dim catalog As Object
Dim remoteComps As Object
Dim util As Object
```

2 ƒ‚ƒ\ƒbƒh‚ªŽ¸”s‚Ì HRESULT ‚ð•Ô‚µ‚½ê‡‚Í**On Error** ƒXƒe[ƒgƒ‚ƒf“ƒg‚ðŽg‚Á‚Äƒ‰ƒf“ƒ^ƒCƒ€ ƒGƒ‰[‚ðˆ—‚µ‚Ü‚·B**On Error** ƒXƒe[ƒgƒ‚ƒf“ƒg‚Æ **Err** ƒIƒuƒWƒFƒNƒg‚ðŽg‚Á‚Ä MTS ‚ªƒgƒ‰ƒbƒv‚Å‚«‚éƒGƒ‰[‚ðƒeƒXƒg‚µ‚AƒGƒ‰[‚É‰ž“š‚·‚é‚±‚Æ‚ª‚Ü‚Ü‚Ü‚·B

```
On Error GoTo failed
```

3 **CreateObject** ƒ‚ƒ\ƒbƒh‚ŒÄ‚Ñ‚o‚µ‚Ä **Catalog** ƒIƒuƒWƒFƒNƒg‚ÌƒCƒ“ƒXƒ^ƒ“ƒX‚ð¶¬‚µ‚Ü‚·B**GetCollection** ƒ‚ƒ\ƒbƒh‚ŒÄ‚Ñ‚o‚µ‚Ä **RemoteComponents** ƒRƒŒƒNƒVƒ‡ƒ“‚ðŽæ“¾‚µ‚Ü‚·BŽŸ‚É**GetUtilInterface** ƒ‚ƒ\ƒbƒh‚ðŽg‚Á‚Ä**RemoteComponentUtil** ƒIƒuƒWƒFƒNƒg‚ÌƒCƒ“ƒXƒ^ƒ“ƒX‚ð¶¬‚µ‚Ü‚·BƒŠƒ‚[ƒg ƒRƒ“ƒ|[ƒlƒ“ƒg‚ð‘g‚Ý‚ž‚Þ‚½‚ß‚É**InstallRemoteComponentByName** ƒ‚ƒ\ƒbƒh‚ŒÄ‚Ñ‚o‚µ‚ÄƒTƒ[ƒo[ƒRƒ“ƒsƒ…[ƒ^–¼ƒAƒTƒ[ƒo[ãƒpƒbƒP[ƒW–¼A¨æ‚ÑƒRƒ“ƒ|[ƒlƒ“ƒg–¼‚ð“n‚µ‚Ü‚·B

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set remoteComps = catalog.GetCollection("RemoteComponents")
Set util = remoteComps.GetUtilInterface
util.InstallRemoteComponentByName "remote1", "New","Bank.CreateTable"
Exit Sub
```

4 ƒRƒ“ƒ|[ƒlƒ“ƒg‚ÌƒCƒ“ƒXƒg[ƒ‹‚ªŽ¸”s‚µ‚½ê‡‚Í**Err** ƒIƒuƒWƒFƒNƒg‚ðŽg‚Á‚ÄƒGƒ‰[ƒ‚ƒbƒZ[ƒW‚ð•\Ž¦‚µ‚Ü‚·B

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A€–Ú

Using MTS Administration Objects‚ÆUsing MTS Collection Types‚ÆMTS Administration Object Methods‚ÆVisual Basic ‚É‚æ‚é MTS ŠÇ—‚ÌŽ©“®‰»

ƒŠƒ‚□[ƒg ƒT□[ƒo□[□ã‚Å MTS ƒpƒbƒP□[ƒW‚Ìƒvƒ□ƒpƒeƒB‚ð□X□V‚·‚é□ì‹Æ‚ðŽ©“® ‰»‚·‚é

"remote1" ‚Æ‚¢‚¤–¼‘O‚ÌƒŠƒ‚□[ƒg ƒRƒ“ƒsƒ…□[ƒ^□ã‚ÅƒpƒbƒP□[ƒW‚Ìƒvƒ□ƒpƒeƒB‚ð□X□V‚·‚é‚É‚Í

1 ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ì”z’u‚ÆŠÇ—□‚ð□s‚¤‚æ‚¤‚É (MTS ‚ðŽÀ□s‚µ‚Ä‚¢‚é) ƒNƒ‰ƒCƒAƒ“ƒg‚ð□\□¬‚·‚é‚Æ‚«‚ÉŽg‚¤ƒIƒuƒWƒFƒNƒg‚ð□éŒ¾‚µ‚Ü‚·□B

```
Dim catalog As Object
Dim packages As Object
Dim pack As Object
Dim root As Object
```

2 ƒ□ƒ\ƒbƒh‚ªŽ¸”s‚Ì HRESULT ‚ð•Ô‚µ‚½□ê□‡‚Í□A**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg‚ðŽg‚Á‚Äƒ‰ƒ“ƒ^ƒCƒ€ ƒGƒ‰□[‚ð□ˆ—□‚µ‚Ü‚·□B**On Error** ƒXƒe□[ƒgƒ□ƒ“ƒg‚Æ **Err** ƒIƒuƒWƒFƒNƒg‚ðŽg‚Á‚Ä MTS ‚ªƒgƒ‰ƒbƒv‚Å‚«‚éƒGƒ‰□[‚ðƒeƒXƒg‚µ□AƒGƒ‰□[‚É‰ž“š‚·‚é‚±‚Æ‚ª‚Å‚«‚Ü‚·□B

```
On Error GoTo failed
```

3 **CreateObject** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚Ä **Catalog** ƒIƒuƒWƒFƒNƒg‚ÌƒCƒ“ƒXƒ^ƒ“ƒX‚ð□¶□¬‚µ‚Ü‚·□B**Connect** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚Ä "remote1" ‚Æ‚¢‚¤–¼‘O‚ÌƒRƒ“ƒsƒ…□[ƒ^□ã‚Ì "ƒ‹□[ƒg ƒRƒŒƒNƒVƒ‡ƒ“" ‚ÉƒAƒNƒZƒX‚µ‚Ü‚·□Bƒ‹□[ƒg ƒRƒŒƒNƒVƒ‡ƒ“‚Í□AƒRƒ“ƒsƒ…□[ƒ^□ã‚Ì□Å□ã‚Ê‚ÌƒRƒŒƒNƒVƒ‡ƒ“‚ÉƒAƒNƒZƒX‚·‚é‚½‚ß‚ÌƒRƒŒƒNƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg‚Å‚·□Bƒ‹□[ƒg ƒRƒŒƒNƒVƒ‡ƒ“‚É‚ÍƒIƒuƒWƒFƒNƒg‚ªŠÜ‚Ü‚ê‚Ä‚¢‚Ü‚¹‚ñ□B‚Ü‚½□Aƒ‹□[ƒg ƒRƒŒƒNƒVƒ‡ƒ“‚É‚Íƒvƒ□ƒpƒeƒB‚ª‚ ‚è‚Ü‚¹‚ñ□Bƒ‹□[ƒg ƒRƒŒƒNƒVƒ‡ƒ“‚©‚ç **GetCollection** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚·‚Æ‚«‚Í□AƒL□['l‚ðŽg‚¢‚Ü‚¹‚ñ□B**GetCollection** ƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚µ‚ÄƒŠƒ‚□[ƒg ƒRƒ“ƒsƒ…□[ƒ^□ã‚Ì **Packages** ƒRƒŒƒNƒVƒ‡ƒ“‚ðŽæ“¾‚µ‚Ü‚·□BŽŸ‚É□A**Populate** ƒ□ƒ\ƒbƒh‚ðŽg‚Á‚Ä□AƒpƒbƒP□[ƒW ƒRƒŒƒNƒVƒ‡ƒ“‚Éƒf□[ƒ^‚ð‘}“ü‚µ‚Ü‚·□B

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set root = catalog.Connect("remote1")
Set packages = root.GetCollection("Packages", "")
packages.Populate
```

4 "My Package" ‚Ì SecurityEnabled ƒvƒ□ƒpƒeƒB‚ð "Y" ‚É□Ý’è‚µ□A•Ï□X‚ðƒpƒbƒP□[ƒW ƒRƒŒƒNƒVƒ‡ƒ“‚É•Û‘¶‚µ‚Ü‚·□B

```
For Each pack In packages
    If pack.Name = "My Package" Then
        pack.Value("SecurityEnabled") = "Y"
        Exit For
    End If
Next
packages.savechanges
Exit Sub
```

5 ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ÌƒCƒ“ƒXƒg□[ƒ‹‚ªŽ¸”s‚µ‚½□ê□‡‚Í□A**Err** ƒIƒuƒWƒFƒNƒg‚ðŽg‚Á‚ÄƒGƒ‰□[ƒ□ƒbƒZ□[ƒW‚ð•\Ž¦‚µ‚Ü‚·□B

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

ŠÖ˜A□€–Ú

Using MTS Administration Objects□AUsing MTS Collection Types□AMTS Administration Object Methods□AVisual Basic ‚É‚æ‚é MTS ŠÇ—□‚ÌŽ©“®‰»

MTS Overview and Concepts

How Does MTS Work?

Explains the elements of Microsoft Transaction Server and how they work together to provide the infrastructure and programming model to develop and deploy components.

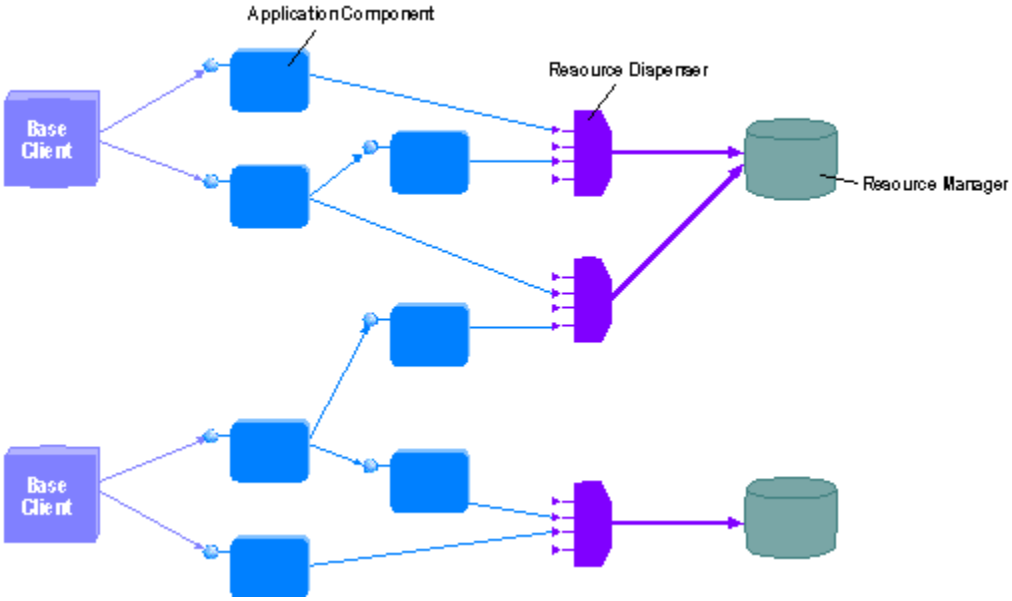
MTS Programming Concepts

Explains the key concepts that a component application developer needs to understand for developing applications.

How Does MTS Work?

This topic describes the elements of Microsoft Transaction Server (MTS) and explains how they provide the infrastructure and programming model to develop and deploy MTS components.

Elements of MTS



Contents

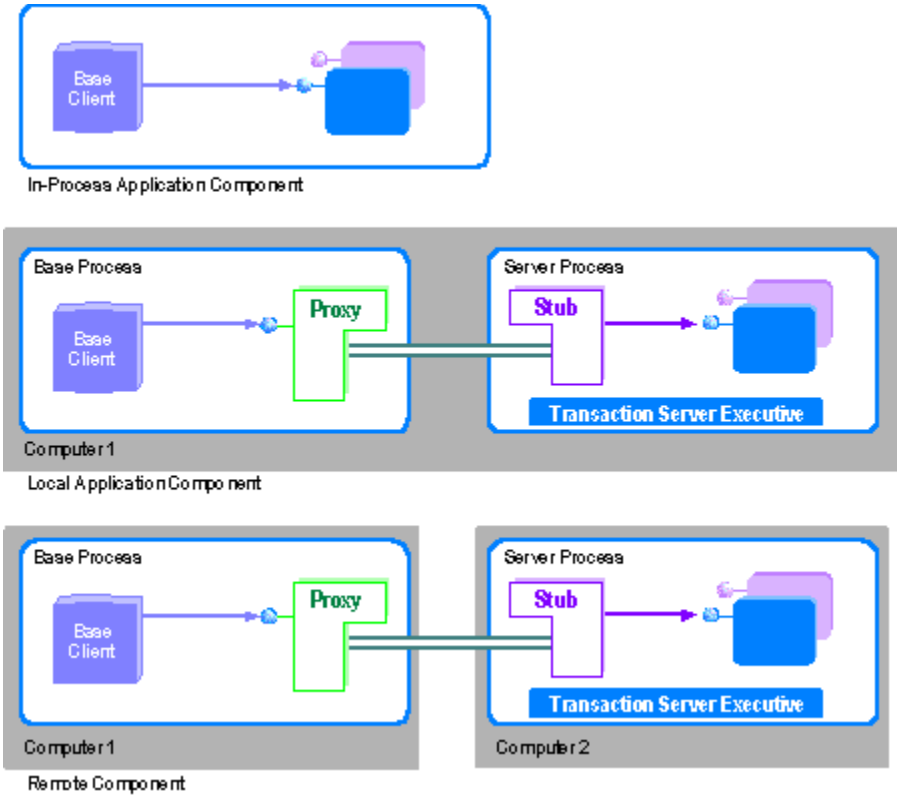
Application Components

Application components model the activity of a business. These components implement the business rules, providing views and transformations of the application state. Consider, for example, the case of an online bank. Records in one or more database systems represent the durable state of the business, such as the amount of money in an account. The application components update that state to reflect such changes as debits and credits.

MTS shelters developers from complex server issues, allowing them to focus on implementing business functions. Because components running in the MTS run-time environment can take advantage of transactions, developers can write applications as if they run in isolation. MTS handles the concurrency, resource pooling, security, context management, and other system-level complexities. The transaction system, working in cooperation with database servers and other types of resource managers, ensures that concurrent transactions are atomic, consistent, have proper isolation, and that, once committed, the changes are durable. For more information on the benefits of transactions, see Transactions.

MTS also makes it easier to build distributed applications by providing location transparency. MTS automatically loads the component into a process environment. An MTS component can be loaded into a client application process (in-process component), or into a separate surrogate server process environment, either on the client's computer (local component) or on another computer (remote component).

In-process, local, and remote components



MTS Components and COM

MTS components are COM in-process server components contained in (DLLs). They are distinguished from other COM components in that they execute in the MTS run-time environment. You can create and implement these components with Visual Basic, Visual C++, Visual J++, or any ActiveX-compatible development tool.

Note that the term *component* represents the code that implements a COM object. For example, Visual C++ components are implemented as classes. Likewise, Visual Basic components are implemented by class modules.

MTS imposes specific requirements on components beyond those required by COM (see MTS Component Requirements). This allows MTS to provide services to the component that would not otherwise be possible. These include increased scalability and robustness and simplified system management.

See Also

Base Clients vs. MTS Components, MTS Objects, Business Logic in MTS Components, Server Processes

MTS Executive

The MTS Executive is a dynamic-link library (DLL) that provides run-time services for MTS components, including thread and context management. This DLL loads into the processes that host application components and runs in the background.

MTS also provides a set of resource dispensers that simplify access to shared resources in a server process. For more information, see Resource Dispensers.

See Also

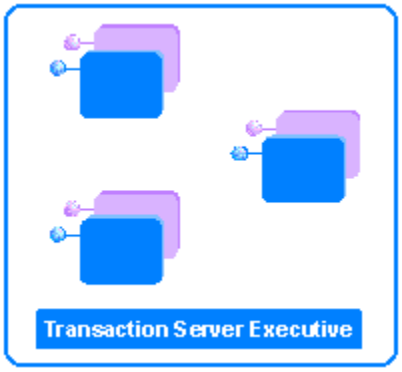
Application Components

Server Processes

A *server process* is a system process that hosts the execution of an application component. A server process can host multiple components and can service tens, hundreds, or potentially thousands of clients. You can configure multiple server processes to execute on a single computer. Each server process reflects a separate trust boundary and fault isolation domain.

Other process environments can also host application components. As a result, you can deploy applications that meet varying distribution, performance, and fault isolation requirements. For example, you can configure MTS components to be loaded directly into Microsoft Internet Information Server (IIS). You can also configure them to load directly into client processes.

Server Process



Resource Managers

A *resource manager* is a system service that manages durable data. Server applications use resource managers to maintain the durable state of the application, such as the record of inventory on hand, pending orders, and accounts receivable. Resource managers work in cooperation with the Microsoft Distributed Transaction Coordinator to guarantee atomicity and isolationto an application.

MTS supports resource managers, such as Microsoft SQL Server version 6.5, that implement the OLE Transactions protocol.

See Also

Resource Dispensers, Microsoft Distributed Transaction Coordinator

Resource Dispensers

A *resource dispenser* manages nondurable shared state on behalf of the application components within a process. Resource dispensers are similar to resource managers, but without the guarantee of durability. MTS provides two resource dispensers:

- The ODBC resource dispenser
- The Shared Property Manager

ODBC Resource Dispenser

The ODBC resource dispenser manages pools of database connections for MTS components that use the standard ODBC interfaces, allocating connections to objects quickly and efficiently. Connections are automatically enlisted on an object's transactions, and the resource dispenser can automatically reclaim and reuse connections. The ODBC 3.0 Driver Manager is the ODBC resource dispenser; the Driver Manager DLL is installed with MTS.

Shared Property Manager

The Shared Property Manager provides synchronized access to application-defined, process-wide properties (variables). For example, you can use it to maintain a Web-page hit counter or to maintain the shared state for a multiuser game.

See Also

ISharedPropertyGroupManager Interface, Creating a Simple ActiveX Component, Sharing State

Microsoft Distributed Transaction Coordinator

The Microsoft Distributed Transaction Coordinator (MS DTC) is a system service that coordinates transactions. Work can be committed as an atomic transaction even if it spans multiple resource managers, potentially on separate computers.

MS DTC was first released as part of Microsoft SQL Server version 6.5 and is included in MTS, providing a low-level infrastructure for transactions. MS DTC implements a two-phase commit protocol to ensure that the transaction outcome (either commit or abort) is consistent across all resource managers involved in a transaction. MS DTC ensures atomicity, regardless of failures.

See Also

Resource Managers

MTS Explorer

You can use the MTS Explorer to deploy application components. You can also use it to view and manipulate items in the MTS run-time environment.

For a complete discussion of using the MTS Explorer for application administration, see the *Administrator's Guide*.

MTS Programming Concepts

This topic explains the key concepts that a component application developer needs to understand to develop Microsoft Transaction Server (MTS) applications.

Contents

Transactions

MTS simplifies the task of developing application components by allowing you to perform work with transactions. This protects applications from anomalies caused by concurrent updates or system failures.

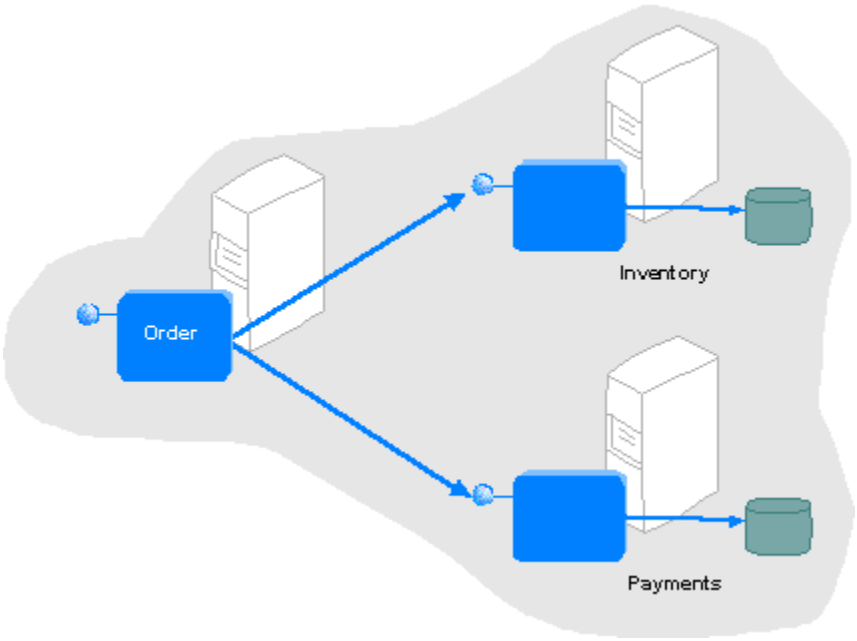
Transactions maintain the ACID properties:

- *Atomicity* ensures that all the updates completed under a specific transaction are committed and made durable, or that they get aborted and rolled back to their previous state.
- *Consistency* means that a transaction is a correct transformation of the system state, preserving the state invariants.
- *Isolation* protects concurrent transactions from seeing each other's partial and uncommitted results, which might create inconsistencies in the application state. Resource managers use transaction-based synchronization protocols to isolate the uncommitted work of active transactions.
- *Durability* means that committed updates to managed resources, such as a database record, survive failures, including communication failures, process failures, and server system failures. Transactional logging even allows you to recover the durable state after disk media failures.

The intermediate states of a transaction are not visible outside the transaction, and either all the work happens or none of it does. This allows you to develop application components as if each transaction executes sequentially and without regard to concurrency. This is a tremendous simplification for application developers.

You can declare that a component is transactional, in which case MTS associates transactions with the component's objects. When an object's method is executed, the services that resource managers and resource dispensers perform on its behalf execute under a transaction. This can also include work that it performs for other MTS objects. Work from multiple objects can be composed into a single atomic transaction.

Multiple objects composed into a transaction



Without transactions, error recovery is extremely difficult, especially when multiple

objects update multiple databases. The possible combinations of failure modes are too great even to consider. Transactions simplify error recovery. Resource managers automatically undo the transaction's work, and the application retries the entire business transaction.

Transactions also provide a simple concurrency model. Because a transaction's isolation prevents one client's work from interfering with other clients, you can develop components as though only a single client executes at a time.

Components Declare Transactional Requirements

Every MTS component has a transaction attribute that is recorded in the MTS catalog. MTS uses this attribute during object creation to determine whether the object should be created to execute within a transaction, and whether a transaction is required or optional. For more information on transaction attributes, see Transaction Attributes.

Components that make updates to multiple transactional resources, such as database records, for example, can ensure that their objects are always created within a transaction. If the object is created from a context that has a transaction, the new context inherits that transaction; otherwise, the system automatically initiates a transaction.

Components that only perform a single transactional update can be declared to support, but not require, transactions. If the object is created from a context that has a transaction, the new context inherits that transaction. This allows the work of multiple objects to be composed into a single atomic transaction. If the object is created from a context that does not have a transaction, the object can rely on the resource manager to ensure that the single update is atomic.

How Work Is Associated with a Transaction

An object's associated context object indicates whether the object is executing within a transaction and, if so, the identity of the transaction.

Resource dispensers can use the context object to provide transaction-based services to the MTS object. For example, when an object executing within a transaction allocates a database connection by using the ODBC resource dispenser, the connection is automatically enlisted on the transaction. All database updates using this connection become part of the transaction, and are either atomically committed or aborted. For more information, see Enlisting Resources in Transactions.

Stateful and Stateless Objects

Like any COM object, MTS objects can maintain internal state across multiple interactions with a client. Such an object is said to be stateful. MTS objects can also be stateless, which means the object does not hold any intermediate state while waiting for the next call from a client.

When a transaction is committed or aborted, all of the objects that are involved in the transaction are deactivated, causing them to lose any state they acquired during the course of the transaction. This helps ensure transaction isolation and database consistency; it also frees server resources for use in other transactions.

Completing a transaction enables MTS to deactivate an object and reclaim its resources, thus increasing the scalability of the application. Maintaining state on an object requires the object to remain activated, holding potentially valuable resources such as database connections. Stateless objects are more efficient and are thus recommended. For more information on object deactivation, see Deactivating Objects.

How Objects Can Participate in Transaction Outcome

You can use methods implemented on the **IObjectContext** interface to enable an MTS object to participate in determining a transaction's outcome. The **SetComplete**, **SetAbort**, **DisableCommit**, and **EnableCommit** methods work in conjunction with the component's transaction attribute to allow one or more objects to be composed simply and safely within transactions.

- **SetComplete** indicates that the object has successfully completed its work for the transaction. The object is deactivated upon return from the method that first entered the context.
- **SetAbort** indicates that the object's work can never be committed. The object is deactivated upon return from the method that first entered the context.
- **EnableCommit** indicates that the object's work is not necessarily done, but that its transactional updates can be committed in their current form.
- **DisableCommit** indicates that the object's transactional updates can not be committed in their current form.

Both **SetComplete** and **SetAbort** deactivate the object on return from the method. MTS reactivates the object on the next call that requires object execution.

Objects that need to retain state across multiple calls from a client can protect themselves from having their work committed prematurely by the client. By calling **DisableCommit** before returning control to the client, the object can guarantee that its transaction cannot successfully be committed without the object doing its remaining work and calling **EnableCommit**.

Client-Controlled vs. Automatic Transactions

Transactions can either be controlled directly by the client, or automatically by the MTS run-time environment.

Clients can have direct control over transactions by using a transaction context object. The client uses the **ITransactionContext** interface to create MTS objects that execute within the client's transactions, and to commit or abort the transactions.

Transactions can automatically be initiated by the MTS run-time environment to satisfy the component's transaction expectations. MTS components can be declared so that their objects always execute within a transaction, regardless of how the objects are created. This feature simplifies component development, because you do not need to write application logic to handle the special case where an object is created by a client not using transactions.

This feature also reduces the burden on client applications. Clients do not need to initiate a transaction simply because the component that they are using requires it.

MTS automatically initiates transactions as needed to satisfy a component's requirements. This event occurs, for example, when a client that is not using transactions creates an object in an MTS component that is declared to require transactions.

MTS completes automatic transactions when the MTS object that triggered their creation has completed its work. This event occurs when returning from a method call on the object after it has called **SetComplete** or **SetAbort**. **SetComplete** causes the transaction to be committed; **SetAbort** causes it to be aborted.

A transaction cannot be committed while any method is executing in an object that is participating in the transaction. The system behaves as if the object disables the commit for the duration of each method call.

See Also

Building Transactional Components, Multiple Transactions

Transaction Attributes

Every MTS component has a transaction attribute. The transaction attribute can have one of the following values:

- **Requires a transaction.** This value indicates that the component's objects must execute within the scope of a transaction. When a new object is created, its object context inherits the transaction from the context of the client. If the client does not have a transaction, MTS automatically creates a new transaction for the object.
- **Requires a new transaction.** This value indicates that the component's objects must execute within their own transactions. When a new object is created, MTS *automatically* creates a new transaction for the object, regardless of whether its client has a transaction.
- **Supports transactions.** This value indicates that the component's objects can execute within the scope of their client's transactions. When a new object is created, its object context inherits the transaction from the context of the client. If the client does not have a transaction, the new context is also created without one.
- **Does not support transactions.** This value indicates that the component's objects do not run within the scope of transactions. When a new object is created, its object context is created without a transaction, regardless of whether the client has a transaction.

Most MTS components are declared as either **Supports transactions** or **Requires a transaction**. These values allow an object to execute within the scope of its client's transaction. You can see the difference between these values when an object is created from a context that does not have a transaction. If the component's transaction attribute is **Supports transactions**, the new object runs without a transaction. If it is declared as **Requires a transaction**, MTS automatically initiates a transaction for the new object.

Declaring a component as **Requires a new transaction** is similar to using **Requires a transaction** in that the component's objects are guaranteed to execute within transactions. However, when you declare the transaction attribute this way, an object never runs inside the scope of its client's transaction. Instead, the system always creates independent transactions for the new objects. For example, you can use this for auditing components that record work done on behalf of another transaction regardless of whether the original transaction commits or aborts.

Specifying **Does not support transactions** ensures that an object's context does not contain a transaction. This value is the default setting and is primarily used with versions of COM components that precede MTS.

Setting the Transaction Attribute

The transaction attribute is part of a component definition. The component developer determines it, and changes to it are not recommended.

You can set a transaction attribute at development time by:

- Using values defined in Mtxattr.h. You can specify these values in an .odl file to encode them into the component's type library.
- Using the MTS Explorer to create a package file for deploying your components.

Because Because Microsoft Visual Basic automatically generates a type library, Visual Basic developers must use the MTS Explorer to set the transaction attribute.

See Also

Application Components, Transactions

Enlisting Resources in Transactions

MTS provides automatic transactions, which means that transaction requirements are declared as component properties. MTS automatically begins transactions and commits or aborts these transactions on behalf of the component, based on the component's transaction property.

Automatic transactions work because resource dispensers pass transactions to the resource manager. For example, the ODBC Driver Manager is a resource dispenser for ODBC database connections. When a database connection is requested from a transactional component, the ODBC Driver Manager obtains the transaction from the object's context. The ODBC Driver Manager then associates (enlists) the database connection with the transaction.

If you do not have a resource dispenser, you can build your own using the Microsoft Transaction ServerSoftware Development Kit (SDK), which is located in the Microsoft Platform SDK. For more information, see the *Resource Dispenser Guide* and the samples included with the MTS SDK.

For a resource manager to participate in an MTS transaction, it must support one of the following protocols:

- OLE Transactions
- The X/Open DTP XA standard

OLE Transactions, the object-oriented, *two-phase commit* protocol defined by Microsoft, is the preferred protocol. OLE Transactions is based on the Component Object Model (COM) and is used by resource managers in order to participate in distributed transactions coordinated by Microsoft Distributed Transaction Coordinator (DTC). OLE Transactions is supported by Microsoft SQL Server version 6.5. For more information on supporting OLE Transactions, see the *Resource Manager Guide* and the samples included with the MTS SDK.

XA is the two-phase commit protocol defined by the X/Open DTP group. XA is natively supported by many Unix databases, including Informix, Oracle, and DB2.

For MTS to work with XA-compliant resource managers, an OLE Transactions-to-XA mapper, provided by the MTS SDK, makes it relatively straightforward for XA-compliant resource managers to provide resource dispensers that can accept OLE Transactions from MTS and translate to XA. For more information, see "Mapping OLE Transactions to the XA Protocol" in the MTS SDK, or contact your resource manager vendor.

See Also

Resource Dispensers, Transactions

Determining Transaction Outcome

This section discusses how applications determine whether a transaction will commit or abort.

First, it is important to understand that the MTS objects involved in a transaction do not need to know the transaction outcome. All objects involved in a transaction are automatically deactivated. Deactivation causes the objects to lose any state that they acquired during the transaction. Consequently, their behavior is not affected by the outcome of the transaction.

Each object can participate in determining the outcome of a transaction. Objects call **SetComplete**, **SetAbort**, **EnableCommit**, and **DisableCommit,** based on the desired component behavior. For example, an object would typically call **SetAbort** after receiving an error from a database operation, on a method call to another object, or due to a violation of a business rule such as an overdrawn account.

The client of the transaction determines its success or failure (commit or abort) based on values returned from the method call that caused the transaction to complete. The client can be either a base client or another MTS object that exists outside the transaction. The client must know which methods cause transactions to complete and how the method output values can be used to determine success (commit) or failure (abort).

An object method that intends to commit a transaction typically returns an HRESULT value of S_OK after calling **SetComplete**. On return, MTS automatically completes the transaction. If the transaction commits, the S_OK value is returned to the client. If it aborts, the HRESULT value is changed to CONTEXT_E_ABORTED. The client can use these two values to determine the outcome.

An object method typically notifies its client that it has forced the transaction to abort by calling **SetAbort** in one of two ways:

- Return S_OK and use an output parameter to indicate the failure.
- Return an HRESULT error code. Different codes could be used to distinguish different causes, or the generic CONTEXT_E_ABORTED error could be used.

For example, the Sample Bank application uses an output parameter to indicate failure:

```
Public Function Perform(lngPrimeAccount As Long, _
    lngSecondAccount As Long, lngAmount As Long, _
    strTranType As String, ByRef strResult As String) _
    As Long

' get our object context
Dim ctxObject As ObjectContext
Set ctxObject = GetObjectContext()

On Error GoTo ErrorHandler

' check for security
If (lngAmount > 500 Or lngAmount < -500) Then
     If Not ctxObject.IsCallerInRole("Managers") Then
          Err.Raise Number:=ERROR_NUMBER, _
          Description:="Need 'Managers' role " + _
              "for amounts over $500"
     End If
End If
.
```

```
.
.
ctxObject.SetComplete          ' we are finished and happy
Perform = 0
Exit Function

ErrorHandler:

ctxObject.SetAbort             ' we are unhappy
strResult = Err.Description    ' return the error message
Perform = -1              ' indicate that an error occured

End Function
```

It is also important to note that there are failure scenarios where the client cannot determine the transaction outcome. This situation results, for example, when a call failure occurs due to a transport error such as RPC_E_CONNECTION_TERMINATED). In such cases, it is necessary to use an application-defined protocol to determine the transaction outcome.

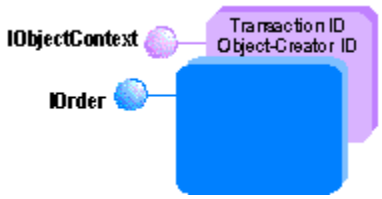
On clustered servers, MTS will not automatically reconnect to MS DTC in the event of a failover. Not enough information exists about the transaction composition and state to determine the appropriate course of action. Retries remain the responsibility of the client application. The client cannot differentiate an error caused by failover from other errors.

Resource managers are guaranteed to get transaction outcomes as part of the two-phase commit protocol managed by the Microsoft Distributed Transaction Coordinator. This feature allows resource managers to manage locks and to determine whether it is necessary to make state changes permanent or to discard them.

MTS Objects

An MTS object is an instance of an MTS component. MTS maintains context for each object. This context, which is implicitly associated with the object, contains information about the object's execution environment, such as the identity of the object's creator and, optionally, the transaction encompassing the work of the object. The object context is similar in concept to the process context that an operating system maintains for an executing program.

An MTS object and its associated context object



An MTS object and its associated context object have corresponding lifetimes. MTS creates the context before it creates the MTS object. MTS destroys the context after it destroys the MTS object.

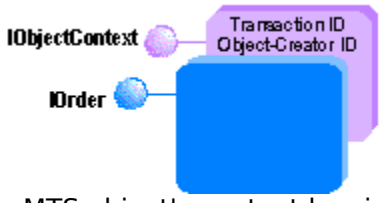
See Also

Application Components, MTS Component Requirements, Context Objects, asconCreatingTransactionServerObjects, Passing Object References, Deactivating Objects

Context Objects

Each MTS object has an associated context object. A *context object* is an extensible MTS object that provides context for the execution of an instance, including transaction, activity, and security properties. When an MTS object is created, MTS automatically creates a context object for it. When the MTS object is released, MTS automatically releases the context object.

An MTS object and its associated context object



An MTS object's context has intrinsic properties that are determined during object creation. These properties include the identity of the client that initiated the object's creation and whether or not the object executes within the scope of a transaction.

The properties established for the new object context are determined by a combination of:

- The transaction attributes of the component as specified in the MTS catalog.
- The properties of the context from which the new object is created. For example, the client's context may contain a transaction.

If your application uses Microsoft Internet Information Server (IIS), you can retrieve IIS intrinsic objects as follows:

- Using Visual Basic, by calling the **Item** method of the context object.
- Using Microsoft Visual C++ or Microsoft Visual J++, by calling the **GetProperty** method of the **IGetContextProperties** interface.

For more information on IIS intrinsic objects, see the IIS documentation.

Contexts Are Implicit

MTS maintains an implicit relationship between an MTS object and its context. This feature eliminates the need for you to pass explicitly a context object through your application.

You can access an MTS object's context by calling the **GetObjectContext** function. This function returns a reference to the **IObjectContext** interface. Resource dispensers and other context-aware services can also access the object's context. This permits the ODBC resource dispenser automatically to enlist connections on the object's transaction.

Before a method of an MTS object is dispatched for execution, that object's context becomes the current context for the thread. This context remains current as long as the object remains within the context. Calling a method in a different context causes that context to become current; the caller's context is automatically restored on return from the method.

Managing References to the Context Object

You must not pass a reference to the context object outside the object. You must explicitly release every reference that you acquire on the object context.

The context object is not available during calls to the component's class factory. This

means, for example, that a Visual C++ class implementation cannot access a context object during calls to a constructor or destructor. Objects that require access to the context object during initialization or destruction should implement **IObjectControl**. For more information, see Deactivating Objects.

See Also

MTS Objects, **GetObjectContext**, **IObjectContext**

Creating MTS Objects

You can create MTS objects by:

- Using context objects.
- Using transaction context objects.
- Using standard COM functions like **CoCreateInstance** or **CreateObject**.

Note If you are using Visual C++ and running an MTS component in-process, you must:

- Call **CoInitialize(**NULL**)** before requesting services from MTS or creating an MTS object.
- Call **CoInitializeSecurity** to initialize process-specific security. MTS security is disabled when loading an MTS object in-process.
- Call **CoUninitialize** only after you have finished using MTS or MTS objects, preferably just prior to terminating your application. You cannot call **CoInitialize** again and invoke more MTS services. Once **CoUninitialize** has been called, your application no longer executes in the MTS run-time environment.

Creating Objects Using a Context Object

You can create an MTS object by calling the **CreateInstance** method on the **IObjectContext** interface of an object's context object. The new MTS object's context inherits the activity, possibly a transaction, and all security identities from the creating object's context.

Creating an object using a context object

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.

Creating Objects Using a Transaction Context Object

If you want your base client to control transaction boundaries, use a transaction context object. You can create an MTS object by calling the **CreateInstance** method of the **ITransactionContext** interface. The new MTS object's context inherits the activity, possibly a transaction, and the identity of the initial client from the transaction context object. You can call the **Commit** method to commit an object's work and the **Abort** method to abort its work.

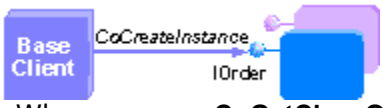
Creating an object using a transaction context object

Creating Objects Using CoCreateInstance

You can create MTS objects by using **CoCreateInstance** or any equivalent method based

on **CoGetClassObject** and **IClassFactory::CreateInstance**. While this approach should suffice for many base client applications, there are some significant limitations for the client, including the inability to control transaction boundaries. Base clients that need this additional level of control can use a transaction context object.

Instantiating an object with CoCreateInstance



When you use **CoGetClassObject** with a component that is registered to run under MTS, it returns a reference to an MTS-provided class factory. This allows MTS to participate in the client's calls to **IClassFactory**::**CreateInstance**. The MTS class factory creates the context object and then calls the component's real class factory.

For clustered servers, if you are using the **CoCreateInstanceEx** function, use the name of the virtual server containing the MSDTC resource in the *pwszName* field of the COSERVERINFO structure. (See the Microsoft Platform SDK documentation for more details about **CoCreateInstanceEx**.)

Important It is recommended that you do not call **CoCreateInstance** to create MTS objects from within MTS objects. When you do so, the new object's context cannot inherit any properties from its client's context. In particular, the new object cannot execute within the scope of its client's transaction.

Aggregation

You cannot use an MTS object as part of an aggregate of other objects. **CoCreateInstance** returns CLASS_E_NOAGGREGATION to indicate an attempt to create an MTS object with another controlling **IUnknown**.

You can, however, create an MTS object that is implemented as an aggregation of objects.

Creating Objects Using Visual Basic

You can use the following object creation methods in Microsoft Visual Basic to create MTS objects:

- The **CreateObject** function
- The **GetObject** function
- The **New** keyword (see the **Important** note for limitations)
- Automatically if the object is an Application object

Using Visual Basic object creation methods results in the same limitations as using **CoCreateInstance**. To inherit a transaction from the creating object's context, use **CreateInstance**.

Important Do not use the **New** operator, or a variable declared **As New**, to create an instance of a class that is part of the active project. In this situation, Visual Basic uses an implementation of object creation that does not use COM. To prevent this occurrence, it is recommended that you mark all objects passed out from a Visual Basic componentas **Public Creatable**, or its equivalent, and created with either the **CreateObject** function or the **CreateInstance** method of the **ObjectContext** object.

See Also

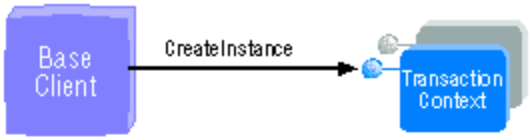
MTS Objects, Calling MTS Components, Context Objects, Passing Object References, Deactivating Objects, **CreateInstance**

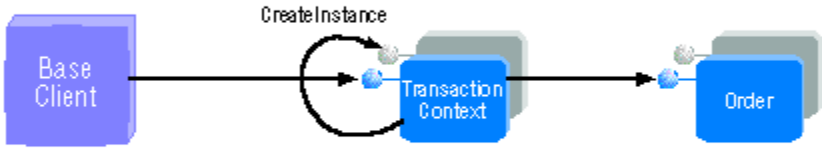
Transaction Context Objects

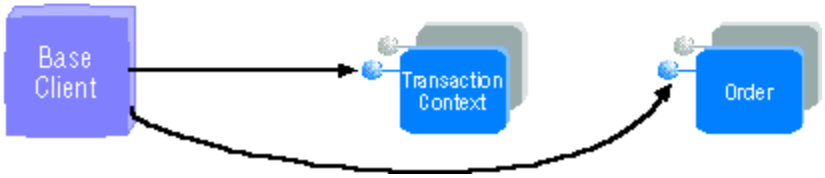
The *transaction context object* allows base clients to combine the work of multiple MTS objects into a single transaction, without having to develop a new component specifically for that purpose.

The transaction context object's methods use its context object as follows:

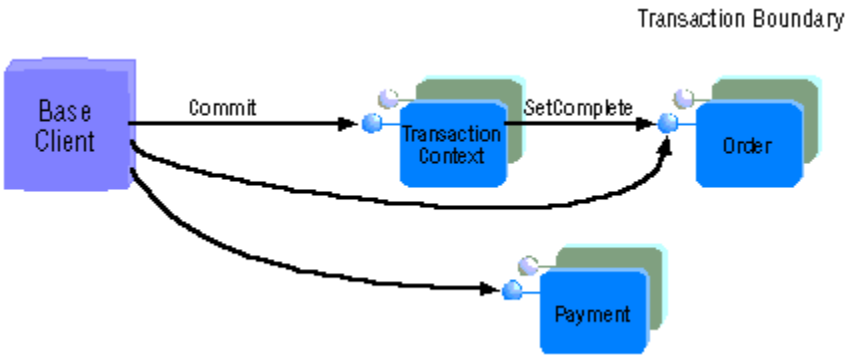
- **CreateInstance** – Calls **CreateInstance** and returns a reference to the newly created object.



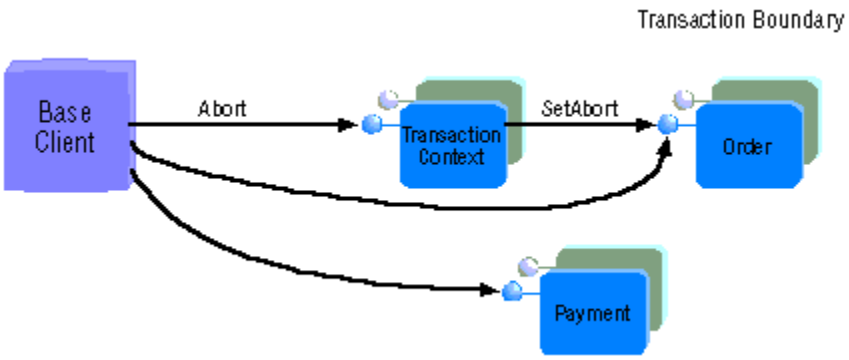




- **Commit** – Calls **SetComplete** and returns.



- **Abort** – Calls **SetAbort** and returns.



The transaction context component is defined as **Requires a New Transaction**. You cannot use the transaction context object to enlist in an existing transaction.

For example, suppose you have two components, Walk and ChewGum. Each component is defined as **Supports Transactions** and calls **SetComplete** when it is finished with its work. A base client could compose the work done by each component in a single transaction.

```
Dim objTxCtx As TransactionContext
Dim objWalk As MyApp.Walk
Dim objChewGum As MyApp.ChewGum

' Get TransactionContext
Set objTxCtx = _
    CreateObject("TxCtx.TransactionContext")

' Create instances of Walk and ChewGum
Set objWalk = _
    objTxCtx.CreateInstance("MyApp.Walk")
Set objChewGum = _
    objTxCtx.CreateInstance("MyApp.ChewGum")

' Both components do work
objWalk.Walk
objChewGum.ChewGum

' Commit the transaction
objTxCtx.Commit
```

Transaction Context Object Limitations

Note the following limitations when using a transaction context object:

- Transaction composition
- Location transparency
- Base client does not have context

Transaction Composition

When using a transaction context object, the application logic that composes the work into a single transaction is tied to a specific base client implementation and the advantages of using MTS components are lost. These implementations include:

- Ability to reuse the application logic as part of an even larger transaction
- Imposition of declarative security

- Flexibility to run the logic remotely from the client

Location Transparency

The transaction context object runs *in-process* with the base client, which means that MTS must exist on the base client computer. This may not be a problem, for example when the transaction context object is used from an Active Server Page (ASP) that is running on the same server as MTS.

Base Client Does Not Have Context

You do not get a context for the base client when you create a transaction context object. Transactional work can only be done indirectly, through MTS objects created by using the transaction context object. In particular, the base client cannot use MTS *resource dispensers* (such as ODBC) and have the work included in of the transaction. For example, developers may be familiar with the following syntax for doing transactional work on relational database systems:

```
BEGIN TRANSACTION
    DoWork
COMMIT TRANSACTION
```

Using the transaction context object in a similar way does not yield the desired result:

```
Set objTxCtx = CreateObject("TxCtx.TransactionContext")
    DoWork
    objTxCtx.Commit
Set objTxCtx = Nothing
```

The call to DoWork in this example will not be enlisted in a transaction. You must build an MTS component that calls DoWork, create an object instance of that component using the transaction context object, then call that object from the base client in order for the work to be part of the client-controlled transaction.

See Also

Transactions, **TransactionContext** Object

Passing Parameters

This topic covers the following:

- Parameter types and marshaling
- Objects as parameters
- Passing large data

Parameter Types and Marshaling

MTS object interfaces must be able to be marshaled. Marshaling interfaces allows calls across thread, process, and machine boundaries.

Standard COM marshaling is used. This means MTS object interfaces must either:

- Have method parameters which are Automation data types and be described in a type library, or
- Use custom interfaces with a MIDL-generated proxy-stub DLL.

For more information on type libraries and proxy-stub DLLs, see MTS Component Requirements.

Custom marshaling is not used. Even if a component supports the **IMarshal** interface, its **IMarshal** methods are never called by the MTS run-time environment.

VBScript Parameters

Components that are intended for use from Active Server Pages (ASPs) using Microsoft Visual Basic® Scripting Edition (VBScript) should support **IDispatch** and limit method parameter types as follows:

- **VBScript version 1.0--**Any Automation type may be passed by value, but not by reference. Method return values must be of type **Variant**.
- **VBScript version 2.0--**Same as VBScript version 1.0, except parameters of type **Variant** may now be passed by reference.

Objects as Parameters

Whether an object is passed *by value* or *by reference* is not specified by the client, but is a characteristic of the object itself. Basic COM objects can either be passed by reference or by value, depending on their implementation. If the COM object uses standard marshaling, then it is passed by reference. COM objects can also implement **IMarshal** to copy data by value. MTS objects are always passed by reference.

Additionally, the function of the object affects how it should be passed as a parameter. When deciding whether to pass objects by value or by reference, it is useful to classify the objects as follows:

- **Recordset Objects--**Encapsulate raw data, such an ADO recordset. Recordset objects are not registered as MTS objects.
- **Business Object--**Encapsulate business logic; for example, an order-processing component. Business objects should be registered as MTS objects.

The following table describes when to pass recordset objects by value or by reference:

Pass Parameter	If	Client Requirements
By value	Data is relatively small	Recipient requires all data

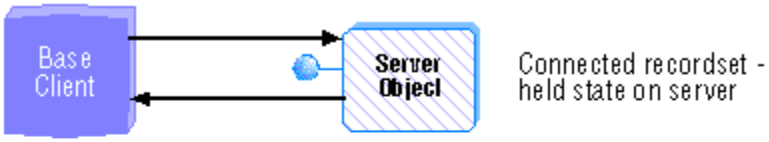
		and can get data without reaccessing caller.
By reference	Data is relatively large	Recipient does not require all data and must reaccess caller, possibly many times.

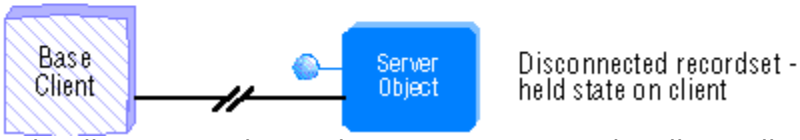
Note Whether data is "small" or "large" also depends on the speed of the connection. For example, if the component is accessed over a corporate intranet, a much larger recordset can be passed to the client in one call than in a call made by a client accessing the component on an Internet server over a modem.

Because business objects are MTS objects, they are always passed by reference.

Passing Large Data

When returning a large amount of data, consider using a Microsoft Active Data Objects (ADO) **Recordset** object. In particular, the Microsoft Advanced Data Connector (ADC) provides a recordset implementation that can be disconnected from the server and marshaled by value to the client.





The disconnected recordset moves state to the client, allowing server resources to be freed. The client can make changes to the recordset and reconnect to the server to submit updates. For more information on state, see Holding State in Objects.

Another method of packaging large amounts of data is to use *safe arrays*. For example, when using Microsoft Remote Data Objects (RDO), you can use the **rdoResultSet.GetRows** method to copy rows into an array, and then pass the array back to the client. This requires fewer calls and is more efficient than issuing **MoveNext** calls across the network for each row.

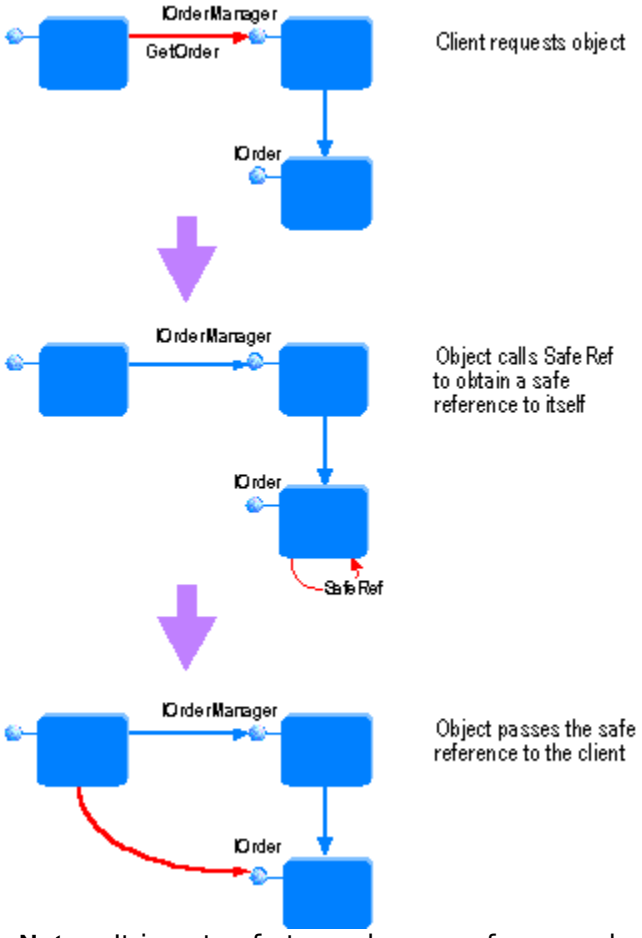
Passing Object References

You must ensure that MTS object references are only exchanged in the following ways:

- Through return from an object creation interface, such as **CoCreateInstance** (or its equivalent), **ITransactionContext::CreateInstance**, or **IObjectContext::CreateInstance**.
- Through a call to **QueryInterface**.
- Through a method that has called **SafeRef** to obtain the object reference.

An object reference that is obtained in these ways is called a *safe reference*. MTS ensures that methods invoked using safe references execute within the correct context.

Using SafeRef to pass a reference to an object



Note It is not safe to exchange references by any other means. In particular, do not pass interfaces outside the object by using global variables. These restrictions are similar to those imposed by COM for references passed between apartments.

Calls that use safe references always pass through the MTS run-time environment. This allows MTS to manage context switches and allows MTS objects to have lifetimes that are independent of client references. For more information, see Deactivating Objects.

Callbacks

It is possible to make *callbacks* to clients and to other MTS components. For example, you can have an object that creates another object. The creating object can pass a reference to itself to the created object; the created object can then use this reference to call the

creating object.

If you choose to use callbacks, note the following restrictions:

- Calling back to the base client or another package requires Access-level security on the client. Additionally, the client must be a DCOM server.
- Intervening firewalls may block calls back to the client.
- Work done on the callback executes in the environment of the object being called. It may be part of the same transaction, a different transaction, or no transaction.
- The creating object must call **SafeRef** and pass the returned reference to the created object in order to call back to itself.

See Also

SafeRef

Deactivating Objects

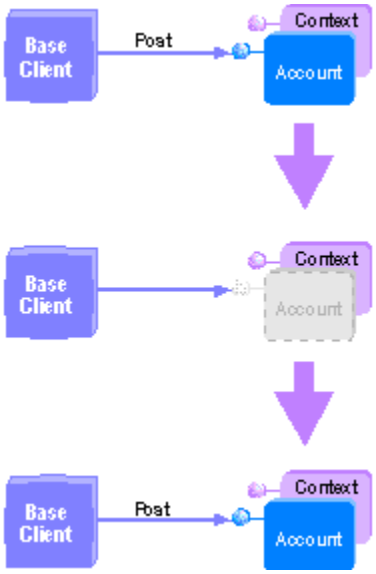
MTS extends COM to allow object deactivation, even while client references are maintained. This makes server applications more scalable by allowing server resources to be used more efficiently.

MTS objects are initially created in the deactivated state. When a client invokes a method on an object that is in a deactivated state, MTS automatically activates the object. During activation, the object is put into its initial state.

Note For Visual C++ developers, calls to **QueryInterface**, **AddRef**, or **Release** do not cause activation.

This ability for an object to be deactivated and reactivated while clients hold references to it is called *just-in-time activation*. From the client's perspective, only a single instance of the object exists from the time the client creates it to the time it is finally released. In actuality, it is possible that the object has been deactivated and reactivated many times.

Just-in-time activation



The context object exists for the entire lifetime of its MTS object, even across one or more deactivation and reactivation cycles.

Object deactivation allows clients to hold references for long periods of time with limited consumption of server resources. Consider, for example, a client application that spends 99 percent of its time between transactions. In this case, the MTS objects are activated less than 1 percent of the time.

When is an object deactivated?

An MTS object is deactivated when any of the following occurs:

- The object requests deactivation.

  An object can request deactivation by using the **IObjectContext** interface. You can use the **SetComplete** method to indicate that the object has successfully completed its work and that the internal object state doesn't need to be retained for the next call from the client. Similarly, **SetAbort** indicates that the object cannot successfully complete its work and that its state does not need to be retained.

  You can develop stateless objects by using MTS objects that deactivate on return from

every method.

- A transaction is committed or aborted.

  MTS does not allow an object to maintain private state that it acquired during a transaction. When an object's transaction is committed or aborted, the object is deactivated. Of these deactivated objects, the only ones that can continue to exist are the ones that have references from clients outside the transaction. Subsequent calls to these objects reactivate them and cause them to execute in the next transaction.

- The last client releases the object.

  This occurrence is listed here for completeness. The object is deactivated and never reactivated. The object's context is also released.

How are objects deactivated?

MTS deactivates an object by releasing all its references to the object. This causes properly developed components to destroy the object; this feature also requires the component to follow the MTS reference passing rules (see Passing Object References) and the COM reference counting rules.

Note MTS writes an Informational message to the event log when objects that do not report their reference count are deactivated.

Application components are responsible for releasing object resources on deactivation. This includes:

- Resources that are allocated with MTS resource dispensers, such as ODBC database connections.
- All other resources, including references to other objects (including MTS objects and context objects) and memory held by any instances of the component, such as using **delete this** in C++).

Doing Additional Work on Activation and Deactivation

If an object is not already activated and it supports **IObjectControl**, MTS calls the **Activate** method prior to initiating the client request. Components can use the **Activate** method to initialize objects. This is especially important for initialization that requires access to the context object. Here, keep in mind that the context is not available during calls to the component's class factory. Having access to the context object through **Activate** allows you to pass a reference to the context object to other methods; this reference can then be released in the **Deactivate** method. The **Activate** method is also useful for objects that support pooling (see Object Pooling and Recycling).

For objects that support the **IObjectControl** interface, MTS calls the **Deactivate** method when it deactivates the object. You can use this method to free resources held by the object. The **Deactivate** method is also useful for objects that support pooling. Like the **Activate** method, the **Deactivate** method has access to the object context.

See Also

Building Scalable Components, Stateful Components, Object Pooling and Recycling, **SetComplete**, **SetAbort**, **IObjectControl**

Object Pooling and Recycling

Objects that support the **IObjectControl** interface can participate in object recycling and pooling, which can increase the efficiency of activation and deactivation. After MTS calls the **Deactivate** method, it calls the **CanBePooled** method, allowing the object to be pooled for reuse. If the object returns TRUE, the object is added to the object pool. Objects that return FALSE or that do notsupport the **IObjectControl** interface are destroyed.

On activation, MTS uses an object from the pool if one is available. Only if the pool is empty will it use the component's class factory to create a new object.

Components that support object pooling must ensure that an object activated using an object from the pool is indistinguishable to the client from an object that is activated by creating a new object. Component developers must provide the appropriate code in the **Activate** and **Deactivate** method implementations to ensure this behavior.

The following table summarizes MTS run-time actions for client call processing.

	IObjectControl implemented by component	
	No	**Yes**
Step 1		
Activate the object if needed (just-in-time activation).	Use the component class factory to create an object.	Allocate an object from the pool. If pooling is not supported or the pool is empty, then use the component class factory to create an object. Call the object's **Activate** method.
Step 2		
Execute the call.	Call the object method.	Call the object method.
Step 3		
Deactivate the object if requested (**SetComplete** or **SetAbort** called before return).		Call object's **Deactivate** method
	Release the last reference held by the MTS run-time environment.	If the system supports pooling, then call **CanBePooled**. If it returns TRUE, then add the object to the pool. Otherwise, release the last reference held by the MTS run-time environment.

Important Object pooling and recycling is not available in this release. MTS calls **CanBePooled** as described here, but no pooling takes place. This forward-compatibility encourages developers to use **CanBePooled** in their applications now in order to benefit from a future release without having to modify their applications later.

See Also

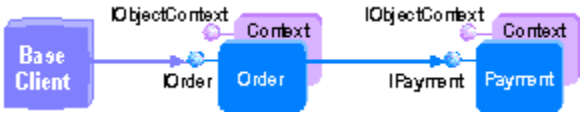
Deactivating Objects, **IObjectControl**

MTS Clients

An application or object that uses an MTS object is referred to as a *client* of the object. It is important to understand that this is a relative term, and describes a *relationship* with a specific object. For example, when MTS object A uses MTS object B, the object A, while still an object, is also a client.

Clients that run outside the direct control of the MTS run-time environment are referred to as *base clients*.

Clients and base clients: The Order object is a client of the Payment object



See Also

Base Clients, Base Clients vs. MTS Components

Base Clients

Base clients are the primary consumers of MTS objects, although base clients execute outside of the MTS run-time environment. One common role of a base client is to provide the application's user interface, and to map the end-user's requests to the business functions exposed by the MTS components.

You can create base clients with a variety of programming languages, including Microsoft Visual C++, Microsoft Visual Basic, Microsoft Visual J++, COBOL, and even Transact SQL. Base client programs can execute in a variety of environments, from application processes to general system services such as Microsoft Internet Information Server (IIS) or SQL Server. In fact, you can use any programming environment in which you can create COM objects and invoke methods on them.

See Also

MTS Clients, Base Clients vs. MTS Components

Base Clients vs. MTS Components

The following table contrasts MTS components with base client applications.

MTS components	Base clients
MTS components are contained in COM dynamic-link libraries (DLLs); MTS loads DLLs into processes on demand.	Base clients can be written as executable files (.exe) or dynamic-link libraries (.dll); MTS is not involved in their initiation or loading.
MTS manages server processes that host MTS components.	MTS does not manage base client processes.
MTS creates and manages the threads used by components.	MTS does not create or manage the threads used by base client applications.
Every MTS object has an associated context object. MTS automatically creates, manages, and releases context objects.	Base clients do not have implicit context objects. They can use transaction context objects, but they must explicitly create, manage, and release them.
MTS objects can use resource dispensers. Resource dispensers have access to the context object, allowing acquired resources to be automatically associated with the context.	Base clients cannot use resource dispensers.

See Also

MTS Clients, Application Components

Activities

An *activity* is a set of objects executing on behalf of a base client application. Every MTS object belongs to one activity. This is an intrinsic property of the object and is recorded in the object's context. The association between an object and an activity cannot be changed. An activity includes the MTS object created by the base client, as well as any MTS objects created by that object and its descendants. These objects can be distributed across one or more processes, executing on one or more computers.

For example, an online banking application may have an MTS object dispatch credit and debit requests to various accounts, each represented by a different object. This dispatch object may use other objects as well, such as a receipt object to record the transaction. This results in several MTS objects that are either directly or indirectly under the control of the base client. These objects all belong to the same activity.

MTS tracks the flow of execution through each activity, preventing inadvertent parallelism from corrupting the application state. This feature results in a single logical thread of execution throughout a potentially distributed collection of objects. By having one logical thread, applications are significantly easier to write.

Whenever you use **CoCreateInstance** or its equivalent to create an MTS object, a new activity is created; note that this includes a base client creating a transaction context object.

When an MTS object is created from an existing context, using either a transaction context object or an MTS object context, the new object becomes a member of the same activity. In other words, the new context inherits the activity identifier of the context used to create it.

MTS only allows a single logical thread of execution within an activity. This is similar in behavior to a COM apartment, except that the objects can be distributed across multiple processes. When a base client calls into an activity, all other requests for work in the activity (such as from another client thread) are blocked until after the initial thread of execution returns back to the client.

Callbacks and Reentrancy

While MTS does not allow multiple threads of execution within an MTS object, reentrancy is possible via callbacks. Suppose you have an object that creates another object. If the creating object passes a reference to itself to the created object, either directly or indirectly, cycles can occur in the call graph. MTS objects that do this must be prepared to receive a method invocation while blocked waiting for a call to complete. MTS ensures that the incoming call belongs to the same logical thread by using the COM logical thread identifier. COM uses the logical thread identifier for a similar purpose in apartment objects.

Limitation

While no parallel execution can exist within the activity on any individual computer, the MTS run-time environment does not protect against clients entering into the same activity through objects on two different computers. This can result in two parallel threads of execution on different computers. However, if the thread of execution on one computer calls an object in the same activity on the other, the call will be blocked.

This behavior is based on the belief that the cost outweighs the benefits of providing fully distributed activity protection, both in terms of development and run-time performance.

See Also

Components and Threading, Passing Object References

Components and Threading

The MTS run-time environment manages threads for you. MTS components need not, and, in fact, should not, create threads. Components must never terminate a thread that calls into a DLL.

Every MTS component has a **ThreadingModel** Registry attribute, which you can specify when you develop the component. This attribute determines how the component's objects are assigned to threads for method execution. You can view the threading-model attribute in the MTS Explorer by clicking the Property view in the Components folder. The possible values are Single, Apartment, and Both.

Single-Threaded Components

All objects of a single-threaded component execute on the main thread. This is compatible with the default COM threading model, which is used for components that do not have a **ThreadingModel** Registry attribute.

The main threading model provides compatibility with COM components that are not reentrant. Because objects always execute on the main thread, method execution is serialized across all objects in the component. In fact, method execution is serialized across all components in a process that uses this policy. This allows components to use libraries that are not reentrant, but it has very limited scalability.

Limitations for Single-Threaded Components

Single-threaded, stateful components are prone to deadlocks. You can eliminate this problem by using stateless objects and calling **SetComplete** before returning from any method.

The following scenario describes how such a deadlock can occur. Suppose you have a single-threaded Account component, which is written to be both transactional and stateful. The following scenario describes how two clients, Client 1 and Client 2, could call objects in a way that causes an application deadlock:

- Client 1 creates Account object A and calls it to update an account record. The database update is done under object A's transaction (Transaction 1). Because the object does not call **SetComplete** before returning to the client, Transaction 1 remains active.
- Next, Client 2 creates Account object B and calls it to update the same account. Because its work is done under a different transaction (Transaction 2), the attempt to update the account record will block while waiting for Transaction 1 to complete.
- Client 1 makes another call to object A, for instance, to have it make another change to the account record and complete the transaction. However, the call must wait for the main thread, which is still busy servicing the call from Client 2.
- The two clients are now deadlocked. Client 1 holds a lock on the account record, while waiting for the server's main thread so that it can complete Transaction 1. Client 2 holds the server's main thread, while waiting for Transaction 1 to complete so that it can update the account record.

Note that this is not a deadlock from the SQL perspective because A's and B's connections are in different transactions.

Apartment-Threaded Components

Each object of an apartment-threaded component is assigned to a thread its *apartment*, for the life of the object; however, multiple threads can be used for multiple objects. This is a standard COM

concurrency model. Each apartment is tied to a specific thread and has a Windows message pump.

The apartment threading model provides significant concurrency improvements over the main threading model. Activities determine apartment boundaries; two objects can execute concurrently as long as they are in different activities. These objects may be in the same component or in different components.

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.

See Also

Activities

Programmatic Security

The MTS security model consists of declarative security and programmatic security. Developers can build both declarative and programmatic security into their components prior to deploying them on a Windows NT security domain.

Important Security is not supported on Windows 98. Note the following application behavior when running MTS on Windows 98:

- All identities are mapped to "Windows 98".
- Role configuration is not supported.
- Checking roles always return success.

Roles are central to the MTS security model. A role is an abstraction that defines a logical group of users. At development time, you use roles to define declarative authorization and programmatic security logic. At deployment time, you bind these roles to specific groups and users.

You can administer package security with the MTS Explorer. This is a form of declarative security, which does no't require any component programming is based on standard Windows NT security.

MTS also allows component applications to implement additional access control programmatically. MTS security is integrated with DCOM and Windows NT security. See the Microsoft Platform SDK for further information on COM security.

See Also

Basic Security Methods, Advanced Security Methods

Basic Security Methods

The **IObjectContext** interface provides two methods for basic programmatic security:

- **IsCallerInRole**
- **IsSecurityEnabled**

The key to understanding how MTS security works is to understand roles, as discussed in the following sections.

Roles from a Development Perspective

A role is a symbolic name that defines a logical group of users for a package of components. For example, an online banking application might define roles for Manager and Teller.

You can define authorization for each component and component interface by assigning roles. For example, in the online banking application, only the Manager may be authorized to perform bank transactions above a certain amount of money.

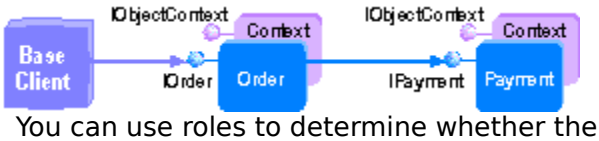
Roles are defined during application development. These roles are then assigned to specific users at deployment time.

Important Roles on a dual interface are not enforced when **IDispatch** (late-binding) is used.

Checking If a Caller Is in a Role

The **IsCallerInRole** method determines if a caller is assigned to a role. The caller is the direct caller, which is the identity of the process (base client or server process) calling into the current server process.

The following illustration shows an application used to order supplies for a business.



You can use roles to determine whether the base client has access to objects in the server process. In this scenario, the server process would check to see if the base client is allowed to place an order. Calling **IsCallerInRole** on the Order object context checks if the direct caller, which is in this case the base client, is in a given role. Such a role might be Purchaser, to restrict the placing of orders to employees within that role.

Security checks are made when a process boundary is crossed. If the Payment object accesses a database, the access rights to the database are derived from the identity of the server process, not the base client. The database would use its own proprietary authorization checking.

Server-process security does not use impersonation. **IsCallerInRole** has the same semantics regardless of how many calls have taken place within the server process. The identity of the direct caller is always used to make the check. For more information on impersonation, see Advanced Security Methods.

Security for In-Process Components

Because the level of trust is process-wide, running MTS components in-process is not recommended for secure applications. Access checks are not made on calls between components in the same server process. Configuring MTS components to run in-process with the base client gives the base client access to all components within that server

process.

The **IsSecurityEnabled** method determines if security checking is enabled. This method returns FALSE when running in-process. **IsSecurityEnabled** can be a useful check to make before using **IsCallerInRole**. **IsCallerInRole** will always return TRUE when called on an object that is running in-process, which may have unintended effects.

When an MTS component is part of a Library package (in-process), it effectively becomes part of the hosting Server package that creates it. If you create Library packages with components that call **IsCallerInRole**, you should instruct installers of your Library packages to define the Library package's roles on the hosting Server package. Otherwise, **IsCallerInRole** will always fail.

See Also

IsCallerInRole, **IsSecurityEnabled**, Secured Components

Advanced Security Methods

MTS objects can use the **ISecurityProperty** interface to obtain security-related information from the object context, including the identity of the client that created the object, as well as the identity of the current calling client. Applications can use this information to implement custom access control, such as using the Win32 security interfaces.

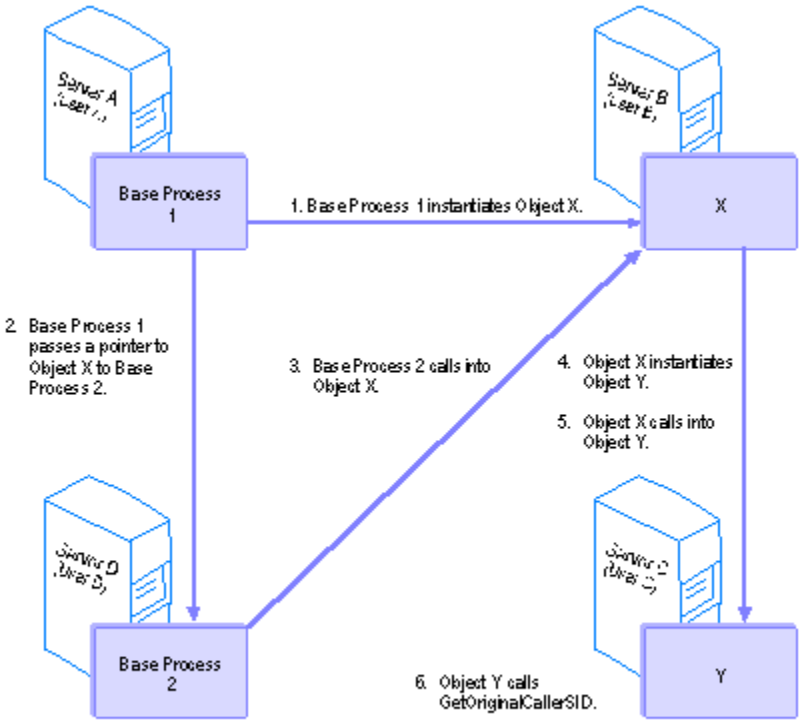
Note Visual Basic programmers can use the **SecurityProperty** object. The methods for **SecurityProperty** return user name strings instead of security identifiers (SIDs).

Security Identifiers (SIDs)

A Windows NT security identifier (SID) is a unique value that identifies a user or group. You can use SIDs to determine the exact identity of a user. Because of their uniqueness, SIDs do not have the flexibility of roles.

Callers and Creators

The following figure shows which SIDs are returned by the various methods on **ISecurityProperty** after a certain sequence of method calls.

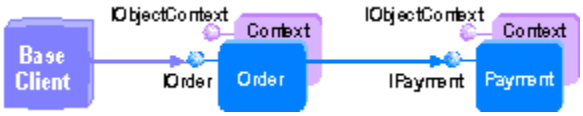


Calling the following methods on Object Y returns SIDs associated with these users:

- **GetDirectCallerSID** returns the SID associated with User B.
- **GetDirectCreatorSID** returns the SID associated with User B.
- **GetOriginalCallerSID** returns the SID associated with User D.
- **GetOriginalCreatorSID** returns the SID associated with User A.

Impersonation

Impersonation allows a thread to execute in a security context different from that of the process that owns the thread. Consider the following application scenario.



Basic Security Methods described an order-entry scenario in which the base client represents an employee submitting an order. In this scenario, the client is not authorized to use the Payment object and its associated database directly.

Suppose the base client were a report writer for an accounting program. In this case, you want to allow access to the Payment object's database. One way to accomplish this is for the Order object to impersonate the base client, allowing the database to use its own security checking to determine access privileges.

MTS does not promote the use of impersonation, but encourages role-based security. Security is simplified by the single-level of authorization provided by MTS, whereas the impersonation model has an *n*-level authorization architecture. The report-writer scenario can be simplified by defining a role, such as Accountant, to allow access to the database.

Error Handling

Fault Isolation and Failfast

MTS performs extensive internal integrity and consistency checks. If MTS encounters an unexpected internal error condition, it immediately terminates the process. This policy, called failfast, facilitates fault containment and results in more reliable and robust systems.

Consider a case in which MTS detects that one of its data structures is in a corrupted state. At this point, both the cause and the magnitude of the corruption are unknown. Unfortunately, MTS cannot tell how far the damage has spread. Certainly MTS is in an indeterminate state. But it does not run in isolation. Like other DLLs, it is hosted in a process environment and shares a single address space with the main program executable and many other DLLs. Consequently, it is safe to assume that the entire process has been corrupted. The process is immediately terminated to prevent it from spreading potentially corrupted information to other processes or, worse yet, from allowing corrupted data to be committed and made durable.

As a developer or administrator, you should inspect the Windows NT Event Viewer Application Log for details on any failfast or serious application errors.

Exceptions in MTS Objects

MTS does not allow exceptions to propagate outside of a context. If an exception occurs while executing within an MTS context and the application doesn't catch the exception before returning from the context, MTS catches the exception and terminates the process. Using the failfast policy in this case is based on the assumption that the exception has put the process into an indeterminate state--it is not safe to continue processing.

MTS Object Method Error Return Codes

MTS never changes the value of an HRESULT error code, such as E_UNEXPECTED or E_FAIL, returned by an MTS object method.

When an MTS object returns an HRESULT status code, such as S_OK or S_FALSE, MTS may convert the status code into an MTS error code before it returns to the caller. This occurs, for example, when the application returns S_OK after calling **SetComplete**; if the object is the root of an automatic transaction that fails to commit, the HRESULT is converted to CONTEXT_E_ABORTED.

When MTS converts a status code to an error code, it clears all of the method's output parameters. Returned references are released and the values of the returned object pointers are set to NULL.

See Also

MTS Error Diagnosis, MTS Error Codes

Developing Applications for MTS

Building MTS Applications

Explains the key concepts that a component application developer needs to understand for developing Microsoft Transaction Server (MTS) applications.

Creating a Simple ActiveX Component

Demonstrates how to create a component and register the component in the Microsoft Transaction Server run-time environment.

Building Scalable Components

Demonstrates how to use just-in-time activation to use server resources efficiently, resulting in more scalable applications and improved performance.

Building Transactional Components

Introduces transactional components and the benefits of running components within the same transaction.

Sharing State

Demonstrates how to use the Shared Property Manager to share state among multiple Transaction Server objects running in the same process.

Stateful Components

Discusses stateful components and outlines some of the issues associated with writing stateful application components.

Multiple Transactions

Explains the benefits of distributing work among multiple transactions.

Secured Components

Shows how to use Microsoft Transaction Server's security features to restrict the use of application features to designated users.

Building MTS Applications

This topic contains information that a component application developer needs to understand before building Microsoft Transaction Server (MTS) applications.

MTS Component Requirements

Business Logic in MTS Components

Packaging MTS Components

Calling MTS Components

Holding State in Objects

Database Access Interfaces with MTS

Developing MTS Components with Java

Debugging MTS Components

Automating MTS Deployment

MTS Error Diagnosis

MTS Component Requirements

An MTS component is a type of COM component that executes in the MTS run-time environment. In addition to the COM requirements, MTS requires that the component must be a dynamic-link library (DLL). Components that are implemented as executable files (.exe files) cannot execute in the MTS run-time environment. For example, if you build a Remote Automation server executable file with Microsoft Visual Basic, you must rebuild it as a DLL.

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

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.

Additional Requirements for Visual C++ Components

- The component must have a standard class factory.

  The component DLL must implement and export the standard DllGetClassObject function and support the IClassFactory interface. MTS uses this interface to create objects. IClassFactory::CreateInstance must return a unique instance of an MTS object.
- The component must only export interfaces that use standard marshaling. For more information, see Passing Parameters.
- All component interfaces and coclasses must be described by a type library. The information in the type library is used by the MTS Explorer to extract information about the installed components.
- For custom interfaces that cannot be marshaled using standard Automation support, you must build the proxy-stub DLL with MIDL version 3.00.44 or later (provided with in the Microsoft Platform SDK for Windows NT version 4.0); use the -Oicf and /MD compiler options; and link the DLL with the mtxih.lib, LE32.lib, and ADVAPI32.lib libraries provided by MTS. The mtxih.lib library must be the first file that you link into your proxy-stub DLL. If the component has both a type library and a proxy-stub DLL, MTS will use the proxy-stub DLL.
- The component must export the DllRegisterServer function and perform self-registration of its CLSID, ProgID, interfaces, and type library in this routine.

Development tools such as Visual Basic and the Active Template Library, which is available with Microsoft Visual C++, allow you to generate interfaces that COM can marshal automatically. These interfaces, known as dual interfaces, are derived from IDispatch and use the built-in Automation marshaling support.

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.

Registering MTS Components

You manage MTS components by using the MTS Explorer. Before a component can run with context in the MTS run-time environment, you must use the MTS Explorer to define the component in the MTS catalog. In addition to keeping track of a component's basic COM attributes, such as the name of the implementation DLL, the MTS catalog maintains a set of MTS-specific attributes. MTS uses these attributes to provide capabilities in addition to those provided by COM. For example, the transaction attribute controls the transactional characteristics of a component.

The MTS Explorer assigns components to a package that controls the assignment of components to server processes and control client access to components.

Note MTS allows only a single server process associated with a given package to run on a computer at a time. MTS writes a warning event message to the log if you attempt to start a second instance of an already active package. However, COM does not explicitly disallow multiple servers running the same COM classes. MTS writes a warning message to the log in the event that two threads try to start the package at the same time. This event is especially likely on a symmetric multiprocessing (SMP) computer where the two package invocations are concurrent. In these cases, MTS enforces a rule of one server process for each package by terminating one of the extra packages. COM then connects to the one server process still running and successfully returns.

Registration for Components Developed with Visual Basic

Whenever you recompile a COM DLL (or "ActiveX DLL") project in Visual Basic, 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, 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 works with Visual Basic versions 4.0, 5.0 and 6.0. 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 Binary Compatibility setting in the Visual Basic Project Properties 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, 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 without selecting the Binary Compatibility 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.

Running COM Components Under MTS

Exercise caution when registering a standard COM component (one developed without regard to MTS) to execute under MTS control.

First, ensure that references are safely passed between contexts (see Passing Object References).

Second, if the component uses other components, consider running them under MTS. Rewrite the code for creating objects in these components to use **CreateInstance** (see Creating MTS Objects).

Third, you can effectively use automatic transactions only with components that indicate the completion of their work by calling either the **SetComplete** or **SetAbort** methods. If a component does not use these methods, an automatic transaction can only be completed when the client releases the object. MTS will attempt to commit the transaction, but there is no way for the client to determine whether the transaction has been committed or aborted. Therefore, it is recommended that you do not register components as Requires a **transaction** or Requires a new **transaction** unless they use **SetComplete** and **SetAbort**.

Including Multiple Components in DLLs

You can implement multiple components in the same DLL. The MTS Explorer allows components from the same DLL to be installed in separate packages.

Including Type Libraries and Proxy-Stub DLLs in MTS Components

Development tools supporting ActiveX components can merge your type library or proxy-stub DLL with your implementation DLL. If you do not want to distribute your implementation DLL to client computers, keep your type libraries and proxy-stub DLLs separate from your implementation DLLs. The client only needs a type library or custom proxy-stub DLL to use your server application remotely.

Business Logic in MTS Components

This topic describes how to enact business logic in MTS components.

Granularity is determined by the number of tasks performed by a component. The granularity of a component affects the performance, debugging, and reusability of your MTS components. A fine-grained component performs a single task, such as calculating tax on a sales order. Fine-grained components consume and release resources quickly after completing a task. A component that enacts a single business rule can facilitate testing packages, because isolating individual tasks in components makes testing your applications easier. In addition, fine-grained components are easily reused in other packages. In the following example, a component performs a single task: adding a customer record to the database.

```
Function Update(ByVal strEmail As String, _
ByVal bNewCust As Boolean, ByVal strContact As String,_
ByVal strPhoneNumber As String, _
ByVal strNightPhoneNumber As String)

    Dim ctxObject As ObjectContext
    Set ctxObject = GetObjectContext

    On Error GoTo ErrorHandler

' Code accesses the customer row from the database.
' Customer information is updated with information
' that was passed in.
'
    ctxObject.SetComplete

    Exit Function
```

This simple component uses system resources efficiently (passing parameters by value), is easy to debug (single function), and also reusable in any other application that maintains customer data.

A coarse-grained component performs multiple tasks. Coarse-grained components are generally harder to debug and reuse in applications. For example, a PlaceOrder component might add a new order, update inventory, and update customer information. PlaceOrder is a more coarsely grained component because it performs more "work" by adding, updating, and deleting customer, inventory and order information.

For more information about components' shared resources, see Holding State in Objects.

Packaging MTS Components

This document describes how you should package your MTS components. Consider the following design issues when defining package boundaries:

- Activation
- Shared resources
- Fault isolation
- Security isolation

Activation

You can select either of the following Activation levels for your packages:

- Library (running within the same process as the client that creates the object)
- Server (on the same computer but in a different process)

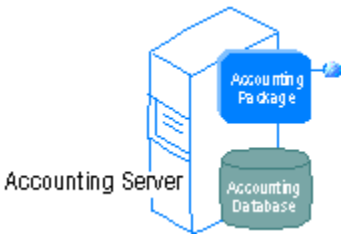
MTS provides a way to set up remote components by using the Remote Computer and Remote Component folders in the MTS Explorer hierarchy. For more information about "pulling" or "pushing" components between computers, see the Administrator's Guide.

By default, components run in a server process on the local computer. If you run your components within the MTS server process, you enable resource sharing, security, and easier administration by using the MTS Explorer for your component. Running components in-process provides an immediate performance benefit, because you do not have to marshal parameters cross-process. However, in-process components do not support declarative security and you lose fault isolation.

Sidebar: In-process Components and Security

Note that in-process components do not support declarative security or offer the benefits of process isolation. In-process components will run in any process that creates the component. Role checking is disabled between in-process components because IsCallerInRole returns True. In other words, the direct caller always passes the authorization check.

Also, it is recommended that you place your components as close as possible to the data source. If you are building a distributed application with a number of packages running on local and remote servers, try to group your components according to the location of your data. For example, in the following figure, the Accounting server hosts an Accounting package and Accounting database.



Shared Resources

Sharing resources in a multiuser environment results in faster applications that scale more easily. Note that only components marked with the Local activation setting can share resources. Package your components to take advantage of the resource sharing and pooling that MTS provides for your application.

Pool your resources by server process.Note that MTS runs each hosted package in a separate server process. The fewer pools you have running on your server, the more efficiently you pool

resources, so try to group components that share "expensive" resources, such as connections to a specific database. If you reuse the expensive resources within your package, you can greatly improve the performance and scaling of your application. For example, if you have a database lookup and a database update component running in a customer maintenance application, package those components together so that they can share database connections.

Fault Isolation

Fault isolation requires separating components into packages that can operate in their own server process. Components in the same package share the same server process if all the activation settings are the same. By placing components in separate packages, you can mitigate the impact of a component failure because each package runs in a separate server process.

You can also use fault isolation to test new components. You can stage updates to MTS applications by introducing new components. Fault isolation for packages greatly reduces the risk of your local server package failing when you introduce a new component to a shared environment.

Security Isolation

MTS security roles represent a logical group of user accounts which are mapped to Microsoft Windows NT® domain users and groups during the deployment of the package. You can use the MTS Explorer to define declarative authorization checking by applying roles to components and component interfaces. Applying a security role to a component defines access privileges for any user assigned as a member of that security role. Users not assigned to a role with access privileges to a package cannot use the package. Because security authorization occurs between packages rather than between components within a package, it is recommended that you consider the MTS security model when determining your package boundaries. Note that security isolation only applies to packages with components running under the Server activation setting.

Security authorization is checked when a method call crosses a package boundary, such as when a client calls into a package or one package calls another. When you package your components, make sure you group components that can safely call each other without requiring security checks within one package.

All components within a package run under the identity established for the package. If you run under different identities, separate them into two different packages.

You can use declarative security between the client and server, and database security based on package identity between the server and data source. You can restrict access to a data source by assigning an identity to a package, and configuring the database to accept updates according to package identity.

If you use package identity to set up your database security, the database recognizes the package identity as a single user. If database access occurs under an identity set by the package, the database connection set up for the package identity name can be used by all the users mapped to role or roles for that package. This kind of resource sharing improves application performance and scalability.

Calling MTS Components

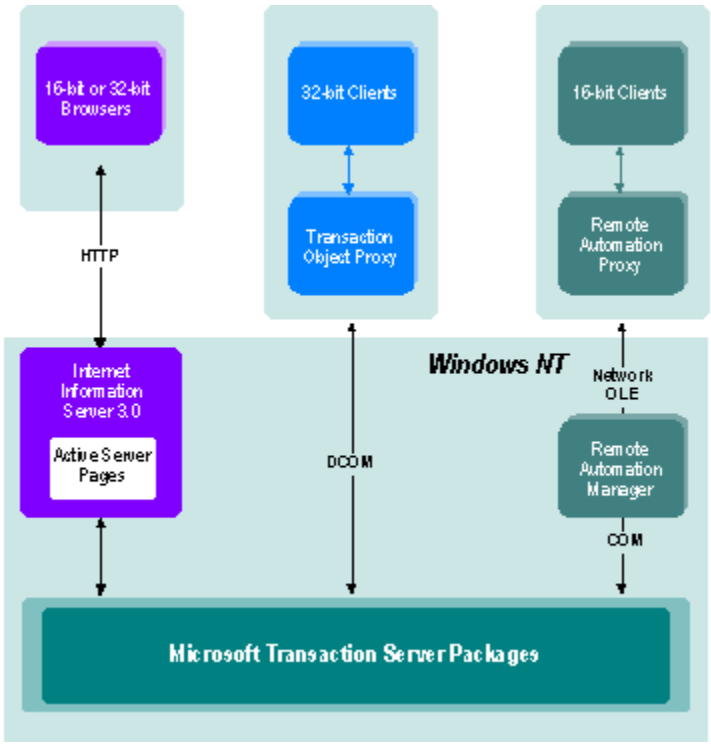
This topic covers the following:

- Calling MTS Components using DCOM
- Calling MTS Components from an Active Server Page
- Calling MTS Components from a Web Browser-Resident Component

For more information on the methods for creating MTS objects, see Creating MTS Objects.

Calling MTS Components from a Client Application

MTS components can be located on a separate computer from the client. A client can call a remote MTS component using DCOM, HTTP, or Remote Automation. To run an MTS component on the client computer, the client computer must have MTS installed.



Calling MTS Components through DCOM

DCOM is the standard transport for calling MTS components. To enable DCOM calls to MTS components, you must configure the following:

Client Registry Settings -- The easiest way to configure your client application to call a remote MTS component is to use the application executable utility, which automatically configures client registry settings. For more information, see the Administrator's Guide.

DCOM Security Settings -- You may have to configure the Impersonation Level and Authentication Level on both client and server computers. MTS works properly using the default values for these settings: Identify for Impersonation Level and Connect for Authentication Level. Make the necessary changes in the MTS Explorer at the package level. Changing default settings by using the DCOM configuration utility (dcomcnfg.exe) is not recommended.

Windows 98 includes DCOM support. However, if you want to use Microsoft Windows 95 clients with MTS, install DCOM for Windows® 95. For the latest information on DCOM support for Windows 95,

see http://www.microsoft.com/oledev on the World Wide Web.

Calling MTS Components through Remote Automation

Remote Automation was introduced with Visual Basic version 4.0, before the introduction of DCOM. It is useful for 16-bit clients, because DCOM works only in 32-bit environments. To use Remote Automation with MTS, the Remote Automation Manager (RACMAN) must be running on the server where the MTS components are installed. For more information, see the Visual Basic documentation.

Note You cannot use MTS security Remote Automation since all calls are made using the RACMAN identity. Because RACMAN does not impersonate when calling the components on the server, the client identity cannot be determined.

Calling MTS Components through HTTP

There are two ways a client can call an MTS component through HTTP:

- Call an Active Server Page (ASP), which in turn calls the MTS component using DCOM.
- Call the MTS component from a Web browser - resident component using the ActiveX Data Objects (ADO) Remote Data Service (RDS), which in turn uses HTTP. For more information about RDS, see http://www.microsoft.com/adc.

Calling MTS Components from an Active Server Page

You can call MTS components from Active Server Pages (ASPs). You can create an MTS object from an ASP by calling Server.CreateObject. Note that if the MTS component has implemented the OnStartPage and OnEndPage methods, the OnStartPage method is called at this time.

You can run your MTS components in-process with or out-of-process with Internet Information Server (IIS). If you run your MTS components in-process with IIS, be aware that if MTS encounters an unexpected internal error condition or an unhandled application error such as a general-protection fault inside a component method call, it immediately results in a failfast, thus terminating the process and IIS.

By default, IIS 3.0 disables calling out-of-process components. To enable calling out-of-process components, modify the following registry entry

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\ASP\Parameters

by setting the AllowOutOfProcCmpnts key to 1.

Calling MTS Components from a Web Browser-Resident Component

You can call an MTS component from a Web browser - resident component. Use the application executable utility to configure that client, and then use the HTML **<OBJECT>** tag to call that component. You can also use the **<OBJECT>** tag to create an MTS object in-process with the browser client. Remember that MTS must be installed on the client computer for an MTS component to run in-process.

The component should be made safe for scripting, either through a component category entry in the registry, or by supporting the IObjectSafety interface.

Remote Data Service (RDS) also allows you to create web browser - resident components using the **<OBJECT>** tag. RDS supports the following:

- HTTP
- HTTPS (HTTP over Secure Socket Layer)

- DCOM
- In-process server

Except for in-process objects, the CreateObject method of the DataSpace object creates a proxy for the MTS object that runs in a local or remote server process.

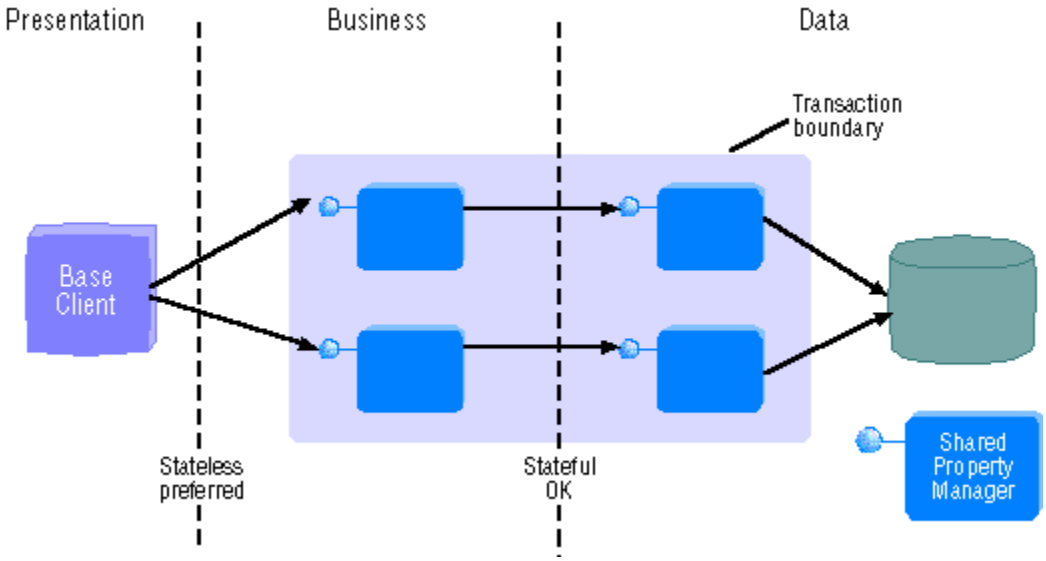
You must configure the following registry key to the Prog ID of the object that you want to call:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\ADCLaunch

Holding State in Objects

Although there are many benefits to using stateless MTS objects, there are cases where holding state is desirable. This topic provides some guidelines in deciding where state is held in your application.

The following diagram shows a three-tier architecture:



Typically, the latency between tiers differs greatly. Calls between the presentation tier and business tier are often an order of magnitude slower than calls between the business tier and data tier. As a result, held state is more costly when calling into the business tier.

However, it often makes sense to hold state within the transaction boundary itself. For example, the objects in the data tier may represent a complex join across many tables in separate databases. Reconstructing the data object state is potentially more inefficient than the cost of the resources held by those objects while they remain active.

Since objects lose state on transaction boundaries, if you need to hold state across transactions, use the Shared Property Manager or store the state in a database.

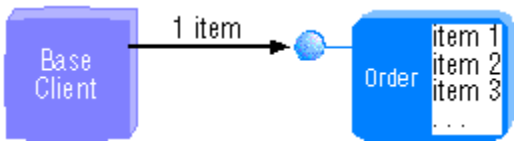
Example: Order-Entry Application

There are two separate issues when considering the effects of holding state in an application:

- Network roundtrips -- More frequent network roundtrips and slower connections extend the lifetime of the called MTS object.
- Held resources -- Holding state often means holding onto a resource, such as a database connection, and potentially, locks on the database.

Consider the example of an online shopping application. The client chooses items from a catalog and submits an order. Order processing is handled by a business object, which in turn stores the order in a database (not shown).

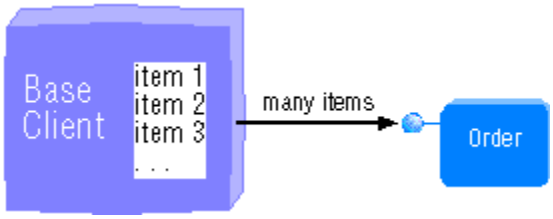
One way of building the application is for the client to call an **Order** object, with each call adding or removing an item from the order:



This application has the following properties:

- Client maintains no state.
- Server maintains state across multiple calls.
- Many network roundtrips.
- High contention for resources. The database connection is held for the lifetime of the **Order** object. This is not a very scalable solution.

You can require that the client cache the items in an array or recordset:



This application has the following properties:

- Stateful client.
- Server is virtually stateless with one call to the server.
- Fewer network roundtrips.
- Less contention for resources. This is a scalable solution.

Concurrency

In addition to network bandwidth and resources, concurrency affects application performance. There are two types of concurrency:

- Pessimistic -- As soon as editing begins, the database locks the records being changed. The records are unlocked when all changes are complete. No two users can access the same record at the same time.
- Optimistic -- The database locks the records being changed only when the changes are committed. Two users can access the same record at the same time, and the database must be able to reconcile, or simply reject, changed records that have been edited by multiple users prior to commit.

Implementing a server cache implies optimistic concurrency. The server does not have to hold locks on the database, thus freeing resources.

However, if there is high contention for the resource, pessimistic concurrency may be preferred. It is easier to reject a request to access a database and have the server try again than it is to reconcile cached, out-of-date data with a rapidly changing database.

Database Access Interfaces with MTS

This topic describes the database access interface options for MTS applications. You can use the Open Database Connectivity (ODBC) Application Programming Interface (API) to access a resource manager (which is a system service that manages durable data), or a data access model that functions over the ODBC layer. Because the ODBC version 3.0 Driver Manager is an MTS resource dispenser, data accessed via ODBC is automatically protected by your object's transaction. For object transactions, an ODBC-compliant database must support the following:

- The database's ODBC driver must be thread safe. It also must be able to connect to the driver from one thread, use the connection from another thread, and disconnect from another thread.
- If ODBC is used from within a transactional component, then the ODBC driver must also support the SQL_ATTR_ENLIST_IN_DTC connection attribute. This is how the ODBC Driver Manager asks the ODBC driver to enlist a connection on a transaction. You can make your component transactional by setting the transaction property for your component in the MTS Explorer. If you are using a database without a resource dispenser that can recognize MTS transactions, contact your database vendor to obtain the required support.

The following table summarizes database requirements for full MTS support.

Requirements	Description	Resources (if applicable)
Support for the OLE transactions specification, or support for XA protocol	Enables direct interaction with Distributed Transaction Coordinator (DTC). Use the XA Mapper to interact with DTC	MTS SDK in the Microsoft Platform SDK.
ODBC driver	Platform requirement for MTS server components	ODBC version 3.0 SDK
ODBC driver support for the ODBC version 3.0 SetConnectAttr SQL_ATTR_ENLIST_ IN_DTC call.	MTS uses this call to pass the transaction identifier to the ODBC driver. The ODBC driver then passes the transaction identifier to the database engine.	ODBC version 3.0 SDK
Fully thread-safe ODBC driver	ODBC driver must be able to handle concurrent calls from any thread at any time.	ODBC version 3.0 SDK
ODBC driver must not require thread affinity	ODBC driver must be able to connect to the driver from one thread, use the connection from another thread, and disconnect from another thread.	ODBC version 3.0 SDK

If a memory access violation in the mtx.exe process occurs within the driver after 60 seconds of inactivity, you may be using an ODBC driver that is not thread safe or requires thread affinity. The fault occurs in the driver when the inactive connections are being disconnected.

MTS Distributed Transaction Coordinator

MTS uses the services of the Microsoft Distributed Transaction Coordinator (DTC) for transaction coordination. DTC is a system service that coordinates transactions that span multiple resource

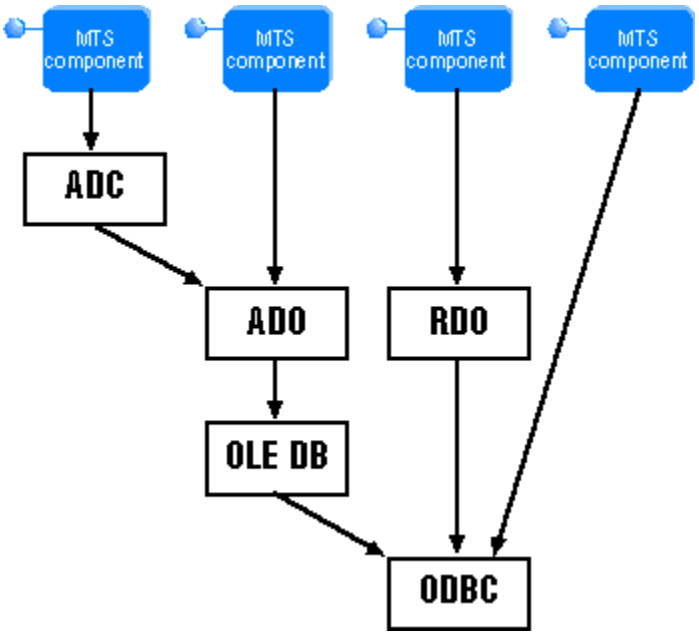
managers. Work can be committed as a single transaction, even if it spans multiple resource managers, potentially on separate computers. DTC was initially released as part of Microsoft SQL Server version 6.5, and is included as part of MTS. DTC implements a two-phase commit protocol that ensures that the transaction outcome (either commit or abort) is consistent across all resource managers involved in a transaction. DTC supports resource managers that implement OLE Transactions, X/Open XA protocols, and LU 6.2 Sync Level 2.

Choosing your Data Access Model

The following table summarizes commonly used data access models supported by MTS.

Interface	Description
Microsoft ActiveX Data Objects (ADO), Remote Data Service (RDS)	ADO offers one common yet extensible programming model for accessing data. ADO includes the ability to pass query results (Recordsets) between server and client, and the ability to pass updated Recordsets from client to server using RDS.
OLE DB	OLE DB is a low-level interface that provides uniform access to any tabular data source. You cannot call OLE DB interfaces directly from Microsoft® Visual Basic® because OLE DB is a pointer-based interface. A Visual Basic client can access an OLE DB data source through ADO.
Open DataBase Connectivity (ODBC)	ODBC is a recognized standard interface to relational data sources. ODBC is fast and provides a universal interface that is not optimized for any specific data source.
Remote Data Objects (RDO)	RDO is a thin object layer interface to the ODBC API. It is specifically designed to access remote ODBC relational data sources. Note that adding the rdExecDirect flag to the .Execute method ensures that a temporaray stored procedure is not created when using two DSN's from the same transaction. See the RDO documentation for more details.

The diagram below illustrates how MTS components interact with the different data access interfaces:



ADO is not specifically designed for relational or ISAM databases, but as an object interface to *any* data source. ADO can access relational databases, ISAM, text, hierarchical, or any type of data source, as long as a data access provider exists. ADO is built around a set of core functions that all data sources are expected to implement. ADO can access native OLE DB data sources, including a specific OLE DB provider that provides access to ODBC drivers. ADO ships with the OLE DB Software Development Kit (SDK).

RDO does have some functionality that is not currently implemented in ADO, including the following:

- Events on the Engine, Connection, Resultset, and Column objects
- Asynchronous operations
- Queries as methods
- Enhanced batch-mode error and contingency handling
- Tight integration with Visual Basic, as in the Query Connection designer and TSQL debugger.

Future versions of ADO will provide a superset of RDO version 2.0 functionality and provide a far more sophisticated interface, in addition to an easier programming model. Because ADO is an Automation-based component, any application or language capable of working with Automation objects can use it.

Developing MTS Components with Java

You can develop Java MTS components using tools provided by MTS and Visual J++. It is also recommended that you install the latest version of the Microsoft SDK for Java, available at http://www.microsoft.com/java.

This section contains the following topics:

- Using Visual J++ 6.0
- Using the Microsoft SDK for Java
- Using an MTS Component from Java
- Using the Java Sample Bank Components

Using Visual J++ 6.0

If you are using Microsoft Visual J++, not only can you create MTS components completely within the integrated development environment (IDE), but the steps required to build a component for MTS are greatly simplified over earlier versions of Visual J++.

Because Visual J++ 6.0 has many new features to support building and debugging components for MTS, some of the steps in the following sections do not apply to Visual J++ 6.0. For more information about building an MTS component in Visual J++ 6.0, see the Visual J++ 6.0 documentation.

Using the Microsoft SDK for Java

Note The following steps assume you are using the Microsoft SDK for Java along with Visual J++ 1.0 or 1.1. (If you have Visual J++ 6.0, these steps are not necessary. For more information, see the Visual J++ 6.0 documentation.)

To implement a component in Java, follow these steps

1. Run the ActiveX Wizard for Java (available with Visual J++) for each Java class file to create an IDL file. Follow the instructions when the wizard asks you to modify your class declaration.
2. Modify the IDL files to add JAVACLASS and PROGID to the coclass attributes. For more information, see *Using IDL Files with Java Components*.
3. Run the ActiveX Component Wizard for Java again. Use the IDL files that you created in Step 1 to create type libraries for your components.

   This will create a set of Java class files, typically under \%systemroot%\Java\Trustlib. It will create one class file for each custom interface, and one class file for each coclass in the library.
4. Run JAVAGUID against each class file generated in Step 3. For more information, see *Working with GUIDs in Java*.
5. Recompile your Java implementation classes.
6. Run EXEGEN to convert the type libraries and class files into a DLL. For more information, see *Using EXEGEN to Create DLLs*.
7. Use the MTS Explorer to install the DLL.

Using IDL Files with Java Components

To specify the custom attributes in a type library, add the following in your IDL or ODL file:

```
#include <JavaAttr.h>
```

Within the attributes section of a coclass, specify the JAVACLASS:

```
JAVACLASS("package.class")
```

You may optionally specify a PROGID:

```
PROGID("Progid")
```

For example:

```
[
    uuid(a2cda060-2d38-11d0-b94b-0080c7394688),
    helpstring("Account Class"),
    JAVACLASS("Account.AccountObj"),
    PROGID("Bank.Account.VJ"),
    TRANSACTION_REQUIRED
]
coclass CAccount
{
    [default] interface IAccount;
};
```

Using EXEGEN to Create DLLs

EXEGEN is the Java executable file generator. To use this file, copy it to the appropriate destination folders (usually \JavaSDK\bin). This version of EXEGEN.EXE is capable of creating DLL files from Java classes, and can also include user-specified resources in its output files. This version of EXEGEN no longer supports the /base: directive. Class files are always included with the proper name. It supports a new /D directive that causes it to generate a DLL file instead of an EXE.

EXEGEN is now capable of reading five types of input files:

- Java class files
- RES files containg resources to be included
- Executable files containg resources to be included
- TLB files containing type libraries to be included
- Text files describing which classes should be registered (DLL only).

If you use EXEGEN to create a DLL, the DLL can self-register any included Java classes that implement COM objects. There are two ways to tell EXEGEN which classes should be registered:

- Include a type library that contains custom attributes for the classes. This is the preferred method.

  - or -

- Include a text file as input that gives EXEGEN the necessary directions. Each line of the text file describes one Java class, using the following keywords:
  - class:JavaClassName

    Required keyword.
  - clsid:{.....}

    Optional keyword that specifies the clsid GUID. If omitted, EXEGEN chooses a unique GUID.
  - progid:ProgId

    Optional keyword. If omitted, the class will be registered without a progid.

Working with GUIDs in Java

When an MTS method uses a GUID parameter, you must pass an instance of class com.ms.com._Guid. Do not use class Guid, CLSID or IID from package com.ms.com; they will not

work and they are deprecated. The definition of class _Guid is:

```
package com.ms.com;
public final class _Guid {

   // Constructors
   public _Guid (String s);
   public _Guid (byte[] b);
   public _Guid (int a, short b, short c,
       byte b0, byte b1, byte b2, byte b3,
       byte b4, byte b5, byte b6, byte b7);
   public _Guid ();

   // methods
   public void set(byte[] b);
   public void set(String s);
   public void set(int a, short b, short c,
       byte b0, byte b1, byte b2, byte b3,
       byte b4, byte b5, byte b6, byte b7);

   public byte[] toByteArray();
   public String toString();
}
```

Instances of this class can be constructed from a String (in the form "{00000000-0000-0000-0000-000000000000}"), from an array of 16 bytes, or from the usual parts of a Guid. Once constructed, the value can also be changed. Method toByteArray will return an array of 16 bytes as stored in the Guid, and method toString will return a string in the same form used by the constructor.

JAVAGUID.EXE

Microsoft Transaction Server supplies a tool, JAVAGUID.EXE, that will post-process the output of JAVATLB. If you are using Visual J++ 6.0 or later, the use of this tool is not required.

JAVAGUID performs the following for each class file:

- If any method takes a GUID as a parameter, the class of that parameter will be changed to com.ms.com._Guid.
- If the class file is an interface derived from a type library, a public static final member named **iid** will be added to the class. This member will contain the interface ID of the interface.
- If the class file represents a coclass derived from a type library, a public static final member named **clsid** will be added to the class. This member will contain the CLSID of the class.

JAVAGUID can only be executed from the command line. It takes one or more parameters which are names of class files to update.

The functionality in JAVAGUID has been included with the JACTIVEX tool from Visual J++ 6.0. Additionally, JACTIVEX automatically adds **clsid** and **iid** members to COM wrapper classes.

The **clsid** and **iid** members are useful as parameters to **IObjectContext.CreateInstance** and **ITransactionContextEx.CreateInstance**.

Using an MTS Component from Java

To use an MTS component from Java, run the Java Type Library Wizard against the type library for the component. This will create several Java class files, typically under \%systemroot%\Java\

TrustLib. It will create one class file for each custom interface, and one class file for each coclass in the library.

Assume, for example, that the type library contained one interface named IMyInterface, and one coclass, named CMyClass.

From Java, you can create a new instance of the component by executing

```
new CMyClass()
```

If you want to control transaction boundaries in the class, you can execute

```
ITransactionContextEx.CreateInstance ( CMyClass.clsid, IMyInterface.iid )
```

You should never call Java's new operator on the class that you implemented. Instead, use one of the following techniques:

- Use Java's new operator on the class created by the Java Type Library Wizard. This will cause the Java VM to call **CoCreateInstance**.
- Call **MTx.GetObjectContext().CreateInstance** (*clsid*, *iid*);
  This will create a new instance in the same activity as the current instance. This only works if the calling code is itself an MTS component.
- If you have a reference to an **ITransactionContextEx** object, call its **CreateInstance** method. This will create a new instance in the transaction owned by the **ITransactionContextEx** object.

All of these techinques will result in the creation of a new instance of the class that you implemented.

Using the Java Sample Bank Components

The Java Sample Bank components are automatically configured by MTS Setup and require no additional steps in order to run them.

If you want to recompile the Java Sample Bank components, follow these steps:

1. Run the SetJavaDev.bat file located in the \mts\Samples\Account.VJ folder. Javatlb.exe must be in your path for this batch file to run properly.
2. Recompile your Java component implementation classes.
3. After you recompile the component classes, use the mkdll.bat file located in the \mts\Samples\Account.VJ folder to generate and register vjacct.dll. Exegen.exe must be in your path for this batch file to run properly. You can also add running mkdll.bat as a build step to your Visual J++ project to simplify recompiling.
4. Using the MTS Explorer, import the new components into the Sample Bank package.

Debugging MTS Components

This document describes techniques for debugging MTS components written in Microsoft® Visual Basic®, Microsoft Visual C++®, and Microsoft Visual J++™. These techniques are just suggestions for successfully debugging MTS components; you can choose your debugging environment and techniques according to your application needs.

If you are using MTS components in a distributed environment, it is recommended that you debug your components on a single computer before deploying to multiple servers. Components that function without error in a package on a local computer usually run successfully over a distributed network. If you do encounter problems with distributed components, you must test and debug both the client and server machines to determine the problem. It is also recommended that you stress test your application with as many clients as possible. You can build a test client that simulates multiple clients to perform the stress test on your application.

To help you debug your components, you should set the MTS transaction timeout to a higher number than the default 60 seconds; otherwise, during debugging, MTS aborts the transaction after this time has lapsed. All subsequent calls to the component return immediately with CONTEXT_E_ABORTED. Any changes to the transaction timeout value made during debugging may not take effect until you restart your debugging environment.

For debugging MTS components written in a specific language, see the following topics:

Debugging Visual Basic MTS Components

Debugging Visual C++ MTS Components

Debugging Java Classes

Debugging Visual Basic MTS Components

Using Visual Basic 6.0

Microsoft Transaction Server components written in Microsoft Visual Basic version 6.0 can be debugged directly in the Visual Basic Integrated Development Environment (IDE).

The following steps describing how to debug a Visual Basic COM component do not apply to Visual Basic version 6.0 or later.

For more information about debugging MTS components using Visual Basic version 6.0, see the Visual Basic version 6.0 documentation.

Using Visual Basic 5.0

Microsoft Transaction Server components written in Microsoft Visual Basic version 5.0 or Visual C++ version 5.0 can be debugged in the Microsoft Visual Studio 97 Integrated Development Environment (IDE).

If you want to debug your components after they are compiled, you cannot use the Visual Basic 5.0 debugger, which only debugs at design time. To debug a compiled Visual Basic component, you will need to use the functionality of the Visual Studio 97 debugger.

Follow these steps to configure Visual Studio to debug MTS components built with Visual Basic 5.0:

1. In Visual Basic, click **Properties** on the **Project** menu and then click the **Compile** tab to select the **Compile to Native Code** and the **Create Symbolic Debug Info** checkbox. It is also recommended that you select the **No Optimization** checkbox while debugging.
2. In the MTS Explorer, right-click the package in which your component is installed, and select the **Properties** option. Place your cursor over the Package ID, and select and copy the GUID to the clipboard.
3. Open Visual Studio. On the **File** menu, click **Open** and select the DLL containing the component that you want to debug.
4. Select **Project Settings**, and then click the **Debug** tab. Select the MTS executable for the debug session (winnt\system32\mtx.exe). Enter the program arguments as /p:{<package GUID>} for the package GUID that you copied from the package properties. MTS 2.0 allows for the package name to be used in place of the GUID. Open the .cls files containing the code that you want to debug and then set your breakpoints. If you also want to display variable information in the debug environment, go to the Visual Studio Tools menu, select **Options**, and then select the **Debug** tab. In the **Debug** tab, place a check next to **Display Unicode strings**.
5. In the MTS Explorer, shut down all server processes.
6. In Visual Studio, select **Build**, then select **Start Debug**. Then select **Go** to run the server process that will host your component(s), and set breakpoints to step through your code.
7. Run your client application to access and debug your components in Visual Studio.
8. Before you deploy your application, remember to select one of the optimizing options in the **Compile** tab on the **Project** menu of Visual Basic (set to **No Optimization** in Step 1), clear the **Create Symbolic Debug Info** checkbox, and recompile the project.

To facilitate application debugging using Visual Basic 5.0, a component that uses ObjectContext can be debugged by enabling a special version of the object context. This debug-only version is enabled by creating the registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Transaction Server\Debug\RunWithoutContext

Note that when running in debug mode, none of the functionality of MTS is enabled.

GetObjectContext will return the debug ObjectContext rather than returning Nothing.

When running in this debug mode, the ObjectContext operates as follows:

- **ObjectContext.CreateInstance** - calls COM CoCreateInstance (no context flows, no transactions, and so on)
- **ObjectContext.SetComplete** - no effect
- **ObjectContext.SetAbort** - no effect
- **ObjectContext.EnableCommit** - no effect
- **ObjectContext.DisableCommit** - no effect
- **ObjectContext.IsInTransaction** - returns FALSE
- **ObjectContext.IsSecurityEnabled** - returns FALSE
- **ObjectContext.IsCallerInRole** - returns TRUE (same as normal when **IsSecurityEnabled** is FALSE)

You can also develop your own testing message box functions to generate an assert in an MTS Visual Basic component. The following sample code can to used to display error messages while debugging Visual Basic code. You can also use this in conjunction with the Microsoft Windows NT® debugger (WinDbg.exe), which is a 32-bit application that, along with a collection of DLLs, is used for debugging the Kernel, device drivers, and applications. Note that you must enter DEBUGGING = -1 in the Conditional Compilation dialog box (located on the **Make** tab of the **Project Properties** dialog box) to enable the assert.

The following code provides an example.

```
#If DEBUGGING Then
    'API Functions
    Private Declare Sub OutputDebugStringA _
        Lib "KERNEL32" (ByVal strError As String)
    Private Declare Function MessageBoxA _
        Lib "USER32" (ByVal hwnd As Long, _
        ByVal lpText As String, _
        ByVal lpCaption As String, _
        ByVal uType As Long) As Long
    'API Constants
    Private Const API_NULL                  As Long = 0
    Private Const MB_ICONERROR              As Long = &H10
    Private Const MB_SERVICE_NOTIFICATION As Long = &H200000

Public Sub DebugPrint(ByVal strError As String)
    Call OutputDebugStringA(strError)
End Sub

Public Sub DebugMessage(ByVal strError As String)
   Dim lngReturn As Long
        lngReturn = MessageBoxA(API_NULL, strError, "Error In Component", _
            MB_ICONERROR Or MB_SERVICE_NOTIFICATION)
End Sub
#End If
```

You can then run checks through your code to aid stress debugging, such as in the following code:

```
SetobjObjectContext=GetObjectContext()
```

```
#If DEBUGGING Then
If objObjectContext Is Nothing Then Call DebugMessage("Context is Not Available")
#End If
```

Debugging Visual C++ MTS Components

You can use the Visual C++ Integrated Development Environment (IDE) to debug MTS components written in Visual C++, including components that call SQL Server functions or stored procedures. For more information, see Debugging Visual Basic MTS Components.

The following information applies to components that have their activation property set to In a dedicated server process.

Microsoft Transaction Server supports the COM transparent remote debugging infrastructure. If transparent remote debugging is enabled, then stepping into a client process will automatically stop at the actual object's code in the server process, even if the server is on a different computer on the network. A debugging session is automatically started on the server process if necessary. Similarly, single stepping past the return address of code in a server object will automatically stop just past the corresponding call site in the client's process.

In Microsoft Visual C++, selecting the **OLE RPC debugging** check box (on the **Tools** menu, select the **Options** submenu and choose the **Debug** property sheet) enables transparent remote debugging. It is not known at this time whether other debuggers support this infrastructure.

You can also debug your Microsoft Transaction Server component DLL in Visual C++ by performing the following steps. Each of these steps is made either inside the MTS Explorer or inside of a Visual C++ session with your MTS DLL project.

1. Shutdown server processes using the MTS Explorer. To do this, right-click My Computer, and select Shutdown Server Process.
2. In your Visual C++ session, under **Project**, **Settings**, **Debug**, **General**, set the program arguments to the following string: **"/p:** *PackageName***"**, for example:

   ```
   /p:"Sample Bank"
   ```

3. In the same property sheet, set the executable to the full path of the Mtx.exe process, for example: **"c:\winnt\system32\MTx.exe"**.
4. Set breakpoints in your component DLL, and you are ready to debug.
5. Run the server process (in the Build menu, select **Start** Debug and click Go.)

The following information applies to in-process component DLLs that have their activation property set to In the creator's process.

You can debug your in-process MTS component DLL in Visual C++ by performing the following steps. Each of these steps is made inside a Visual C++ session with your base process project.

1. Set the component DLL under **Build, Settings, Debug, Additional DLLs**.
2. Now you are ready to step into or set breakpoints in your component DLL at will.

If you are using Visual Studio and Microsoft Foundation Classes (MFC) to debug, the TRACE macro can facilitate your debugging. The TRACE macro is an output debug function that traces debugging output to evaluate argument validity. The TRACE macro expressions specify a variable number of arguments that are used in exactly the same way that a variable number of arguments are used in the run-time function printf. The TRACE macro provides similar functionality to the printf function by sending a formatted string to a dump device such as a file or debug monitor. Like printf for C programs under MS-DOS, the TRACE macro is a convenient way to track the value of variables as your program executes. In the Debug environment, the TRACE macro output goes to afxDump. In the Release environment, the TRACE macro output does nothing.

Example:

```
// example for TRACE
int i = 1;
char sz[] = "one";
TRACE( "Integer = %d, String = %s\n", i, sz );
// Output: 'Integer = 1, String = one'
```

The TRACE macro is available only in the debug version of MFC, but a similar function could be written for use without MFC. For more information on using the TRACE macro, see the "MFC Debugging Support" section in Microsoft Visual C++ Programmer's Guide.

Note that you should avoid using standard ASSERT code in Visual C++. Instead, it is recommended that you write assert macros like a **MessageBox** using the MB_SERVICE_NOTIFICATION flag, and TRACE macro statements using the OutputDebugString function call.

Debugging Java Classes

Using Visual J++ 6.0

Because you can use Visual J++ version 6.0 to debug MTS components completely within the Visual J++ Integrated Development Environment (IDE), the debugging topics below do not apply to Visual J++ 6.0, and are included only for those using earlier versions of Visual J++.

For more information about debugging MTS components in Visual J++ 6.0, please see the Visual J++ 6.0 documentation.

Using Visual J++ 1.1 or 1.0

If you are using Visual J++ 1.1 or earlier, you cannot use the Visual J++ IDE to debug your MTS component. Once your Java class is converted into an MTS component it is not possible to step through the code in the Visual J++ debugger.

Using Visual J++ to Debug Java Classes

Microsoft Visual J++ provides a Java debugger that you can use to set breakpoints in your code. Note that when you are using Visual J++ to debug, if you set a breakpoint in a Java source file before starting the debugging session, Visual J++ may not stop on the breakpoint. For performance reasons, the debugger preloads only the main class of your project. The main class is either the class with the same name as the project or the class you specify in Visual J++. If you use the editor to set breakpoints in other classes before the classes are loaded, the breakpoints are disabled.

You can choose one of the following options to load the correct class so that the debugger stops at breakpoints.

- Select the class in the category Additional Classes, located on the Debug tab of the **Project** Settings dialog box, and make sure the first column is checked. This loads the class when the debugging session starts.
- Right-click a method in the ClassView pane of the Project Workspace and select **Set Breakpoint** from the Shortcut menu. This causes a break when program execution enters the method.
- Set the breakpoint after Visual J++ has loaded the class during debugging. You may need to step through your Java source until the class is loaded.

When a method has one or more overloaded versions and shows up as a called method in the Call Stack window, the type and value for the parameters are not displayed in some cases. It appears as though the method takes no parameters. This occurs when the called method is not defined as the first version of the overloaded method in the class definition. For example, see the following class definition:

```
public class Test
   {
       int method(short s)
       {
           return s;
       }

       int method(int i)
       {
           return i;
       }
   }
```

If you were looking at a call to the second version of the method in the Call Stack window, it would appear without the type and value for the method:

```
method()
```

To view the method's parameters, change the order of the method overloads so that the method that you are currently debugging is first in the class definition.

printf-style Debugging

You can use printf-style debugging to debug your Java classes without using a debugger. printf-style debugging involves including status text messages into your code, allowing you to "step through" your code without a debugger. You can also use printf-style debugging to return error information. The following code shows how you can add a `System.out.println` call to the try clause of the Hellojtx.HelloObj.SayHello sample.

```
try
{
System.out.println("This message is from the HelloObj implementation");
   result[0] = "Hello from simple MTS Java sample";
   MTx.GetObjectContext().SetComplete();
   return 0;
}
```

The client must be a Java client class, and you must use the JVIEW console window to run that class. Note that you need to configure your component to run in the process of its caller, which is in this case JVIEW. Otherwise, this debugging technique results in your component running in the MTS server process (mtx.exe), which would put the println output in the bit bucket rather than the JVIEW console window.

Use the MTS Explorer to configure your component to run in the caller's process by following these steps.

1. Right-click the component.
2. Click the **Properties** option.
3. Click the **Activation** tab and clear the **In a server process on this computer** checkbox.
4. Select the **In the creator's process**... checkbox.
5. Reload the **Client** class. Your component's `println` calls will be visible in the JVIEW console window.

Using the AWT Classes

You can also use the AWT (Abstract Window Toolkit) classes to display intermediate results, even if your component is running in a server process. The java.awt package provides an integrated set of classes to manage user interface components such as windows, dialog boxes, buttons, checkboxes, lists, menus, scrollbars, and text fields.

The following example demonstrates how to use the AWT classes to display intermediate results in a dialog box:

```
import java.awt.*;

public final class MyMessage extends Frame
{
```

```
    private Button closeButton;
    private Label textLabel;

    // constructor
    public MyMessage(String msg)
    {
        super("Debug Window");

        Panel panel;

        textLabel = new Label (msg, Label.CENTER);
        closeButton = new Button ("Close");

        setLayout (new BorderLayout (15, 15));
        add ("Center", textLabel);

        add ("South", closeButton);

        pack();
        show();
    }

    public boolean action (Event e, Object arg)
    {

        if (e.target == closeButton)
        {
            hide();
            dispose();
            return true;
        }

        return false;
    }

}
```

Asynchronous Java Garbage Collection

Note that garbage collection for Java components is asynchronous to program execution and can cause unexpected behavior. This behavior especially affects MTS components that perform functions such as enumerating through the collections in the catalog because the collection count will be too high (garbage collection is not synchronized). To force synchronous release of references to COM or MTS objects, you can use the release method defined in class com.ms.com.ComLib.

Example:

```
Import com.ms.com.ComLib
…
ComLib.release(someMTSObject);
```

This method releases the reference to the object when the call is executed. Release the object reference when you are sure that the reference is no longer needed. Note that if you fail to release

the reference, an application error is not returned. However, an incorrect collection count results because the object reference is released asynchronously when the garbage collector eventually runs.

You can also force the release of your reference and not call that released reference again.

Example:

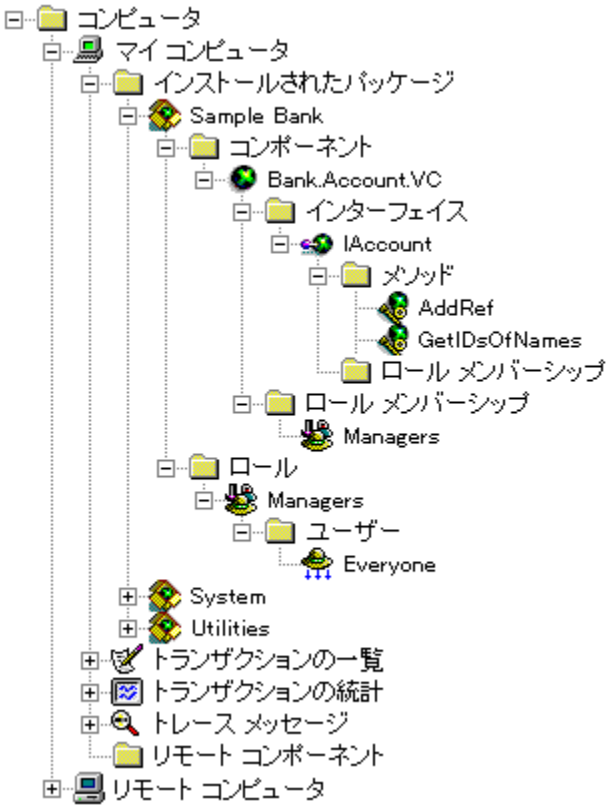
```
myHello = null;
   System.gc();
```

Note that this method of forcing the release of an object reference consumes extensive system resources, and is not guaranteed to work in all cases. For more information, please see Microsoft Knowledge Base article number Q179062 available at http://www.microsoft.com/kb/.

It is recommended that you use the release method defined in com.ms.com.ComLib class to release references to MTS objects in a synchronous fashion.

Automating MTS Deployment

This document describes how you can use the scriptable administration objects to automate deployment and distribution of your MTS packages. The MTS Explorer lets you configure and deploy packages by using a graphical user interface rather than by programming code. However, you can use the scriptable administrative objects to automate administration tasks, such as program configuration and deployment. Note that the scriptable administrative objects support the same collection hierarchy as the MTS Explorer. The following figure shows the MTS Explorer collection hierarchy.



For more information about MTS Explorer functionality, see the Administrator's Guide.

Using the Scriptable Administration Objects

Microsoft Transaction Server contains Automation objects that you can use to program administrative and deployment procedures, including:

- Installing a Pre-Built Package
- Creating a New Package and Installing Components
- Enumerating Through Installed Packages to Update Properties
- Enumerating Through Installed Packages to Delete a Package
- Enumerating Through Installed Components to Delete a Component
- Accessing Related Collection Names
- Accessing Property Information
- Configuring a Role
- Exporting a Package
- Configuring a Client to Use Remote Components

Note that you can use the scriptable administration objects to automate any task in the MTS Explorer.

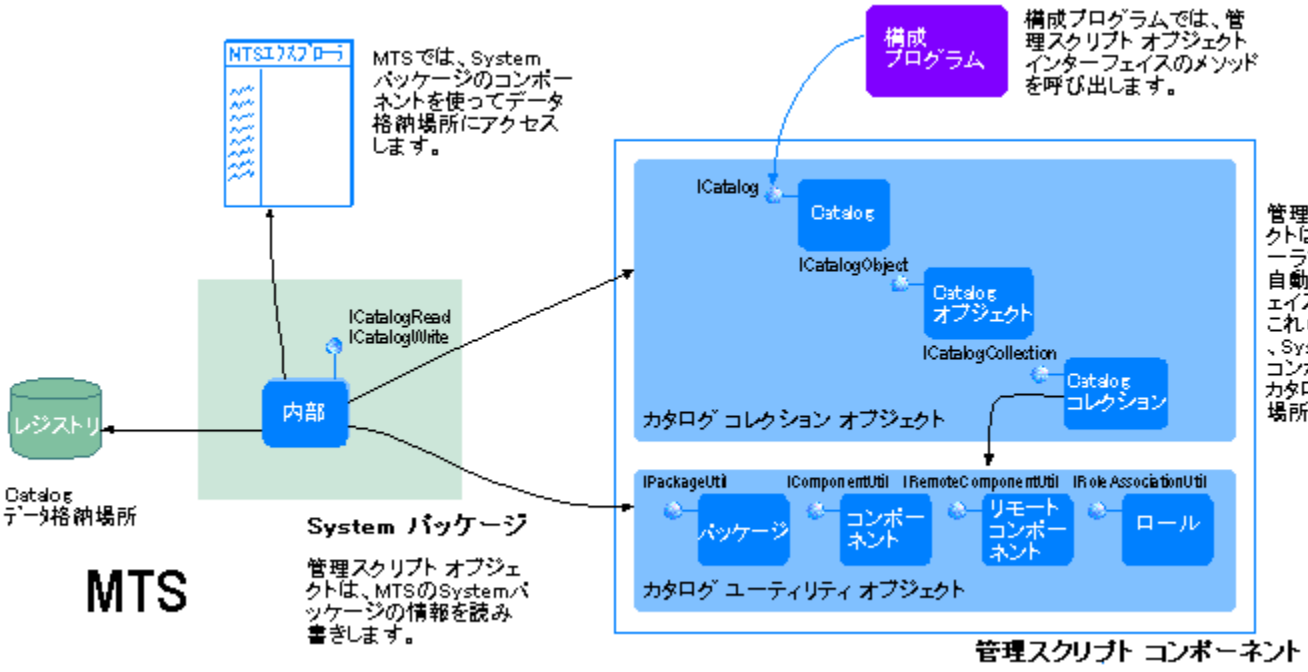
The scriptable administration objects are derived from the **IDispatch** interface, so you can use any Automation language to develop your package, such as Microsoft® Visual Basic® version 5.0, Microsoft Visual C++® version 5.0, Microsoft Visual Basic® Scripting Edition (VBScript), and Microsoft JScript™.

Each folder in the MTS Explorer hierarchy corresponds to a collection stored in the catalog data store. The following scriptable objects are used for administration:

- Catalog
- CatalogObject
- CatalogCollection
- PackageUtil
- ComponentUtil
- RemoteComponentUtil

The Catalog, CatalogObject, and CatalogCollection scriptable objects provide top-level functionality such as creating and modifying objects. The Catalog object enables you to connect to specific servers and access collections. Call the CatalogCollection object to enumerate, create, delete, and modify objects, as well as to access related collections. CatalogObject allows you to retrieve and set properties on an object. The Package, Component, Remote Component, and Role objects enable more specific task automation, such as installing components and exporting packages. This utility layer allows you to program very specific tasks for collection types, such as associating a role with a user or class of users.

The following diagram illustrates how the MTS scriptable administration objects interact with the MTS Explorer catalog:



Interface	Description
ICatalog	The Catalog object enables you to connect to

	specific servers and access collections.
ICatalogCollection	The CatalogCollection object can be used to enumerate objects, create, delete, and modify objects, and access related collections.
ICatalogObject	The CatalogObject object provides methods to get and set properties on an object.
IPackageUtil	The IPackageUtil object enables a package to be installed and exported within the Packages collection.
IComponentUtil	The IComponentUtil object provides methods to install a component in a specific collection and to import components registered as an in-process server.
IRemoteComponentUtil	You can use the IRemoteComponentUtil object to program your application to pull remote components from a package on a remote server.
IRoleAssociationUtil	Call methods on the IRoleAssociationUtil object to associate roles with a component or component interface.

For example, you can automate creating a new package and installing components into the new package by using the scriptable objects in the utility layer (Package, Component, Remote Component, and Role objects).

The following Visual Basic sample shows how to use the scriptable administration objects to create and install components into a new package named "My Package."

1. Declare the objects that you will be using to create and install components into a new package.

```
Dim catalog As Object
Dim packages As Object
Dim newPack As Object
Dim componentsInNewPack As Object
Dim util As Object
```

2. Use the **On Error** statement to handle run-time errors if a method returns a failure HRESULT. You can test and respond to MTS trappable errors using the **On Error** statement and the **Err** object.

```
On Error GoTo failed
```

3. Call the **CreateObject** method to create an instance of the Catalog object. Retrieve the top level Packages collection from the CatalogCollection object by calling the **GetCollection** method. Then call the **Add** method to add a new package.

```
Set catalog = CreateObject("MTSAdmin.Catalog.1")
Set packages = catalog.GetCollection("Packages")
Set newPack = packages.Add
Dim newPackID As String
```

4. Set the package name to "My Package" and save changes to the **Packages** collection.

```
newPackID = newPack.Key
newPack.Value("Name") = "My Package"
packages.savechanges
```

5. Call the **GetCollection** method to access the ComponentsInPackage collection. Then instantiate

the ComponentUtil object in order to call the **InstallComponent** method to populate the new package with components.

```
Set componentsInNewPack =
    packages.GetCollection("ComponentsInPackage",
    newPackID)
Set util = componentsInNewPack.GetUtilInterface
util.InstallComponent"d:\dllfilepath", "", ""
Exit Sub
```

6 Use the **Err** object to display an error message if the installation of the package fails.

```
failed:
    MsgBox "Failure code " + Str$(Err.Number)

End Sub
```

For a complete description of how to program these procedures and more sample code, refer to the Administrator's Guide.

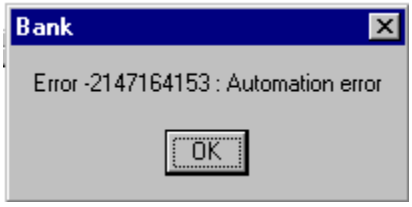
MTS Error Diagnosis

This topic describes how to determine the source of an error in your MTS application. You can diagnose the source and obtain a description of application errors by using a combination of Microsoft® Windows NT®, MTS, and other tools. If you discover that the application error is caused by MTS, you can interpret the error message using the Win32 (Win32.h) or MTS header files (mts.h), or the Microsoft Visual C++® error utility.

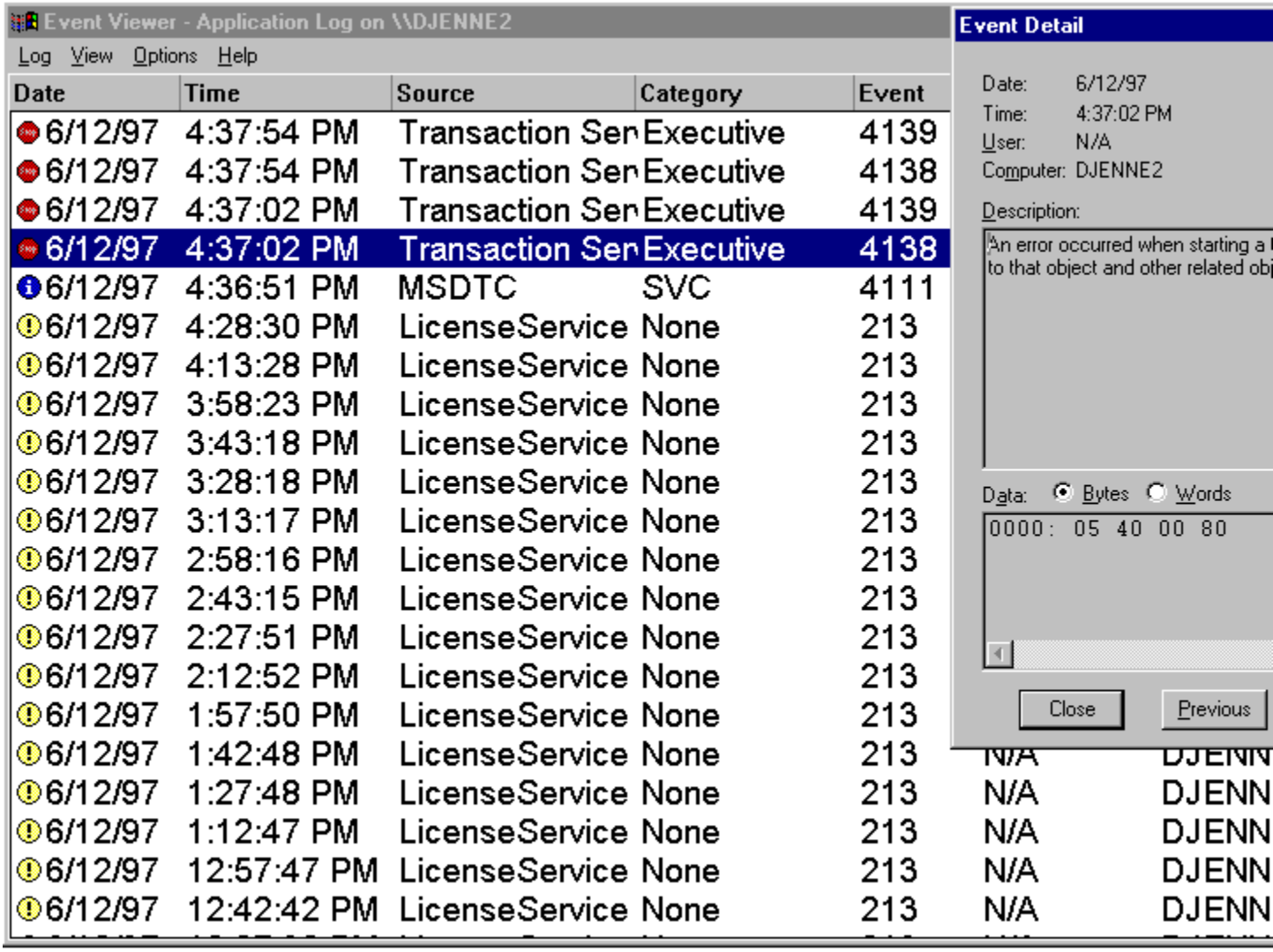
For more information on debugging an MTS application, see Debugging MTS Components.

Finding the Source of the Error

If your server application is failing or causing unexpected behavior, you must first determine where your error occurred. Windows NT provides a system Event Viewer that tracks application, security, and system events. Refer to the Application Log in the Event Viewer first to check the application associated with the event message. (Because you can also archive event logs, you can track an event history of the error.) Selecting an entry in your log activates an Event Detail, which provides further information about the system event. If you attempt to run the Sample Bank client without starting the Microsoft Distributed Transaction Coordinator (MS DTC), you will be returned the following Automation error.



Since this error does not indicate which application caused the failure, you can reference the Application Log in the Event viewer, which shows the error was caused by MTS.



Note If you are using MTS for Windows 98, events are written to text files in the \Windows\MTSLogs directory.

Interpreting Error Messages

The Event Viewer helps you determine the application source of the problem. You can use other tools to interpret individual error messages. Success, warning, and error values are returned using a 32-bit number known as a result handle, or HRESULT. HRESULTs are 32-bit values with several fields encoded in the value. A zero result indicates failure if that bit is set. A non-zero result can be a warning or informational message.

HRESULTs work differently, depending on the platform you are using. On 16-bit platforms, an HRESULT is generated from a 32-bit value known as a status code, or SCODE. On 32-bit platforms, an HRESULT is the same as an SCODE. Note that if you are returning HRESULTs from Java, you should throw an instance of com.ms.com.ComFailException to indicate failure. You can specify a particular HRESULT when constructing the ComFailException object. The HRESULT is used as the return value for the COM method. To indicate successful completion, you do not need to do anything; just return normally. To return S_FALSE, indicating a successful completion but a return value of Boolean False, throw an instance of com.ms.com.ComSuccessException. In Visual Basic, you use the Err.Raise function to set and the On Error… / Err.Number to retrieve HRESULTs.

For a list of the values of common system-defined HRESULTs, see ComFailException. For a complete list of system-defined HRESULT values, see the header file Winerror.h included with the Platform SDK.

MTS never changes the value of an HRESULT error code, such as E_UNEXPECTED or E_FAIL, returned by an MTS object method. When an MTS object returns an HRESULT status code (such as S_OK or S_FALSE), MTS may convert the status code into an MTS error code before it returns to the caller. This occurs, for example, when the application returns S_OK after calling the **SetComplete** function; if the object is the root of an automatic transaction that fails to commit, the HRESULT is converted to CONTEXT_E_ABORTED. When MTS converts a status code to an error code, all the method's output parameters are cleared. Returned references are released and the values of the returned object pointers are set to NULL.

The Mtx.h header file contains the MTS specific error codes. Winerror.h contains the error code definitions for the Win32 API. For an overview of error codes, see "Error Handling" in the COM portion of the Microsoft Platform SDK

You can also use the ERRLOOK utility in Microsoft Visual Studio™ 97 to retrieve a system error message or module error message based on the value entered. ERRLOOK retrieves the error message text automatically if you drag-and-drop a hexadecimal or decimal value from the Visual Studio debugger or other Automation-enabled application. You can also enter a value either by typing it in or pasting it from the IDE clipboard and clicking the **Look Up** option.

Contacting MTS Support

If you run into a problem that you cannot solve, you can contact Microsoft support with the following information:

- Topography of the application where the error occurred, such as a description of packages, components, and interfaces
- Application event log on all computers
- Reproduction of the error, if possible

Troubleshooting

If you are having trouble diagnosing your problem, refer to the list of troubleshooting tips below:

- Make sure that the Distributed Transaction Coordinator (DTC) is running on all servers.
- Check network communication by first testing on a local computer to verify that the application works. If you are running TCP/IP on your network, you can then use the Windows NT Ping.exe utility to verify that the machines are on the network.
- Make sure that SQL and DTC are either located on the same computer or that the DTC Client Configuration program specifies that the DTC is on another computer. If not, SQLConnect will return an error internally when called from a transactional component.
- Set the MTS transaction timeout to a higher number than the default 60 seconds; otherwise, during debugging, MTS aborts the transaction after this time has lapsed. All subsequent calls to the component return immediately with CONTEXT_E_ABORTED.

Note Any changes to the transaction timeout value made during debugging may not take effect until you restart your debugging environment.

- Make sure that your ODBC drivers are thread-safe and do not have thread affinity.
- If you have difficulty getting an application to work over several servers, reboot the client and then verify that your Windows NT domain controller is configured properly.

- Turning off resource pooling may reveal that a resource dispenser used by your application is the source of the problem. You can turn off resource pooling by setting the following registry key:

  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Transaction Server\Local Computer\My Computer:Resource Pooling=N

See the DCOM documentation and DCOM-related Knowledge Base articles if you are experiencing application errors that you suspect are caused by DCOM.

Creating a Simple ActiveX Component

This section gets you started quickly with a simple ActiveX™ component (Account). You then install, run, and monitor Account with the Microsoft Transaction Server (MTS) Explorer and a sample client (Bank).

Scenario: Creating and Using a Simple ActiveX Component

First, create your component (Account) and run it in the MTS run-time environment by using the Bank client. Then, add a database connection to Account and run it again.

Creating the Account Component

Create a new ActiveX component, Account.

Creating the Bank Package

Use the MTS Explorer to create a new package for your component.

Installing the Account Component in the Bank Package

Use the MTS Explorer to install your component in a package.

Running and Monitoring the Account Component

Use the Bank client to run your component, and use the MTS Explorer to monitor it.

Modifying the Account Component: Add a Database Connection

Modify your component so that it connects to a database. The connection will be pooled by the ODBC resource dispenser. Then use the Bank client to re-run the modified component.

Application Design Notes: Sharing Resources

Learn how to efficiently share resources, such as database connections, through the MTS Resource Dispenser Manager and resource dispensers.

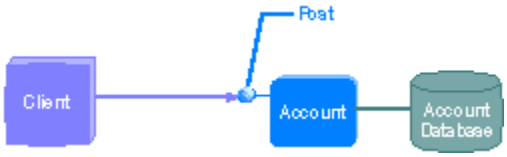
See Also

Programming Concepts, Transaction Server Components, Building Scalable Components

Scenario: Creating and Using a Simple ActiveX Component

This scenario has two implementation stages. The initial stage consists of building a simple component: the Account component. You can implement Account by creating a project (Account.vbp) with a class module (Account.cls). The Account class module exposes one method, **Post**, that passes back a string indicating that it was called successfully. The following illustration depicts the first stage of this scenario.

The second stage of this scenario adds a database connection to get the appropriate account information from a database and update it. This demonstrates using a resource dispenser – in this case, the ODBC resource dispenser, which enables efficient connection pooling. The following illustration depicts the second stage of this scenario.



The rest of this section provides step-by-step instructions for creating, installing, and running the Account component in this scenario. You can find the Microsoft Visual Basic projects for each of these steps in the Step1 through Step8 folders in the \Samples\ Account.VB folder of your Microsoft Transaction Server installation.

See Also

Programming Concepts, Application Design Notes: Sharing Resources, Transaction Server Components, Building Scalable Components, Creating the Account Component, Run and Monitor the Account Component, Modifying the Account Component, Application Design Notes: Resource Usage, Creating a Simple ActiveX Component

Creating the Account Component

The first step toward building a simple application that you can use with Microsoft Transaction Server (MTS) is to create a simple ActiveX™ component DLL (VBAcct.dll); the Account component provides one method, Post.

The information presented here assumes a basic understanding of how to use Microsoft Visual Basic to create ActiveX components.

Note You cannot install executable files (.exe) in MTS. If you have a component built as an executable file, you must rebuild it as a dynamic-link library (DLL).

To create the Account component

1. Start Microsoft® Visual Basic™ and open the \Mts\Samples\Account.VB\Step1\Account.vbp project.

Click here to see the code for the Account component

2. Build the component as a DLL and save it as \Mts\Samples\Account.VB\Step1\VBAcct.dll.

See Also

Programming Concepts, Transaction Server Components, Building Scalable Components, Transaction Server Component Requirements, Run and Monitor the Account Component, Application Design Notes: Sharing Resources, Modifying the Account Component, Application Design Notes: Resource Usage, Creating a Simple ActiveX Component

Creating the Bank Package

To run your component in the MTS run-time environment, you need to create a package. For this scenario, you will create a package with a single component.

A package is a collection of components that you can deploy and manage as a unit. By grouping components into packages, you define the security and process boundaries for components running on a computer. The criteria for deciding how to group components into packages require achieving the optimum balance between performance, fault isolation, and load balancing.

You will create a package called Bank that will contain the Account component.

To create the Bank package

1. On the **Start** menu, point to **Programs**, point to **Microsoft Transaction Server**, and then click **Transaction Server Explorer**.
2. Create a new package named **Bank**. In the **Set Package Identity** dialog box, select **Interactive user**.

How?

You can use the **General**, **Security**, **Advanced**, **Identity, and Activation** tabs to configure a package. For this scenario, you will use the default settings for the package you just created.

See Also

What Does Creating a Package Mean?, Package Properties , Programming Concepts, Installing the Account Component in the Bank Package, Creating a Simple ActiveX Component

Installing the Account Component

To run your components in the Microsoft Transaction Server run-time environment, you first need to install them in a package. This means you need to install the Account component in the Bank package.

To install the Account component

- Install the Account component into the Bank package you created in Creating the Bank Package. Use the Account component that you built in \Mts\Samples\Step1\Account.VB\VBAcct.dll.

How?

You can use the **General**, **Transaction**, and **Security** tabs to configure a component. For this scenario, you will use the default settings for the component you just installed.

See Also

Adding A Component to a Package, Component Properties , Creating the Bank Package, Creating a Simple ActiveX Component

Running and Monitoring the Account Component

Now that you have created the Bank package and installed the Account component in the Microsoft Transaction Server Explorer, you can run the Account component with the Bank client and monitor the component status in the Explorer.

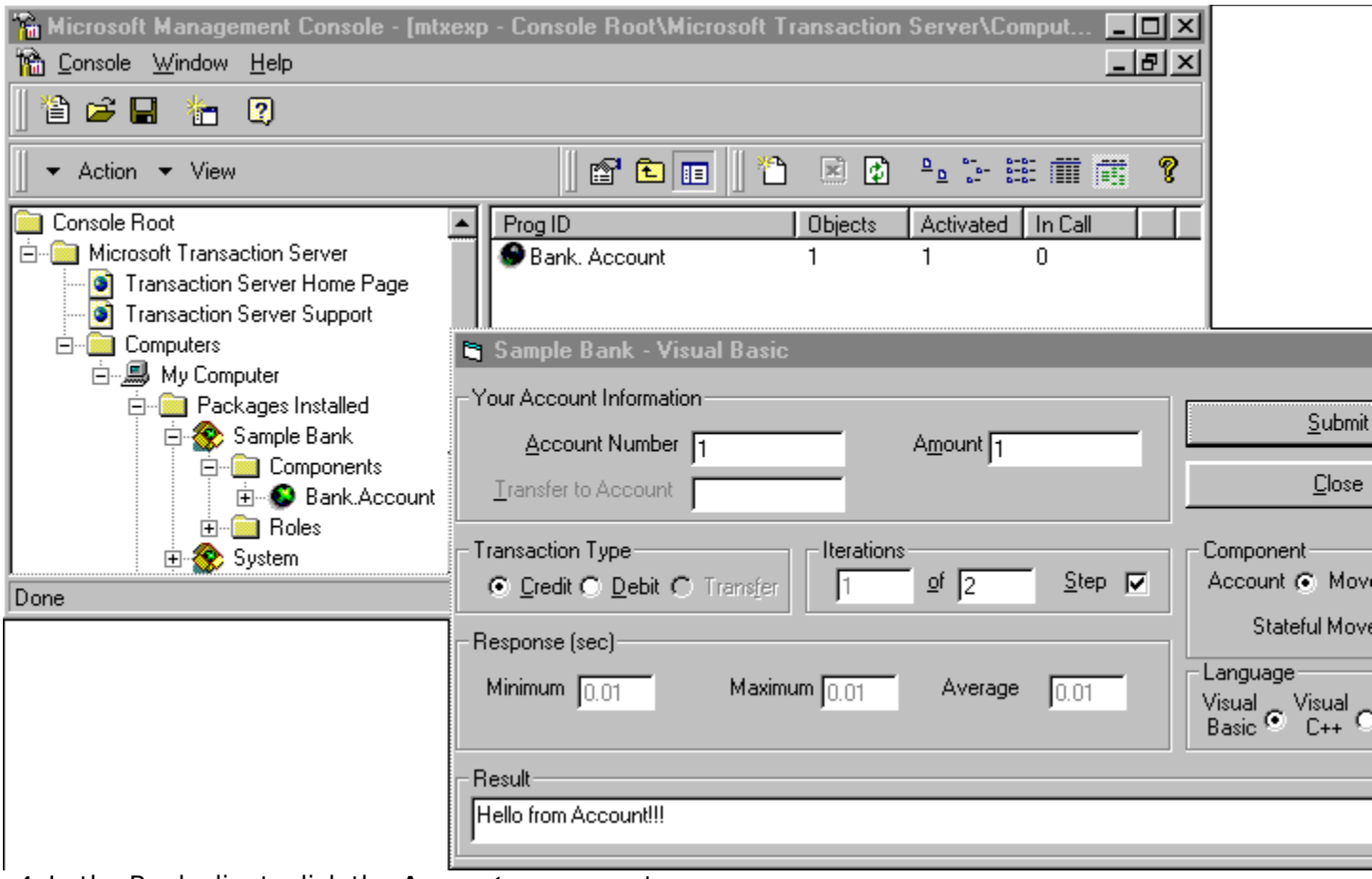
Running your component in MTS brings immediate benefits to your application, even if it doesn't implement any of the MTS APIs. Such benefits include:

- Simplified management of your components though an easy-to-use graphical tool, the MTS Explorer.
- Location transparency--the ability to run your components in-process, locally, or remotely.
- Thread management and component tracking.
- Database connection pooling through the Resource Dispenser Manager and the ODBC resource dispenser (automatically provided if you call the ODBC API).

To run and monitor your component

1. In the left pane of the MTS Explorer, click the Components folder where you installed the Bank.Account component.
2. On the **View** menu, click **Status view** to display usage information for the Bank.Account component.
3. On the **Start** menu, point to **Programs**, point to **Microsoft Transaction Server**, and then click **Bank Client**.

   Arrange the windows so that you can see the Bank Client window and the MTS Explorer window simultaneously. The form will default to credit $1 to account number 1.



4 In the Bank client, click the **Account** component.

5 Click **Submit**.

You should see the response **Hello from Account**.

6 In the Bank client, change the iterations from **1 of 0** to **1 of 100** and click **Submit**.

In the right pane of the MTS Explorer, you should see the values under the **Objects** and **Activated** columns change to 1 and back to 0.

See Also

Status View, Creating the Bank Package, Installing the Account Component in the Bank Package, Microsoft Transaction Server APIs, Maintaining MTS Packages, Creating a Simple ActiveX Component

Modifying the Account Component to Use the ODBC Resource Dispenser

In this section, you enhance the Account component by adding a database connection. You revise the Post method to access a database using ActiveX Data Objects (ADO) to obtain the appropriate account information. This demonstrates using a resource dispenser – in this case, the ODBC resource dispenser, which enables connections to be pooled for efficiency. You will also add a new class module, CreateTable, to the Account project.

To modify the Account component

1. Open the \Mts\Samples\Account.VB\Step2\Account.vbp project.

Click here to see the modified Account component

2. Build the component as a DLL and save it as \Mts\Samples\Account.VB\Step2\VBAcct.dll.

By adding a new class module, you have added a new COM component to this DLL. Therefore, you will need to delete the Account component in the Microsoft Transaction Server Explorer and then install the Account and the MoveMoney components.

To reinstall your components

1. Remove the Account component.

How?

2. Add the new components.

How?

Use the DLL you created in \Mts\Samples\Account.VB\Step2\VBAcct.dll.

You can now run the new component by using the Bank client. You should see a response **Credit, balance is $ 1. (VB)**.

See Also

Creating the Account Component, Application Design Notes: Resource Usage, Programming Concepts, Creating a Simple ActiveX Component

Application Design Notes: Sharing Resources

Each time the **Account** object's **Post** method is called, it obtains, uses, and then releases its database connection. A database connection is a valuable resource. The most efficient model for resource usage in scalable applications is to use them sparingly, acquire them only when you really need them, and return them as soon as possible.

Historically, acquiring resources has been an expensive operation in terms of system performance. Many programs have adopted a strategy of acquiring resources and holding onto them until program termination. While this strategy is effective for single-user systems, building scalable server applications requires sharing these resources.

Microsoft Transaction Server provides an architecture for resource sharing through its Resource Dispenser Manager and resource dispensers. The Resource Dispenser Manager works with specific resource dispensers to automatically pool and recycle resources. The ODBC version 3.0 Driver Manager is a Microsoft Transaction Server resource dispenser, also referred to as the ODBC resource dispenser.

Although the Account component hasn't implemented any of the MTS-specific APIs, when you run it, MTS uses the ODBC resource dispenser. This happens automatically when the Post method uses ActiveX Data Objects (ADO) to access the database, because ADO in turn uses ODBC. Whenever any component running in the MTS run-time environment uses ODBC directly or indirectly, the component automatically uses the ODBC resource dispenser.

When the **Account** object releases the database connection, the connection is returned to a pool. When the **Post** method is called again, it requests the same database connection. Instead of creating a new connection, the ODBC resource dispenser recycles the pooled connection, which saves time and server resources.

The topic Building Scalable Components, shows you how to use just-in-time activation to use server resources even more efficiently, resulting in more scalable applications and improved performance.

See Also

Application Design Notes: Resource Usage, Building Scalable Components, Modifying the Account Component, Creating the Account Component, Creating a Simple ActiveX Component

Building Scalable Components

In this section, you'll learn how you can use just-in-time activation to use server resources efficiently, resulting in more scalable applications and improved performance. You'll also see how a simple change to the Account component allows it to scale efficiently and support a large number of clients, without requiring you to make any changes to the client.

Scenario: Adding Just-In-Time Activation to the Account Component

Add code to your Account component to take advantage of just-in-time activation, which releases the component's resources when Account has completed its work.

Adding Code to Call GetObjectContext, SetComplete, and SetAbort

Add code to call **GetObjectContext**, **SetComplete**, and **SetAbort**.

Application Design Notes: Just-In-Time Activation

Learn how to reuse resources efficiently so you can build scalable applications.

See Also

Context Objects, Deactivating Objects, Creating a Simple ActiveX Component, **GetObjectContext** method, **SetAbort** method, **SetComplete** method

Scenario: Adding Just-In-Time Activation to the Account Component

In this scenario, the Bank client creates an Account object from the Account component and calls its Post method, just as in Creating a Simple ActiveX Component. This time, Account obtains a reference to its context object. When it successfully completes its work on behalf of the client, it uses its context object to call **SetComplete**. If Account encounters an error and is unable to complete its work successfully, it uses its context object to call **SetAbort**. When Account calls either **SetComplete** or **SetAbort**, it indicates that it's finished with its work and that it doesn't need to maintain any private state for its client. This allows the MTS run-time environment to reclaim and reuse the Account object's resources.

See Also

Context Objects, Deactivating Objects, Application Design Notes: Just-In-Time Activation, **GetObjectContext** method, **SetAbort** method

Adding Code to Call GetObjectContext, SetComplete, and SetAbort

Every Transaction Server object has a context object associated with it. The context object is automatically created at the same time the object itself is created. You can use an object's context to declare when the object's work is complete, as shown in the following illustration.

Calling either of these methods notifies the MTS run-time environment that it can safely deactivate the object, making its resources available for reuse.

To implement the scenario for this chapter, you will modify the Post method to use the Account object's context object. Then you will use **SetComplete** and **SetAbort** to enable just-in-time activation.

First, you call **GetObjectContext** to get a reference to the context object. When an object has completed its work successfully, it should call **SetComplete**:

```
GetObjectContext.SetComplete
```

SetComplete notifies the MTS run-time environment that the Account object should be deactivated as soon as it returns control to the Bank client.

If the object encountered an error, it should call **SetAbort**. **SetAbort** also notifies the MTS run-time environment that the Account object should be deactivated as soon as it returns control to the Bank client.

```
GetObjectContext.SetAbort
```

To obtain a reference to an object's context

1. Open the \Mts\Samples\Account.VB\Step3\Account.vbp project.

Click here to see the Post method

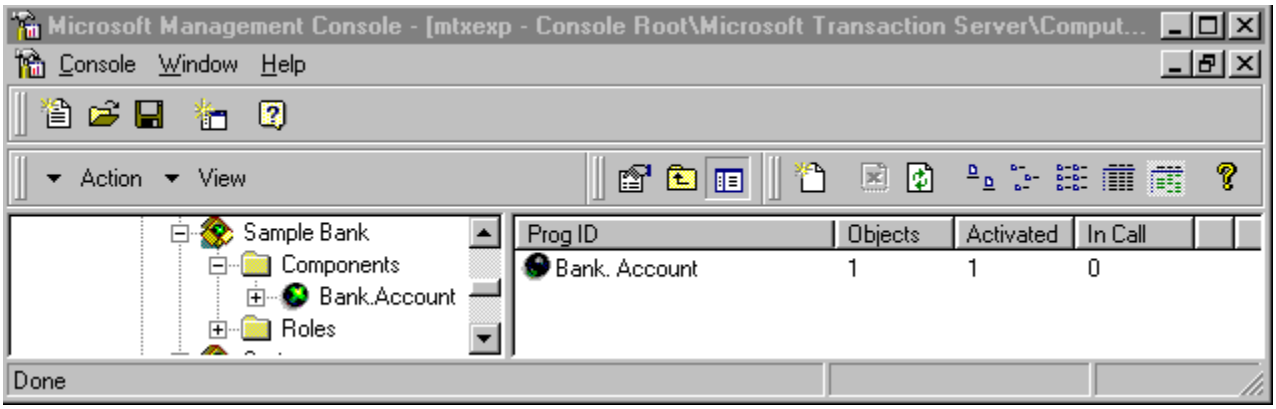
2. Build the component as a DLL and save it as \Mts\Samples\Account.VB\Step3\VBAcct.dll.

Before you can run your new component again in MTS, the registry needs to be updated with the new component information. To do this, refresh the MTS Explorer window.

If you install the Development version of Microsoft Transaction Server, you will get a Visual Basic - compatible add-in that automates this process for you (select the VB Addin box during Setup). The next time you run Visual Basic, the add-in is automatically installed in Visual Basic. The add-in automatically refreshes all of your MTS component DLLs whenever you recompile your project.

You can also turn this feature on and off on a per-project basis by using the toggle command on the Visual Basic **Add-Ins** menu. To turn it on, on the Visual Basic **Add-Ins** menu, point to **MS Transaction Server**, and click **AutoRefresh after compile of active project**. This puts a check mark next to the command, indicating that the feature is activated. If you want to refresh all of your MTS components at any given time, on the Visual Basic **Add-Ins** menu, point to **MS Transaction Server**, and then click **Refresh all components now**.

Now you'll run the Account component again from the Bank client, and monitor its execution in the MTS Explorer's Status window. Follow the same steps as in "Running and Monitoring the Account Component."



When the Bank client creates the Account object, the number 1 will appear under **Objects** and **Activated**. This indicates that one object is executing in the MTS run-time environment, and that it is currently activated. When the client calls the Post method, the number 1 appears, briefly, under **In Call**. This indicates that one object is currently executing a method call. When the Post method returns control to the client, the number under **Objects** is still 1, but the numbers under **Activated** and **In Call** return to 0. This is because after calling **SetComplete**, the object is deactivated as soon as it returns from the current method call.

Note Because the Post method executes so quickly, you may not actually see this sequence appear.

See Also

Context Objects, Deactivating Objects, Creating a Simple ActiveX Component, **GetObjectContext** method, **SetAbort** method, **SetComplete** method

Application Design Notes: Just-In-Time Activation

When you design a traditional application, you have two options:

- A client can create, use, and release an object. The next time it needs the object, it creates it again.

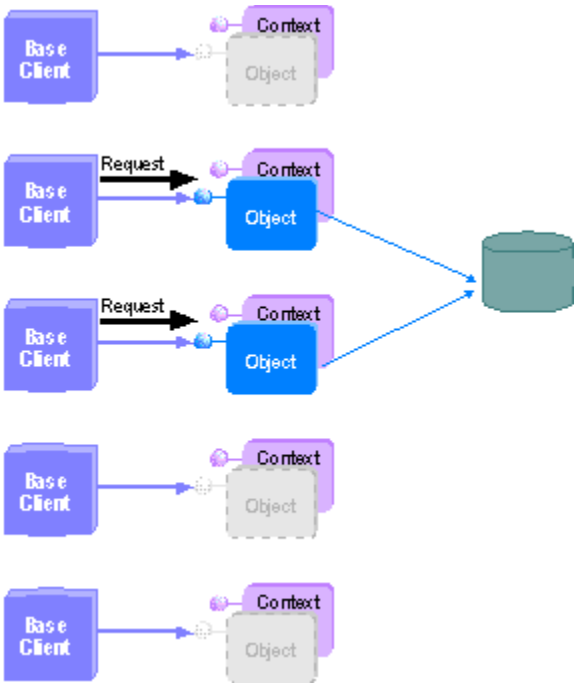
  The advantage to this technique is that it conserves server resources. The disadvantage is that, as your application scales up, your performance slows down. If the object is on a remote computer, each time an object is created, there must be a network round-trip, which negatively affects performance.
- A client can create an object and hold onto it until the client no longer needs it.

  The advantage of this approach is that it's faster. The problem with it is that, in a large-scale application, it quickly becomes expensive in terms of server resources.

While either of these approaches might be fine for a small-scale application, as your application scales up, they're both inefficient. Just-in-time activation provides the best of both approaches, while avoiding the disadvantages of each.

In Creating a Simple ActiveX Component, the Bank client controlled the Account object's life cycle. Clients held onto server resources even when the clients were idle. As you added more clients, you saw a proportional increase in the number of allocated objects and database connections. A quick look at the Account component shows that each call to the Post method is independent of any previous calls. An Account object doesn't need to maintain any private state to correctly process new requests from its client. It also doesn't need to maintain its database connection between calls. The only problem is that, in this scenario, the MTS run-time environment can't reclaim the object's resources until the client explicitly releases the object. If you have to depend on your clients to manage your object's resources, you can't build a scalable component.

By adding just a few lines of code, you were able to implement just-in-time activation in the Account component. When an Account object calls **SetComplete**, it notifies the MTS run-time environment that it should be deactivated as soon as it returns control to the client. This allows the MTS run-time environment to release the object's resources, including any database connection it holds prior to the object's release. The Bank client continues to hold a reference to the deactivated Account object.



When a client calls a method on a deactivated object, the client's reference is automatically bound to a new object. Thus, the client has the illusion of a continuous reference to a single object, without tying up server resources unnecessarily.

Although the call to **SetAbort** has a similar effect, it isn't apparent in this scenario why it is used when errors occur. The next chapter, Building Transactional Components, shows you how transactions can make your applications more robust in the event of an error.

See Also

Context Objects, Deactivating Objects, Creating a Simple ActiveX Component, **GetObjectContext** method, **SetAbort** method, **SetComplete** method

Building Transactional Components

This section introduces transactional components and the benefits of running components within the same transaction.

Scenario: Composing Work from Multiple Components Under the Same Transaction

Add new functionality to transfer money between accounts by adding a new component, MoveMoney, which uses the existing Account component.

Creating the MoveMoney Component

Use the **CreateInstance** method to run the MoveMoney and Account components within the same transaction.

Monitoring Transactions

Use the Bank client to run your components, and use the Microsoft Transaction Server Explorer to monitor transactions.

Application Design Notes: Using Context and Transactions

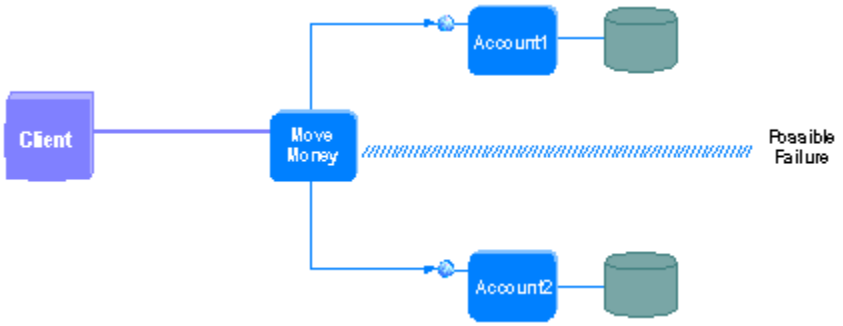
Using transactional components provides atomicity and simplified error recovery.

See Also

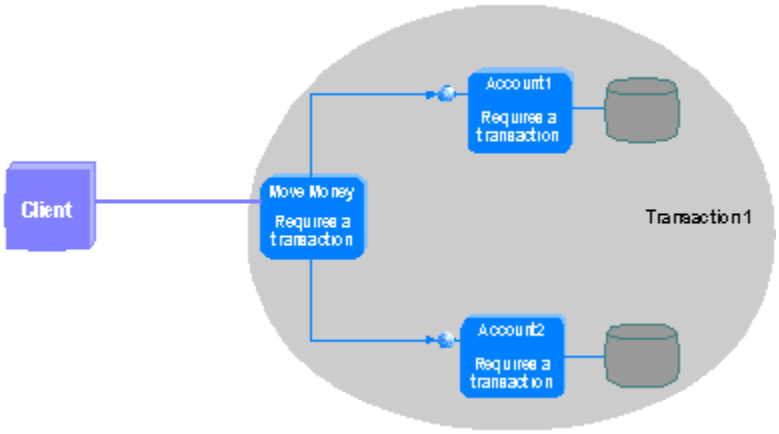
Transactions, Transaction Attributes, **ObjectContext** object, **CreateInstance** method

Scenario: Composing Work from Multiple Components Under the Same Transaction

For this scenario, you will add new functionality that allows you to transfer money between two accounts. To implement this, you add a new component, MoveMoney. MoveMoney creates Account and then calls it once for a credit or debit, or twice for a transfer, as shown here.



Because either MoveMoney or Account could fail at any point, all database updates need to be in the same transaction to ensure that the database remains consistent. To do this, you configure the MoveMoney and Account components in the Microsoft Transaction Server Explorer to require a transaction. Transaction Server ensures that all their objects' work is automatically done in the same transaction.



See Also

Transactions, Transaction Attributes

Creating the MoveMoney Component

To implement this scenario, you will add a new class module, MoveMoney, to the Account project. MoveMoney has a single method, Perform, which creates an Account object to perform the credit, debit, or transfer.

To create the MoveMoney component

1. Open the \Mts\Samples\Account.VB\Step4\Account.vbp project.

Click here to see the Perform method

2. Build the component as a dynamic-link library (DLL) and save it as \Mts\Samples\Account.VB\Step4\VBAcct.dll.

By adding a new class module, you have added a new COM component to this DLL. Therefore, you will need to delete the Account component in the Microsoft Transaction Server Explorer and then install the Account and the MoveMoney components.

To reinstall your components

1. Remove the Account and CreateTable components.

How?

2. Add the new components.

How?

Use the DLL you created in the previous procedure. You can find it in \Mts\Samples\Account.VB\Step4\VBAcct.dll.

MTS enlists a component in a transaction as specified by the component's transaction attribute. For this scenario, Account and MoveMoney run within the same transaction.

To set the transaction attributes for your components

1. For the Account and MoveMoney components, set the transaction attribute to **Requires a transaction**.
2. For the CreateTable component, set the transaction attribute to **Requires a new transaction**.

How?

The MoveMoney object uses **CreateInstance** to create the Account object. **CreateInstance** is a method on the context object. By using **CreateInstance**, the Account object created by MoveMoney shares context with MoveMoney.

```
Dim objAccount As Bank.Account
Set objAccount = _
    GetObjectContext.CreateInstance("Bank.Account")
```

Transactions are associated with an object's context. Because both MoveMoney and Account have a transaction attribute of **Requires a transaction**, the Account object will be enlisted within the same transaction as MoveMoney.

In Building Scalable Components, you learned how to use **SetComplete** to indicate that an object has finished its work and can be deactivated. For transactional components, calling **SetComplete** indicates that a transaction can be committed.

```
GetObjectContext.SetComplete
```

When the Perform method returns, the transaction attempts to commit. There is no guarantee that it will commit, however. If an error occurs, Perform instead calls **SetAbort**.

```
GetObjectContext.SetAbort
```

SetAbort also indicates that an object has finished its work, but that it isn't in a consistent state. When the Perform method returns after calling **SetAbort**, the attempt to commit the transaction won't succeed.

See Also

Transactions, Transaction Attributes, Context Objects, Creating MTS Objects, **ObjectContext** object, **CreateInstance** method, **GetObjectContext** method, **SetAbort** method, **SetComplete** method

Monitoring Transaction Statistics

You can use the Microsoft Transaction Server Explorer to monitor commit and abort statistics for transactions.

You can experiment with the MoveMoney and Account components to see how transactions are committed and aborted as you provide user input with the Bank client.

To monitor transactions

1. On the **Window** menu of the Transaction Server Explorer, click **New Window**.
2. In the left pane, click **Transaction Statistics**.
3. On the **Action** menu, click **Scope Pane** to hide the left pane of the Explorer.
4. Make sure that the Microsoft Distributed Transaction Coordinator (MS DTC) is running on your SQL Server computer. You can start MS DTC from the Transaction Server Explorer or from SQL Server.
5. Also make sure that you have the ODBC data source set up, and that SQL Server is running. Click ▶ to get information on how to do this.
6. Start the Bank client.

   Rearrange the windows so that you see the two Microsoft Transaction Server Explorer windows and the Bank client window.

To monitor a commit, click **Submit** in the Bank client. The Transaction Statistics window first indicates that one transaction is active, and indicates that one transaction was committed.

To monitor an abort, click **Debit** in the Bank client, and enter an amount for the transaction that is greater than the balance on your account. Click **Submit**. The Transaction Statistics window first indicates that one transaction is active, and then indicates that one transaction was aborted.

Try experimenting with **Transfer**. Verify that both objects are running within the same transaction by checking the balance of two accounts and performing a transfer that would overdraw from an account. Notice that both the credit and the debit are aborted.

See Also

Monitoring Transactions in MTS

Application Design Notes: Using Context and Transactions

Context simplifies defining transactions. A transaction is automatically started when a component is declared as transactional. Components don't need to add additional code to indicate the start and end of a transaction. Using context allows you to define the scope of a transaction.

Besides simplifying building components, automatic transaction enlistment also allows for reuse of existing components. Changing the transaction attribute is the only change to the Account component from the previous section, Building Scalable Components.

Creating the Account object from MoveMoney establishes MoveMoney as the root of the transaction. The root transaction attempts to commit after it has completed its work. If an Account object calls **SetAbort** to indicate that it cannot successfully commit its work, then when the root transaction attempts to commit, the entire transaction will fail.

In the case of a money transfer, this provides atomicity. If a credit succeeds, but insufficient funds prevent the debit from succeeding, then the credit will be rolled back from the database automatically. Thus, **SetAbort** provides simplified error recovery.

Context simplifies the development of the component. Each object independently acquires its own resources, performs its work, and indicates its own internal state by using **SetComplete** or **SetAbort** before returning.

See Also

Transactions, Transaction Attributes, Context Objects, **CreateInstance** method, **ObjectContext** object, **SetAbort** method, **SetComplete** method

Sharing State

This chapter shows you how to use the Shared Property Manager to share state among multiple Microsoft Transaction Server objects running in the same process.

Scenario: A Receipt Number Generator That Uses the Shared Property Manager

Create a component that uses the Shared Property Manager to generate a unique receipt number for each bank transaction.

Creating the Receipt Component

Create a **SharedProperty** object to get a new receipt number, with appropriate isolation and release modes for this scenario.

Application Design Notes: Sharing State by Using the Shared Property Manager

Learn some of the advantages of using the Shared Property Manager to manage shared state within a process.

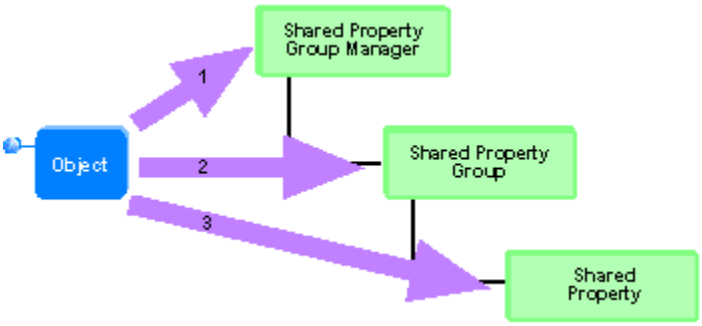
See Also

Resource Dispensers, Stateful Components, **SharedPropertyGroupManager** object

Scenario: A Receipt Number Generator That Uses the Shared Property Manager

This chapter introduces the Receipt component, which dispenses unique receipt numbers for fund transfers. When a bank transaction takes place, the MoveMoney object creates a Receipt object. The Receipt component contains a single method, GetNextReceipt.

GetNextReceipt uses the Shared Property Manager to get a unique receipt number. The Shared Property Manager has an object hierarchy as shown in the following figure:



Within a server process, there is only one instance of the **SharedPropertyGroupManager** object. The value of the receipt number is maintained by a **SharedProperty** object, which provides locking mechanisms to ensure that no two calls to GetNextReceipt retrieve the same value.

See Also

Resource Dispensers, Creating the Receipt Component, Application Design Notes: Sharing State by Using the Shared Property Manager, **SharedPropertyGroupManager** object, **SharedPropertyGroup** object, **SharedProperty** object

Creating the Receipt Component

The Receipt component contains a single method, GetNextReceipt. The Receipt object itself doesn't maintain the value of the receipt number between calls. The Shared Property Manager maintains these values. The Receipt object calls a **SharedProperty** object to get a new receipt number.

You will also add code to the MoveMoney component to call the Receipt component.

To create the Receipt Component

1. Start Microsoft Visual Basic and open the \Mts\Samples\Account.VB\Step5\Account.vbp project.

Click here to see the code for the Receipt component

Click here to see the code for the MoveMoney component

2. Build the component as a DLL and save it as \Mts\Samples\Account.VB\Step5\VBAcct.dll.

By adding a new class module, you add a new COM component to this DLL. Therefore, you need to delete the existing components in the Microsoft Transaction Server Explorer and then install the new components.

To reinstall your components

1. Remove the Account, MoveMoney, and CreateTable components from the Transaction Server Explorer.

How?

2. Add the new components. Use the DLL you created in \Mts\Samples\Account.VB\Step5\VBAcct.dll.

How?

To set the transaction attributes for your components

1. For the Account and MoveMoney components, set the transaction attribute to **Requires a transaction**.

How?

2. For the CreateTable component, set the transaction attribute to **Requires a new transaction**.
3. For the Receipt component, set the transaction attribute to **Does not support transactions**. This is the default value.

   Note that the Receipt component is not transactional because the receipts are maintained as properties in memory and aren't durable.

When you run the Bank Client, select the MoveMoney button under **Component**. You should see the response **Credit, balance is $ 1. (VB); Receipt No: #####**.

The various object creation methods for Shared Property Manager objects are designed for simplified coding. If the object doesn't exist, it will be created. If it already exists, the object is returned. GetNextReceipt makes the following method call to access the shared property group manager:

```
Set spmMgr = CreateObject _
    ("MTxSpm.SharedPropertyGroupManager.1")
```

This code works every time it is called. There is no need to check if the shared property group manager has already been created. Such behavior also ensures that only one instance of the **SharedPropertyGroupManager** object exists per server process.

For the **SharedPropertyGroup** and **SharedProperty** objects, a flag is returned to indicate whether the property group or property already exists. The following code shows how this flag is used to determine if the property needs to be initialized:

```
Set spmPropNextReceipt = _
    spmGroup.CreateProperty("Next", bResult)

' Set the initial value of the SharedProperty to
' 0 if the SharedProperty didn't already exist.
If bResult = False Then
```

```
    spmPropNextReceipt.Value = 0
End If
```

Access to shared properties is controlled through the **CreatePropertyGroup** method:

```
Set spmGroup = _
    spmMgr.CreatePropertyGroup("Receipt", _
    LockMethod, Process, bResult)
```

CreatePropertyGroup has two parameters, isolation mode and release mode. The isolation mode for the Receipt property group is set to **LockMethod**, which ensures that two instances of the Receipt object can't read or write to the same property during a call to GetNextReceipt. The release mode for the Receipt property group is set to **Process**, which maintains the property group until the server process is terminated.

See Also

Application Design Notes: Sharing State by Using the Shared Property Manager, **SharedPropertyGroupManager** object, **CreateProperty** methodasmthCreatePropertyvb, **CreatePropertyGroup** methodasmthCreatePropertyGroupvb

Application Design Notes: Sharing State by Using the Shared Property Manager

Using the Shared Property Manager makes sharing state in a multiuser environment as easy as it is in a single-user environment. Without the use of the Shared Property Manager, the application would require much more code that has nothing to do with the business problem at hand.

One alternate way to create the same functionality would be to maintain the receipt number as a member variable of the Receipt component. However, this complicates coding the Receipt component immensely. The Receipt component would have to remain persistent in memory during the life of the server process. This would require the following additional code:

- Referencing all instances of MoveMoney to the Receipt object.
- Maintaining a locking mechanism to prevent concurrent access to the Receipt object.

Even after adding this code, the application wouldn't be extensible for additional shared properties. The Shared Property Manager is another example of how Microsoft Transaction Server provides the infrastructure for server applications so that you can concentrate on coding business logic.

Location Transparency and the Shared Property Manager

For objects to share state, they all must be running in the same server process with the Shared Property Manager.

To maintain location transparency, it's a good idea to limit the use of a shared property group to objects created by the same component, or to objects created by components implemented within the same DLL. When components provided by different DLLs use the same shared property group, you run the risk of an administrator moving one or more of those components into a different package. Because components in different packages generally run in different processes, objects created by those components would no longer be able to share properties.

See Also

Resource Dispensers, Stateful Components, **SharedPropertyGroupManager** object

Stateful Components

This section discusses stateful components and outlines some of the issues associated with writing stateful application components.

Scenario: Holding State in the MoveMoney Component

Consider the design alternative of holding state within objects.

Adding a New Method to the MoveMoney Component

Add the StatefulPerform method, which uses MoveMoney to maintain account number values.

Application Design Notes: The Trade-offs of Using Stateful Objects

Learn how holding state in objects affects the application behavior within the Microsoft Transaction Server run-time environment.

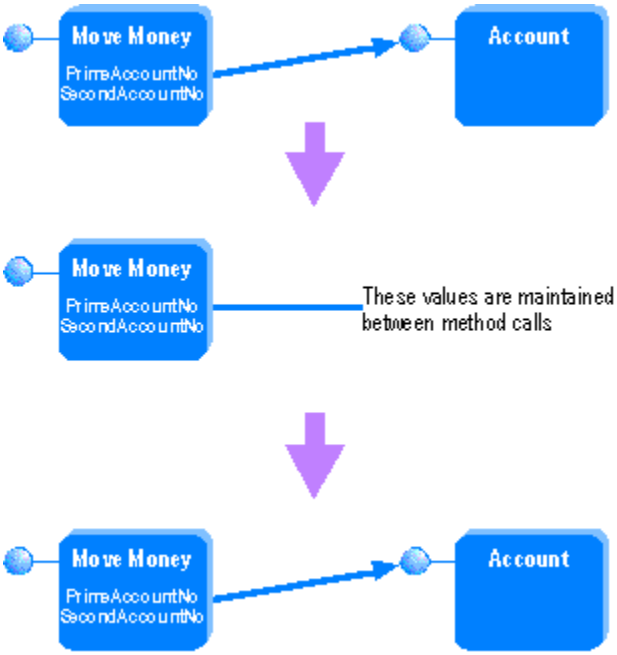
See Also

Transactions

Scenario: Holding State in the MoveMoney Component

Building stateful objects is a useful approach in application design. However, such design can have performance trade-offs. This section demonstrates how holding state in objects affects the application behavior within the Microsoft Transaction Server run-time environment.

You will modify the MoveMoney component to be stateful by adding the StatefulPerform function to MoveMoney. StatefulPerform is called when you click **Stateful MoveMoney** on the Sample Bank client. This new function causes MoveMoney to retain data in member variables between method calls.



See Also

Transactions, Deactivating Objects, Context Objects, Stateful Components

Adding a New Method to the MoveMoney Component

To implement the scenario for this section, you will add a method similar to Perform, named StatefulPerform, which uses class member variables to set account numbers. Thus, MoveMoney becomes a stateful object when StatefulPerform is called.

To add a new function to the MoveMoney component

1. Open the \Mts\Samples\Account.VB\Step6\Account.vbp project.

Click here to see the StatefulPerform method

2. Build the component as a dynamic-link library (DLL) and save it as \Mts\Samples\Account.VB\Step6\VBAcct.dll.

The code for StatefulPerform calls the Perform method. The methods differ in how the account numbers are set. Class member variables for each account must be set before calling StatefulPerform, whereas Perform passes the account numbers by value through function parameters.

When you click the **MoveMoney** option in the Sample Bank client, it calls the following code to initialize the function:

```
StatefulPerform = Perform(PrimeAccount, SecondAccount, lngAmount,
lngTranType)
```

When you click the **Stateful MoveMoney** option, the Sample Bank client calls the following code to initialize the function:

```
obj.PrimeAccount = PrimeAcct
obj.SecondAccount = lSecondAcct
Res = obj.StatefulPerform(CLng(Amount), TranType)
```

The PrimeAccount and SecondAccount properties are actually separate class member variables on the MoveMoney object. Note that the PrimeAccount and SecondAccount properties aren't accessed through the Shared Property Manager properties; the MoveMoney object controls getting and setting the account number values, thus making the MoveMoney object stateful.

Run the Bank client with the **MoveMoney** option. Then run it again with the **Stateful MoveMoney** option. You should notice that the stateless version is slightly faster. Try running multiple Bank clients with concurrent transactions. You should notice that the stateless version performs significantly better. The next section explains why.

See Also

Transactions, Deactivating Objects, Context Objects, Stateful Components

Application Design Notes: The Trade-offs of Using Stateful Objects

This section explains the trade-offs of using stateful objects in your applications.

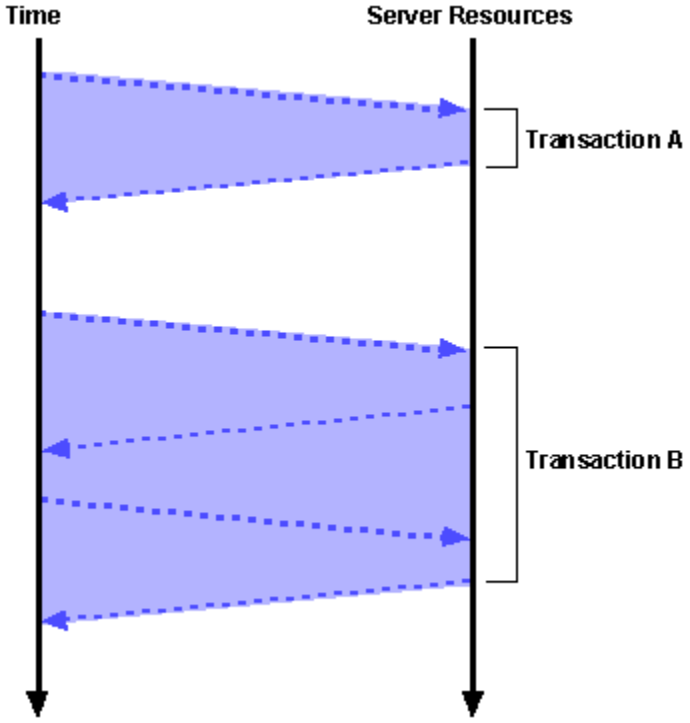
Why does MoveMoney outperform Stateful MoveMoney?

In the previous section, you saw that the time per transaction in **MoveMoney** and **Stateful MoveMoney** using a single Sample Bank client is nearly the same. However, as the number of concurrent transactions increases, **MoveMoney** begins to outperform **Stateful MoveMoney** significantly . At first glance, the code doesn't seem to account for the lag.

Class member variables for each account must be set before calling StatefulPerform, whereas Perform passes the account numbers by value through function parameters. The call to return the value of the account number in the MoveMoney object isn't an intensive operation. So what explains the performance degradation?

The reason is that Microsoft Transaction Server cannot commit transactions until it completes a method call. To maintain internal state, additional method calls are made on the MoveMoney object, thereby delaying the object from completing its work. This delay may cause server resources, such as database connections, to be held longer, therefore decreasing the amount of resources available for other clients. In other words, the application won't scale well.

The following diagram illustrates this point. The arrow on the left indicates time, which translates into performance. The arrow on the right indicates the server resources consumed, which translates into throughput. Transaction A represents a call made to stateless objects. On return from the method call, Transaction Server determines that the transaction can be committed, allowing the object to release its resources and be deactivated. On the other hand, Transaction B holds state between method calls, which increases the time that the server holds onto resources for that transaction. As the number of clients increases, so does the time required for transactions to be completed.



Another Pitfall When Using Stateful Objects

Examine the following excerpt from the Sample Bank client code (some code has been omitted for clarity).

```
For i = 1 To nTrans
   .
   .
   .
   obj.PrimeAccount = PrimeAcct
   obj.SecondAccount = lSecondAcct
   Res = obj.StatefulPerform(CLng(Amount), TranType
   .
   .
   .
Next i
```

Because the account numbers don't change, you might be inclined to rearrange the code as follows:

```
obj.PrimeAccount = PrimeAcct
obj.SecondAccount = lSecondAcct

For i = 1 To nTrans
   Res = obj.StatefulPerform(CLng(Amount), TranType
Next i
```

If you modify the code and then run the Sample Bank client for multiple transactions, the application fails on the second transaction. Why?

The answer is subtle. MoveMoney uses **SetComplete** to notify Transaction Server that it has completed its work. At this point, the MoveMoney object is deactivated. In the process of deactivation, all of the object's member variables are reinitialized. The next call to MoveMoney causes just-in-time activation. The activated object is now in its initial state, meaning the values of PrimeAccountNo and SecondAccountNo are both zero. Thus, the next call to StatefulPerform fails because of an invalid account number.

This is yet another reason to be careful when maintaining state in objects. Clients of application objects must be aware of how an object uses **SetComplete** to ensure that any state the object maintains won't be needed after the object undergoes just-in-time activation.

See Also

Transactions, Deactivating Objects, Context Objects, Stateful Components, **ObjectContext** object, **SetComplete** method

Multiple Transactions

This section explains the benefits of distributing work among multiple transactions.

Scenario: Storing Receipt Numbers in a Database

Add the UpdateReceipt component, which stores a maximum receipt number in a database and runs in a new transaction.

Creating the UpdateReceipt Component

Create the UpdateReceipt component, and modify the Receipt component to use UpdateReceipt.

Application Design Notes: Using Separate Transactions

Learn why this scenario requires separate transactions.

See Also

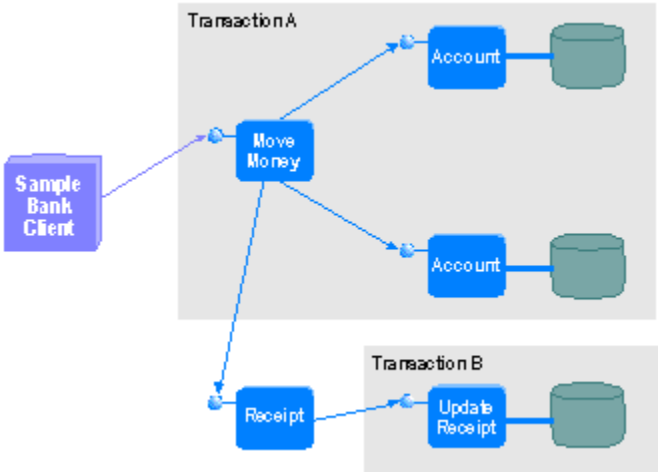
Transactions, Transaction Attributes, Activities

Scenario: Storing Receipt Numbers in a Database

In Sharing State, you added the Receipt component, which assigns a unique receipt number to each monetary transaction. The Shared Property Manager maintains these values, so they exist for the duration of the server process. In this section, you will add code to store a maximum receipt number in a database. Storing them makes receipt numbers unique beyond the life of the server process.

You will create the UpdateReceipt component, which stores a maximum receipt number in a database. When this maximum is reached, which happens on every one-hundredth transaction, UpdateReceipt adds 100 to the maximum receipt value and updates the database.

You will install the UpdateReceipt component so that it creates a new transaction, separate from those of the Account objects. The section will then discuss the advantages of using multiple transactions in this scenario. The application looks like the following figure (the Shared Property Manager and its associated objects are omitted for clarity):



See Also

Transactions, Transaction Attributes, Activities, Sharing State

Creating the UpdateReceipt Component

To implement the scenario for this section, you will build the UpdateReceipt component. You will also modify the Receipt component's Update method to use UpdateReceipt. Update adds 100 to the maximum receipt value stored in the database.

Click here to see the Update method

You also need to add code to the GetNextReciept method of the Receipt component to check whether the maximum receipt value has been reached. If so, the Update method is called.

Click here to see the GetNextReceipt method

To create the UpdateReceipt component

1. Open the \Mts\Samples\Account.VB\Step7\Account.vbp project.
2. Build the component as a dynamic-link library (DLL) and save it as \Mts\Samples\Account.VB\Step7\VBAcct.dll.

By adding a new class module, you add a new COM component to this DLL. Therefore, you need to delete the existing components in the Microsoft Transaction Server Explorer and then install the new components.

To reinstall your components

1. Remove the Account, MoveMoney, CreateTable, and Receipt components from the Transaction Server Explorer.

How?

2. Add the new components. Use the DLL you created in \Mts\Samples\Account.VB\Step7\VBAcct.dll.

How?

To set the transaction attributes for your components

1. For the Account and MoveMoney components, set the transaction attribute to **Requires a transaction**.

How?

2. For the Receipt component, set the transaction attribute to **Does not support transactions**. This is the default value.
3. For the CreateTable and UpdateReceipt components, set the transaction attribute to **Requires a new transaction**.

The code you added here is similar to the code you added in "Building Transactional Components." However, choosing **Requires a new transaction** causes the UpdateReceipt component to run in a new transaction. The next section discusses how this affects application behavior.

See Also

Transactions, Transaction Attributes

Application Design Notes: Using Separate Transactions

In Building Transactional Components, you saw the benefits of composing work under a transaction. The scenario in this section demonstrates a case in which using multiple transactions within an activity is required.

The major functional change in this scenario is the addition of the UpdateReceipt component, which makes the maximum receipt number durable by storing it in a database. As in Sharing State, the Shared Property Manager stores the receipt number. On every 100 transactions, the value in the database is incremented by 100. This dispenses a block of receipt numbers that are assigned to the next 100 transactions.

The UpdateReceipt component has a transaction attribute of **Requires a new transaction**. This guarantees that UpdateReceipt's work happens in a separate transaction. Thus, there is no connection between the success or failure of Account's work and UpdateReceipt's work.

This might appear to lower the fault tolerance of the application. For example, if the Account object aborts the transaction, a receipt number is still assigned. Therefore, skips in the receipt number sequence are possible. However, the application doesn't really need consecutively increasing receipt numbers--it just requires that there be no duplicate receipts. In this scenario, it's more important for the monetary transaction to be completed properly. Furthermore, requesting an update on every one-hundredth transaction improves performance by conserving calls to the database.

Composing both database updates under a single transaction would reduce the application's scalability. Even though UpdateReceipt is a simple update, it would consume more server resources because the database connection would have to be maintained until the Account object has completed its work. Thus, locks would be held longer than necessary, preventing other clients from writing to the database. Only when all work has been completed could these resources be freed.

See Also

Transactions, Transaction Attributes

Secured Components

This chapter shows how to use Microsoft Transaction Server's security features to restrict the use of application features to designated users.

Scenario: Adding Role Checking to the MoveMoney and Account Components

Add role checking to the MoveMoney and Account components to limit the transaction amount for designated users.

Using IsCallerInRole in the MoveMoney and Account Components

Use the **IsCallerInRole** method in the MoveMoney and Account components to verify that the user running the Bank client is a manager.

Application Design Notes: Using Roles

Learn how roles are useful in building secured components and how security boundaries are determined.

See Also

Programmatic Security

Scenario: Adding Role Checking to the MoveMoney and Account Components

For this scenario, you will limit which users have the ability to perform transactions of more than $500. You will add code to the MoveMoney and Account components that checks to see if the user of the Bank client is a manager. This is accomplished by defining a Manager role. Roles provide the flexibility a developer needs to build secured components without tying the implementation to a specific deployment domain.

See Also

Programmatic Security, Application Design Notes: Using Roles

Using IsCallerInRole in the MoveMoney and Account Components

You will add the **IsCallerInRole** method to the MoveMoney and Account components to verify that the user running the Bank client is a manager. This additional code is the same for both components. You must modify both components because clicking **Account** in the Bank client doesn't use the MoveMoney component when the Sample Bank application runs.

To use IsCallerInRole in the MoveMoney and Account components

1. Open the \Mts\Samples\Account.VB\Step8\Account.vbp project.

Click here to see the modified MoveMoney component

Click here to see the modified Account component

2. Build the component as a DLL and save it as \Mts\Samples\Account.VB\Step8\VBAcct.dll.

IsCallerInRole is a method on an object's context. **IsCallerInRole** returns TRUE if the direct caller of that object is assigned to a given role. You will use **IsCallerInRole** in the MoveMoney and Account components to verify if the caller of an object – in this case the user running the Bank client – is a manager.

```
If (lngAmount > 500 Or lngAmount < -500) Then
    If Not GetObjectContext.IsCallerInRole("Managers") Then
        Err.Raise Number:=APP_ERROR, _
        Description:="Need 'Managers' role for amounts over $500"
    End If
End If
```

Before you can use the new MoveMoney and Account components, you must create the role. The Manager role must exist before the call to **IsCallerInRole**; otherwise, you will get an error.

Note that the source code is bound to a role name scoped to a package. This creates a dependancy between the source and the package definition that must be considered when making modifications to a Package's security configuration, such as deleting a role.

To define a role for the Sample Bank package

1. Start the Microsoft Transaction Server Explorer.

   If you are currently running Sample Bank, you must shut down the associated server process to change security properties.

How?

2. Create a role named Manager.

How?

3. Assign users to the role. If you have access to more than one Windows NT account, you may want to exclude some user accounts from the Manager role to see the role checking in effect.

How?

Run the Bank client. If you are logged on as a user in the Manager role, you will be able to perform transactions of any amount. However, if you are logged on as a user who isn't in the Manager role, you will get a warning message when attempting a transaction of more than $500. The transaction will then abort. If you don't have access to more than one account, try removing your user account from the role to see the role checking enforced.

How?

See Also

Programmatic Security, Enabling MTS Package Security, Application Design Notes: Using Roles, **IsCallerInRole** method

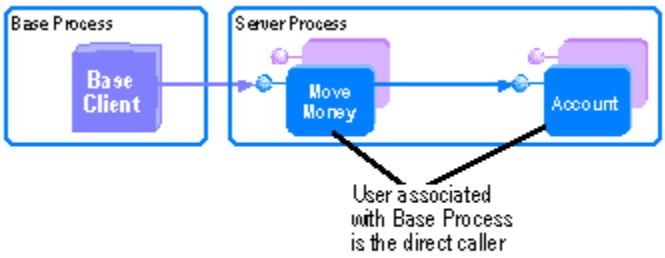
Application Design Notes: Using Roles

This section explains how roles are useful in building secured components. It also discusses how deployment configuration determines security boundaries.

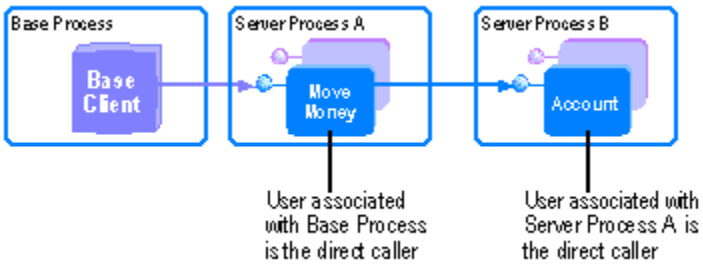
Roles

A role is a symbolic name that defines a group of users for a package of components. Roles extend Windows NT security to allow a developer to build secured components in a distributed application. In this scenario, the Manager role is defined at development time, but not yet bound to specific users. For each deployment of the application, the administrator can then assign the users and groups to a role in order to customize the application for his or her business.

In this scenario, role checking is done by the MoveMoney and Account objects. When both MoveMoney and Account objects are used, the second check (on the Account object) is redundant. However, it yields the same result, because **IsCallerInRole** applies to the direct caller, and both the MoveMoney and Account objects run in the same server process.



If you place the MoveMoney and Account components into separate packages, the components run in separate server processes. In this scenario, calling **IsCallerInRole** on the Account object context would check if the MoveMoney object's associated server process is running in the Manager role. MoveMoney is now the direct caller because a process boundary has been crossed.



The Account object runs under a package identity that gives that process full access to the bank account database. Account objects have the authority to update any account for any amount. Roles provide a means of permitting and denying access to objects. Once this permission is granted, the client, in effect, has the same access rights as the server process.

When you configured the package, you chose a package identity of **Interactive User**. In a real-world scenario, packages are more likely to run as a specific user, such as SampleBank, which has access rights to the database.

Returning to the scenario where you split the MoveMoney and Account components into separate packages, running as the SampleBank user solves the role checking problem. Adding the SampleBank user to the Manager role would allow the second **IsCallerInRole** check (on the Account object) to always succeed.

Security Boundaries Are Process-Wide

Transaction Server security is enabled only within a server process. Because the MoveMoney component is configured to run within a server process, role checking is enabled.

If you configure the Sample Bank components to run in-process, role checking would be disabled. In this case, **IsCallerInRole** always returns TRUE, which means the direct caller would always pass the authorization check.

You could use the **IsSecurityEnabled** method to check if Transaction Server security can be used. **IsSecurityEnabled** returns FALSE when the object runs in-process. Using **IsSecurityEnabled**, you could rewrite the role-checking code to disable transactions when objects aren't running in a secured environment.

In-process components share the same level of trust as the base client. Because of this, it isn't recommended that you deploy your secured components to be loaded in-process with their clients.

See Also

Programmatic Security, Enabling MTS Package Security, **IsCallerInRole** method, **IsSecurityEnabled** method

MTS Reference

Visual Basic

- Functions
- Methods
- Objects
- Properties
- Language Summary

Visual C++

- Functions
- Interfaces
- Methods
- Language Summary

Visual J++

- Interfaces
- Methods
- Language Summary

Functions (Visual Basic)

GetObjectContext

SafeRef

Objects (Visual Basic)

You use the following object in base clients:

TransactionContext

You use the following objects in components:

ObjectContext

SecurityProperty

SharedProperty

SharedPropertyGroup

SharedPropertyGroupManager

Methods (Visual Basic)

Abort

Activate

CanBePooled

Commit

Count

CreateInstance

CreateInstance (Transaction Context)

CreatePropertyGroup

CreateProperty

CreatePropertyByPosition

Deactivate

DisableCommit

EnableCommit

GetDirectCallerName

GetDirectCreatorName

GetOriginalCallerName

GetOriginalCreatorName

IsCallerInRole

IsInTransaction

IsSecurityEnabled

Item

SetAbort

SetComplete

Properties (Visual Basic)

Group

Property

PropertyByPosition

Value

Language Summary (Visual Basic)

Functions

GetObjectContext Function

SafeRef Function

ObjectContext Object

ObjectContext Object

ObjectContext Methods

Count

CreateInstance

DisableCommit

EnableCommit

IsCallerInRole

IsInTransaction

IsSecurityEnabled

Item

Security

SetAbort

SetComplete

ObjectControl Object

ObjectControl Object

ObjectControl Methods

Activate

CanBePooled

Deactivate

SecurityProperty Object

SecurityProperty Object

SecurityProperty Methods

GetDirectCallerName

GetDirectCreatorName

GetOriginalCallerName

GetOriginalCreatorName

SharedProperty Object

SharedProperty Object

SharedProperty Properties

Value

SharedPropertyGroup Object

SharedPropertyGroup Object

SharedPropertyGroup Properties and Methods

CreateProperty

CreatePropertyByPosition

Property

PropertyByPosition

SharedPropertyGroupManager Object

SharedPropertyGroupManager Object

SharedPropertyGroupManager Methods

CreatePropertyGroup

Group

TransactionContext Object

TransactionContext Object

TransactionContext Methods

Abort

Commit

CreateInstance

Functions (Visual C++)

GetObjectContext

SafeRef

Interfaces (Visual C++)

You use the following object in base clients:

ITransactionContext

You use the following objects in components:

IGetContextProperties

IObjectContext

IObjectContextActivity

IObjectControl

ISecurityProperty

ISharedProperty

ISharedPropertyGroup

ISharedPropertyGroupManager

Methods (Visual C++)

Abort

Activate

CanBePooled

Commit

Count

CreateInstance

CreateInstance (Transaction Context)

CreatePropertyGroup

CreateProperty

CreatePropertyByPosition

Deactivate

DisableCommit

EnableCommit

EnumNames

get__NewEnum

get_Group

get_Property

get_PropertyByPosition

get_Value

GetActivityId

GetDirectCallerSID

GetDirectCreatorSID

GetOriginalCallerSID

GetOriginalCreatorSID

GetProperty

IsCallerInRole

IsInTransaction

IsSecurityEnabled

put_Value

ReleaseSID

SetAbort

SetComplete

Language Summary (Visual C++)

Functions

GetObjectContext Function

SafeRef Function

IGetContextProperties Interface

IGetContextProperties Interface

IGetContextProperties Methods

Count

EnumNames

GetProperty

IObjectContext Interface

IObjectContext Interface

IObjectContext Methods

CreateInstance

DisableCommit

EnableCommit

IsCallerInRole

IsInTransaction

IsSecurityEnabled

SetAbort

SetComplete

IObjectContextActivity Interface

IObjectContextActivity Interface

IObjectContextActivity Methods

IObjectContextActivity::GetActivityId Method

IObjectControl Interface

IObjectControl Interface

IObjectControl Interface Methods

Activate

CanBePooled

Deactivate

ISecurityProperty Interface

ISecurityProperty Interface

ISecurityProperty Interface Methods

GetDirectCallerSID

GetDirectCreatorSID

GetOriginalCallerSID

GetOriginalCreatorSID

ReleaseSID

ISharedProperty Interface

ISharedProperty Interface

ISharedProperty Interface Methods

put_Value

get_Value

ISharedPropertyGroup Interface

ISharedPropertyGroup Interface

ISharedPropertyGroup Methods

CreateProperty

CreatePropertyByPosition

get_Property

get_PropertyByPosition

ISharedPropertyGroupManager Interface

ISharedPropertyGroupManager Interfacea

ISharedPropertyGroupManager Methods

CreatePropertyGroup

get_Group

get__NewEnum

ITransactionContextEx Interface

ITransactionContextEx Interface

ITransactionContextEx Methods

Abort

Commit

CreateInstance

Interfaces (Visual J++)

You use the following object in base clients:

ITransactionContext

You use the following objects in components:

IGetContextProperties

IObjectContext

IObjectControl

ISharedProperty

ISharedPropertyGroup

ISharedPropertyGroupManager

Methods (Visual J++)

Abort

Activate

CanBePooled

Commit

Count

CreateInstance

CreateInstance (Transaction Context)

CreatePropertyGroup

CreateProperty

CreatePropertyByPosition

Deactivate

DisableCommit

EnableCommit

EnumNames

get_NewEnum

getGroup

getProperty

GetProperty

getPropertyByPosition

getValue

IsCallerInRole

IsInTransaction

IsSecurityEnabled

MTx.GetObjectContextasfctGetObjectContextvj

MTx.SafeRef

putValueasmthPutValuevj

SetAbort

SetComplete

Language Summary (Visual J++)

Context Class

Context Class

Context Methods

createInstance

disableCommit

enableCommit

getPropertyNames

getProperty

isCallerInRole

isInTransaction

isSecurityEnabled

setAbort

setComplete

Mtx.GetObjectContext

MTx.GetObjectContext

MTx.GetObjectContext Method

MTx.SafeRef Method

IGetContextProperties Interface

IGetContextProperties Interface

IGetContextProperties Methods

Count

EnumNames

GetProperty

IObjectContext Interface

IObjectContext Interface

IObjectContext Methods

CreateInstance

DisableCommit

EnableCommit

IsCallerInRole

IsInTransaction

IsSecurityEnabled

SetAbort

SetComplete

IObjectControl Interface

IObjectControl Interface

IObjectControl Methods

Activate

CanBePooled

Deactivate

ISharedProperty Interface

ISharedProperty Interface

ISharedProperty Methods

putValue

getValue

ISharedPropertyGroup Interface

ISharedPropertyGroup Interface

ISharedPropertyGroup Methods

CreateProperty

CreatePropertyByPosition

getProperty

getPropertyByPosition

ISharedPropertyGroupManager Interface

ISharedPropertyGroupManager Interface

ISharedPropertyGroupManager Methods

CreatePropertyGroup

getGroup

get_NewEnum

ITransactionContextEx Interface

ITransactionContextEx Interface

ITransactionContextEx Methods

Abort

Commit

CreateInstance

IGetContextProperties Interface

The **IGetContextProperties** interface provides access to context object properties.

Remarks

The header file for the **IGetContextProperties** interface is mtx.h.

The **IGetContextProperties** interface exposes the following methods.

Method	Description
Count	Returns the number of properties associated with the context object.
EnumNames	Returns a reference to an enumerator that you can use to iterate through all the context object properties.
GetProperty	Returns a context object property.

See Also

Context Objects

IGetContextProperties::Count Method

Returns the number of context object properties.

Provided By

IGetContextProperties

HRESULT IGetContextProperties::Count (
long * *plCount*);

Parameters

plCount
[out] The number of properties.

Return Values

S_OK
A valid count is returned in the *plCount* parameter.

E_INVALIDARG
The valid count could not be returned.

Example

IGetContextProperties::EnumNames Method

Returns a reference to an enumerator that you can use to iterate through all the context object properties.

Provided By

IGetContextProperties

HRESULT IGetContextProperties::EnumNames (
IEnumNames ** *ppenum*);

Parameters

ppenum
[out] A reference to the **IEnumNames** interface on a new enumerator object that you can use to iterate through all the context object properties.

Return Values

S_OK
A reference to the requested enumerator is returned in the *ppenum* parameter.

E_INVALIDARG
The requested enumerator could not be returned.

Remarks

You use the **EnumNames** method to obtain a reference to an enumerator object. The returned **IEnumNames** interface exposes several methods you can use to iterate through a list of BSTRs representing context object properties. Once you have a name, you can use the **GetProperty** method to obtain a reference to the context object property it represents. See the COM documentation for information on enumerators.

As with any COM object, you must release an enumerator object when you're finished using it.

Example

IGetContextProperties::GetProperty Method

Returns a context object property.

Provided By

IGetContextProperties

HRESULT IGetContextProperties::GetProperty (
    BSTR *name*
    VARIANT * *pProperty*);

Parameters

name
[in] The name of the context object property to be retrieved.

pProperty
[out] The returned pointer to the property.

Return Values

S_OK
The property was returned successfully.

S_FALSE
The property was not found.

E_INVALIDARG
The *name* parameter was invalid.

Remarks

You can use **GetProperty** to retrieve the following Microsoft Internet Information Server (IIS) intrinsic objects:

- Request
- Response
- Server
- Application
- Session

For more information, see the IIS documentation.

To retrieve an IIS object, call **QueryInterface** using the VT_DISPATCH member of the returned VARIANT for the interface to the IIS object (for example **IReponse** for the Response object).

Note Context properties are not available in MTS 1.0 server processes.

Example

Count, EnumNames, GetProperty Methods Example

```
#include "stdafx.h"
#include <initguid.h>
#include "asptlb.h"
#include "mtx.h"

HRESULT hr = NOERROR;

// Get the context object
CComPtr<IObjectContext> pObjectContext;
hr = GetObjectContext(&pObjectContext);

// Get the Response object
CComVariant v;
CComBSTR bstr(L"Response");
CComQIPtr<IGetContextProperties,   &IID_IGetContextProperties>
pProps(pObjectContext);
hr = pProps->GetProperty(bstr, &v);
CComPtr<IDispatch> pDisp;
pDisp = V_DISPATCH(&v);
CComQIPtr<IResponse, &IID_IResponse> pResponse(pDisp);

// Print number of properties
long lCount;
hr = pProps->Count(&lCount);
bstr = L"<p>Number of properties: ";
CComBSTR bstrCount;
VarBstrFromI4(lCount, 0, 0, &bstrCount);
bstr.Append(bstrCount);
bstr.Append(L"</p>");
v = bstr;
hr = pResponse->Write(v);

// Iterate over properties collection and print the
// names of the properties
CComPtr<IEnumNames> pEnum;
CComBSTR bstrTemp;
pProps->EnumNames(&pEnum);
for ( int i = 0; i < lCount; ++i )
{
   pEnum->Next(1, &bstr, NULL);
   bstrTemp = L"<p>";
   bstrTemp.Append(bstr);
   bstrTemp.Append(L"</p>");
   v = bstrTemp;
   hr = pResponse->Write(v);
}
```

IGetContextProperties Interface

The **IGetContextProperties** interface provides access to context object properties.

Remarks

New applications should use the **Context** class.

The **IGetContextProperties** interface is declared in the package `com.ms.mtx`.

The **IGetContextProperties** interface exposes the following methods.

Method	Description
Count	Returns the number of properties associated with the context object.
EnumNames	Returns a reference to an enumerator that you can use to iterate through all the context object properties.
GetProperty	Returns a context object property.

See Also

Context Objects

IGetContextProperties.Count Method

Returns the number of context object properties.

Provided By

IGetContextProperties

int Count( );

Return Value

The number of properties.

Example

IGetContextProperties.EnumNames Method

Returns a reference to an enumerator that you can use to iterate through all the context object properties.

New applications should call **getPropertyNames**.

Provided By

IGetContextProperties

IEnumNames EnumNames ( );

Return Value

A reference to the **IEnumNames** interface on a new enumerator object that you can use to iterate through the list of all the shared property groups in the process.

Remarks

You use the **EnumNames** method to obtain a reference to an enumerator object. The returned **IEnumNames** interface exposes several methods you can use to iterate through a list of string expressions representing context object properties. Once you have a name, you can use the **GetProperty** method to obtain a reference to the context object property it represents. See the COM documentation for information on enumerators.

Example

IGetContextProperties.GetProperty Method

Returns a context object property.

New applications should call **getProperty**.

Provided By

IGetContextProperties

Variant GetProperty(
String *name***);**

Parameters

name
[in] The name of the context object property to be retrieved.

Return Value

The requested context object property.

Remarks

You can use **GetProperty** to retrieve the following Microsoft Internet Information Server (IIS) intrinsic objects:

- Request
- Response
- Server
- Application
- Session

For more information, see the IIS documentation.

To retrieve an IIS object, use the **getDispatch** method of returned Variant and cast this value to the interface to the IIS object (for example **IReponse** for the Response object).

Note Context properties are not available in MTS 1.0 server processes.

Example

Count, EnumNames, GetProperty Methods Example

```
import com.ms.com.*;
import com.ms.mtx.*;
import asp.*;

// Get the context object
IGetContextProperties icp;
IResponse iresp;
Variant av = new Variant();
icp = (IGetContextProperties)MTx.GetObjectContext();

// Get the Response object
av = icp.GetProperty("Response");
iresp = (IResponse) av.getDispatch();
av.VariantClear();

// Print number of properties
String str;
int pc;
pc = icp.Count();
str = "<p>Number of properties: " + pc + "</p>";
av.putString(str);
iresp.Write (av);
av.VariantClear();

// Iterate over properties collection and print the
// names of the properties
IEnumNames iEnum = null;
int howmany = 1;
String[] names = new String[1];
int fetched;
iEnum = icp.EnumNames();
for ( int i = 0; i < pc; ++i )
{
   fetched = iEnum.Next( howmany, names);
   str = "<p>" + names[0] + "</p>";
   av.putString(str);
   iresp.Write (av);
   av.VariantClear();
}
```

ObjectControl Object

You implement the **ObjectControl** interface when you want to define context-specific initialization and cleanup procedures for your MTS objects and specify whether or not the objects can be recycled. Implementing the **ObjectControl** interface is optional.

Remarks

To use the **ObjectControl** object, you must set a reference to the Microsoft Transaction Server Type Library (MTxAS.dll).

If you implement the **ObjectControl** interface in your component, the MTS run-time environment automatically calls the **ObjectControl** methods on your objects at the appropriate times.

When an object supports the **ObjectControl** interface, MTS calls its **Activate** method once for each time the object is activated. The **Activate** method is called before any of the object's other methods are called. You can use this method to perform any context-specific initialization an object may require.

MTS calls the object's **Deactivate** method each time the object is deactivated. This can be the result of the object returning from a method in which it calls **SetComplete** or **SetAbort**, or due to the root of the object's transaction, causing the transaction to complete. You use the **Deactivate** method to clean up state that you initialized in the **Activate** method.

After calling the **Deactivate** method, the MTS run-time environment calls the **CanBePooled** method. If this method returns **True**, the deactivated object is placed in an object pool for reuse. If the **CanBePooled** method returns **False**, the object is released in the usual way.

Note On systems that don't support object pooling, the value returned by this method is ignored.

The **ObjectControl** interface is not accessible to an object's clients or to the object itself. Only the MTS run-time environment can invoke the **ObjectControl** methods.

The **ObjectControl** interface provides the following methods.

Method	Description
Activate	Allows an object to perform context-specific initialization whenever it's activated. This method is called by the MTS run-time environment before any other methods are called on the object.
CanBePooled	Allows an object to notify the MTS run-time environment of whether it can be pooled for reuse. Return **True** if you want instances of this component to be pooled, or **False** if not.
Deactivate	Allows an object to perform whatever cleanup is necessary before it's recycled or destroyed. This method is called by the MTS run-time environment whenever an object is deactivated.

See Also

Deactivating Objects, Object Pooling and Recycling

Activate Method

Implementing this method allows an MTS object to perform context-specific initialization whenever it's activated. This method is called by the MTS run-time environment before any other methods are called on the object.

Applies To

ObjectContext Object

Syntax

*objectcontrol.***Activate**

The *objectcontrol* placeholder represents is an object variable that evaluates to an **ObjectControl** object.

Remarks

To use the **ObjectControl** object, you must set a reference to the Microsoft Transaction Server Type Library (MTxAS.dll).

Whenever a client calls an MTS object that isn't already activated, the MTS run-time environment automatically activates the object. This is called just-in-time activation. For components that support the **ObjectControl** interface, MTS invokes the object's **Activate** method before passing the client's method call on to the object. This allows objects to do context-specific initialization.

If you need to perform any initialization that involves the **ObjectContext**, you should implement the **Activate** method and place all your context-specific initialization procedures there.

For example, you can use the **Activate** method to obtain a reference to an object's context and store it in a member variable. Then the context is available to any method that requires it, and you don't have to acquire a new one and then release it every time you use it. Once you have a reference to the object's context, you can use the **IObjectContext** methods to check whether security is enabled, whether the object is executing in a transaction, or whether the caller is in a particular role.

If you're enabling object recycling (returning **True** from the **CanBePooled** method), the **Activate** method must be able to handle both newly created and recycled objects. When the **Activate** method returns, there should be no distinguishable difference between a new object and a recycled one.

Example

See Also

Deactivating Objects, Object Pooling and Recycling

Activate Method Example

```
Implements ObjectControl
Dim context As ObjectContext
Option Explicit

Private Sub ObjectControl_Activate()
    ' Get a reference to the object's context here,
    ' so it can be used by any method that may be
    ' called during this activation of the object.
   Set context = GetObjectContext()
End Sub
```

CanBePooled Method

Implementing this method allows an MTS object to notify the MTS run-time environment of whether it can be pooled for reuse. Return **True** if you want the object to be pooled for reuse, or **False** if not.

Applies To

ObjectContext Object

Syntax

objectcontrol.**CanBePooled (True | False**)

The *objectcontrol* placeholder represents is an object variable that evaluates to an **ObjectControl** object.

Return Values

True
Notifies the MTS run-time environment that it can pool this object on deactivation for later reuse.

False
Notifies the MTS run-time environment that it should not pool this object on deactivation, but should release its last reference to the object so that the object will be destroyed.

Remarks

To use the **ObjectControl** object, you must set a reference to the Microsoft Transaction Server Type Library (MTxAS.dll).

When an object returns **True** from the **CanBePooled**method, it indicates to the MTS run-time environment that it can be added to an object pool after deactivation rather than being destroyed. Whenever an instance is required, one is drawn from the pool rather than created.

The way recycling works is that an object cleans itself up in its **Deactivate** method and is returned to an object pool. Later, when an instance of the same component is needed, the cleaned up object can be reused. For this to work, an object must be accessible on different threads each time it's activated. Recycling isn't possible under the apartment threading model because, in that model, although an object can be instantiated on any thread, it can only be used by the thread on which it was instantiated. If you want instances of a component to be recyclable, you should register the component with the **ThreadingModel** Registry value set to **Both**. This indicates to MTS that the component's objects can be called from different threads.

In MTS, these objects will run under the apartment threading model and won't be recycled even if they return **True** from the **CanBePooled** method. However, if you configure a component to support both threading models, the component will run under the current version of MTS and will also be able to take advantage of recycling as soon as it becomes available, without any changes to the code.

Deciding whether to enable recycling is a matter of weighing the costs and benefits. Recycling requires writing additional code, and there's a risk that some state may be inadvertently retained from one activation to the next. When you allow objects to be pooled, you have to be very careful in your **Activate** and **Deactivate** methods to ensure that a recycled object is always restored to a state that's equivalent to the state of a newly created object. Another consideration to take into account is the amount of resources required to maintain an object pool. Objects that hold a lot of resources can be expensive to pool. However, in certain situations, recycling can be extremely efficient, resulting in improved performance and increased scalability. The trade-off is between the cost of holding onto resources while objects are pooled (and inactive) versus the cost of creating and destroying the resources.

It's usually best to enable recycling for objects that cost more to create than they cost to reinitialize.

For example, if a component contains a complex structure, and that structure can be reused, it could save a lot of time if the structure didn't have to be recreated every time an instance of the component was activated. This is a case in which you might want to enable recycling, which you would do by returning **True** from the **CanBePooled** method.

The **Activate** method is called whether a new instance is created or a recycled instance is drawn from the pool. Similarly, the **Deactivate** method is called every time the object is deactivated, whether it's being destroyed or returned to the pool for recycling.

So, in this example, you'd use the object's **Activate** method to initialize, or reinitialize, the structure that's being reused, and you'd use the **Deactivate** method to restore the object to a state that the **Activate** method can handle. (The **Activate** method must be able to handle both new objects and reused objects drawn from the pool.) This combined use of the **Activate**, **Deactivate**, and **CanBePooled** methods eliminates the need to recreate reusable resources every time an instance is activated.

For some objects, recycling isn't efficient. For example, if an object acquires a lot of state during its lifetime that isn't reusable, and has little to do during its construction, it's usually cheaper to create a new instance whenever one is needed. In that case, you would return **False** from the **CanBePooled** method.

Note Returning **True** from the **CanBePooled** method doesn't guarantee that objects will be recycled; it only gives the MTS run-time environment permission to recycle them. On systems that don't support object pooling, a return value of **True** is ignored. Returning **False** from the **CanBePooled** method guarantees that instances of a component aren't recycled.

Example

See Also

Deactivating Objects, Object Pooling and Recycling

CanBePooled Method Example

```
Implements ObjectControl
Dim context As ObjectContext
Option Explicit

Private Function ObjectControl_CanBePooled() As Boolean
   ' This object should not be recycled,
   ' so return false.
   ObjectControl_CanBePooled = False
End Function
```

Deactivate Method

Implementing this method allows an MTS object to perform any cleanup required before it's recycled or destroyed. This method is called by the MTS run-time environment whenever an object is deactivated.

Applies To

ObjectContext Object

Syntax

objectcontrol.**Deactivate**

The *objectcontrol* placeholder represents is an object variable that evaluates to an **ObjectControl** object.

Remarks

To use the **ObjectControl** object, you must set a reference to the Microsoft Transaction Server Type Library (MTxAS.dll).

The MTS run-time environment calls the **Deactivate** method whenever an object that supports the **ObjectControl** interface is deactivated. An object is deactivated when it returns from a method in which it called **SetComplete** or **SetAbort**, when the transaction in which it executed is committed or aborted, or when the last client to hold a reference on it releases its reference.

If the component supports recycling (returns **True** from the **CanBePooled** method), you should use the **Deactivate** method to reset the object's state to the state in which the **Activate** method expects to find it. You can also use the **Deactivate** method to release the object's **ObjectContext** or to do other context-specific cleanup. Even if an object supports recycling, it can be beneficial to release certain reusable resources in its **Deactivate** method. For example, ODBC provides its own connection pooling. It's more efficient to pool a database connection in a general connection pool where it can be used by other objects than it is to keep it tied to a specific object in an object pool.

Example

See Also

Deactivating Objects, Object Pooling and Recycling

Deactivate Method Example

```
Implements ObjectControl
Dim objcontext As ObjectContext
Option Explicit

Private Sub ObjectControl_Deactivate()
    ' Perform any necessary cleanup here.
   Set objcontext = Nothing
End Sub
```

IObjectControl Interface

You implement the **IObjectControl** interface when you want to define context-specific initialization and cleanup procedures for your MTS objects and specify whether or not the objects can be recycled. Implementing the **IObjectControl** interface is optional.

Remarks

The header file for the **IObjectControl** interface is mtx.h.

If you implement the **IObjectControl** interface in your component, the MTS run-time environment automatically calls the **IObjectControl** methods on your objects at the appropriate times.

When an object supports the **IObjectControl** interface, MTS calls its **Activate** method once for each time the object is activated. The **Activate** method is called before any of the object's other methods are called. You can use this method to perform any context-specific initialization an object may require.

Note An object's context isn't available during object construction (from the object's class factory), so context-specific initialization can't be performed in an object's constructor.

MTS calls the object's **Deactivate** method each time the object is deactivated. This can be the result of the object returning from a method in which it calls **SetComplete** or **SetAbort**, or due to the root of the object's transaction, causing the transaction to complete. You use the **Deactivate** method to clean up state that you initialized in the **Activate** method.

After calling the **Deactivate** method, the MTS run-time environment calls the **CanBePooled** method. If this method returns TRUE, the deactivated object is placed in an object pool for reuse. If the **CanBePooled** method returns FALSE, the object is released in the usual way and its destructor is invoked.

Note On systems that don't support object pooling, the value returned by this method is ignored.

The **IObjectControl** interface is not accessible to an object's clients or to the object itself. Only the MTS run-time environment can invoke the **IObjectControl** methods. If a client queries for the **IObjectControl** interface, **QueryInterface** will return E_NOINTERFACE.

The **IObjectControl** interface exposes the following methods.

Method	Description
Activate	Allows an object to perform context-specific initialization whenever it's activated. This method is called by the MTS run-time environment before any other methods are called on the object.
CanBePooled	Allows an object to notify the MTS run-time environment of whether it can be pooled for reuse. Return TRUE if you want instances of this component to be pooled, or FALSE if not.
Deactivate	Allows an object to perform whatever cleanup is necessary before it's recycled or destroyed. This method is called by the MTS run-time environment whenever an object is deactivated.

See Also

Deactivating Objects, Object Pooling and Recycling

IObjectControl::Activate Method

Implementing this method allows an MTS object to perform context-specific initialization whenever it's activated. This method is called by the MTS run-time environment before any other methods are called on the object.

Provided By

IObjectContext

HRESULT IObjectControl::Activate ( );

Return Values

S_OK
The Activate method succeeded.

Failure HRESULT
Any error code indicating why an object was unable to activate itself.

Remarks

Whenever a client calls an MTS object that isn't already activated, the MTS run-time environment automatically activates the object. This is called just-in-time activation. For components that support the **IObjectControl** interface, MTS invokes the object's **Activate** method before passing the client's method call on to the object. This allows objects to do context-specific initialization.

Because an object's context isn't available during object construction, you can't perform any initialization that involves the **ObjectContext**inside the constructor. Instead, you should implement the **Activate** method and place all your context-specific initialization procedures there.

For example, you can use the **Activate** method to obtain a reference to an object's context and store it in a member variable. Then the context reference is available to any method that requires it, and you don't have to acquire a new one and then release it every time you use it. Once you have a reference to the object's context, you can use the **IObjectContext** methods to check whether security is enabled, whether the object is executing in a transaction, or whether the caller is in a particular role. You can also use the **Activate** method to obtain a reference to the object's **SecurityProperty** and check the security ID of the object's creator before any methods are called.

If you're enabling object recycling (returning TRUE from the **CanBePooled** method), the **Activate** method must be able to handle both newly created and recycled objects. When the **Activate** method returns, there should be no distinguishable difference between a new object and a recycled one.

Example

See Also

Deactivating Objects, Object Pooling and Recycling

IObjectControl::Activate Method Example

```
#include <mtx.h>

IObjectContext* m_pObjectContext;

HRESULT MyObject::Activate()
{
   // Get a reference to the object's context here,
   // so it can be used by any method that may be
   // called during this activation of the object.
   HRESULT hr = GetObjectContext(&m_pObjectContext);
   if (SUCCEEDED(hr))
      return S_OK;
   return hr;
}
```

IObjectControl::CanBePooled Method

Implementing this method allows an MTS object to notify the MTS run-time environment of whether it can be pooled for reuse. Return TRUE if you want the object to be pooled for reuse, or FALSE if not.

Provided By

IObjectContext

BOOL IObjectControl::CanBePooled ( );

Return Values

TRUE
Notifies the MTS run-time environment that it can pool this object on deactivation for later reuse.

FALSE
Notifies the MTS run-time environment that it should not pool this object on deactivation, but should release its last reference to the object so that the object will be destroyed.

Remarks

When an object returns TRUE from the **CanBePooled**method, it indicates to the MTS run-time environment that it can be added to an object pool after deactivation rather than being destroyed. Whenever an instance is required, one is drawn from the pool rather than created by the class factory.

The way recycling works is that an object cleans itself up in its **Deactivate** method and is returned to an object pool. Later, when an instance of the same component is needed, the cleaned up object can be reused. For this to work, an object must be accessible on different threads each time it's activated. Recycling isn't possible under the apartment threading model because, in that model, although an object can be instantiated on any thread, it can only be used by the thread on which it was instantiated. If you want instances of a component to be recyclable, you should register the component with the **ThreadingModel** Registry value set to **Both**. This indicates to MTS that the component's objects can be called from different threads.

In MTS, these objects will run under the apartment threading model and won't be recycled even if they return TRUE from the **CanBePooled** method. However, if you configure a component to support both threading models, the component will run under the current version of MTS and will also be able to take advantage of recycling as soon as it becomes available, without any changes to the code.

Deciding whether to enable recycling is a matter of weighing the costs and benefits. Recycling requires writing additional code, and there's a risk that some state may be inadvertently retained from one activation to the next. When you allow objects to be pooled, you have to be very careful in your **Activate** and **Deactivate** methods to ensure that a recycled object is always restored to a state that's equivalent to the state of a newly created object. Another consideration to take into account is the amount of resources required to maintain an object pool. Objects that hold a lot of resources can be expensive to pool. However, in certain situations, recycling can be extremely efficient, resulting in improved performance and increased scalability. The trade-off is between the cost of holding onto resources while objects are pooled (and inactive) versus the cost of creating and destroying the resources.

It's usually best to enable recycling for objects that cost more to create than they cost to reinitialize. For example, if a component contains a complex structure, and that structure can be reused, it could save a lot of time if the structure didn't have to be recreated every time an instance of the component was activated. This is a case in which you might want to enable recycling, which you would do by returning TRUE from the **CanBePooled** method.

You could still create the structure in the object's constructor and release it in the destructor. The constructor is only called once, when a new object is created by the class factory. When recycling is

enabled, that only happens when the object pool is empty. The **Activate** method, on the other hand, is called whether a new instance is created by the class factory or a recycled instance is drawn from the pool. Similarly, the object's destructor is only called when no further instances are needed, for example, when the server is shutting down. The **Deactivate** method is called every time the object is deactivated, whether it's being destroyed or returned to the pool for recycling.

So, in this example, you'd use the object's **Activate** method to initialize, or reinitialize, the structure that's being reused, and you'd use the **Deactivate** method to restore the object to a state that the **Activate** method can handle. (The **Activate** method must be able to handle both new objects created by the class factory and reused objects drawn from the pool.) This combined use of the **Activate**, **Deactivate**, and **CanBePooled** methods eliminates the need to recreate reusable resources every time an instance is activated.

For some objects, recycling isn't efficient. For example, if an object acquires a lot of state during its lifetime that isn't reusable, and has little to do during its construction, it's usually cheaper to create a new instance whenever one is needed. In that case, you would return FALSE from the **CanBePooled** method.

Note Returning TRUE from the **CanBePooled** method doesn't guarantee that objects will be recycled; it only gives the MTS run-time environment permission to recycle them. On systems that don't support object pooling, a return value of TRUE is ignored. Returning FALSE from the **CanBePooled** method guarantees that instances of a component aren't recycled.

Example

See Also

Deactivating Objects, Object Pooling and Recycling

IObjectControl::CanBePooled Method Example

```
#include <mtx.h>

BOOL MyObject::CanBePooled()
{
   // This object should not be recycled,
   // so return false.
   return FALSE;
}
```

IObjectControl::Deactivate Method

Implementing this method allows an MTS object to perform any cleanup required before it's recycled or destroyed. This method is called by the MTS run-time environment whenever an object is deactivated.

Provided By

IObjectContext

void IObjectControl::Deactivate ( );.

Remarks

The MTS run-time environment calls the **Deactivate** method whenever an object that supports the **IObjectControl** interface is deactivated. An object is deactivated when it returns from a method in which it called **SetComplete** or **SetAbort**, when the transaction in which it executed is committed or aborted, or when the last client to hold a reference on it releases its reference.

If the component supports recycling (returns TRUE from the **CanBePooled** method), you should use the **Deactivate** method to reset the object's state to the state in which the **Activate**method expects to find it. You can also use the **Deactivate** method to release the object's **ObjectContext** reference or to do other context-specific cleanup. You can't do this in the **Release** method or the destructor, because the **IObjectContext** interface isn't accessible from the destructor or any of the **IUnknown** methods. Even if an object supports recycling, it can be beneficial to release certain reusable resources in its **Deactivate** method. For example, ODBC provides its own connection pooling. It's more efficient to pool a database connection in a general connection pool where it can be used by other objects than it is to keep it tied to a specific object in an object pool.

Example

See Also

Deactivating Objects, Object Pooling and Recycling

IObjectControl::Deactivate Method Example

```
#include <mtx.h>

void MyObject::Deactivate(void)
{
   // This object is no longer associated with this
   // context, so release the reference it acquired in
   // its Activate method.
   m_pObjectContext->Release();
}
```

IObjectControl Interface

You implement the **IObjectControl** interface when you want to define context-specific initialization and cleanup procedures for your MTS objects and specify whether or not the objects can be recycled. Implementing the **IObjectControl** interface is optional.

Remarks

The **IObjectControl** interface is declared in the package `com.ms.mtx`.

If you implement the **IObjectControl** interface in your component, the MTS run-time environment automatically calls the **IObjectControl** methods on your objects at the appropriate times.

When an object supports the **IObjectControl** interface, MTS calls its **Activate** method once for each time the object is activated. The **Activate** method is called before any of the object's other methods are called. You can use this method to perform any context-specific initialization an object may require.

Note An object's context isn't available during object construction (from the object's class factory), so context-specific initialization can't be performed in an object's constructor.

MTS calls the object's **Deactivate** method each time the object is deactivated. This can be the result of the object returning from a method in which it calls **SetComplete** or **SetAbort**, or due to the root of the object's transaction, causing the transaction to complete. You use the **Deactivate** method to clean up state that you initialized in the **Activate** method.

After calling the **Deactivate** method, the MTS run-time environment calls the **CanBePooled** method. If this method returns `true`, the deactivated object is placed in an object pool for reuse. If the **CanBePooled** method returns `false`, the object is released in the usual way.

Note On systems that don't support object pooling, the value returned by this method is ignored.

The **IObjectControl** interface is not accessible to an object's clients or to the object itself. Only the MTS run-time environment can invoke the **IObjectControl** methods.

The **IObjectControl** interface exposes the following methods.

Method	Description
Activate	Allows an object to perform context-specific initialization whenever it's activated. This method is called by the MTS run-time environment before any other methods are called on the object.
CanBePooled	Allows an object to notify the MTS run-time environment of whether it can be pooled for reuse. Return `true` if you want instances of this component to be pooled, or `false` if not.
Deactivate	Allows an object to perform whatever cleanup is necessary before it's recycled or destroyed. This method is called by the MTS run-time environment whenever an object is deactivated.

See Also

Deactivating Objects, Object Pooling and Recycling

IObjectControl.Activate Method

Implementing this method allows an MTS object to perform context-specific initialization whenever it's activated. This method is called by the MTS run-time environment before any other methods are called on the object.

Provided By

IObjectContext

void Activate ( );

Remarks

Whenever a client calls an MTS object that isn't already activated, the MTS run-time environment automatically activates the object. This is called just-in-time activation. For components that support the **IObjectControl** interface, MTS invokes the object's **Activate** method before passing the client's method call on to the object. This allows objects to do context-specific initialization.

Because an object's context isn't available during object construction, you can't perform any initialization that involves the **ObjectContext**inside the constructor. Instead, you should implement the **Activate** method and place all your context-specific initialization procedures there.

For example, you can use the **Activate** method to obtain a reference to an object's context and store it in a member variable. Then the context reference is available to any method that requires it, and you don't have to acquire a new one and then release it every time you use it. Once you have a reference to the object's context, you can use the **IObjectContext** methods to check whether security is enabled, whether the object is executing in a transaction, or whether the caller is in a particular role.

If you're enabling object recycling (returning `true` from the **CanBePooled** method), the **Activate** method must be able to handle both newly created and recycled objects. When the **Activate** method returns, there should be no distinguishable difference between a new object and a recycled one.

Example

See Also

Deactivating Objects, Object Pooling and Recycling

IObjectControl.Activate Method Example

```
import com.ms.mtx.*;

void Activate()
{
   // Get a reference to the object's context here,
   // so it can be used by any method that may be
   // called during this activation of the object.
   m_ObjectContext = MTx.GetObjectContext()
}
```

IObjectControl.CanBePooled Method

Implementing this method allows an MTS object to notify the MTS run-time environment of whether it can be pooled for reuse. Return `true` if you want the object to be pooled for reuse, or `false` if not.

Provided By

IObjectContext

boolean CanBePooled ( );

Return Values

`true`
Notifies the MTS run-time environment that it can pool this object on deactivation for later reuse.

`false`
Notifies the MTS run-time environment that it should not pool this object on deactivation, but should release its last reference to the object so that the object will be destroyed.

Remarks

When an object returns `true` from the **CanBePooled**method, it indicates to the MTS run-time environment that it can be added to an object pool after deactivation rather than being destroyed. Whenever an instance is required, one is drawn from the pool rather than created by the class factory.

The way recycling works is that an object cleans itself up in its **Deactivate** method and is returned to an object pool. Later, when an instance of the same component is needed, the cleaned up object can be reused. For this to work, an object must be accessible on different threads each time it's activated. Recycling isn't possible under the apartment threading model because, in that model, although an object can be instantiated on any thread, it can only be used by the thread on which it was instantiated. If you want instances of a component to be recyclable, you should register the component with the **ThreadingModel** Registry value set to **Both**. This indicates to MTS that the component's objects can be called from different threads.

In MTS, these objects will run under the apartment threading model and won't be recycled even if they return `true` from the **CanBePooled** method. However, if you configure a component to support both threading models, the component will run under the current version of MTS and will also be able to take advantage of recycling as soon as it becomes available, without any changes to the code.

Deciding whether to enable recycling is a matter of weighing the costs and benefits. Recycling requires writing additional code, and there's a risk that some state may be inadvertently retained from one activation to the next. When you allow objects to be pooled, you have to be very careful in your **Activate** and **Deactivate** methods to ensure that a recycled object is always restored to a state that's equivalent to the state of a newly created object. Another consideration to take into account is the amount of resources required to maintain an object pool. Objects that hold a lot of resources can be expensive to pool. However, in certain situations, recycling can be extremely efficient, resulting in improved performance and increased scalability. The trade-off is between the cost of holding onto resources while objects are pooled (and inactive) versus the cost of creating and destroying the resources.

It's usually best to enable recycling for objects that cost more to create than they cost to reinitialize. For example, if a component contains a complex structure, and that structure can be reused, it could save a lot of time if the structure didn't have to be recreated every time an instance of the component was activated. This is a case in which you might want to enable recycling, which you would do by returning `true` from the **CanBePooled** method.

You could still create the structure in the object's constructor because the constructor is only called once, when a new object is created by the class factory. When recycling is enabled, that only happens

when the object pool is empty. The **Activate** method, on the other hand, is called whether a new instance is created by the class factory or a recycled instance is drawn from the pool. Similarly, the **Deactivate** method is called every time the object is deactivated, whether it's being destroyed or returned to the pool for recycling.

So, in this example, you'd use the object's **Activate** method to initialize, or reinitialize, the structure that's being reused, and you'd use the **Deactivate** method to restore the object to a state that the **Activate** method can handle. (The **Activate** method must be able to handle both new objects created by the class factory and reused objects drawn from the pool.) This combined use of the **Activate**, **Deactivate**, and **CanBePooled** methods eliminates the need to recreate reusable resources every time an instance is activated.

For some objects, recycling isn't efficient. For example, if an object acquires a lot of state during its lifetime that isn't reusable, and has little to do during its construction, it's usually cheaper to create a new instance whenever one is needed. In that case, you would return `false` from the **CanBePooled** method.

Note Returning `true` from the **CanBePooled** method doesn't guarantee that objects will be recycled; it only gives the MTS run-time environment permission to recycle them. On systems that don't support object pooling, a return value of `true` is ignored. Returning `false` from the **CanBePooled** method guarantees that instances of a component aren't recycled.

Example

See Also

Deactivating Objects, Object Pooling and Recycling

IObjectControl.CanBePooled Method Example

```
import com.ms.mtx.*;

boolean CanBePooled()
{
   // This object should not be recycled,
   // so return false.
   return false;
}
```

IObjectControl.Deactivate Method

Implementing this method allows an MTS object to perform any cleanup required before it's recycled or destroyed. This method is called by the MTS run-time environment whenever an object is deactivated.

Provided By

IObjectContext

void Deactivate ( );.

Remarks

The MTS run-time environment calls the **Deactivate** method whenever an object that supports the **IObjectControl** interface is deactivated. An object is deactivated when it returns from a method in which it called **SetComplete** or **SetAbort**, when the transaction in which it executed is committed or aborted, or when the last client to hold a reference on it releases its reference.

If the component supports recycling (returns `true` from the **CanBePooled** method), you should use the **Deactivate** method to reset the object's state to the state in which the **Activate**method expects to find it. You can also use the **Deactivate** method to release the object's **ObjectContext** reference or to do other context-specific cleanup. Even if an object supports recycling, it can be beneficial to release certain reusable resources in its **Deactivate** method. For example, ODBC provides its own connection pooling. It's more efficient to pool a database connection in a general connection pool where it can be used by other objects than it is to keep it tied to a specific object in an object pool.

Example

See Also

Deactivating Objects, Object Pooling and Recycling

IObjectControl.Deactivate Method Example

```
import com.ms.mtx.*;

void Deactivate( )
{
   // Perform any necessary cleanup here.
}
```

SafeRef Function

Used by an object to obtain a reference to itself that's safe to pass outside its context.

Syntax

Set *safeobject* = **SafeRef**(**Me**)

Part

safeobject
An object variable representing the current object that's safe to pass to another object or client.

Remarks

When an MTS object wants to pass a self-reference to a client or another object (for example, for use as a callback), it should always call **SafeRef** first and then pass the reference returned by this call.

To use the **SafeRef** function, you must set a reference to Microsoft Transaction Server Type Library (mtxas.dll).

Example

See Also

Passing Object References

SafeRef Function Example

```
Dim anotherObject As New ObjectThatCallsBack
Dim safeMe As My.Class

' Get a safe reference.
Set safeMe = SafeRef(Me)

' Invoke a method on another object, passing the
' safe reference so it can call back.
If Not safeMe Is Nothing Then
   Call anotherObject.CallMeBack(safeMe)
Set safeMe = Nothing
```

GetObjectContext Function

Obtains a reference to the **ObjectContext** that's associated with the current MTS object.

To use the **GetObjectContext** function, you must set a reference to Microsoft Transaction Server Type Library (mtxas.dll).

Syntax

Set *objectcontext* = **GetObjectContext ( )**

Parameters

objectcontext
An object variable that evaluates to the **ObjectContext** belonging to the current object.

Remarks

An object should never attempt to pass its **ObjectContext** reference to another object. If you pass an **ObjectContext** reference to another object, it will no longer be a valid reference.

When an object obtains a reference to its **ObjectContext,** it must release the **ObjectContext** object when it's finished with it.

Example

See Also

Building Scalable Components, Context Objects, **CreateInstance** Method

GetObjectContext Function, CreateInstance Method Example

```
Dim ctxObject As ObjectContext
Dim objAccount As Bank.Account

' Get the object's ObjectContext.
Set ctxObject = GetObjectContext()

' Use it to instantiate another object.
Set objAccount = _
   ctxObject.CreateInstance("Bank.Account")
```

ObjectContext Object

The **ObjectContext** object provides access to the current object's context.

Remarks

To use the **ObjectContext** object, you must set a reference to Microsoft Transaction Server Type Library (mtxas.dll).

You obtain a reference to the **ObjectContext** object by calling the **GetObjectContext** function. As with any COM object, you must release an **ObjectContext** object when you're finished using it, unless it's a local variable.

You can use an object's **ObjectContext** to:

- 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.

The **ObjectContext** object provides the following methods.

Method	Description
Count	Returns the number of context object properties.
CreateInstance	Instantiates another MTS object.
DisableCommit	Declares that the object hasn't finished its work and that its transactional updates are in an inconsistent state. The object retains its state across method calls, and any attempts to commit the transaction before the object calls **EnableCommit** or **SetComplete** will result in the transaction being aborted.
EnableCommit	Declares that the object's work isn't necessarily finished, but its transactional updates are in a consistent state. This method allows the transaction to be committed, but the object retains its state across method calls until it calls **SetComplete** or **SetAbort**, or until the transaction is completed.
IsCallerInRole	Indicates whether the object's direct caller is in a specified role (either directly or as part of a group).
IsInTransaction	Indicates whether the object is executing within a transaction.
IsSecurityEnabled	Indicates whether security is enabled. MTS security is enabled unless the object is running in the client's process.
Item	Returns a context object property.
Security	Returns a reference to an object's **SecurityProperty** object.
SetAbort	Declares that the object has completed its work and can be deactivated on returning from the currently executing method, but that its transactional updates

	are in an inconsistent state or that an unrecoverable error occurred. This means that the transaction in which the object was executing must be aborted. If any object executing within a transaction returns to its client after calling **SetAbort**, the entire transaction is doomed to abort.
SetComplete	Declares that the object has completed its work and can be deactivated on returning from the currently executing method. For objects that are executing within the scope of a transaction, it also indicates that the object's transactional updates can be committed. When an object that is the root of a transaction calls **SetComplete**, MTS attempts to commit the transaction on return from the current method.

Note When an object calls **DisableCommit**, **EnableCommit**, **SetComplete**, or **SetAbort** from within a method, two flags (Done and Consistent) are set in its **ObjectContext**. (See the following table for an explanation.) These flags aren't evaluated by the MTS run-time environment until the object's currently executing method returns to its caller. This means that an object can call these methods any number of times from within one of its own methods, but the last call before the object returns to its client is the one that will be in effect.

Method	Done	Consistent
SetAbort	TRUE	FALSE
SetComplete	TRUE	TRUE
DisableCommit	FALSE	FALSE
EnableCommit	FALSE	TRUE

The **Done** flag, which allows an object to be deactivated and its transaction to commit or abort, is only evaluated after the object returns from the call that first entered its context. For example, suppose client A calls into object B. Object B calls **SetComplete** and then calls into object C (passing it a safe reference for a callback). Object C calls back to object B, and then object B returns to client A. Object B won't be deactivated when it returns to object C; it will be deactivated when it returns to client A.

See Also

Basic Security Methods, Passing Object References, Context Objects, Transactions, Deactivating Objects

Count Method

Returns the number of context object properties.

Applies To

ObjectContext Object

Syntax

objectcontext.**Count**

The *objectcontext* placeholder represents an object variable that evaluates to the **ObjectContext** associated with the current object.

Return Value

The number of properties.

Example

CreateInstance Method

Instantiates an MTS object.

Applies To

ObjectContext Object

Syntax

Set *object* = *objectcontext*.**CreateInstance**("*progID*")

Part

object
An object variable that evaluates to an MTS object.

objectcontext
An object variable that represents the **ObjectContext** from which to instantiate the new *object*.

progID
A string expression that is the programmatic ID of the new object's component.

Remarks

CreateInstance creates a COM object. However, the object will have context only if its component is registered with MTS.

When you create an object by using **CreateInstance**, the new object's context is derived from the current object's **ObjectContext** and the declarative properties of the new object's component. The new object always executes within the same activity as the object that created it. If the current object has a transaction, the transaction attribute of the new object's component determines whether or not the new object will execute within the scope of that transaction.

If the component's transaction attribute is set to either **Requires a transaction** or **Supports transactions**, the new object inherits its creator's transaction. If the component's transaction attribute is set to **Requires a new transaction**, MTS initiates a new transaction for the new object. If the component's transaction attribute is set to **Does not support transactions**, the new object doesn't execute under any transaction.

Example

See Also

Creating MTS Objects, Transaction Attributes, MTS Component Requirements

DisableCommit Method

Declares that the object's transactional updates are inconsistent and can't be committed in their present state.

Applies To

ObjectContext Object

Syntax

objectcontext.**DisableCommit**

The *objectcontext* placeholder represents an object variable that evaluates to the **ObjectContext** associated with the current object.

Remarks

An object that invokes **DisableCommit** is stateful.

You can use the **DisableCommit** method to prevent a transaction from committing prematurely between method calls in a stateful object. When an object invokes **DisableCommit**, it indicates that its work is inconsistent and that it can't complete its work until it receives further method invocations from the client. It also indicates that it needs to maintain its state to perform that work. This prevents the MTS run-time environment from deactivating the object and reclaiming its resources on return from a method call. Once an object has called **DisableCommit**, if a client attempts to commit the transaction before the object has called **EnableCommit** or **SetComplete**, the transaction will abort.

For example, suppose you have a General Ledger component that updates a database. A client makes multiple calls to a General Ledger object to post entries to various accounts. There's an integrity constraint that says the debits must equal the credits when the final method invocation returns, or the transaction must abort. The General Ledger object has an initialization method in which the client informs it of the sequence of calls the client is going to make, and the General Ledger object calls **DisableCommit**. The object maintains its state between calls so that after the final call in the sequence is made the object can make sure the integrity constraint is satisfied before allowing its work to be committed.

Example

See Also

Transactions

DisableCommit Method Example

```
Dim objContext As ObjectContext

Set objContext = GetObjectContext()
objContext.DisableCommit
```

EnableCommit Method

Declares that the current object's work is not necessarily finished, but that its transactional updates are consistent and could be committed in their present form.

Applies To

ObjectContext Object

Syntax

objectcontext.**EnableCommit**

The *objectcontext* placeholder represents an object variable that evaluates to the **ObjectContext** associated with the current object.

Remarks

When an object calls **EnableCommit**, it allows the transaction in which it's participating to be committed, but it maintains its internal state across calls from its clients until it calls **SetComplete** or **SetAbort** or until the transaction completes.

EnableCommit is the default state when an object is activated. This is why an object should always call **SetComplete** or **SetAbort** before returning from a method, unless you want the object to maintain its internal state for the next call from a client.

Example

See Also

Transactions

EnableCommit Method Example

```
Dim objContext As ObjectContext

Set objContext = GetObjectContext()
objContext.EnableCommit
```

IsCallerInRole Method

Indicates whether an object's direct caller is in a specified role (either individually or as part of a group).

Applies To

ObjectContext Object

Syntax

objectcontext.**IsCallerInRole**(*role*)

The *objectcontext* placeholder represents an object variable that evaluates to the **ObjectContext** associated with the current object.

Parameters

objectcontext
An object variable that represents the **ObjectContext** belonging to the current object.

role
A string expression that contains the name of the role in which to determine if the caller is acting.

Return Values

True
Either the caller is in the specified role, or security is not enabled.

False
The caller is not in the specified role.

Remarks

You use this method to determine whether the direct caller of the currently executing method is associated with a specific role. A role is a symbolic name that represents a user or group of users who have specific access permissions to all components in a given package. Developers define roles when they create a component, and roles are mapped to individual users or groups at deployment time.

IsCallerInRole only applies to the direct caller of the currently executing method. (The direct caller is the process calling into the current server process. It can be either a base client process or a server process.) **IsCallerInRole** doesn't apply to the process that initiated the call sequence from which the current method was called, or to any other callers in that sequence.

Because **IsCallerInRole** returns **True** when the object that invokes it is executing in a client's process, it's a good idea to call **IsSecurityEnabled** before calling **IsCallerInRole**. If security isn't enabled, **IsCallerInRole** won't return an accurate result.

Example

See Also

Programmatic Security, Basic Security Methods, Secured Components

IsCallerInRole, IsSecurityEnabled Methods Example

```
Dim objContext As ObjectContext
Set objContext = GetObjectContext()

If Not objContext Is Nothing Then
   ' Find out if Security is enabled.
   If objContext.IsSecurityEnabled Then
      ' Find out if the caller is in the right role.
      If Not objContext.IsCallerInRole("Manager") Then
         ' If not, do something appropriate here.
      Else
         ' If so, execute the call normally.
      End If
   Else
   ' If security's not enabled, do something
   ' appropriate here.
   End If
End If
```

IsInTransaction Method

Indicates whether the current object is executing in a transaction.

Applies To

ObjectContext Object

Syntax

objectcontext.**IsInTransaction**

The *objectcontext* placeholder represents an object variable that evaluates to the **ObjectContext** associated with the current object.

Return Values

True
The current object is executing within a transaction.

False
The current object is not executing within a transaction.

Remarks

You can use this method to make sure that an object that requires a transaction never runs without one. For example, if a component that requires a transaction is improperly configured in the MTS Explorer, you can use this method to determine that the object doesn't have a transaction. Then you can return an error to alert the user to the problem, or take whatever action is appropriate.

Example

See Also

Transaction Attributes, Transactions

IsInTransaction Method Example

```
Dim objContext As ObjectContext
Set objContext = GetObjectContext()

If Not objContext Is Nothing Then
   ' Find out if the object is in a transaction.
   If Not objContext.IsInTransaction Then
      ' If not, do something appropriate here.
   End If
End If
```

IsSecurityEnabled Method

Indicates whether or not security is enabled for the current object. MTS security is enabled unless the object is running in the client's process.

Applies To

ObjectContext Object

Syntax

objectcontext.**IsSecurityEnabled**

The *objectcontext* placeholder represents an object variable that evaluates to the **ObjectContext** associated with the current object.

Return Values

True
Security is enabled for this object.

False
Security is not enabled for this object.

Remarks

MTS security is enabled only if an object is running in a server process. This could be either because the object's component was configured to run in a client's process, or because the component and the client are in the same package. If the object is running in the client's process, there is no security checking and **IsSecurityEnabled** will always return **False**.

Example

See Also

Programmatic Security, Basic Security Methods, Secured Components

Item Method

Returns a context object property.

Applies To

ObjectContext Object

Syntax

objectcontext.**Item(***name***)**

Part

objectcontext
An object variable that evaluates to the **ObjectContext** associated with the current object.

name
The name of the context object property to be retrieved.

Return Value

The requested context object property.

Remarks

You can use **Item** to retrieve the following Microsoft Internet Information Server (IIS) built-in objects:

- Request
- Response
- Server
- Application
- Session

For more information, see the IIS documentation.

The **Item** method is the default method for a collection. Therefore, the following lines of code are equivalent:

```
oc("Response").Write "<p>" & prop & "</p>"
oc.Item("Response").Write "<p>" & prop & "</p>"
```

Example

Count, Item Methods Example

```
' Get the context object
Dim oc As ObjectContext
Dim str As String
Set oc = GetObjectContext()

' Get the Response object
' Print number of properties
oc("Response").Write "<p>Number of properties: " & oc.Count & "</p>"

' Iterate over properties collection and print the
' names of the properties
For Each prop In oc
    oc("Response").Write "<p>" & prop & "</p>"
Next
```

Security Property

Returns a reference to an object's **SecurityProperty** object.

Applies To

ObjectContext Object

Syntax

Set *objectsecurity* = *objectcontext*.**Security**

Part

objectcontext
An object variable that evaluates to the **ObjectContext** associated with the current object.

objectsecurity
An object variable that evaluates to the **SecurityProperty** object associated with the current object.

Example

See Also

Programmatic Security, Advanced Security Methods

Security Property Example

```
Public Function UsingSecurityMethod() As String

    Dim objCtx As ObjectContext
    Dim objSP As SecurityProperty

    Set objCtx = GetObjectContext()
    Set objSP = objCtx.Security
    UsingSecurityMethod = _
        objSP.GetOriginalCreatorName()

End Function
```

SetAbort Method

Declares that the transaction in which the object is executing must be aborted, and that the object should be deactivated on returning from the currently executing method call.

Applies To

ObjectContext Object

Syntax

objectcontext.**SetAbort**

The *objectcontext* placeholder represents an object variable that evaluates to the **ObjectContext** associated with the current object.

Remarks

The object is deactivated automatically on return from the method in which it called **SetAbort**. If the object is the root of an automatic transaction, MTS aborts the transaction. If the object is transactional, but not the root of an automatic transaction, the transaction in which it's participating is doomed to abort. (An object is the root of a transaction if the MTS run-time environment has to initiate a new transaction for it. This is the case when the component that provides the object is configured to require a transaction and the object's creator doesn't have one, or when the component is configured to require a new transaction.)

You can call **SetAbort** in error handlers to ensure that a transaction aborts when an error occurs. You can also call **SetAbort** at the beginning of a method to protect your object from committing prematurely in the event of an unexpected return and then call **SetComplete** just before the method returns, if all goes well.

Example

See Also

Transactions, Context Objects, Deactivating Objects

SetAbort, SetComplete Methods Example

```
Dim ctxObject As ObjectContext
Set ctxObject = GetObjectContext()
On Error GoTo ErrorHandler

' Do some work here. If the work was successful,
' call SetComplete.
ctxObject.SetComplete
Set ctxObject = Nothing
Perform = 0
Exit Function

' If an error occurred, call SetAbort in the error
' handler.
ErrorHandler:
   ctxObject.SetAbort
   Set ctxObject = Nothing
   Perform = -1
   Exit Function
```

SetComplete Method

Declares that the current object has completed its work and should be deactivated when the currently executing method returns to the client. For objects that are executing within the scope of a transaction, it also indicates that the object's transactional updates can be committed.

Applies To

ObjectContext Object

Syntax

objectcontext.**SetComplete**

The *objectcontext* placeholder represents an object variable that evaluates to the **ObjectContext** associated with the current object.

Remarks

The object is deactivated automatically on return from the method in which it called **SetComplete**. If the object is the root of an automatic transaction, MTS attempts to commit the transaction. However, if any object that was participating in the transaction has called **SetAbort**, or has called **DisableCommit** and has not subsequently called **EnableCommit** or **SetComplete**, the transaction will be aborted. (An object is the root of a transaction if the MTS run-time environment has to initiate a new transaction for it. This is the case when the component that provides the object is configured to require a transaction and the object's creator doesn't have one, or when the component is configured to require a new transaction.)

If an object doesn't need to maintain its state after it returns from a method call, it should call **SetComplete** so that it can be automatically deactivated as soon as it returns and its resources can be reclaimed.

Example

See Also

Transactions, Context Objects, Deactivating Objects

SafeRef Function

Used by an object to obtain a reference to itself that's safe to pass outside its context.

The header file for the **SafeRef** function is mtx.h.

```
void* SafeRef (
    REFIID riid
    UNKNOWN* pUnk
);
```

Parameter

riid
[in] A reference to the interface ID of the interface that the current object wants to pass to another object or client.

pUnk
[in] A reference to an interface on the current object.

Return Values

Non-NULL
A pointer to the interface specified in the *riid* parameter that's safe to pass outside the current object's context.

NULL
The object is requesting a safe reference on an object other than itself, or the interface requested in the *riid* parameter is not implemented.

Remarks

When an MTS object wants to pass a self-reference to a client or another object (for example, for use as a callback), it should always call **SafeRef** first and then pass the reference returned by this call. An object should never pass a **this** pointer, or a self-reference obtained through an internal call to **QueryInterface**, to a client or to any other object. Once such a reference is passed outside the object's context, it's no longer a valid reference.

Calling **SafeRef** on a reference that is already safe returns the safe reference unchanged, except that the reference count on the interface is incremented.

When a client calls **QueryInterface** on a reference that's safe, MTS automatically ensures that the reference returned to the client is also a safe reference.

An object that obtains a safe reference must release the safe reference when it's finished with it.

Note Safe references have different pointer values than their unsafe counterparts. For example, **this** and the safe version of **this** do not have the same value. It's important to be aware of this when testing whether two pointers refer to the same object. Calling **QueryInterface** for **IID_IUnknown** on each of the pointers and comparing the value of the returned pointers may result in the wrong conclusion. It's possible that both pointers refer to the same object, but that one is a safe reference and the other isn't. If both references are safe references, they can be compared in the usual way. This is only a consideration for MTS objects, because clients should never have access to unsafe references.

Example

See Also

Passing Object References

SafeRef Function Example

```
#include <mtx.h>


IMyInterface* pSafeMyObject = NULL;
IAnotherObject* pOtherObject = NULL;
HRESULT hr;

// Get a safe reference.
pSafeMyObject = SafeRef(IID_IMyInterface,
   (IUnknown*)this);

// Invoke a method on another object, passing the
// safe reference so it can call back.
hr = pOtherObject->CallMeBack(pSafeMyObject);

// Release the safe reference.
safeMyObject->Release();
```

GetObjectContext Function

Obtains a reference to the **IObjectContext** interface on the **ObjectContext** that's associated with the current MTS object.

The header file for the **GetObjectContext** function is mtx.h.

HRESULT GetObjectContext (
  IObjectContext** *ppInstanceContext*
);

Parameters

ppInstanceContext
  [out] A reference to the **IObjectContext** interface on the object's context. If the object's component hasn't been imported into an MTS package, or if **GetObjectContext** is called from a constructor or an **IUnknown** method, this will be set to a NULL pointer.

Return Values

S_OK
  A reference to the **IObjectContext** interface on the current object's context is returned in the *ppInstanceContext* parameter.

E_INVALIDARG
  The argument passed in the *ppInstanceContext* parameter is invalid.

E_UNEXPECTED
  An unexpected error occurred.

CONTEXT_E_NOCONTEXT
  The current object doesn't have a context associated with it, because either the component wasn't imported into a package or the object wasn't created with one of the MTS **CreateInstance** methods. This error will also be returned if the **GetObjectContext** method was called from the constructor or from an **IUnknown** method.

Remarks

An object's context is not accessible from an object's constructor or from any **IUnknown** method (**QueryInterface**, **AddRef**, or **Release**).

An object should never attempt to pass its **ObjectContext** reference to another object. If you pass an **ObjectContext** reference to another object, it will no longer be a valid reference.

When an object obtains a reference to its **ObjectContext,** it must release the **ObjectContext** object when it's finished with it.

Example

See Also

Building Scalable Components, Context Objects, **IObjectContext::CreateInstance** Method

GetObjectContext Function, IObjectContext::CreateInstance Method Example

```
#include <mtx.h>


IObjectContext* pObjectContext = NULL;
IAccount* pAccount = NULL;
HRESULT hr;

// Get the object's ObjectContext.
hr = GetObjectContext(&pObjectContext);

// Use it to instantiate another object.
hr = pObjectContext->CreateInstance(CLSID_CAccount,
   IID_IAccount, (void**)&pAccount);
```

IObjectContext Interface

The **IObjectContext** interface provides access to the current object's context.

Remarks

The header file for the **IObjectContext** interface is mtx.h.

You obtain a reference to the **IObjectContext** interface by calling the **GetObjectContext** function. As with any COM object, you must release an **ObjectContext** object when you're finished using it.

You can use an object's **ObjectContext** to:

- 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.

The **IObjectContext** interface exposes the following methods.

Method	Description
CreateInstance	Instantiates another MTS object.
DisableCommit	Declares that the object hasn't finished its work and that its transactional updates are in an inconsistent state. The object retains its state across method calls, and any attempts to commit the transaction before the object calls **EnableCommit** or **SetComplete** will result in the transaction being aborted.
EnableCommit	Declares that the object's work isn't necessarily finished, but its transactional updates are in a consistent state. This method allows the transaction to be committed, but the object retains its state across method calls until it calls **SetComplete** or **SetAbort**, or until the transaction is completed.
IsCallerInRole	Indicates whether the object's direct caller is in a specified role (either directly or as part of a group).
IsInTransaction	Indicates whether the object is executing within a transaction.
IsSecurityEnabled	Indicates whether security is enabled. MTS security is enabled unless the object is running in the client's process.
SetAbort	Declares that the object has completed its work and can be deactivated on returning from the currently executing method, but that its transactional updates are in an inconsistent state or that an unrecoverable error occurred. This means that the transaction in which the object was executing must be aborted. If any object executing within a transaction returns to its client after calling **SetAbort**, the entire transaction is doomed to abort.
SetComplete	Declares that the object has completed its work and

can be deactivated on returning from the currently executing method. For objects that are executing within the scope of a transaction, it also indicates that the object's transactional updates can be committed. When an object that is the root of a transaction calls **SetComplete**, MTS attempts to commit the transaction on return from the current method.

Note When an object calls **DisableCommit**, **EnableCommit**, **SetComplete**, or **SetAbort** from within a method, two flags (Done and Consistent) are set in its **ObjectContext**. (See the following table for an explanation.) These flags aren't evaluated by the MTS run-time environment until the object's currently executing method returns to its caller. This means that an object can call these methods any number of times from within one of its own methods, but the last call before the object returns to its client is the one that will be in effect.

Method	Done	Consistent
SetAbort	TRUE	FALSE
SetComplete	TRUE	TRUE
DisableCommit	FALSE	FALSE
EnableCommit	FALSE	TRUE

The **Done** flag, which allows an object to be deactivated and its transaction to commit or abort, is only evaluated after the object returns from the call that first entered its context. For example, suppose client A calls into object B. Object B calls **SetComplete** and then calls into object C (passing it a safe reference for a callback). Object C calls back to object B, and then object B returns to client A. Object B won't be deactivated when it returns to object C; it will be deactivated when it returns to client A.

See Also

Basic Security Methods, Passing Object References, Context Objects, Transactions, Deactivating Objects

IObjectContext::CreateInstance Method

Instantiates an MTS object.

Provided By

IObjectContext

HRESULT IObjectContext::CreateInstance (
REFCLSID *rclsid*,
REFIID *riid*,
LPVOID FAR* *ppvObj*
);

Parameter

rclsid
[in] A reference to the CLSID of the type of object to instantiate.

riid
[in] A reference to the interface ID of the interface through which you want to communicate with the new object.

ppvObj
[out] A reference to the requested interface on the new object.

Return Values

S_OK
The object was created and a reference to it is returned in the *ppvObj* parameter.

REGDB_E_CLASSNOTREG
The component specified by *rclsid* is not registered as a COM component.

E_OUTOFMEMORY
There's not enough memory available to instantiate the object.

E_INVALIDARG
The argument passed in the *ppvObj* parameter is invalid.

E_UNEXPECTED
An unexpected error occurred. This can happen if one object passes its **IObjectContext** pointer to another object, and the other object calls **CreateInstance** using this pointer. An **IObjectContext** pointer is not valid outside the context of the object that originally obtained it.

Remarks

CreateInstance creates a COM object. However, the object will have context only if its component is registered with MTS.

When you create an object by using **CreateInstance**, the new object's context is derived from the current object's **ObjectContext** and the declarative properties of the new object's component. The new object always executes within the same activity as the object that created it. If the current object has a transaction, the transaction attribute of the new object's component determines whether or not the new object will execute within the scope of that transaction.

If the component's transaction attribute is set to either **Requires a transaction** or **Supports transactions**, the new object inherits its creator's transaction. If the component's transaction attribute is set to **Requires a new transaction**, MTS initiates a new transaction for the new object. If the component's transaction attribute is set to **Does not support transactions**, the new object doesn't execute under any transaction.

If the Microsoft Distributed Transaction Coordinator is not running and the object is transactional, the object is successfully created. However, method calls to that object will fail with

CONTEXT_E_TMNOTAVAILABLE. Objects cannot recover from this condition and should be released.

MTS always uses standard marshaling. Even if a component exposes the **IMarshal** interface, its **IMarshal** methods will never be called by the MTS run-time environment.

You can't create MTS objects as part of an aggregation. In this respect, using **CreateInstance** is like using **CoCreateInstance** and specifying NULL for the controlling **IUnknown** interface (*pUnkOuter*).

Example

See Also

Creating MTS Objects, Transaction Attributes, MTS Component Requirements

IObjectContext::DisableCommit Method

Declares that the object's transactional updates are inconsistent and can't be committed in their present state.

Provided By

IObjectContext

HRESULT IObjectContext::DisableCommit ( );

Return Values

S_OK
The call to **DisableCommit** succeeded. The object's transactional updates can't be committed until the object calls either **EnableCommit** or **SetComplete**.

E_UNEXPECTED
An unexpected error occurred. This can happen if one object passes its **IObjectContext** pointer to another object and the other object calls **DisableCommit** using this pointer. An **IObjectContext** pointer is not valid outside the context of the object that originally obtained it.

CONTEXT_E_NOCONTEXT
The current object doesn't have a context associated with it. This is probably because it wasn't created with one of the MTS **CreateInstance** methods.

Remarks

An object that invokes **DisableCommit** is stateful.

You can use the **DisableCommit** method to prevent a transaction from committing prematurely between method calls in a stateful object. When an object invokes **DisableCommit**, it indicates that its work is inconsistent and that it can't complete its work until it receives further method invocations from the client. It also indicates that it needs to maintain its state to perform that work. This prevents the MTS run-time environment from deactivating the object and reclaiming its resources on return from a method call. Once an object has called **DisableCommit**, if a client attempts to commit the transaction before the object has called **EnableCommit** or **SetComplete**, the transaction will abort.

For example, suppose you have a General Ledger component that updates a database. A client makes multiple calls to a General Ledger object to post entries to various accounts. There's an integrity constraint that says the debits must equal the credits when the final method invocation returns, or the transaction must abort. The General Ledger object has an initialization method in which the client informs it of the sequence of calls the client is going to make, and the General Ledger object calls **DisableCommit**. The object maintains its state between calls so that after the final call in the sequence is made the object can make sure the integrity constraint is satisfied before allowing its work to be committed.

Example

See Also

Transactions

IObjectContext::DisableCommit Method Example

```
#include <mtx.h>


IObjectContext* pObjectContext = NULL;
HRESULT hr;

hr = GetObjectContext(&pObjectContext);
hr = pObjectContext->DisableCommit();
```

IObjectContext::EnableCommit Method

Declares that the current object's work is not necessarily finished, but that its transactional updates are consistent and could be committed in their present form.

Provided By

IObjectContext

HRESULT IObjectContext::EnableCommit ( );

Return Values

S_OK
: The call to **EnableCommit** succeeded and the object's transactional updates can now be committed.

E_UNEXPECTED
: An unexpected error occurred. This can happen if one object passes its **IObjectContext** pointer to another object and the other object calls **EnableCommit** using this pointer. An **IObjectContext** pointer is not valid outside the context of the object that originally obtained it.

Remarks

When an object calls **EnableCommit**, it allows the transaction in which it's participating to be committed, but it maintains its internal state across calls from its clients until it calls **SetComplete** or **SetAbort** or until the transaction completes.

EnableCommit is the default state when an object is activated. This is why an object should always call **SetComplete** or **SetAbort** before returning from a method, unless you want the object to maintain its internal state for the next call from a client.

Example

See Also

Transactions

IObjectContext::EnableCommit Method Example

```
#include <mtx.h>

IObjectContext* pObjectContext = NULL;
HRESULT hr;

hr = GetObjectContext(&pObjectContext);
hr = pObjectContext->EnableCommit();
```

IObjectContext::IsCallerInRole Method

Indicates whether an object's direct caller is in a specified role (either individually or as part of a group).

Provided By

IObjectContext

```
HRESULT IObjectContext::IsCallerInRole (
    BSTR bstrRole,
    BOOL* pfIsInRole
);
```

Parameters

bstrRole
[in] The name of the role in which you want to determine whether the caller is acting.

pfIsInRole
[out] TRUE if the caller is in the specified role, FALSE if not. This parameter will also be set to TRUE if security is not enabled.

Return Values

S_OK
The role specified in the *bstrRole* parameter is a recognized role, and the Boolean result returned in the *pfIsInRole* parameter indicates whether or not the caller is in that role.

CONTEXT_E_ROLENOTFOUND
The role specified in the *bstrRole* parameter does not exist.

E_INVALIDARG
One or more of the arguments passed in is invalid.

E_UNEXPECTED
An unexpected error occurred. This can happen if one object passes its **IObjectContext** pointer to another object and the other object calls **IsCallerInRole** using this pointer. An **IObjectContext** pointer is not valid outside the context of the object that originally obtained it.

Remarks

You use this method to determine whether the direct caller of the currently executing method is associated with a specific role. A role is a symbolic name that represents a user or group of users who have specific access permissions to all components in a given package. Developers define roles when they create a component, and roles are mapped to individual users or groups at deployment time.

IsCallerInRole only applies to the direct caller of the currently executing method. (The direct caller is the process calling into the current server process. It can be either a base client process or a server process.) **IsCallerInRole** doesn't apply to the process that initiated the call sequence from which the current method was called, or to any other callers in that sequence.

Because **IsCallerInRole** returns TRUE when the object that invokes it is executing in a client's process, it's a good idea to call **IsSecurityEnabled** before calling **IsCallerInRole**. If security isn't enabled, **IsCallerInRole** won't return an accurate result.

Example

See Also

Programmatic Security, Basic Security Methods, Secured Components

IObjectContext::IsCallerInRole, IObjectContext::IsSecurityEnabled Methods Example

```
#include <mtx.h>

IObjectContext* pObjectContext = NULL;
BSTR stRole = SysAllocString(L"Manager");
VARIANT_BOOL fIsInRole;
HRESULT hr;

hr = GetObjectContext(&pObjectContext);

// Find out if security is enabled.
if (pObjectContext->IsSecurityEnabled()) {
   //Then find out if the caller is in the right role.
   fIsInRole = pObjectContext->IsCallerInRole
      (stRole, &fIsInRole)
   SysFreeString(stRole);
   if (!fIsInRole) {
      // If not, do something appropriate here.
   }
}
else {
   // If security's not enabled, do something
   // appropriate here.
}
```

IObjectContext::IsInTransaction Method

Indicates whether the current object is executing in a transaction.

Provided By

IObjectContext

BOOL IObjectContext::IsInTransaction ( );

Return Values

TRUE
The current object is executing within a transaction.

FALSE
The current object is not executing within a transaction.

Remarks

You can use this method to make sure that an object that requires a transaction never runs without one. For example, if a component that requires a transaction is improperly configured in the MTS Explorer, you can use this method to determine that the object doesn't have a transaction. Then you can return an error to alert the user to the problem, or take whatever action is appropriate.

Example

See Also

Transaction Attributes, Transactions

IObjectContext::IsInTransaction Method Example

```
#include <mtx.h>

IObjectContext* pObjectContext = NULL;
VARIANT_BOOL fInTransaction;
HRESULT hr;

hr = GetObjectContext(&pObjectContext);

// Find out if the object is in a transaction.
fInTransaction = pObjectContext->IsInTransaction();

if (!fInTransaction) {
   // If not, do something appropriate here.
}
```

IObjectContext::IsSecurityEnabled Method

Indicates whether or not security is enabled for the current object. MTS security is enabled unless the object is running in the client's process.

Provided By

IObjectContext

BOOL IObjectContext::IsSecurityEnabled ( );

Return Values

TRUE
Security is enabled for this object.

FALSE
Security is not enabled for this object.

Remarks

MTS security is enabled only if an object is running in a server process. This could be either because the object's component was configured to run in a client's process, or because the component and the client are in the same package. If the object is running in the client's process, there is no security checking and **IsSecurityEnabled** will always return FALSE.

Example

See Also

Programmatic Security, Basic Security Methods, Secured Components

IObjectContext::SetAbort Method

Declares that the transaction in which the object is executing must be aborted, and that the object should be deactivated on returning from the currently executing method call.

Provided By

IObjectContext

HRESULT IObjectContext::SetAbort ( );

Return Values

S_OK
: The call to **SetAbort** succeeded and the transaction will be aborted.

E_UNEXPECTED
: An unexpected error occurred. This can happen if one object passes its **IObjectContext** pointer to another object and the other object calls **SetAbort** using this pointer. An **IObjectContext** pointer is not valid outside the context of the object that originally obtained it.

Remarks

The object is deactivated automatically on return from the method in which it called **SetAbort**. If the object is the root of an automatic transaction, MTS aborts the transaction. If the object is transactional, but not the root of an automatic transaction, the transaction in which it's participating is doomed to abort. (An object is the root of a transaction if the MTS run-time environment has to initiate a new transaction for it. This is the case when the component that provides the object is configured to require a transaction and the object's creator doesn't have one, or when the component is configured to require a new transaction.)

You can call **SetAbort** in error handlers to ensure that a transaction aborts when an error occurs. You can also call **SetAbort** at the beginning of a method to protect your object from committing prematurely in the event of an unexpected return and then call **SetComplete** just before the method returns, if all goes well.

Example

See Also

Transactions, Context Objects, Deactivating Objects

IObjectContext::SetAbort, IObjectContext::SetComplete Methods Example

```
#include <mtx.h>


IObjectContext* pObjectContext = NULL;
HRESULT hr;

hr = GetObjectContext(&pObjectContext);
// Do some work here.
// If the work was successful, call SetComplete.
if (SUCCEEDED(hr)) {
   if (pObjectContext)
      pObjectContext->SetComplete();
}
// Otherwise, call SetAbort.
else {
   if (pObjectContext)
      pObjectContext->SetAbort();
}
```

IObjectContext::SetComplete Method

Declares that the current object has completed its work and should be deactivated when the currently executing method returns to the client. For objects that are executing within the scope of a transaction, it also indicates that the object's transactional updates can be committed.

Provided By

IObjectContext

HRESULT IObjectContext::SetComplete ( );

Return Values

S_OK
The call to **SetComplete** succeeded.

E_UNEXPECTED
An unexpected error occurred. This can happen if one object passes its **IObjectContext** pointer to another object and the other object calls **SetComplete** using this pointer. An **IObjectContext** pointer is not valid outside the context of the object that originally obtained it.

Remarks

The object is deactivated automatically on return from the method in which it called **SetComplete**. If the object is the root of an automatic transaction, MTS attempts to commit the transaction. However, if any object that was participating in the transaction has called **SetAbort**, or has called **DisableCommit** and has not subsequently called **EnableCommit** or **SetComplete**, the transaction will be aborted. (An object is the root of a transaction if the MTS run-time environment has to initiate a new transaction for it. This is the case when the component that provides the object is configured to require a transaction and the object's creator doesn't have one, or when the component is configured to require a new transaction.)

If an object doesn't need to maintain its state after it returns from a method call, it should call **SetComplete** so that it can be automatically deactivated as soon as it returns and its resources can be reclaimed.

Example

See Also

Transactions, Context Objects, Deactivating Objects

MTx.SafeRef Method

Used by an object to obtain a reference to itself that's safe to pass outside its context.

SafeRef is a static method of the **MTx** class, which is declared in the package `com.ms.mtx`.

Note The class **MTx** has only static methods and has no public constructor. You can't create an instance of this class.

New applications should call **getSafeRef**.

```
IUnknown SafeRef (
    IUnknown obj
);
```

Parameter

obj
[in] A reference to an interface on the current object.

Return Value

A reference to the **IUnknown** interface on the current object that's safe to pass outside the current object's context.

Remarks

When an MTS object wants to pass a self-reference to a client or another object (for example, for use as a callback), it should always call **SafeRef** first and then pass the reference returned by this call. An object should never pass a reference to **this** to a client or to any other object. Once such a reference is passed outside the object's context, it's no longer a valid reference.

Regardless of the interface ID you pass to **SafeRef**, it always returns the **IUnknown** interface on the object that calls it. You should immediately cast the returned value to the interface that you want to pass outside the object.

Calling **SafeRef** on a reference that is already safe returns the safe reference unchanged, except that the reference count on the interface is incremented.

Example

See Also

Passing Object References

MTx.SafeRef Method Example

```
import com.ms.mtx.*;


IMyInterface safeMyObject = null;
IAnotherObject someOtherObject = null;

// Get a safe reference.
safeMyObject = (IMyInterface) MTx.SafeRef(this);

// Invoke a method on another object, passing the
// safe reference so it can call back.
someOtherObject.CallMeBack(safeMyObject);
```

MTx.GetObjectContext Method

Obtains a reference to the **IObjectContext** interface on the **ObjectContext** that's associated with the current MTS object.

GetObjectContext is a static method of the **MTx** class, which is declared in the package `com.ms.mtx`.

Note The **MTx** class has only static methods and has no public constructor. You can't create an instance of this class.

IObjectContext GetObjectContext ( );

Return Value

A reference to the **IObjectContext** interface on the current object's context. **GetObjectContext** will return null if it is called from a constructor or finalizer, or if the object's component hasn't been imported into an MTS package.

Remarks

An object should never attempt to pass its **ObjectContext** reference to another object. If you pass an **ObjectContext** reference to another object, it will no longer be a valid reference.

Example

See Also

Building Scalable Components, Context Objects, **IObjectContext.CreateInstance** Method

MTx.GetObjectContext Method, IObjectContext.CreateInstance Method Example

```
import com.ms.mtx.*;


IAccount account = null;

// Get the object's ObjectContext and
// use it to instantiate another object.
account = (IAccount) MTx.GetObjectContext().
   CreateInstance(CAccount.clsid, IAccount.iid);
```

IObjectContext Interface

The **IObjectContext** interface provides access to the current object's context.

Remarks

New applications should use the **Context** class.

The **IObjectContext** interface is declared in the package `com.ms.mtx`.

You obtain a reference to the **IObjectContext** interface by calling the **MTx.GetObjectContext** method. As with any COM object, you must release an **ObjectContext** object when you're finished using it.

You can use an object's **ObjectContext** to:

- 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.

The **IObjectContext** interface exposes the following methods.

Method	Description
CreateInstance	Instantiates another MTS object.
DisableCommit	Declares that the object hasn't finished its work and that its transactional updates are in an inconsistent state. The object retains its state across method calls, and any attempts to commit the transaction before the object calls **EnableCommit** or **SetComplete** will result in the transaction being aborted.
EnableCommit	Declares that the object's work isn't necessarily finished, but its transactional updates are in a consistent state. This method allows the transaction to be committed, but the object retains its state across method calls until it calls **SetComplete** or **SetAbort**, or until the transaction is completed.
IsCallerInRole	Indicates whether the object's direct caller is in a specified role (either directly or as part of a group).
IsInTransaction	Indicates whether the object is executing within a transaction.
IsSecurityEnabled	Indicates whether security is enabled. MTS security is enabled unless the object is running in the client's process.
SetAbort	Declares that the object has completed its work and can be deactivated on returning from the currently executing method, but that its transactional updates are in an inconsistent state or that an unrecoverable error occurred. This means that the transaction in which the object was executing must be aborted. If any object executing within a transaction returns to its client after calling **SetAbort**, the entire transaction is

	doomed to abort.
SetComplete	Declares that the object has completed its work and can be deactivated on returning from the currently executing method. For objects that are executing within the scope of a transaction, it also indicates that the object's transactional updates can be committed. When an object that is the root of a transaction calls **SetComplete**, MTS attempts to commit the transaction on return from the current method.

Note When an object calls **DisableCommit**, **EnableCommit**, **SetComplete**, or **SetAbort** from within a method, two flags (Done and Consistent) are set in its **ObjectContext**. (See the following table for an explanation.) These flags aren't evaluated by the MTS run-time environment until the object's currently executing method returns to its caller. This means that an object can call these methods any number of times from within one of its own methods, but the last call before the object returns to its client is the one that will be in effect.

Method	Done	Consistent
SetAbort	TRUE	FALSE
SetComplete	TRUE	TRUE
DisableCommit	FALSE	FALSE
EnableCommit	FALSE	TRUE

The **Done** flag, which allows an object to be deactivated and its transaction to commit or abort, is only evaluated after the object returns from the call that first entered its context. For example, suppose client A calls into object B. Object B calls **SetComplete** and then calls into object C (passing it a safe reference for a callback). Object C calls back to object B, and then object B returns to client A. Object B won't be deactivated when it returns to object C; it will be deactivated when it returns to client A.

See Also

Basic Security Methods, Passing Object References, Context Objects, Transactions, Deactivating Objects

IObjectContext.CreateInstance Method

Instantiates an MTS object.

New applications should call **createInstance**.

Provided By

IObjectContext Interface

```
IUnknown CreateInstance (
   _Guid clsid,
   _Guid iid,
);
```

Parameter

clsid
   [in] The clsid of the type of object to instantiate.

iid
   [in] Any interface that's implemented by the object you want to instantiate.

Return Value

A reference to the **IUnknown** interface on the newly created object.

Remarks

CreateInstance creates a COM object. However, the object will have context only if its component is registered with MTS.

When you create an object by using **CreateInstance**, the new object's context is derived from the current object's **ObjectContext** and the declarative properties of the new object's component. The new object always executes within the same activity as the object that created it. If the current object has a transaction, the transaction attribute of the new object's component determines whether or not the new object will execute within the scope of that transaction.

If the component's transaction attribute is set to either **Requires a transaction** or **Supports transactions**, the new object inherits its creator's transaction. If the component's transaction attribute is set to **Requires a new transaction**, MTS initiates a new transaction for the new object. If the component's transaction attribute is set to **Does not support transactions**, the new object doesn't execute under any transaction.

CreateInstance always returns the **IUnknown** interface on the newly instantiated object. You should immediately cast the returned value to the interface through which you want to communicate with the new object. The interface ID you pass in the *iid* parameter doesn't have to be the same interface as the one to which you cast the returned value, but it must be an interface that's implemented by the object you're instantiating.

MTS always uses standard marshaling. Even if a component exposes the **IMarshal** interface, its **IMarshal** methods will never be called by the MTS run-time environment.

You can't create MTS objects as part of an aggregation.

Example

See Also

Creating MTS Objects, Transaction Attributes, MTS Component Requirements

IObjectContext.DisableCommit Method

Declares that the object's transactional updates are inconsistent and can't be committed in their present state.

New applications should call **disableCommit**.

Provided By

IObjectContext

void DisableCommit ( );

Remarks

An object that invokes **DisableCommit** is stateful.

You can use the **DisableCommit** method to prevent a transaction from committing prematurely between method calls in a stateful object. When an object invokes **DisableCommit**, it indicates that its work is inconsistent and that it can't complete its work until it receives further method invocations from the client. It also indicates that it needs to maintain its state to perform that work. This prevents the MTS run-time environment from deactivating the object and reclaiming its resources on return from a method call. Once an object has called **DisableCommit**, if a client attempts to commit the transaction before the object has called **EnableCommit** or **SetComplete**, the transaction will abort.

For example, suppose you have a General Ledger component that updates a database. A client makes multiple calls to a General Ledger object to post entries to various accounts. There's an integrity constraint that says the debits must equal the credits when the final method invocation returns, or the transaction must abort. The General Ledger object has an initialization method in which the client informs it of the sequence of calls the client is going to make, and the General Ledger object calls **DisableCommit**. The object maintains its state between calls so that after the final call in the sequence is made the object can make sure the integrity constraint is satisfied before allowing its work to be committed.

Example

See Also

Transactions

IObjectContext.DisableCommit Method Example

```
import com.ms.mtx.*;

MTx.GetObjectContext().DisableCommit();
```

IObjectContext.EnableCommit Method

Declares that the current object's work is not necessarily finished, but that its transactional updates are consistent and could be committed in their present form.

New applications should call **enableCommit**.

Provided By

IObjectContext

void EnableCommit ( );

Remarks

When an object calls **EnableCommit**, it allows the transaction in which it's participating to be committed, but it maintains its internal state across calls from its clients until it calls **SetComplete** or **SetAbort** or until the transaction completes.

EnableCommit is the default state when an object is activated. This is why an object should always call **SetComplete** or **SetAbort** before returning from a method, unless you want the object to maintain its internal state for the next call from a client.

Example

See Also

Transactions

IObjectContext.EnableCommit Method Example

```
import com.ms.mtx.*;

MTx.GetObjectContext().EnableCommit();

}
```

IObjectContext.IsCallerInRole Method

Indicates whether an object's direct caller is in a specified role (either individually or as part of a group).

New applications should call **isCallerInRole**.

Provided By

IObjectContext

```
boolean IsCallerInRole (
   String bstrRole
);
```

Parameters

bstrRole
   [in] The name of the role in which you want to determine whether the caller is acting.

Return Values

`true`
   Either the caller is in the specified role, or security is not enabled.

`false`
   The caller is not in the specified role.

Remarks

You use this method to determine whether the direct caller of the currently executing method is associated with a specific role. A role is a symbolic name that represents a user or group of users who have specific access permissions to all components in a given package. Developers define roles when they create a component, and roles are mapped to individual users or groups at deployment time.

IsCallerInRole only applies to the direct caller of the currently executing method. (The direct caller is the process calling into the current server process. It can be either a base client process or a server process.) **IsCallerInRole** doesn't apply to the process that initiated the call sequence from which the current method was called, or to any other callers in that sequence.

Because **IsCallerInRole** returns `true` when the object that invokes it is executing in a client's process, it's a good idea to call **IsSecurityEnabled** before calling **IsCallerInRole**. If security isn't enabled, **IsCallerInRole** won't return an accurate result.

Example

See Also

Programmatic Security, Basic Security Methods, Secured Components

IObjectContext.IsCallerInRole, IObjectContext.IsSecurityEnabled Methods Example

```
import com.ms.mtx.*;

IObjectContext objContext = null;

objContext = MTx.GetObjectContext();

// Find out if Security is enabled.
if (objContext.IsSecurityEnabled()) {
   //Then find out if the caller is in the right role.
   if (!objContext.IsCallerInRole("Manager")) {
      // If not, do something appropriate here.
   }
}
else {
   // If security's not enabled, do something
   // appropriate here.
}
```

IObjectContext.IsInTransaction Method

Indicates whether the current object is executing in a transaction.

New applications should call **isInTransaction**.

Provided By

IObjectContext

boolean IsInTransaction ( );

Return Values

`true`
The current object is executing within a transaction.

`false`
The current object is not executing within a transaction.

Remarks

You can use this method to make sure that an object that requires a transaction never runs without one. For example, if a component that requires a transaction is improperly configured in the MTS Explorer, you can use this method to determine that the object doesn't have a transaction. Then you can return an error to alert the user to the problem, or take whatever action is appropriate.

Example

See Also

Transaction Attributes, Transactions

IObjectContext.IsInTransaction Method Example

```
import com.ms.mtx.*;


// Find out if the object is in a transaction.
if (!MTx.GetObjectContext().IsInTransaction()) {
   // If not, do something appropriate here.
}
```

IObjectContext.IsSecurityEnabled Method

Indicates whether or not security is enabled for the current object. MTS security is enabled unless the object is running in the client's process.

New applications should call **isSecurityEnabled**.

Provided By

IObjectContext

boolean IsSecurityEnabled ( );

Return Values

`true`
Security is enabled for this object.

`false`
Security is not enabled for this object.

Remarks

MTS security is enabled only if an object is running in a server process. This could be either because the object's component was configured to run in a client's process, or because the component and the client are in the same package. If the object is running in the client's process, there is no security checking and **IsSecurityEnabled** will always return `false`.

Example

See Also

Programmatic Security, Basic Security Methods, Secured Components

IObjectContext.SetAbort Method

Declares that the transaction in which the object is executing must be aborted, and that the object should be deactivated on returning from the currently executing method call.

New applications should call **setAbort**.

Provided By

IObjectContext

void SetAbort ( );

Remarks

The object is deactivated automatically on return from the method in which it called **SetAbort**. If the object is the root of an automatic transaction, MTS aborts the transaction. If the object is transactional, but not the root of an automatic transaction, the transaction in which it's participating is doomed to abort. (An object is the root of a transaction if the MTS run-time environment has to initiate a new transaction for it. This is the case when the component that provides the object is configured to require a transaction and the object's creator doesn't have one, or when the component is configured to require a new transaction.)

You can call **SetAbort** in error handlers to ensure that a transaction aborts when an error occurs. You can also call **SetAbort** at the beginning of a method to protect your object from committing prematurely in the event of an unexpected return and then call **SetComplete** just before the method returns, if all goes well.

Example

See Also

Transactions, Context Objects, Deactivating Objects

IObjectContext.SetAbort, IObjectContext.SetComplete Methods Example

```
import com.ms.mtx.*;


boolean success = false;
// Do some work here.
// If the work was successful, call SetComplete
if (success)
   MTx.GetObjectContext().SetComplete();
// Otherwise, call SetAbort.
else
   MTx.GetObjectContext().SetAbort();
```

IObjectContext.SetComplete Method

Declares that the current object has completed its work and should be deactivated when the currently executing method returns to the client. For objects that are executing within the scope of a transaction, it also indicates that the object's transactional updates can be committed.

New applications should call **setComplete**.

Provided By

IObjectContext

void SetComplete ( );

Remarks

The object is deactivated automatically on return from the method in which it called **SetComplete**. If the object is the root of an automatic transaction, MTS attempts to commit the transaction. However, if any object that was participating in the transaction has called **SetAbort**, or has called **DisableCommit** and has not subsequently called **EnableCommit** or **SetComplete**, the transaction will be aborted. (An object is the root of a transaction if the MTS run-time environment has to initiate a new transaction for it. This is the case when the component that provides the object is configured to require a transaction and the object's creator doesn't have one, or when the component is configured to require a new transaction.)

If an object doesn't need to maintain its state after it returns from a method call, it should call **SetComplete** so that it can be automatically deactivated as soon as it returns and its resources can be reclaimed.

Example

See Also

Transactions, Context Objects, Deactivating Objects

getSafeRef Method

Used by an object to obtain a reference to itself that's safe to pass outside its context.

getSafeRef is a static method of the **Context** class, which is declared in the package `com.ms.mtx`.

Note The class **Context** has only static methods and has no public constructor. You can't create an instance of this class.

Provided By

Context class

```
IUnknown getSafeRef (
    IUnknown unknown
);
```

Parameter

unknown
[in] A reference to the current object.

Return Value

A reference to the current object that's safe to pass outside the current object's context.

Remarks

When an MTS object wants to pass a self-reference to a client or another object (for example, for use as a callback), it should always call **getSafeRef** first and then pass the reference returned by this call. An object should never pass a reference to to a client or to any other object. Once such a reference is passed outside the object's context, it's no longer a valid reference.

Calling **getSafeRef** on a reference that is already safe returns the safe reference unchanged.

See Also

Passing Object References

getObjectContext Method

Obtains a reference to the **ObjectContext** that's associated with the current MTS object.

getObjectContext is a static method of the **Context** class, which is declared in the package `com.ms.mtx.`

Note The class **Context** has only static methods and has no public constructor. You can't create an instance of this class.

Return Value

A reference to the **IObjectContext** interface on the current object's context. **getObjectContext** will return null if it is called from a constructor or finalizer, or if the object's component hasn't been imported into an MTS package.

Remarks

An object should never attempt to pass its **ObjectContext** reference to another object. If you pass an **ObjectContext** reference to another object, it will no longer be a valid reference.

See Also

Building Scalable Components, Context Objects

Context Class

The **Context** class provides access to the current object's context.

Remarks

The **Context** class is in the package `com.ms.mtx`.

You can use an object's **Context** to:

- 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.

The **Context** class exposes the following methods.

Method	Description
createObject	Instantiates another MTS object.
disableCommit	Declares that the object hasn't finished its work and that its transactional updates are in an inconsistent state. The object retains its state across method calls, and any attempts to commit the transaction before the object calls **enableCommit** or **setComplete** will result in the transaction being aborted.
enableCommit	Declares that the object's work isn't necessarily finished, but its transactional updates are in a consistent state. This method allows the transaction to be committed, but the object retains its state across method calls until it calls **setComplete** or **setAbort**, or until the transaction is completed.
getProperty	Returns a context object property.
getPropertyNames	Returns the names of the context object properties.
isCallerInRole	Indicates whether the object's direct caller is in a specified role (either directly or as part of a group).
isInTransaction	Indicates whether the object is executing within a transaction.
isSecurityEnabled	Indicates whether security is enabled. MTS security is enabled unless the object is running in the client's process.
getSafeRef	Used by an object to obtain a reference to itself that's safe to pass outside its context.
setAbort	Declares that the object has completed its work and can be deactivated on returning from the currently executing method, but that its transactional updates are in an inconsistent state or that an unrecoverable error occurred. This means that the transaction in which the object was executing must be aborted. If any object executing within a transaction returns to its

	client after calling **setAbort**, the entire transaction is doomed to abort.
setComplete	Declares that the object has completed its work and can be deactivated on returning from the currently executing method. For objects that are executing within the scope of a transaction, it also indicates that the object's transactional updates can be committed. When an object that is the root of a transaction calls **setComplete**, MTS attempts to commit the transaction on return from the current method.
getDirectCallerName	Retrieves the user name associated with the external process that called the currently executing method.
getDirectCreatorName	Retrieves the user name associated with the external process that directly created the current object.
getOriginalCallerName	Retrieves the user name associated with the base process that initiated the call sequence from which the current method was called.
getOriginalCreatorName	Retrieves the user name associated with the base process that initiated the activity in which the current object is executing.

Note When an object calls **disableCommit**, **enableCommit**, **setComplete**, or **setAbort** from within a method, two flags (Done and Consistent) are set in its **Context**. (See the following table for an explanation.) These flags aren't evaluated by the MTS run-time environment until the object's currently executing method returns to its caller. This means that an object can call these methods any number of times from within one of its own methods, but the last call before the object returns to its client is the one that will be in effect.

Method	Done	Consistent
setAbort	TRUE	FALSE
setComplete	TRUE	TRUE
disableCommit	FALSE	FALSE
enableCommit	FALSE	TRUE

The **Done** flag, which allows an object to be deactivated and its transaction to commit or abort, is only evaluated after the object returns from the call that first entered its context. For example, suppose client A calls into object B. Object B calls **setComplete** and then calls into object C (passing it a safe reference for a callback). Object C calls back to object B, and then object B returns to client A. Object B won't be deactivated when it returns to object C; it will be deactivated when it returns to client A.

See Also

Basic Security Methods, Passing Object References, Context Objects, Transactions, Deactivating Objects

createObject Method

Instantiates an MTS object.

Provided By

Context class

IUnknown createObject (String *progID***);**
IUnknown createObject (com.ms.com._Guid *rclsid*)

Parameters

progID
[in] The programmatic identifier (ProgID) of the object to be created.

rclsid
[in] Reference to the class identifier (CLSID) of the object to be created.

Return Value

A reference to the **IUnknown** interface on the newly created object.

Remarks

createObject creates a COM object. However, the object will have context only if its component is registered with MTS.

When you create an object by using **createObject**, the new object's context is derived from the current object's **Context** and the declarative properties of the new object's component. The new object always executes within the same activity as the object that created it. If the current object has a transaction, the transaction attribute of the new object's component determines whether or not the new object will execute within the scope of that transaction.

If the component's transaction attribute is set to either **Requires a transaction** or **Supports transactions**, the new object inherits its creator's transaction. If the component's transaction attribute is set to **Requires a new transaction**, MTS initiates a new transaction for the new object. If the component's transaction attribute is set to **Does not support transactions**, the new object doesn't execute under any transaction.

MTS always uses standard marshaling. Even if a component exposes the **IMarshal** interface, its **IMarshal** methods will never be called by the MTS run-time environment.

You can't create MTS objects as part of an aggregation.

See Also

Creating MTS Objects, Transaction Attributes, MTS Component Requirements

disableCommit Method

Declares that the object's transactional updates are inconsistent and can't be committed in their present state.

Provided By

Context class

void disableCommit ( );

Remarks

An object that invokes **disableCommit** is stateful.

You can use the **disableCommit** method to prevent a transaction from committing prematurely between method calls in a stateful object. When an object invokes **disableCommit**, it indicates that its work is inconsistent and that it can't complete its work until it receives further method invocations from the client. It also indicates that it needs to maintain its state to perform that work. This prevents the MTS run-time environment from deactivating the object and reclaiming its resources on return from a method call. Once an object has called **disableCommit**, if a client attempts to commit the transaction before the object has called **enableCommit** or **setComplete**, the transaction will abort.

For example, suppose you have a General Ledger component that updates a database. A client makes multiple calls to a General Ledger object to post entries to various accounts. There's an integrity constraint that says the debits must equal the credits when the final method invocation returns, or the transaction must abort. The General Ledger object has an initialization method in which the client informs it of the sequence of calls the client is going to make, and the General Ledger object calls **disableCommit**. The object maintains its state between calls so that after the final call in the sequence is made the object can make sure the integrity constraint is satisfied before allowing its work to be committed.

See Also

Transactions

enableCommit Method

Declares that the current object's work is not necessarily finished, but that its transactional updates are consistent and could be committed in their present form.

Provided By

Context class

void enableCommit ( );

Remarks

When an object calls **enableCommit**, it allows the transaction in which it's participating to be committed, but it maintains its internal state across calls from its clients until it calls **etComplete** or **setAbort** or until the transaction completes.

enableCommit is the default state when an object is activated. This is why an object should always call **setComplete** or **setAbort** before returning from a method, unless you want the object to maintain its internal state for the next call from a client.

See Also

Transactions

getProperty Method

Returns a context object property.

Provided By

Context class

Variant getProperty(
String *propertyName*);

Parameters

propertyName
[in] The name of the context object property to be retrieved.

Return Value

The requested context object property.

Remarks

You can use **getProperty** to retrieve the following Microsoft Internet Information Server (IIS) intrinsic objects:

- Request
- Response
- Server
- Application
- Session

For more information, see the IIS documentation.

To retrieve an IIS object, use the **getDispatch** method of returned Variant and cast this value to the interface to the IIS object (for example, **IReponse** for the Response object).

Note Context properties are not available in MTS 1.0 server processes.

getPropertyNames Method

Returns the names of the context object properties.

Provided By

Context class

String[] getPropertyNames ( );

Return Value

An array of context object property names.

isCallerInRole Method

Indicates whether an object's direct caller is in a specified role (either individually or as part of a group).

Provided By

Context class

boolean isCallerInRole (
  String *role*
);

Parameters

role
  [in] The name of the role in which you want to determine whether the caller is acting.

Return Values

`true`
  Either the caller is in the specified role, or security is not enabled.

`false`
  The caller is not in the specified role.

Remarks

You use this method to determine whether the direct caller of the currently executing method is associated with a specific role. A role is a symbolic name that represents a user or group of users who have specific access permissions to all components in a given package. Developers define roles when they create a component, and roles are mapped to individual users or groups at deployment time.

isCallerInRole only applies to the direct caller of the currently executing method. (The direct caller is the process calling into the current server process. It can be either a base client process or a server process.) **isCallerInRole** doesn't apply to the process that initiated the call sequence from which the current method was called, or to any other callers in that sequence.

Because **isCallerInRole** returns `true` when the object that invokes it is executing in a client's process, it's a good idea to call **isSecurityEnabled** before calling **isCallerInRole**. If security isn't enabled, **isCallerInRole** won't return an accurate result.

See Also

Programmatic Security, Basic Security Methods, Secured Components

isInTransaction Method

Indicates whether the current object is executing in a transaction.

Provided By

Context class

boolean isInTransaction ( );

Return Values

`true`
The current object is executing within a transaction.

`false`
The current object is not executing within a transaction.

Remarks

You can use this method to make sure that an object that requires a transaction never runs without one. For example, if a component that requires a transaction is improperly configured in the MTS Explorer, you can use this method to determine that the object doesn't have a transaction. Then you can return an error to alert the user to the problem, or take whatever action is appropriate.

See Also

Transaction Attributes, Transactions

isSecurityEnabled Method

Indicates whether or not security is enabled for the current object. MTS security is enabled unless the object is running in the client's process.

Provided By

Context class

boolean isSecurityEnabled ( );

Return Values

`true`
Security is enabled for this object.

`false`
Security is not enabled for this object.

Remarks

MTS security is enabled only if an object is running in a server process. This could be either because the object's component was configured to run in a client's process, or because the component and the client are in the same package. If the object is running in the client's process, there is no security checking and **isSecurityEnabled** will always return `false`.

See Also

Programmatic Security, Basic Security Methods, Secured Components

setAbort Method

Declares that the transaction in which the object is executing must be aborted, and that the object should be deactivated on returning from the currently executing method call.

Provided By

Context class

void setAbort ( );

Remarks

The object is deactivated automatically on return from the method in which it called **setAbort**. If the object is the root of an automatic transaction, MTS aborts the transaction. If the object is transactional, but not the root of an automatic transaction, the transaction in which it's participating is doomed to abort. (An object is the root of a transaction if the MTS run-time environment has to initiate a new transaction for it. This is the case when the component that provides the object is configured to require a transaction and the object's creator doesn't have one, or when the component is configured to require a new transaction.)

You can call **setAbort** in error handlers to ensure that a transaction aborts when an error occurs. You can also call **setAbort** at the beginning of a method to protect your object from committing prematurely in the event of an unexpected return and then call **setComplete** just before the method returns, if all goes well.

See Also

Transactions, Context Objects, Deactivating Objects

setComplete Method

Declares that the current object has completed its work and should be deactivated when the currently executing method returns to the client. For objects that are executing within the scope of a transaction, it also indicates that the object's transactional updates can be committed.

Provided By

Context class

void setComplete ( );

Remarks

The object is deactivated automatically on return from the method in which it called **setComplete**. If the object is the root of an automatic transaction, MTS attempts to commit the transaction. However, if any object that was participating in the transaction has called **setAbort**, or has called **disableCommit** and has not subsequently called **enableCommit** or **setComplete**, the transaction will be aborted. (An object is the root of a transaction if the MTS run-time environment has to initiate a new transaction for it. This is the case when the component that provides the object is configured to require a transaction and the object's creator doesn't have one, or when the component is configured to require a new transaction.)

If an object doesn't need to maintain its state after it returns from a method call, it should call **setComplete** so that it can be automatically deactivated as soon as it returns and its resources can be reclaimed.

See Also

Transactions, Context Objects, Deactivating Objects

IObjectContextActivity Interface

The **IObjectContextActivity** interface is used to retrieve a unique identifier associated with the current activity. This activity identifier is a GUID, and is only valid for the lifetime of the current activity.

Remarks

The header file for the **IObjectContextActivity** is mtx.h. You must also link mtxguid.lib to your project to use this interface.

You obtain a reference to an object's **IObjectContextActivity** interface by calling **QueryInterface** on the object's **ObjectContext**. For example:

```
m_pIObjectContext->QueryInterface
    (IID_IObjectContextActivity,
    (void**)&m_pIObjectContextActivity));
```

The **IObjectContextActivity** interface provides the following methods.

Method	Description
GetActivityId	Retrieves the GUID associated with the current activity.

IObjectContextActivity::GetActivityId Method

Retrieves the GUID associated with the current activity.

Provided By

IObjectContextActivity

HRESULT IObjectContextActivity::GetActivityId(
GUID * *pActivityId*
);

Parameters

pActivityId
[out] A reference to the GUID associated with the current activity.

Return Values

S_OK
The GUID of the current activity is returned in the parameter *pActivityId*.

E_INVALIDARG
The argument passed in the *pActivityId* parameter is a NULL pointer.

E_UNEXPECTED
An unexpected error occurred.

Example

GetActivityId Method Example

```
#include <mtx.h>

HRESULT hr = S_OK;
IObjectContext *pIObjectContext = NULL;
IObjectContextActivity *pIObjectContextActivity = NULL;
GUID activityId;

// Get object context
hr = GetObjectContext(&pIObjectContext);
// Get IObjectContextActivity interface
hr = pIObjectContext->
    QueryInterface(IID_IObjectContextActivity,
    (void**)&pIObjectContextActivity);
// Use IObjectContextActivity to retrieve
// the activity GUID.
hr = pIObjectContextActivity->
    GetActivityId(&activityId);

// Do something with the activity GUID here.

// Release the IObjectContextActivity
// and the IObjectContext pointers
pIObjectContextActivity->Release();
pIObjectContext->Release();
```

SharedPropertyGroupManager Object

The **SharedPropertyGroupManager** object is used to create shared property groups and to obtain access to existing shared property groups.

Remarks

To use the **SharedPropertyGroupManager** object, you must set a reference to the Shared Property Manager Type Library (mtxspm.dll).

You can access the **SharedPropertyGroupManager** object by using either the **CreateObject** function or the **CreateInstance** method of the **ObjectContext** object. It makes no difference which you use.

The Shared Property Manager is a resource dispenser that you can use to share state among multiple objects within a server process. You can't use global variables in a distributed environment because of concurrency and name collision issues. The Shared Property Manager eliminates name collisions by providing shared property groups, which establish unique name spaces for the shared properties they contain. The Shared Property Manager also implements locks and semaphores to protect shared properties from simultaneous access, which could result in lost updates and could leave the properties in an unpredictable state.

Shared properties can be shared only by objects running in the same process. If you want instances of different components to share properties, you have to install the components in the same MTS package. Because there is a risk that administrators will move components from one package to another, it's safest to limit the use of a shared property group to instances of components that are defined in the same DLL.

It's also important for components sharing properties to have the same activation attribute. If two components in the same package have different activation attributes, they generally won't be able to share properties. For example, if one component is configured to run in a client's process and the other is configured to run in a server process, their objects will usually run in different processes, even though they're in the same package.

You should always instantiate the **SharedPropertyGroupManager**, **SharedPropertyGroup**, and **SharedProperty** objects from MTS objects rather than from a base client. If a base client creates shared property groups and properties, the shared properties are inside the base client's process, not

in a server process. This means MTS objects can't share the properties unless the objects, too, are running in the client's process (which is generally not a good idea).

Note When you set the isolation mode to **LockMethod**, the Shared Property Manager requires access to the calling object's **ObjectContext**. You can't use this isolation mode to create a shared property group from within an object's constructor or from a non-MTS object because **ObjectContext** isn't available during object construction and a non-MTS object doesn't have an **ObjectContext**.

The **SharedPropertyGroupManager** object provides the following methods and properties.

Method	Description
CreatePropertyGroup	Creates a new **SharedPropertyGroup** with a string name as an identifier. If a group with the specified name already exists, **CreatePropertyGroup** returns a reference to the existing group.
Group	Returns a reference to an existing shared property group, given a string name by which it can be identified.

See Also

Sharing State

CreatePropertyGroup Method

Creates and returns a reference to a new shared property group. If a property group with the specified name already exists, **CreatePropertyGroup** returns a reference to the existing group.

Applies To

SharedPropertyGroupManager Object

Syntax

Set *propertygroup* = *sharedpropertygroupmanager*.**CreatePropertyGroup**(*name*, *dwIsoMode*, *dwrelmode, fExists*)

Parameters

propertygroup
An object variable that evaluates to a **SharedPropertyGroup** object.

sharedpropertygroupmanager
An object variable that represents the **SharedPropertyGroupManager** with which to create the shared property group.

name
A string expression that contains the name of the shared property group to create.

dwIsoMode
A **Long** value that specifies the isolation mode for the properties in the new shared property group. See the table that lists *dwIsoMode* constants later in this topic. If the value of the *fExists* parameter is set to **True** on return from this method, the *dwIsoMode* value you passed in is ignored and the value returned in this parameter is the isolation mode that was assigned when the property group was created.

dwRelMode
A **Long** value that specifies the release mode for the properties in the new shared property group. See the table that lists *dwRelMode* constants later in this topic. If the value of the *fExists* parameter is set to **True** on return from this method, the *dwRelMode* value you passed in is ignored and the value returned in this parameter is the release mode that was assigned when the property group was created.

fExists
A **Boolean** value that's set to **True** on return from this method if the shared property group specified in the *name* parameter existed prior to this call, and **False** if the property group was created by this call.

Settings

The following constants are used in the *dwIsoMode* parameter to specify the effective isolation mode for a shared property group.

Constant	Value	Description
LockSetGet	0	Default. Locks a property during a **Value** call, assuring that every get or set operation on a shared property is atomic. This ensures that two clients can't read or write to the same property at the same time, but it doesn't prevent other clients from concurrently accessing other properties in the same group.
LockMethod	1	Locks all of the properties in the shared property group for exclusive use by the caller as long as

	the caller's current method is executing.
	This is the appropriate mode to use when there are interdependencies among properties, or in cases where a client may have to update a property immediately after reading it before it can be accessed again.

Note When you set the isolation mode to **LockMethod**, the Shared Property Manager requires access to the calling object's **ObjectContext**. You can't use this isolation mode to create a shared property group from within an object's constructor or from a non-MTS object because **ObjectContext** isn't available during object construction and a base client doesn't have an **ObjectContext**.

The following constants are used in the *dwRelMode* parameter to specify the effective release mode for a shared property group.

Constant	Value	Description
Standard	0	When all clients have released their references on the property group, the property group is automatically destroyed.
Process	1	The property group isn't destroyed until the process in which it was created has terminated. You must still release all **SharedPropertyGroup** objects by setting them to **Nothing**.

Remarks

The **CreatePropertyGroup** method sets the value in *fExists* to **True** if the property group it returns existed prior to the current call. This occurs when another object in the same process has already called **CreatePropertyGroup** with the same property group name. The **CreatePropertyGroup** method sets the value in *fExists* to **False** if the returned property group was created by the current call.

The isolation mode and release mode are assigned when the property group is originally created and aren't changed if a subsequent call passes different values in these parameters. The caller should always check the value of *fExists* on return from this method. If *fExists* is set to **True**, the caller should check the values returned in *dwIsoMode* and *dwRelMode* to determine the isolation and release modes in effect for the property group. For example:

```
Dim isolationMode As Long
Dim releaseMode As Long

Set isolationMode = LockMethod
Set releaseMode = Process
Set spmGroup = spmMgr.CreatePropertyGroup _
   ("Counter", isolationMode, releaseMode, fExists)

If fExists Then
   If isolationMode <> LockMethod _
      Or releaseMode <> Process Then
      ' Do something appropriate.
   EndIf
EndIf
```

You can pass the constants, **LockGetSet** or **LockMethod** as the *dwIsoMode* argument, and **Standard** or **Process** as the *dwRelMode* argument, directly to the **CreatePropertyGroup** method. However, when you pass a constant instead of a variable, the **CreatePropertyGroup** method can't return the isolation and release modes currently in effect if the requested property group already

exists.

Note An object should never attempt to pass a shared property group reference to another object. If the reference is passed outside of the object that acquired it, it's no longer a valid reference.

Example

See Also

Sharing State, **IObjectContext** Interface, **SharedPropertyGroup** Object

Group Property

Returns a reference to an existing shared property group.

Applies To

ISharedPropertyGroupManager Interface

Syntax

Set *propertygroup* = *sharedpropertygroupmanager*.**Group**(*name*)

Parameters

propertygroup
An object variable that evaluates to a **SharedPropertyGroup** object.

sharedpropertygroupmanager
An object variable that represents the **SharedPropertyGroupManager** for the current process.

name
A string expression that contains the name of the shared property group to retrieve.

Example

See Also

Sharing State, **SharedPropertyGroupManager** Object

SharedPropertyGroup Object

The **SharedPropertyGroup** object is used to create and access the shared properties in a shared property group.

Remarks

To use the **SharedPropertyGroup** object, you must set a reference to the Shared Property Manager Type Library (mtxspm.dll)

You can create a **SharedPropertyGroup** object with the **CreatePropertyGroup** method of the **SharedPropertyGroupManager** object.

As with any COM object, you must release a **SharedPropertyGroup** object when you're finished using it, unless it's a local variable. For example:

```
Set myPropertyGroup = Nothing
```

The **SharedPropertyGroup** object provides the following methods and properties.

Method/Property	Description
CreateProperty	Creates a new shared property identified by a string expression that's unique within its property group.
CreatePropertyByPosition	Creates a new shared property identified by a numeric index within its property group.
Property	Returns a reference to a shared property, given the string name by which the property is identified.
PropertyByPosition	Returns a reference to a shared property, given its numeric index in the shared property group.

See Also

Sharing State, **ISharedPropertyGroupManager** Object

CreateProperty Method

Creates and returns a reference to a new **SharedProperty** with a specified name. If a shared property by that name already exists, **CreateProperty** returns a reference to the existing property.

Applies To

SharedPropertyGroup Object

Syntax

Set *property* = *propertygroup*.**CreateProperty**(*name, fExists*)

Parameters

propertygroup
An object variable that represents the **SharedPropertyGroup** to which the new **SharedProperty** object will belong.

property
An object variable that evaluates to a **SharedProperty** object.

name
A string expression that contains the name of the property to create. You can use this name later to obtain a reference to this property.

fExists
A Boolean value that's set to **True** on return from this method if the shared property specified in the *name* parameter existed prior to this call, and **False** if the property was created by this call.

Remarks

When you create a shared property, its value is set to the default, which is 0.

If you create a shared property with the **CreateProperty** method, you can access that property only by using **Property**. You can't assign a numeric index to the same property and then access it by using **PropertyByPosition**.

The same shared property group can contain some shared property objects that are identified by name and others that are identified by position.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreatePropertyByPosition** Method, **ISharedPropertyGroup::get_PropertyByPosition** Method, **ISharedPropertyGroup::get_Property** Method

CreatePropertyByPosition Method

Creates a new shared property identified by a numeric index that's unique within the property group. If a shared property with the specified index already exists, **CreatePropertyByPosition** returns a reference to the existing one.

Applies To

SharedPropertyGroup Object

Syntax

Set *property* = *propertygroup*.**CreatePropertyByPosition** (*index, fExists*)

Parameters

property
An object variable that evaluates to a **SharedProperty** object.

propertygroup
An object variable that represents the **SharedPropertyGroup** to which the new **SharedProperty** object will belong.

index
A **Long** value that represents the numeric index within the **SharedPropertyGroup** by which the new property will be referenced. You can use this index later to retrieve the shared property with **PropertyByPosition**.

fExists
A **Boolean** value. If *fExists* is set to **True** on return from this method, the shared property specified by *index* existed prior to this call. If it's set to **False**, the property was created by this call.

Remarks

When you create a shared property, its value is set to the default, which is 0.

If you create a **SharedProperty** object with the **CreatePropertyByPosition** method, you can access that property only by using **PropertyByPosition**. You can't assign a string name to the same property and then access it by using **Property**. Accessing a property by position is faster than accessing a property by using a string name because it requires less overhead.

The same shared property group can contain some **SharedProperty** objects that are identified by position and others that are identified by name.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreateProperty** Method, **ISharedPropertyGroup::get_PropertyByPosition** Method, **ISharedPropertyGroup::get_Property** Method

CreatePropertyByPosition Method Example

```
Dim spmMgr As SharedPropertyGroupManager
Dim spmGroup As SharedPropertyGroup
Dim spmPropNextNumber As SharedProperty
Dim bExists As Boolean
Dim iNextValue As Integer

' Create the SharedPropertyGroupManager,
' SharedPropertyGroup, and SharedProperty.
Set spmMgr = CreateObject _
   ("MTxSpm.SharedPropertyGroupManager.1")
Set spmGroup = spmMgr.CreatePropertyGroup _
   ("Counter", LockSetGet, Process, bExists)
Set spmPropNextNumber = _
   spmGroup.CreatePropertyByPosition(0, bExists)

' Get the next number and increment it.
iNextValue = spmPropNextNumber.Value
spmPropNextNumber.Value = _
   spmPropNextNumber.Value + 1
```

Property Property

Returns a reference to an existing shared property identified by a string name.

Applies To

SharedPropertyGroup Object

Syntax

Set *property* = *propertygroup*.**Property**(*name)*

Parameters

propertygroup
An object variable that represents the **SharedPropertyGroup** to which the **SharedProperty** object belongs.

property
An object variable that evaluates to a **SharedProperty** object.

name
A string expression that contains the name of the shared property to retrieve.

Remarks

You can use only **Property** to access properties that were created with the **CreateProperty** method. To access properties that were created with the **CreatePropertyByPosition** method, use **PropertyByPosition**.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreateProperty** Method, **ISharedPropertyGroup::CreatePropertyByPosition** Method, **ISharedPropertyGroup::get_PropertyByPosition** Method

Property, Group Properties Example

```
Dim spmMgr As SharedPropertyGroupManager
Dim spmGroup As SharedPropertyGroup
Dim spmPropNextNumber As SharedProperty
Dim iNextValue As Integer

' Get the SharedPropertyGroupManager,
' SharedPropertyGroup, and SharedProperty.
Set spmMgr = CreateObject _
   ("MTxSpm.SharedPropertyGroupManager.1")
Set spmGroup = spmMgr.Group("Counter")
Set spmPropNextNumber = spmGroup.Property("Next")

' Get the next number and increment it.
iNextValue = spmPropNextNumber.Value
spmPropNextNumber.Value = _
   spmPropNextNumber.Value + 1
```

PropertyByPosition Property

Returns a reference to an existing shared property identified by its numeric index within the property group.

Applies To

SharedPropertyGroup Object

Syntax

Set *sharedproperty* = *propertygroup*.**PropertyByPosition**(*index*)

Parameters

propertygroup
An object variable that represents the **SharedPropertyGroup** object to which the **SharedProperty** object belongs.

sharedproperty
An object variable that evaluates to a **SharedProperty** object.

index
A **Long** value that represents the numeric index within the **SharedPropertyGroup** of the property to retrieve.

Remarks

You can use only **PropertyByPosition** to access properties that were created with the **CreatePropertyByPosition** method. To access properties that were created with the **CreateProperty** method, use **Property**.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreateProperty** Method, **ISharedPropertyGroup::CreatePropertyByPosition** Method, **ISharedPropertyGroup::get_Property** Method

PropertyByPosition Property Example

```
Dim spmMgr As SharedPropertyGroupManager
Dim spmGroup As SharedPropertyGroup
Dim spmPropNextNumber As SharedProperty
Dim iNextValue As Integer

' Get the SharedPropertyGroupManager,
' SharedPropertyGroup, and SharedProperty.
Set spmMgr = CreateObject _
   ("MTxSpm.SharedPropertyGroupManager.1")
Set spmGroup = spmMgr.Group("Counter")
Set spmPropNextNumber = spmGroup.PropertyByPosition(0)

' Get the next number and increment it.
iNextValue = spmPropNextNumber.Value
spmPropNextNumber.Value = _
   spmPropNextNumber.Value + 1
```

SharedProperty Object

The **SharedProperty** object is used to set or retrieve the value of a shared property. A shared property can contain any data type that can be represented by a variant.

Remarks

To use the **SharedProperty** object, you must set a reference to the Shared Property Manager Type Library (mtxspm.dll).

You can create a **SharedProperty** object with the **CreateProperty** method or the **CreatePropertyByPosition** method.

A **SharedProperty** object can be created or accessed only from within a **SharedPropertyGroup**.

As with any COM object, you must release a **SharedProperty** object when you're finished using it, unless it's a local variable. For example:

```
Set myProperty = Nothing
```

The **SharedProperty** object provides the following property.

Property	Description
Value	Sets or retrieves the value of a shared property.

See Also

Sharing State, MTS Supported Variant Types

Value Property

Sets or retrieves the value of a shared property.

Applies To

SharedProperty Object

Syntax

property.**Value** = *value*

Parameters

property
An object variable that represents a **SharedProperty** object.

value
A **Variant** containing the value to assign to the **SharedProperty** object, or the **SharedProperty**'s current value.

Example

See Also

MTS Supported Variant Types, Sharing State

CreatePropertyGroup Method, CreateProperty Method, Value Property Example

```
Dim spmMgr As SharedPropertyGroupManager
Dim spmGroup As SharedPropertyGroup
Dim spmPropNextNumber As SharedProperty
Dim bExists As Boolean
Dim iNextValue As Integer

' Create the SharedPropertyGroupManager,
' SharedPropertyGroup, and SharedProperty.
Set spmMgr = CreateObject _
   ("MTxSpm.SharedPropertyGroupManager.1")
Set spmGroup = spmMgr.CreatePropertyGroup _
   ("Counter", LockSetGet, Process, bExists)
Set spmPropNextNumber = _
   spmGroup.CreateProperty("Next", bExists)

' Get the next number and increment it.
iNextValue = spmPropNextNumber.Value
spmPropNextNumber.Value = _
   spmPropNextNumber.Value + 1
```

ISharedPropertyGroupManager Interface

The **ISharedPropertyGroupManager** interface is used to create shared property groups and to obtain access to existing shared property groups.

Remarks

The header file for the **ISharedPropertyGroupManager** interface is mtxspm.h. You must also link mtxguid.lib to your project to use this interface.

You can access the **ISharedPropertyGroupManager** interface by creating an instance of the **SharedPropertyGroupManager** by using either **IObjectContext::CreateInstance** or **CoCreateInstance**. It makes no difference which you use.

CreateInstance method of the **ObjectContext** object. It makes no difference which you use.

The Shared Property Manager is a resource dispenser that you can use to share state among multiple objects within a server process. You can't use global variables in a distributed environment because of concurrency and name collision issues. The Shared Property Manager eliminates name collisions by providing shared property groups, which establish unique name spaces for the shared properties they contain. The Shared Property Manager also implements locks and semaphores to protect shared properties from simultaneous access, which could result in lost updates and could leave the properties in an unpredictable state.

Shared properties can be shared only by objects running in the same process. If you want instances of different components to share properties, you have to install the components in the same MTS package. Because there is a risk that administrators will move components from one package to another, it's safest to limit the use of a shared property group to instances of components that are defined in the same DLL.

It's also important for components sharing properties to have the same activation attribute. If two components in the same package have different activation attributes, they generally won't be able to share properties. For example, if one component is configured to run in a client's process and the other is configured to run in a server process, their objects will usually run in different processes, even though they're in the same package.

You should always instantiate the **SharedPropertyGroupManager**, **SharedPropertyGroup**, and **SharedProperty** objects from MTS objects rather than from a base client. If a base client creates

shared property groups and properties, the shared properties are inside the base client's process, not in a server process. This means MTS objects can't share the properties unless the objects, too, are running in the client's process (which is generally not a good idea).

Note When you set the isolation mode to **LockMethod**, the Shared Property Manager requires access to the calling object's **ObjectContext**. You can't use this isolation mode to create a shared property group from within an object's constructor or from a non-MTS object because **ObjectContext** isn't available during object construction and a non-MTS object doesn't have an **ObjectContext**.

The **ISharedPropertyGroupManager** interface exposes the following methods and properties.

Method	Description
CreatePropertyGroup	Creates a new **SharedPropertyGroup** with a string name as an identifier. If a group with the specified name already exists, **CreatePropertyGroup** returns a reference to the existing group.
get_Group	Returns a reference to an existing shared property group, given a string name by which it can be identified.
get__NewEnum	Returns a reference to an enumerator that iterates through a list of all the shared property groups in a given process.

See Also

Sharing State

ISharedPropertyGroupManager::CreatePropertyGroup Method

Creates and returns a reference to a new shared property group. If a property group with the specified name already exists, **CreatePropertyGroup** returns a reference to the existing group.

Provided By

ISharedPropertyGroupManager Interface

HRESULT ISharedPropertyGroup::CreatePropertyGroup (
BSTR *name*,
LONG* *plIsoMode*,
LONG* *plRelMode*,
VARIANT_BOOL* *pfExists,*
ISharedPropertyGroup** *ppGroup*,
);

Parameters

name
[in] The name of the shared property group to create.

plIsoMode
[in, out] A reference to a LONG that specifies the isolation mode for the properties in the new shared property group. See the table that lists *plIsoMode* constants later in this topic. If the value of the *pfExists* parameter is set to VARIANT_TRUE on return from this method, the *plIsoMode* value you passed in is ignored and the value returned in this parameter is the isolation mode that was assigned when the property group was created.

plRelMode
[in, out] A reference to a LONG that specifies the release mode for the properties in the new shared property group. See the table that lists *plRelMode* constants later in this topic. If the value of the *pfExists* parameter is set to VARIANT_TRUE on return from this method, the *plRelMode* value you passed in is ignored and the value returned in this parameter is the release mode that was assigned when the property group was created.

pfExists
[out] A reference to a BOOL that's set to VARIANT_TRUE on return from this method if the shared property group specified in the *name* parameter existed prior to this call, and VARIANT_FALSE if the property group was created by this call.

ppGroup
[out] A reference to a shared property group identified by the BSTR passed in the *name* parameter, or NULL if an error is encountered.

Settings

The following constants are used in the *plIsoMode* parameter to specify the effective isolation mode for a shared property group.

Constant	Value	Description
LockSetGet	0	Default. Locks a property during a **get_Value** or **put_Value**call, assuring that every get or set operation on a shared property is atomic. This ensures that two clients can't read or write to the same property at the same time, but it doesn't prevent other clients from concurrently accessing other properties in the same group.
LockMethod	1	Locks all of the properties in the shared property

group for exclusive use by the caller as long as the caller's current method is executing.

This is the appropriate mode to use when there are interdependencies among properties, or in cases where a client may have to update a property immediately after reading it before it can be accessed again.

Note When you set the isolation mode to **LockMethod**, the Shared Property Manager requires access to the calling object's **ObjectContext**. You can't use this isolation mode to create a shared property group from within an object's constructor or from a non-MTS object because **ObjectContext** isn't available during object construction and a base client doesn't have an **ObjectContext**.

The following constants are used in the *plRelMode* parameter to specify the effective release mode for a shared property group.

Constant	Value	Description
Standard	0	When all clients have released their references on the property group, the property group is automatically destroyed. (This is the default COM mode.)
Process	1	The property group isn't destroyed until the process in which it was created has terminated. (Objects that hold references on a property group must still call **Release** on their references).

Return Values

S_OK
A reference to the shared property group specified in the *name* parameter is returned in the *ppGroup* parameter.

CONTEXT_E_NOCONTEXT
The caller isn't executing under the MTS run-time environment. A caller must be executing under MTS to use the Shared Property Manager.

E_INVALIDARG
At least one of the parameters is invalid, or the same object is attempting to create the same property group more than once.

Remarks

The **CreatePropertyGroup** method sets the value in *pfExists* to VARIANT_TRUE if the property group it returns in the *ppGroup* parameter existed prior to the current call. This occurs when another object in the same process has already called **CreatePropertyGroup** with the same property group name. The **CreatePropertyGroup** method sets the value in *pfExists* to VARIANT_FALSE if the returned property group was created by the current call.

The isolation mode and release mode are assigned when the property group is originally created and aren't changed if a subsequent call passes different values in these parameters. The caller should always check the value of *pfExists* on return from this method. If *pfExists* is set to VARIANT_TRUE, the caller should check the values returned in *plIsoMode* and *plRelMode* to determine the isolation and release modes in effect for the property group. For example:

```
hr = pPropGpMgr->CreatePropertyGroup(stName,
   &lIsolationMode, &lReleaseMode, &fAlreadyExists,
   &pPropGp);
if (fAlreadyExists) {
```

```
   if ((lIsolationMode != LockMethod) ||
      (lReleaseMode != Process)) {
      // Do something appropriate.
   }
}
If*
```

Note An object should never attempt to pass a shared property group reference to another object. If the reference is passed outside of the object that acquired it, it's no longer a valid reference.

Example

See Also

Sharing State, **IObjectContext** Interface, **ISharedPropertyGroup** Interface

ISharedPropertyGroupManager::get_Group Method

Returns a reference to an existing shared property group.

Provided By

ISharedPropertyGroupManager Interface

HRESULT ISharedPropertyGroupManager::get_Group (
  BSTR *name*,
  ISharedPropertyGroup** *ppGroup*,
);

Parameters

name
  [in] The name of the shared property group to retrieve.

ppGroup
  [out] A reference to the shared property group specified in the *name* parameter, or NULL if the property group doesn't exist.

Return Values

S_OK
  The shared property group exists, and a reference to it is returned in the *ppGroup* parameter.

E_INVALIDARG
  The shared property group with the name specified in the *name* parameter doesn't exist.

Example

See Also

Sharing State, **ISharedPropertyGroupManager** Interface

ISharedPropertyGroupManager::get__NewEnum Method

Returns a reference to an enumerator that you can use to iterate through all the shared property groups in a process.

Provided By

ISharedPropertyGroupManager Interface

HRESULT ISharedPropertyGroupManager::get__NewEnum (
    IUnknown** *ppEnumerator*
);

Parameters

ppEnumerator
    [out] A reference to the **IUnknown** interface on a new enumerator object that you can use to iterate through the list of all the shared property groups in the process.

Return Values

S_OK
    A reference to the requested enumerator is returned in the *ppEnumerator* parameter.

Remarks

You use the **get__NewEnum** method to obtain a reference to an enumerator object. You should immediately call **QueryInterface** on the returned **IUnknown** for the **IEnumVARIANT** interface. This interface exposes several methods you can use to iterate through a list of BSTRs representing shared property group names. Once you have a name, you can use the **get_Group** method to obtain a reference to the shared property group it represents.

As with any COM object, you must release an enumerator object when you're finished using it. When you enumerate the shared property groups, all groups will be included. However, if you then call **CreatePropertyGroup** to add a new group, the existing enumerator won't include the new group even if you call **Reset** or **Clone**. To include the new group, you must create a new enumerator by calling **get__NewEnum** again.

Note **get__NewEnum** has two underscore characters between **get** and **NewEnum**.

Example

See Also

ISharedPropertyGroupManager::get_Group Method, **ISharedPropertyGroup Interface**

ISharedPropertyGroupManager::get__NewEnum Method Example

```
#include <mtxspm.h>

ISharedPropertyGroupManager* pspgm = NULL;
IUnknown* pUnknown = NULL;
IEnumVARIANT* pEnum = NULL;
VARIANT v;
ULONG cElementsFetched;
int i;
HRESULT hr;

// Get the enumerator object.
hr = pspgm->get__NewEnum(&pUnknown);

// Query for the IEnumVARIANT interface.
hr = pUnknown->QueryInterface(IID_IEnumVARIANT, (void**) &pEnum);

// Use the enumerator to iterate through
// the property group names.
for(i = 0; i < 10; i++)
{
   VariantInit(&v);
   pEnum->Next(1, &v, &cElementsFetched);
   // Do something with the returned
   // property group names.
}
```

ISharedPropertyGroup Interface

The **ISharedPropertyGroup** interface is used to create and access the shared properties in a shared property group.

Remarks

The header file for the **ISharedPropertyGroup** interface is mtxspm.h. You must also link mtxguid.lib to your project to use this interface.

You can access the **ISharedPropertyGroup** interface by creating a **SharedPropertyGroup** object with the **ISharedPropertyGroupManager::CreatePropertyGroup** method.

As with any COM object, you must release a **SharedPropertyGroup** object when you're finished using it.

The **ISharedPropertyGroup** interface exposes the following methods.

CreateProperty	Creates a new shared property identified by a string expression that's unique within its property group.
CreatePropertyByPosition	Creates a new shared property identified by a numeric index within its property group.
get_Property	Returns a reference to a shared property, given the string name by which the property is identified.
get_PropertyByPosition	Returns a reference to a shared property, given its numeric index in the shared property group.

See Also

Sharing State, **ISharedPropertyGroupManager** Interface

ISharedPropertyGroup::CreateProperty Method

Creates and returns a reference to a new **SharedProperty** with a specified name. If a shared property by that name already exists, **CreateProperty** returns a reference to the existing property.

Provided By

ISharedPropertyGroup Interface

HRESULT ISharedPropertyGroup::CreateProperty (
  BSTR *name*,
  VARIANT_BOOL* *pfExists;*
  ISharedProperty** *ppProp*,
);

Parameters

name
  [in] The name of the property to create. You can use this name later to obtain a reference to this property by using the **get_Property** method.

pfExists
  [out] A reference to a Boolean value that's set to VARIANT_TRUE on return from this method if the shared property specified in the *name* parameter existed prior to this call, and VARIANT_FALSE if the property was created by this call.

ppProp
  [out] A reference to a **SharedProperty** object with the name specified in the *name* parameter, or NULL if an error is encountered.

Return Values

S_OK
  A reference to a shared property with the name specified in the *name* parameter is returned in the parameter *ppProp*.

E_INVALIDARG
  One or more of the arguments passed in is invalid.

Remarks

When you create a shared property, its value is set to the default, which is a VARIANT of type VT_I4, with a value of 0.

If you create a shared property with the **CreateProperty** method, you can access that property only by using the **get_Property** method. You can't assign a numeric index to the same property and then access it by using the **get_PropertyByPosition** method.

The same shared property group can contain some shared property objects that are identified by name and others that are identified by position.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreatePropertyByPosition** Method, **ISharedPropertyGroup::get_PropertyByPosition** Method, **ISharedPropertyGroup::get_Property** Method

ISharedPropertyGroup::CreatePropertyByPosition Method

Creates a new shared property identified by a numeric index that's unique within the property group. If a shared property with the specified index already exists, **CreatePropertyByPosition** returns a reference to the existing one.

Provided By

ISharedPropertyGroup Interface

HRESULT ISharedPropertyGroup::CreatePropertyByPosition (
  INT *index*,
  VARIANT_BOOL* *pfExists;*
  ISharedProperty** *ppProp*,
);

Parameters

index
  [in] The numeric index within the **SharedPropertyGroup** by which the new property will be referenced. You can use this index later to retrieve the shared property with the **get_PropertyByPosition** method.

pfExists
  [out] A reference to a Boolean value. If *pfExists* is set to VARIANT_TRUE on return from this method, the shared property specified by *index* existed prior to this call. If it's set to VARIANT_FALSE, the property was created by this call.

ppProp
  [out] A reference to a shared property object identified by the numeric index passed in the *index* parameter, or NULL if an error is encountered.

Return Values

S_OK
  A reference to the shared property occupying the position specified in the *index* parameter is returned in the *ppProp* parameter.

E_INVALIDARG
  One or more of the arguments passed in is invalid.

Remarks

When you create a shared property, its value is set to the default, which is a VARIANT of type VT_I4, with a value of 0.

If you create a **SharedProperty** object with the **CreatePropertyByPosition** method, you can access that property only by using the **get_PropertyByPosition** method. You can't assign a string name to the same property and then access it by using the **get_Property** method. Accessing a property by position is faster than accessing a property by using a string name because it requires less overhead.

The same shared property group can contain some **SharedProperty** objects that are identified by position and others that are identified by name.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreateProperty** Method, **ISharedPropertyGroup::get_PropertyByPosition** Method, **ISharedPropertyGroup::get_Property** Method

ISharedPropertyGroup::CreatePropertyByPosition Method Example

```
#include <mtx.h>
#include <mtxspm.h>


IObjectContext* pObjectContext = NULL;
ISharedPropertyGroupManager* pPropGpMgr = NULL;
ISharedPropertyGroup* pPropGp = NULL;
ISharedProperty* pPropNextNum = NULL;
VARIANT_BOOL fAlreadyExists = VARIANT_FALSE;
LONG lIsolationMode = LockMethod;
LONG lReleaseMode = Process;
BSTR stName;
VARIANT vNext;
LONG lNextValue = 0L;
HRESULT hr = S_OK;

hr = GetObjectContext(&pObjectContext);

// Create the SharedPropertyGroupManager,
// SharedPropertyGroup, and SharedProperty.
hr = pObjectContext->CreateInstance
   (CLSID_SharedPropertyGroupManager,
   IID_ISharedPropertyGroupManager,
   (void**)&pPropGpMgr);

stName = SysAllocString(L"Counter");
hr = pPropGpMgr->CreatePropertyGroup(stName,
   &lIsolationMode, &lReleaseMode, &fAlreadyExists,
   &pPropGp);
SysFreeString(stName);

hr = pPropGp->CreatePropertyByPosition
   (0, &fAlreadyExists, &pPropNextNum);

// Get the next number and increment the counter.
VariantInit(&vNext);
vNext.vt = VT_I4;
hr = pPropNextNum->get_Value(&vNext);
lNextValue = vNext.lVal++;
hr = pPropNextNum->put_Value(vNext);
```

ISharedPropertyGroup::get_Property Method

Returns a reference to an existing shared property identified by a string name.

Provided By

ISharedPropertyGroup Interface

HRESULT ISharedPropertyGroup::get_Property (
  BSTR *name*,
  ISharedProperty** *ppProperty*,
);

Parameters

name
  [in] The name of the shared property to retrieve.

ppProperty
  [out] A reference to the shared property specified in the *name* parameter, or NULL if the property doesn't exist.

Return Values

S_OK
  The shared property specified by *name* was found and a reference to it is returned in the *ppProperty* parameter.

E_INVALIDARG
  Either the argument passed in the *ppProperty* parameter was a null pointer, or there is no property in the shared property group with the name specified in the *name* parameter.

Remarks

You can use only the **get_Property** method to access properties that were created with the **CreateProperty** method. To access properties that were created with the **CreatePropertyByPosition** method, use the **get_PropertyByPosition** method.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreateProperty** Method, **ISharedPropertyGroup::CreatePropertyByPosition** Method, **ISharedPropertyGroup::get_PropertyByPosition** Method

ISharedPropertyGroupManager::get_Group, ISharedPropertyGroup::get_Property Methods Example

```
#include <mtx.h>
#include <mtxspm.h>


IObjectContext* pObjectContext = NULL;
ISharedPropertyGroupManager* pPropGpMgr = NULL;
ISharedPropertyGroup* pPropGp = NULL;
ISharedProperty* pPropNextNum = NULL;
BSTR stName, stNextNumber;
VARIANT vNext;
LONG lNextValue = 0L;
HRESULT hr = S_OK;

hr = GetObjectContext(&pObjectContext);

// Get the SharedPropertyGroupManager,
// SharedPropertyGroup, and SharedProperty.
hr = pObjectContext->CreateInstance
   (CLSID_SharedPropertyGroupManager,
   IID_ISharedPropertyGroupManager,
   (void**)&pPropGpMgr);

stName = SysAllocString(L"Counter");
hr = pPropGpMgr->get_Group(stName, &pPropGp);
SysFreeString(stName);

stNextNumber = SysAllocString(L"NextNum");
hr = pPropGp->get_Property
   (stNextNumber, &pPropNextNum);
SysFreeString(stNextNumber);

// Get the next number and increment the counter.
VariantInit(&vNext);
vNext.vt = VT_I4;
hr = pPropNextNum->get_Value(&vNext);
lNextValue = vNext.lVal++;
hr = pPropNextNum->put_Value(vNext);
```

ISharedPropertyGroup::get_PropertyByPosition Method

Returns a reference to an existing shared property identified by its numeric index within the property group.

Provided By

ISharedPropertyGroup Interface

HRESULT ISharedPropertyGroup::get_PropertyByPosition (
  INT *index*,
  ISharedProperty** *ppProperty*,
);

Parameters

index
  [in] The numeric index within the **SharedPropertyGroup** of the property to retrieve.

ppProperty
  [out] A reference to the shared property specified by the *index* parameter, or NULL if the property doesn't exist.

Return Values

S_OK
  The shared property specified by *index* was found and a reference to it is returned in the *ppProperty* parameter.

E_INVALIDARG
  Either the argument passed in the *ppProperty* parameter was a null pointer, or there is no property in the shared property group with the index number specified in the *index* parameter.

Remarks

You can use only the **get_PropertyByPosition** method to access properties that were created with the **CreatePropertyByPosition** method. To access properties that were created with the **CreateProperty** method, use the **get_Property** method.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreateProperty** Method, **ISharedPropertyGroup::CreatePropertyByPosition** Method, **ISharedPropertyGroup::get_Property** Method

ISharedPropertyGroup::get_PropertyByPosition Method Example

```
#include <mtx.h>
#include <mtxspm.h>


IObjectContext* pObjectContext = NULL;
ISharedPropertyGroupManager* pPropGpMgr = NULL;
ISharedPropertyGroup* pPropGp = NULL;
ISharedProperty* pPropNextNum = NULL;
BSTR stName;
VARIANT vNext;
LONG lNextValue = 0L;
HRESULT hr = S_OK;

hr = GetObjectContext(&pObjectContext);

// Get the SharedPropertyGroupManager,
// SharedPropertyGroup, and SharedProperty.
hr = pObjectContext->CreateInstance
   (CLSID_SharedPropertyGroupManager,
   IID_ISharedPropertyGroupManager,
   (void**)&pPropGpMgr);

stName = SysAllocString(L"Counter");
hr = pPropGpMgr->get_Group(stName, &pPropGp);
SysFreeString(stName);

hr = pPropGp->get_PropertyByPosition
   (0, &pPropNextNum);

// Get the next number and increment the counter.
VariantInit(&vNext);
vNext.vt = VT_I4;
hr = pPropNextNum->get_Value(&vNext);
lNextValue = vNext.lVal++;
hr = pPropNextNum->put_Value(vNext);
```

ISharedProperty Interface

The **ISharedProperty** interface is used to set or retrieve the value of a shared property. A shared property can contain any data type that can be represented by a variant.

Remarks

The header file for the **ISharedProperty** interface is mtxspm.h. You must also link mtxguid.lib to your project to use this interface.

You can access the **ISharedProperty** interface by creating a **SharedProperty** object with the **ISharedPropertyGroup::CreateProperty** method or the **ISharedPropertyGroup::CreatePropertyByPosition** method.

A **SharedProperty** object can be created or accessed only from within a **SharedPropertyGroup**.

As with any COM object, you must release a **SharedProperty** object when you're finished using it.

The **ISharedProperty** interface exposes the following methods.

Method	Description
get_Value	Retrieves the value of a shared property.
put_Value	Sets the value of a shared property.

See Also

Sharing State, MTS Supported Variant Types

ISharedProperty::get_Value Method

Retrieves the value of a shared property.

Provided By

ISharedProperty Interface

HRESULT ISharedProperty::get_Value (
VARIANT* *pVal*
);

Parameters

pVal
[out] A reference to a variant in which the value of the shared property will be returned.

Return Values

S_OK
A reference to a VARIANT containing the value of the shared property is returned in the *pVal* parameter.

E_INVALIDARG
The argument passed in the *pval* parameter is invalid.

Example

See Also

MTS Supported Variant Types, Sharing State

ISharedProperty::put_Value Method

Assigns a value to a shared property.

Provided By

ISharedProperty Interface

HRESULT ISharedProperty::put_Value (
  VARIANT *value*
);

Parameters

value
  [in] A VARIANT containing the value to assign to the **SharedProperty** object.

Return Values

S_OK
  The shared property's value has been set to *value*.

E_INVALIDARG
  The argument passed in the *value* parameter has the VT_BYREF bit set.

DISP_E_ARRAYISLOCKED
  The argument passed in the *value* parameter contains an array that's locked.

DISP_E_BADVARTYPE
  The argument passed in the *value* parameter isn't a valid VARIANT type.

Example

See Also

MTS Supported Variant Types, Sharing State

CreatePropertyGroup, CreateProperty, put_Value, get_Value Methods Example

```
#include <mtx.h>
#include <mtxspm.h>


IObjectContext* pObjectContext = NULL;
ISharedPropertyGroupManager* pPropGpMgr = NULL;
ISharedPropertyGroup* pPropGp = NULL;
ISharedProperty* pPropNextNum = NULL;
VARIANT_BOOL fAlreadyExists = VARIANT_FALSE;
LONG lIsolationMode = LockMethod;
LONG lReleaseMode = Process;
LONG lNextValue = 0L;
BSTR stName, stNextNumber;
VARIANT vNext;
HRESULT hr = S_OK;

hr = GetObjectContext(&pObjectContext);

// Create the SharedPropertyGroupManager,
// SharedPropertyGroup, and SharedProperty.
hr = pObjectContext->CreateInstance
   (CLSID_SharedPropertyGroupManager,
   IID_ISharedPropertyGroupManager,
   (void**)&pPropGpMgr);

stName = SysAllocString(L"Counter");
hr = pPropGpMgr->CreatePropertyGroup(stName,
   &lIsolationMode, &lReleaseMode, &fAlreadyExists,
   &pPropGp);
SysFreeString(stName);

stNextNumber = SysAllocString(L"NextNum");
hr = pPropGp->CreateProperty
   (stNextNumber, &fAlreadyExists, &pPropNextNum);
SysFreeString(stNextNumber);

// Get the next number and increment the counter.
VariantInit(&vNext);
vNext.vt = VT_I4;
hr = pPropNextNum->get_Value(&vNext);
lNextValue = vNext.lVal++;
hr = pPropNextNum->put_Value(vNext);
```

ISharedPropertyGroupManager Interface

The **ISharedPropertyGroupManager** interface is used to create shared property groups and to obtain access to existing shared property groups.

Remarks

The **ISharedPropertyGroupManager** interface is declared in the package `com.ms.mtx.`

You can access the **ISharedPropertyGroupManager** interface by creating an instance of the **SharedPropertyGroupManager** by using either **IObjectContext.CreateInstance** or **new SharedPropertyGroupManager**. It makes no difference which you use.

CreateInstance method of the **ObjectContext** object. It makes no difference which you use.

The Shared Property Manager is a resource dispenser that you can use to share state among multiple objects within a server process. You can't use global variables in a distributed environment because of concurrency and name collision issues. The Shared Property Manager eliminates name collisions by providing shared property groups, which establish unique name spaces for the shared properties they contain. The Shared Property Manager also implements locks and semaphores to protect shared properties from simultaneous access, which could result in lost updates and could leave the properties in an unpredictable state.

Shared properties can be shared only by objects running in the same process. If you want instances of different components to share properties, you have to install the components in the same MTS package. Because there is a risk that administrators will move components from one package to another, it's safest to limit the use of a shared property group to instances of components that are defined in the same DLL.

It's also important for components sharing properties to have the same activation attribute. If two components in the same package have different activation attributes, they generally won't be able to share properties. For example, if one component is configured to run in a client's process and the other is configured to run in a server process, their objects will usually run in different processes, even though they're in the same package.

You should always instantiate the **SharedPropertyGroupManager**, **SharedPropertyGroup**, and **SharedProperty** objects from MTS objects rather than from a base client. If a base client creates shared property groups and properties, the shared properties are inside the base client's process, not in a server process. This means MTS objects can't share the properties unless the objects, too, are running in the client's process (which is generally not a good idea).

Note When you set the isolation mode to LOCKMODE_METHOD, the Shared Property Manager requires access to the calling object's **ObjectContext**. You can't use this isolation mode to create a shared property group from within an object's constructor or from a non-MTS object because **ObjectContext** isn't available during object construction and a non-MTS object doesn't have an **ObjectContext**.

The **ISharedPropertyGroupManager** interface exposes the following methods and properties.

Method	Description
CreatePropertyGroup	Creates a new **SharedPropertyGroup** with a string name as an identifier. If a group with the specified name already exists, **CreatePropertyGroup** returns a reference to the existing group.
getGroup	Returns a reference to an existing shared property group, given a string name by which it can be identified.
get_NewEnum	Returns a reference to an enumerator that iterates through a list of all the shared property groups in a given process.

See Also

Sharing State

ISharedPropertyGroupManager.CreatePropertyGroup Method

Creates and returns a reference to a new shared property group. If a property group with the specified name already exists, **CreatePropertyGroup** returns a reference to the existing group.

Provided By

ISharedPropertyGroupManager Interface

ISharedPropertyGroup CreatePropertyGroup (
String *name*,
int[] *lockmode*,
int[] *releasemode*,
boolean[] e*xists,*
);

Parameters

name
[in] The name of the shared property group to create.

lockmode
[in, out] An array of one integer that specifies the isolation mode for the properties in the new shared property group. See the table that lists *lockmode* constants later in this topic. If the value of the *exists* parameter is set to `true` on return from this method, the *lockmode* value you passed in is ignored and the value returned in this parameter is the isolation mode that was assigned when the property group was created.

releasemode
[in, out] An array of one integer that specifies the release mode for the properties in the new shared property group. See the table that lists *releasemode* constants later in this topic. If the value of the *exists* parameter is set to `true` on return from this method, the *releasemode* value you passed in is ignored and the value returned in this parameter is the release mode that was assigned when the property group was created.

exists
[out] An array of one boolean that's set to `true` on return from this method if the shared property group specified in the *name* parameter existed prior to this call, and `false` if the property group was created by this call.

Settings

The following constants are used in the *lockmode* parameter to specify the effective isolation mode for a shared property group. These constants are static final members of the **ISharedPropertyGroupManager** interface.

Constant	Value	Description
LOCKMODE_SETGET	0	Default. Locks a property during a **getValue** or **putValue**call, assuring that every get or set operation on a shared property is atomic. This ensures that two clients can't read or write to the same property at the same time, but it doesn't prevent other clients from concurrently accessing other properties in the same group.
LOCKMODE_METHOD	1	Locks all of the properties in the shared property group for exclusive use by the caller as long as the caller's current method is executing. This is the appropriate mode to use when there

are interdependencies among properties, or in cases where a client may have to update a property immediately after reading it before it can be accessed again.

Note When you set the isolation mode to LOCKMODE_METHOD, the Shared Property Manager requires access to the calling object's **ObjectContext**. You can't use this isolation mode to create a shared property group from within an object's constructor or from a non-MTS object because **ObjectContext** isn't available during object construction and a base client doesn't have an **ObjectContext**.

The following constants are used in the *releasemode* parameter to specify the effective release mode for a shared property group. These constants are static final members of the **ISharedPropertyGroupManager** interface.

Constant	Value	Description
RELEASE MODE_ STANDARD	0	When all clients have released their references on the property group, the property group is automatically destroyed. (This is the default COM mode.)
RELEASE MODE_ PROCESS	1	The property group isn't destroyed until the process in which it was created has terminated.

Return Value

A reference to a shared property group identified by the string expression passed in the *name* parameter, or `null` if an error is encountered.

Remarks

The **CreatePropertyGroup** method sets the value in e*xists* to `true` if the property group it returns existed prior to the current call. This occurs when another object in the same process has already called **CreatePropertyGroup** with the same property group name. The **CreatePropertyGroup** method sets the value in e*xists* to `false` if the returned property group was created by the current call.

The isolation mode and release mode are assigned when the property group is originally created and aren't changed if a subsequent call passes different values in these parameters. The caller should always check the value of e*xists* on return from this method. If *exists* is set to `true`, the caller should check the values returned in *lockmode* and *releasemode* to determine the isolation and release modes in effect for the property group. For example:

```
propGp = propGpMgr.CreatePropertyGroup
   ("Counter", aiIsolationMode,
   aiReleaseMode, afAlreadyExists);
if (afAlreadyExists[0]) {
   if ((aiIsolationMode[0] !=
      ISharedPropertyGroupManager.LOCKMODE_METHOD) ||
      (aiReleaseMode[0] ISharedPropertyGroupManager.
      RELEASEMODE_PROCESS)) {
      // Do something appropriate.
   }
}
If*
```

Note An object should never attempt to pass a shared property group reference to another object. If the reference is passed outside of the object that acquired it, it's no longer a valid reference.

Example

See Also

Sharing State, **IObjectContext** Interface, **ISharedPropertyGroup** Interface

ISharedPropertyGroupManager.getGroup Method

Returns a reference to an existing shared property group.

Provided By

ISharedPropertyGroupManager Interface

ISharedPropertyGroup getGroup (
    String *name*,
);

Parameters

name
    [in] The name of the shared property group to retrieve.

Return Value

A reference to the shared property group specified in the *name* parameter, or `null` if the property group doesn't exist.

Example

See Also

Sharing State, **ISharedPropertyGroupManager** Interface

ISharedPropertyGroupManager.get_NewEnum Method

Returns a reference to an enumerator that you can use to iterate through all the shared property groups in a process.

Provided By

ISharedPropertyGroupManager Interface

IUnknown get_NewEnum ( );

Return Value

A reference to the **IUnknown** interface on a new enumerator object that you can use to iterate through the list of all the shared property groups in the process.

Remarks

You use the **get_NewEnum** method to obtain a reference to an enumerator object. You should immediately cast the returned value to the **IEnumVariant** interface. This interface exposes several methods you can use to iterate through a list of string expressions representing shared property group names. Once you have a name, you can use the **getGroup** method to obtain a reference to the shared property group it represents. When you enumerate the shared property groups, all groups will be included. However, if you then call **CreatePropertyGroup** to add a new group, the existing enumerator won't include the new group even if you call **Reset** or **Clone**. To include the new group, you must create a new enumerator by calling **NewEnum** again.

Example

See Also

ISharedPropertyGroupManager.getGroup Method, **ISharedPropertyGroup Interface**

get_NewEnum Method Example

```
import com.ms.mtx.*;

ISharedPropertyGroupManager spgm = null;
IEnumVariant myEnum = null;
Variant v;
int i;

// Get the enumerator object and
// cast the returned interface to IEnumVariant.
myEnum = (IEnumVariant)spgm.get_NewEnum();

// Use the enumerator to iterate through
// the property group names.
for(i = 0; i < 10; i++){
   v = Enum.Next(1);
   // Do something with the returned
   // property group names.
}
```

ISharedPropertyGroup Interface

The **ISharedPropertyGroup** interface is used to create and access the shared properties in a shared property group.

Remarks

The **ISharedPropertyGroup** interface is declared in the package `com.ms.mtx`.

You can access the **ISharedPropertyGroup** interface by creating a **SharedPropertyGroup** object with the **ISharedPropertyGroupManager.CreatePropertyGroup** method.

The **ISharedPropertyGroup** interface exposes the following methods.

CreateProperty	Creates a new shared property identified by a string expression that's unique within its property group.
CreatePropertyByPosition	Creates a new shared property identified by a numeric index within its property group.
getProperty	Returns a reference to a shared property, given the string name by which the property is identified.
getPropertyByPosition	Returns a reference to a shared property, given its numeric index in the shared property group.

See Also

Sharing State, **ISharedPropertyGroupManager** Interface

ISharedPropertyGroup.CreateProperty Method

Creates and returns a reference to a new **SharedProperty** with a specified name. If a shared property by that name already exists, **CreateProperty** returns a reference to the existing property.

Provided By

ISharedPropertyGroup Interface

ISharedProperty CreateProperty (
  String *name*,
  boolean[] *exists;*
);

Parameters

name
  [in] The name of the property to create. You can use this name later to obtain a reference to this property by using the **getProperty** method.

exists
  [out] An array of one Boolean value that's set to `true` on return from this method if the shared property specified in the *name* parameter existed prior to this call, and `false` if the property was created by this call.

Return Value

A reference to a shared property object with the name specified in the *name* parameter, or `null` if an error is encountered.

Remarks

When you create a shared property, its value is set to the default, which is a Variant with an integer value of 0.

If you create a shared property with the **CreateProperty** method, you can access that property only by using the **getProperty** method. You can't assign a numeric index to the same property and then access it by using the **getPropertyByPosition** method.

The same shared property group can contain some shared property objects that are identified by name and others that are identified by position.

Example

See Also

Sharing State, I**SharedPropertyGroup::CreatePropertyByPosition** Method, **ISharedPropertyGroup::get_PropertyByPosition** Method, **ISharedPropertyGroup::get_Property** Method

ISharedPropertyGroup.CreatePropertyByPosition Method

Creates a new shared property identified by a numeric index that's unique within the property group. If a shared property with the specified index already exists, **CreatePropertyByPosition** returns a reference to the existing one.

Provided By

ISharedPropertyGroup Interface

ISharedProperty CreatePropertyByPosition (
  int *index*,
  boolean[] *exists;*
);

Parameters

index
  [in] The numeric index within the **SharedPropertyGroup** by which the new property will be referenced. You can use this index later to retrieve the shared property with the **getPropertyByPosition** method.

exists
  [out] An array of one boolean value. If *exists* is set to `true` on return from this method, the shared property specified by *index* existed prior to this call. If it's set to `false`, the property was created by this call.

Return Value

A reference to a shared property object that's identified by the numeric index passed in the *index* parameter, or `null` if an error is encountered.

Remarks

When you create a shared property, its value is set to the default, which is a Variant with an integer value of 0.

If you create a **SharedProperty** object with the **CreatePropertyByPosition** method, you can access that property only by using the **getPropertyByPosition** method. You can't assign a string name to the same property and then access it by using the **getProperty** method. Accessing a property by position is faster than accessing a property by using a string name because it requires less overhead.

The same shared property group can contain some **SharedProperty** objects that are identified by position and others that are identified by name.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreateProperty** Method, **ISharedPropertyGroup::get_PropertyByPosition** Method, **ISharedPropertyGroup::get_Property** Method

ISharedPropertyGroup.CreatePropertyByPosition Method Example

```
import com.ms.mtx.*;


ISharedPropertyGroupManager propGpMgr = null;
ISharedPropertyGroup propGp = null;
ISharedProperty propNextNum = null;
Variant vNext;
int iNextValue = 0;
boolean[] afAlreadyExists = new boolean[1];
int[] aiIsolationMode = new int[1];
aiIsolationMode[0] =
   ISharedPropertyGroupManager.LOCKMODE_SETGET;
int[] aiReleaseMode = new int[1];
aiReleaseMode[0] =
   ISharedPropertyGroupManager.RELEASEMODE_PROCESS;

// Create the SharedPropertyGroupManager,
// SharedPropertyGroup, and SharedProperty.
propGpMgr = new SharedPropertyGroupManager();
propGp = propGpMgr.CreatePropertyGroup
   ("Counter", aiIsolationMode,
   aiReleaseMode, afAlreadyExists);
propNextNum = propGp.CreatePropertyByPosition
   (0, fAlreadyExists);

// Get the next number and increment it.
VariantInit(&vNext);
vNext = propNextNum.getValue();
iNextValue = vNext.getInt();
vNext.putInt(iNextValue + 1);
propNextNum.putValue(vNext);
```

ISharedPropertyGroup.getProperty Method

Returns a reference to an existing shared property identified by a string name.

Provided By

ISharedPropertyGroup Interface

ISharedProperty getProperty (
    String *name*,
);

Parameters

name
    [in] A string expression that contains the name of the shared property to retrieve.

Return Value

A reference to the shared property specified in the *name* parameter, or `null` if the property doesn't exist.

Remarks

You can use only the **getProperty** method to access properties that were created with the **CreateProperty** method. To access properties that were created with the **CreatePropertyByPosition** method, use the **getPropertyByPosition** method.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreateProperty** Method, **ISharedPropertyGroup::CreatePropertyByPosition** Method, **ISharedPropertyGroup::get_PropertyByPosition** Method

ISharedPropertyGroupManager.getGroup, ISharedPropertyGroup.getProperty Methods Example

```
import com.ms.mtx.*;


ISharedPropertyGroupManager propGpMgr = null;
ISharedPropertyGroup propGp = null;
ISharedProperty propNextNum = null;
Variant vNext;
int iNextValue = 0;

// Get the SharedPropertyGroupManager,
// SharedPropertyGroup, and SharedProperty.
propGpMgr = new SharedPropertyGroupManager();
propGp = propGpMgr.getGroup("Counter");
propNextNum = propGp.getProperty("NextNum");

// Get the next number and increment it.
VariantInit(&vNext);
vNext = propNextNum.getValue();
iNextValue = vNext.getInt();
vNext.putInt(iNextValue + 1);
propNextNum.putValue(vNext);
```

ISharedPropertyGroup.getPropertyByPosition Method

Returns a reference to an existing shared property identified by its numeric index within the property group.

Provided By

ISharedPropertyGroup Interface

ISharedProperty getPropertyByPosition (
    INT *index*,
);

Parameters

index
    [in] The numeric index within the **SharedPropertyGroup** of the property to retrieve.

Return Value

A reference to the shared property specified by the *index* parameter, or `null` if the property doesn't exist.

Remarks

You can use only the **getPropertyByPosition** method to access properties that were created with the **CreatePropertyByPosition** method. To access properties that were created with the **CreateProperty** method, use the **getProperty** method.

Example

See Also

Sharing State, **ISharedPropertyGroup::CreateProperty** Method, **ISharedPropertyGroup::CreatePropertyByPosition** Method, **ISharedPropertyGroup::get_Property** Method

ISharedPropertyGroup.getPropertyByPosition Method Example

```
import com.ms.mtx.*;


ISharedPropertyGroupManager propGpMgr = null;
ISharedPropertyGroup propGp = null;
ISharedProperty propNextNum = null;
Variant vNext;
int iNextValue = 0;

// Get the SharedPropertyGroupManager,
// SharedPropertyGroup, and SharedProperty.
propGpMgr = new SharedPropertyGroupManager();
propGp = propGpMgr.getGroup("Counter");
propNextNum = propGp.getPropertyByPosition(0);

// Get the next number and increment it.
VariantInit(&vNext);
vNext = propNextNum.getValue();
iNextValue = vNext.getInt();
vNext.putInt(iNextValue + 1);
propNextNum.putValue(vNext);
```

ISharedProperty Interface

The **ISharedProperty** interface is used to set or retrieve the value of a shared property. A shared property can contain any data type that can be represented by a variant.

Remarks

The **ISharedProperty** interface is declared in the package `com.ms.mtx`.

You can access the **ISharedProperty** interface by creating a **SharedProperty** object with the **ISharedPropertyGroup.CreateProperty** method or the **ISharedPropertyGroup.CreatePropertyByPosition** method.

A **SharedProperty** object can be created or accessed only from within a **SharedPropertyGroup**.

The **ISharedProperty** interface exposes the following methods.

Method	Description
getValue	Retrieves the value of a shared property.
putValue	Sets the value of a shared property.

See Also

Sharing State, MTS Supported Variant Types

ISharedProperty.getValue Method

Retrieves the value of a shared property.

Provided By

ISharedProperty Interface

Variant getValue ( );

Return Value

A Variant in which the value of the shared property will be returned.

Example

See Also

MTS Supported Variant Types, Sharing State

ISharedProperty.putValue Method

Assigns a value to a shared property.

Provided By

ISharedProperty Interface

```
void putValue (
    Variant value
);
```

Parameters

value
[in] A Variant containing the value to assign to the **SharedProperty** object.

Example

See Also

MTS Supported Variant Types, Sharing State

CreatePropertyGroup, CreateProperty, putValue, getValue Methods Example

```
import com.ms.mtx.*;


ISharedPropertyGroupManager propGpMgr = null;
ISharedPropertyGroup propGp = null;
ISharedProperty propNextNum = null;
Variant vNext;
int iNextValue = 0;
boolean[] afAlreadyExists = new boolean[1];
int[] aiIsolationMode = new int[1];
aiIsolationMode[0] =
   ISharedPropertyGroupManager.LOCKMODE_SETGET;
int[] aiReleaseMode = new int[1];
aiReleaseMode[0] =
   ISharedPropertyGroupManager.RELEASEMODE_PROCESS;

// Create the SharedPropertyGroupManager,
// SharedPropertyGroup, and SharedProperty.
propGpMgr = new SharedPropertyGroupManager();
propGp = propGpMgr.CreatePropertyGroup
   ("Counter", aiIsolationMode,
   aiReleaseMode, afAlreadyExists);
propNextNum = propGp.CreateProperty
   ("NextNum", afAlreadyExists);

// Get the next number and increment it.
VariantInit(&vNext);
vNext = propNextNum.getValue();
iNextValue = vNext.getInt();
vNext.putInt(iNextValue + 1);
propNextNum.putValue(vNext);
```

TransactionContext Object

The **TransactionContext** object is used by a base client to compose the work of one or more MTS objects into an atomic transaction and to commit or abort the transaction.

Remarks

To use the **TransactionContext** object, you must set a reference to the Transaction Context Type Library (txctx.dll).

You can use a **TransactionContext** object to scope a transaction from a base client. You begin the transaction by instantiating a **TransactionContext** object, and you end the transaction by calling **Commit** or **Abort** on the object. The base client itself never executes within the transaction.

The **TransactionContext** component is a standard MTS component. The component's transaction attribute is set to **Requires a new transaction**, which means that a **TransactionContext** object is always the root of a transaction. When a base client instantiates an object by using the **TransactionContext** object's **CreateInstance** method, the new object and its descendants will participate in the**TransactionContext** object's transaction unless the new object's transaction attribute is set to **Requires a new transaction** or **Does not support transactions**.

You could easily write your own **TransactionContext** component. You would simply create a component that implements the methods **Commit**, **Abort**, and **CreateInstance**, and set the component's transaction attribute to **Requires a new transaction**. The three methods would do nothing more than call **GetObjectContext** and invoke their **ObjectContext** object's **SetComplete**, **SetAbort**, and **CreateInstance** methods, respectively.

Before you use **TransactionContext** to compose the work of existing components in a transaction, you should consider implementing a separate component that not only composes their work but encapsulates it into a reusable unit. This new component would not only serve the needs of the current base client, but other clients could also use it. In one approach, the base client instantiates a **TransactionContext** object, calls its **CreateInstance** method to instantiate other objects, calls various methods on those objects, and finally calls **Commit** or **Abort** on the **TransactionContext** object. In the other approach, you create a new component that requires a transaction. This new component instantiates the other objects using its **ObjectContext** object's **CreateInstance** method, calls the relevant methods on those other objects itself, and then calls **SetComplete** or **SetAbort** on its **ObjectContext** when it's done. Using this approach, the base client only needs to instantiate this one object, and invoke one method on it, and the object does the rest of the work. When other clients require the same functionality, they can reuse the new component.

You obtain a reference to a **TransactionContext** object with **CreateObject**. For example:

```
Set objTransactionContext = _
   CreateObject("TxCtx.TransactionContext")
```

The **TransactionContext** object provides the following methods.

Method	Description
Abort	Aborts the work of all MTS objects participating in the current transaction. The transaction is completed on return from this method.
Commit	Attempts to commit the work of all MTS objects participating in the current transaction. If any of the MTS objects participating in the transaction have called **SetAbort** or **DisableCommit**, or if a system error has occurred, the transaction will be aborted. Otherwise, the transaction will be committed. In either case, the transaction is completed on return from this method.
CreateInstance	Instantiates another MTS object. If the component that provides the object is configured to support or require a transaction, then the

new object runs under the transaction of the **TransactionContext**object.

See Also

Transaction Context Objects, Base Clients, Transactions

Abort Method

Aborts the current transaction.

Applies To

TransactionContext Object

Syntax

transactioncontextobject.**Abort**

The *transactioncontextobject* placeholder represents an object variable that evaluates to a **TransactionContext** object.

Remarks

When a base client calls **Abort**, all objects that participated in the transaction are automatically deactivated. Any database updates made by those objects are rolled back. The transaction is completed on return from this method. If another call is made on the **TransactionContext** object after the **TransactionContext** object has returned from a call in which it called the **Abort** method, a new transaction is started.

Example

See Also

Transaction Context Objects, Base Clients, Transactions, **SetAbort**

Abort, Commit Methods Example

```
Dim objTxCtx As TransactionContext
Dim objMyObject As MyCompany.MyObject
Dim userCanceled As Boolean

' Get TransactionContext.
Set objTxCtx = _
   CreateObject("TxCtx.TransactionContext")

' Create an instance of some component.
Set objMyObject= _
   objTxCtx.CreateInstance("MyCompany.MyObject")

' Do some work here.

' If something goes wrong, abort the transaction.
If userCanceled Then
   objTxCtx.Abort

' Otherwise, commit it.
Else
   objTxCtx.Commit
End If
```

Commit Method

Attempts to commit the current transaction.

Applies To

TransactionContext Object

Syntax

transactioncontextobject.**Commit**

The *transactioncontextobject* placeholder represents an object variable that evaluates to a **TransactionContext** object.

Remarks

Calling **Commit** doesn't guarantee that a transaction will be committed. If any MTS object that was part of the transaction has returned from a method after calling **SetAbort**, the transaction will be aborted. If any object that was part of the transaction has called **DisableCommit** and hasn't yet called **EnableCommit** or **SetComplete**, the transaction will also be aborted. Any error that causes Microsoft Distributed Transaction Coordinator to abort a transaction will also abort an MTS transaction.

When a base client calls **Commit**, regardless of whether the transaction commits or aborts, the transaction is completed on return from this method and all objects that participated in the transaction are automatically deactivated. If another call comes in after the **TransactionContext** object has returned from a call in which it called the **Commit** method, a new transaction is started.

Example

See Also

Transaction Context Objects, Base Clients, Transactions, **SetComplete**

CreateInstance Method

Instantiates an MTS object that will execute within the scope of the transaction that was initiated with the creation of the **TransactionContext** object.

Applies To

TransactionContext Object

Syntax

Set *object* = *transactioncontextobject*.**CreateInstance**(*programmaticID*)

Part

object
An object variable that evaluates to an MTS object.

transactioncontextobject
An object variable that represents the **TransactionContext** object from which to create the new object.

programmaticID
The programmatic Id of the new object's component.

Remarks

When a base client uses the **TransactionContext** object's **CreateInstance** method to instantiate an MTS object, the new object executes within the transaction context object's activity. If the transaction attribute of the new object's component is set to either **Supports transactions** or **Requires a transaction**, the new object also inherits the transaction initiated with the creation of the **TransactionContext** object. However, if the component that provides the new object has its transaction attribute set to **Does not support transactions**, the object neither inherits the transaction nor passes it on to objects it subsequently creates. If the component that provides the new object has its transaction attribute set to **Requires a new transaction**, the MTS run-time environment initiates a new transaction for the new object, and that transaction is the one that's inherited by objects it subsequently creates.

In this respect, using **CreateInstance** is comparable to using **CoCreateInstance** and specifying NULL for the controlling **IUnknown** interface (*pUnkOuter*).

Example

See Also

Transaction Context Objects, Base Clients, Transactions, **CreateInstance**

CreateInstance Method, TransactionContext Object Example

```
Dim objTxCtx As TransactionContext
Dim objMyObject As MyCompany.MyObject

' Get TransactionContext.
Set objTxCtx = _
   CreateObject("TxCtx.TransactionContext")

' Create an instance of MyObject.
Set objMyObject= _
   objTxCtx.CreateInstance("MyCompany.MyObject")
```

ITransactionContextEx Interface

The **ITransactionContextEx** interface is used by a base client to compose the work of one or more MTS objects into an atomic transaction and to commit or abort the transaction.

Remarks

The header file for the **ITransactionContextEx** interface is txctx.h. You must also link mtxguid.lib to your project to use this interface.

You can use a **TransactionContextEx** object to scope a transaction from a base client. You begin the transaction by instantiating a **TransactionContextEx** object, and you end the transaction by calling **Commit** or **Abort** on the object. The base client itself never executes within the transaction.

The **TransactionContextEx** component is a standard MTS component. The component's transaction attribute is set to **Requires a new transaction**, which means that a **TransactionContextEx** object is always the root of a transaction. When a base client instantiates an object by using the **ITransactionContextEx::CreateInstance** method, the new object and its descendants will participate in the**TransactionContextEx** object's transaction unless the new object's transaction attribute is set to **Requires a new transaction** or **Does not support transactions**.

You could easily write your own **TransactionContextEx** component. You would simply create a component that implements the methods **Commit**, **Abort**, and **CreateInstance**, and set the component's transaction attribute to **Requires a new transaction**. The three methods would do nothing more than call **GetObjectContext** and invoke their **ObjectContext** object's **SetComplete**, **SetAbort**, and **CreateInstance** methods, respectively.

Before you use **TransactionContextEx** to compose the work of existing components in a transaction, you should consider implementing a separate component that not only composes their work but encapsulates it into a reusable unit. This new component would not only serve the needs of the current base client, but other clients could also use it. In one approach, the base client instantiates a **TransactionContextEx** object, calls its **CreateInstance** method to instantiate other objects, calls various methods on those objects, and finally calls **Commit** or **Abort** on the **TransactionContextEx** object. In the other approach, you create a new component that requires a transaction. This new component instantiates the other objects using its **ObjectContext** object's **CreateInstance** method, calls the relevant methods on those other objects itself, and then calls **SetComplete** or **SetAbort** on its **ObjectContext** when it's done. Using this approach, the base client only needs to instantiate this one object, and invoke one method on it, and the object does the rest of the work. When other clients require the same functionality, they can reuse the new component.

You obtain a reference to the **ITransactionContextEx** interface by creating a **TransactionContextEx** object with a call to **CoCreateInstance**. For example:

```
CoCreateInstance(CLSID_TransactionContextEx, NULL, CLSCTX_INPROC,
IID_ITransactionContextEx, (void**)&m_pTransactionContext);
```

The **ITransactionContextEx** interface exposes the following methods.

Method	Description
Abort	Aborts the work of all MTS objects participating in the current transaction. The transaction is completed on return from this method.
Commit	Attempts to commit the work of all MTS objects participating in the current transaction. If any of the MTS objects participating in the transaction have called **SetAbort** or **DisableCommit**, or if a system error has occurred, the transaction will be aborted. Otherwise, the transaction will be committed. In either case, the transaction is completed on return from this method.
CreateInstance	Instantiates another MTS object. If the component that provides the

object is configured to support or require a transaction, then the new object runs under the transaction of the **TransactionContextEx**object.

See Also

Transaction Context Objects, Base Clients, Transactions

ITransactionContextEx::Abort Method

Aborts the current transaction.

Provided By

ITransactionContextEx Interface

HRESULT ITransactionContextEx::Abort ( );

Return Values

S_OK
The transaction was aborted.

E_FAIL
The **TransactionContextEx** object isn't running under a MTS process. This could happen if the **TransactionContextEx** component's Registry entry has been corrupted.

E_UNEXPECTED
An unexpected error occurred.

Remarks

When a base client calls **Abort**, all objects that participated in the transaction are automatically deactivated. Any database updates made by those objects are rolled back. The transaction is completed on return from this method. If another call is made on the **TransactionContextEx** object after the **TransactionContextEx** object has returned from a call in which it called the **Abort** method, a new transaction is started.

Example

See Also

Transaction Context Objects, Base Clients, Transactions, **SetAbort**

ITransactionContextEx::Abort, ITransactionContextEx::Commit Methods Example

```
#include <Txctx.h>

ITransactionContextEx* pTransactionContext = NULL;
IMyObject* pMyObject = NULL;
boolean bUserCanceled = FALSE;
HRESULT hr;

// Get TransactionContextEx.
hr = CoCreateInstance(CLSID_ITransactionContextEx,
   NULL, CLSCTX_INPROC, IID_ITransactionContextEx,
   (void**) &pTransactionContext);

// Create an instance of MyObject.
hr = pTransactionContext->CreateInstance
   (CLSID_CMyObject, IID_IMyObject,
   (void**) &pMyObject);

// Do some work here.

// If something goes wrong, abort the transaction.
if (bUserCanceled)
   pTransactionContext->Abort();

// Otherwise, commit it.
else
   pTransactionContext->Commit();
```

ITransactionContextEx::Commit Method

Attempts to commit the current transaction.

Provided By

ITransactionContextEx Interface

HRESULT ITransactionContextEx::Commit ( );

Return Values

S_OK
The transaction was committed.

E_FAIL
The **TransactionContextEx** object isn't running under a MTS process. This could happen if the **TransactionContextEx** component's Registry entry has been corrupted.

E_UNEXPECTED
An unexpected error occurred.

CONTEXT_E_ABORTED
The transaction was aborted.

Remarks

Calling **Commit** doesn't guarantee that a transaction will be committed. If any MTS object that was part of the transaction has returned from a method after calling **SetAbort**, the transaction will be aborted. If any object that was part of the transaction has called **DisableCommit** and hasn't yet called **EnableCommit** or **SetComplete**, the transaction will also be aborted. Any error that causes Microsoft Distributed Transaction Coordinator to abort a transaction will also abort an MTS transaction.

When a base client calls **Commit**, regardless of whether the transaction commits or aborts, the transaction is completed on return from this method and all objects that participated in the transaction are automatically deactivated. If another call comes in after the **TransactionContextEx** object has returned from a call in which it called the **Commit** method, a new transaction is started.

Example

See Also

Transaction Context Objects, Base Clients, Transactions, **SetComplete**

ITransactionContextEx::CreateInstance Method

Instantiates an MTS object that will execute within the scope of the transaction that was initiated with the creation of the **TransactionContextEx** object.

Provided By

ITransactionContextEx Interface

HRESULT ITransactionContextEx::CreateInstance (
   REFCLSID *rclsid*,
   REFIID *riid*,
   LPVOID FAR* *ppvObj*
);

Parameter

rclsid
   [in] A reference to the CLSID of the type of object to instantiate.

riid
   [in] A reference to the interface ID of the interface through which you want to communicate with the new object.

ppvObj
   [out] A reference to a new object of the type specified by the *rclsid* argument, through the interface specified by the *riid* argument.

Return Values

S_OK
   A reference to the object is returned in the *ppvObj* parameter.

REGDB_E_CLASSNOTREG
   The component specified by *rclsid* is not registered as a COM component.

E_OUTOFMEMORY
   There's not enough memory available to instantiate the object.

E_INVALIDARG
   The argument passed in the *ppvObj* parameter is invalid.

E_UNEXPECTED
   An unexpected error occurred.

Remarks

When a base client uses the **ITransactionContextEx::CreateInstance** method to instantiate an MTS object, the new object executes within the transaction context object's activity. If the transaction attribute of the new object's component is set to either **Supports transactions** or **Requires a transaction**, the new object also inherits the transaction initiated with the creation of the **TransactionContextEx** object. However, if the component that provides the new object has its transaction attribute set to **Does not support transactions**, the object neither inherits the transaction nor passes it on to objects it subsequently creates. If the component that provides the new object has its transaction attribute set to **Requires a new transaction**, the MTS run-time environment initiates a new transaction for the new object, and that transaction is the one that's inherited by objects it subsequently creates.

If the Microsoft Distributed Transaction Coordinator is not running and the object is transactional, the object is successfully created. However, method calls to that object will fail with CONTEXT_E_TMNOTAVAILABLE. Objects cannot recover from this condition and should be released.

MTS always uses standard marshaling. Even if a component exposes the **IMarshal** interface, its **IMarshal** methods will never be called by the MTS run-time environment.

Note You can't create MTS objects as part of an aggregation. In this respect, using **CreateInstance** is comparable to using **CoCreateInstance** and specifying NULL for the controlling **IUnknown** interface (*pUnkOuter*).

Example

See Also

Transaction Context Objects, Base Clients, Transactions, **CreateInstance**

ITransactionContextEx::CreateInstance Method Example

```
#include <Txctx.h>

ITransactionContextEx* pTransactionContext = NULL;
IMyObject* pMyObject = NULL;
HRESULT hr;

// Get TransactionContextEx.
hr = CoCreateInstance(CLSID_ITransactionContextEx,
   NULL, CLSCTX_INPROC, IID_ITransactionContextEx,
   (void**) &pTransactionContext);

// Create an instance of MyObject.
hr = pTransactionContext->CreateInstance
   (CLSID_CMyObject, IID_IMyObject,
   (void**) &pMyObject);
```

ITransactionContextEx Interface

The **ITransactionContextEx** interface is used by a base client to compose the work of one or more MTS objects into an atomic transaction and to commit or abort the transaction.

Remarks

The **ITransactionContextEx** interface is declared in the package `com.ms.mtx`.

You can use a **TransactionContextEx** object to scope a transaction from a base client. You begin the transaction by instantiating a **TransactionContextEx** object, and you end the transaction by calling **Commit** or **Abort** on the object. The base client itself never executes within the transaction.

The **TransactionContextEx** component is a standard MTS component. The component's transaction attribute is set to **Requires a new transaction**, which means that a **TransactionContextEx** object is always the root of a transaction. When a base client instantiates an object by using the **ITransactionContextEx.CreateInstance** method, the new object and its descendants will participate in the**TransactionContextEx** object's transaction unless the new object's transaction attribute is set to **Requires a new transaction** or **Does not support transactions**.

You could easily write your own **TransactionContextEx** component. You would simply create a component that implements the methods **Commit**, **Abort**, and **CreateInstance**, and set the component's transaction attribute to **Requires a new transaction**. The three methods would do nothing more than call **GetObjectContext** and invoke their **ObjectContext** object's **SetComplete**, **SetAbort**, and **CreateInstance** methods, respectively.

Before you use **TransactionContextEx** to compose the work of existing components in a transaction, you should consider implementing a separate component that not only composes their work but encapsulates it into a reusable unit. This new component would not only serve the needs of the current base client, but other clients could also use it. In one approach, the base client instantiates a **TransactionContextEx** object, calls its **CreateInstance** method to instantiate other objects, calls various methods on those objects, and finally calls **Commit** or **Abort** on the **TransactionContextEx** object. In the other approach, you create a new component that requires a transaction. This new component instantiates the other objects using its **ObjectContext** object's **CreateInstance** method, calls the relevant methods on those other objects itself, and then calls **SetComplete** or **SetAbort** on its **ObjectContext** when it's done. Using this approach, the base client only needs to instantiate this one object, and invoke one method on it, and the object does the rest of the work. When other clients require the same functionality, they can reuse the new component.

You obtain a reference to the **ITransactionContextEx** interface by creating a **TransactionContextEx** object. For example:

```
new TransactionContextEx();
```

The **ITransactionContextEx** interface exposes the following methods.

Method	Description
Abort	Aborts the work of all MTS objects participating in the current transaction. The transaction is completed on return from this method.
Commit	Attempts to commit the work of all MTS objects participating in the current transaction. If any of the MTS objects participating in the transaction have called **SetAbort** or **DisableCommit**, or if a system error has occurred, the transaction will be aborted. Otherwise, the transaction will be committed. In either case, the transaction is completed on return from this method.
CreateInstance	Instantiates another MTS object. If the component that provides the object is configured to support or require a transaction, then the new object runs under the transaction of the

TransactionContextExobject.

See Also

Transaction Context Objects, Base Clients, Transactions

ITransactionContextEx.Abort Method

Aborts the current transaction.

Provided By

ITransactionContextEx Interface

void Abort ( );

Remarks

When a base client calls **Abort**, all objects that participated in the transaction are automatically deactivated. Any database updates made by those objects are rolled back. The transaction is completed on return from this method. If another call is made on the **TransactionContextEx** object after the **TransactionContextEx** object has returned from a call in which it called the **Abort** method, a new transaction is started.

Example

See Also

Transaction Context Objects, Base Clients, Transactions, **SetAbort**

ITransactionContext.Abort, ITransactionContext.Commit Methods Example

```
import com.ms.mtx.*;

ITransactionContextEx myTransactionContext = null;
IMyObject myObject = null;
boolean userCanceled = false;

// Get TransactionContextEx.
myTransactionContext = new TransactionContextEx();

// Create an instance of MyObject.
myObject = myTransactionContext.CreateInstance (CMyObject.clsid,
IMyObject.iid);

// Do some work here.

// If something goes wrong, abort the transaction.
if (userCanceled)
   myTransactionContext.Abort();

// Otherwise, commit it.
else
   pTransactionContext.Commit();
```

ITransactionContextEx.Commit Method

Attempts to commit the current transaction.

Provided By

ITransactionContextEx Interface

void Commit ( );

Remarks

Calling **Commit** doesn't guarantee that a transaction will be committed. If any MTS object that was part of the transaction has returned from a method after calling **SetAbort**, the transaction will be aborted. If any object that was part of the transaction has called **DisableCommit** and hasn't yet called **EnableCommit** or **SetComplete**, the transaction will also be aborted. Any error that causes Microsoft Distributed Transaction Coordinator to abort a transaction will also abort an MTS transaction.

When a base client calls **Commit**, regardless of whether the transaction commits or aborts, the transaction is completed on return from this method and all objects that participated in the transaction are automatically deactivated. If another call comes in after the **TransactionContextEx** object has returned from a call in which it called the **Commit** method, a new transaction is started.

Example

See Also

Transaction Context Objects, Base Clients, Transactions, **SetComplete**

ITransactionContextEx.CreateInstance Method

Instantiates an MTS object that will execute within the scope of the transaction that was initiated with the creation of the **TransactionContextEx** object.

Provided By

ITransactionContextEx Interface

```
IUnknown CreateInstance (
   _Guid clsid,
   _Guid  iid,
);
```

Parameter

clsid
[in] A reference to the CLSID of the type of object to instantiate.

iid
[in] Any interface that's implemented by the object you want to instantiate.

Return Value

A reference to the **IUnknown** interface on a new instance of the MTS component specified in the *clsid* parameter.

Remarks

When a base client uses the **ITransactionContextEx.CreateInstance** method to instantiate an MTS object, the new object executes within the transaction context object's activity. If the transaction attribute of the new object's component is set to either **Supports transactions** or **Requires a transaction**, the new object also inherits the transaction initiated with the creation of the **TransactionContextEx** object. However, if the component that provides the new object has its transaction attribute set to **Does not support transactions**, the object neither inherits the transaction nor passes it on to objects it subsequently creates. If the component that provides the new object has its transaction attribute set to **Requires a new transaction**, the MTS run-time environment initiates a new transaction for the new object, and that transaction is the one that's inherited by objects it subsequently creates.

CreateInstance always returns the **IUnknown** interface on the newly instantiated object. You should immediately cast the returned value to the interface with which you want to communicate with the new object. The interface ID you pass in the *iid* parameter doesn't have to be the same interface to which you cast the returned value, but it must be an interface that's implemented by the object you want to instantiate.

MTS always uses standard marshaling. Even if a component exposes the **IMarshal** interface, its **IMarshal** methods will never be called by the MTS run-time environment.

Note You can't create MTS objects as part of an aggregation.

Example

See Also

Transaction Context Objects, Base Clients, Transactions, **CreateInstance**

ITransactionContext.CreateInstance Method Example

```
import com.ms.mtx.*;

ITransactionContextEx myTransactionContext = null;
IMyObject myObject = null;

// Get TransactionContextEx.
myTransactionContext = new TransactionContextEx();

// Create an instance of MyObject.
myObject = (IMyObject)
   myTransactionContext.CreateInstance
   (CMyObject.clsid, IMyObject.iid);
```

SecurityProperty Object

The **SecurityProperty** object is used to determine the current object's caller or creator.

Remarks

To use the **SecurityProperty** object, you must set a reference to Microsoft Transaction Server Type Library (mtxas.dll).

You obtain a reference to an object's **SecurityProperty** object by calling **Security** on the object's **ObjectContext**. For example:

```
Set secObject = ctxObject.Security
```

The **SecurityProperty** object provides the following methods.

Method	Description
GetDirectCallerName	Retrieves the user name associated with the external process that called the currently executing method.
GetDirectCreatorName	Retrieves the user name associated with the external process that directly created the current object.
GetOriginalCallerName	Retrieves the user name associated with the base process that initiated the call sequence from which the current method was called.
GetOriginalCreatorName	Retrieves the user name associated with the base process that initiated the activity in which the current object is executing.

See Also

Programmatic Security, Advanced Security Methods, **ObjectContext** Object

GetDirectCallerName Method

Retrieves the user name associated with the external process that called the currently executing method.

Applies To

SecurityProperty Object

Syntax

username = *securityproperty*.**GetDirectCallerName( )**

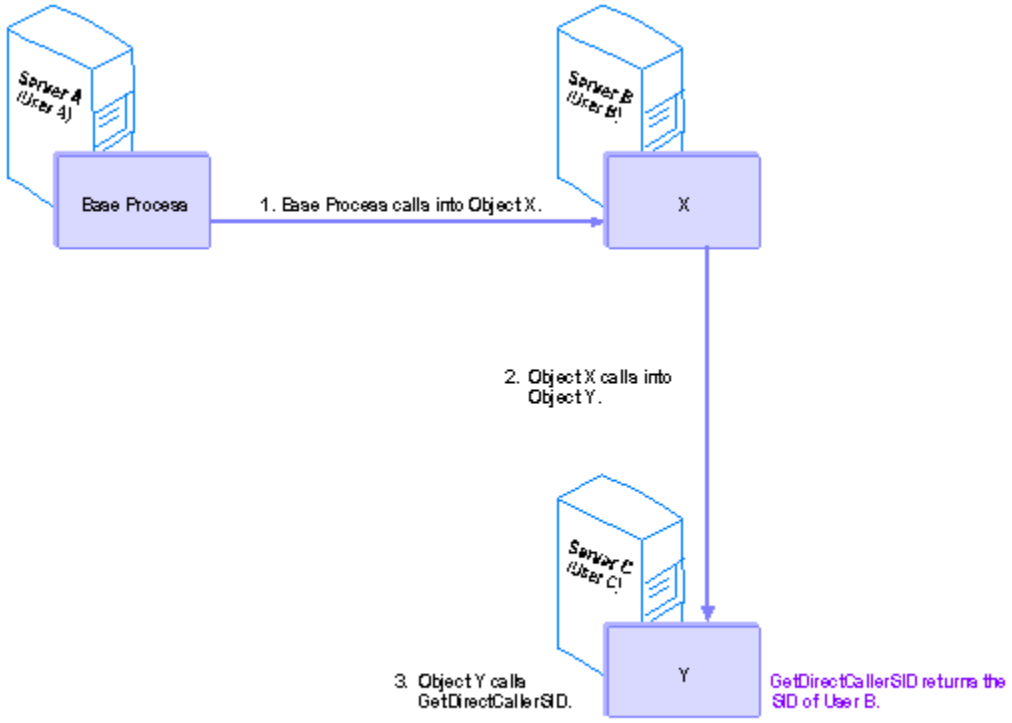
Part

username
The user name associated with the process from which the current method was invoked.

securityproperty
An object variable that evaluates to a **SecurityProperty** object.

Remarks

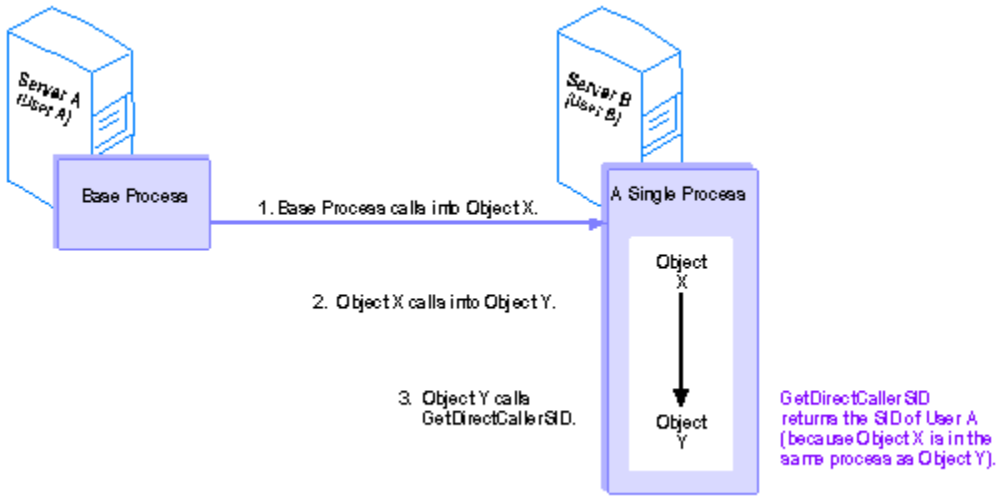
You use the **GetDirectCallerName** method to determine the user name associated with the process that called the object's currently executing method. The following scenarios illustrate the functionality of the **GetDirectCallerName** method.



A base process running on server A, as user A, calls into object X on server B, running as user B. Then object X calls into object Y, running on server C. If object Y calls **GetDirectCallerName**, the name of user B is returned.

Security can only be enforced across process boundaries. This means that the name returned by

GetDirectCallerName is the name associated with the process that called into the process in which the current object is running, not necessarily the immediate caller into the object itself. If an object calls into another object within the same process, when the second object calls **GetDirectCallerName**, it will get the name of the most immediate caller outside its own process boundary, not the name of the object that directly called into it.



A base process, running on server A as user A, calls into object X on server B, running as user B. Then object X calls into object Y, running in the same process as object X, also on server B. When object Y calls **GetDirectCallerName**, the name of user A is returned , not the name of user B.

Example

See Also

Programmatic Security, Advanced Security Methods, **ObjectContext** Object

GetDirectCallerName Method Example

```
Public Function ComponentDirectCaller() As String

    Dim objCtx As ObjectContext

    Set objCtx = GetObjectContext()
    ComponentDirectCaller = _
        objCtx.Security.GetDirectCallerName()

End Function
```

GetDirectCreatorName Method

Retrieves the user name associated with the current object's immediate (out-of-process) creator.

Applies To

SecurityProperty Object

Syntax

username = *securityproperty*.**GetDirectCreatorName( )**

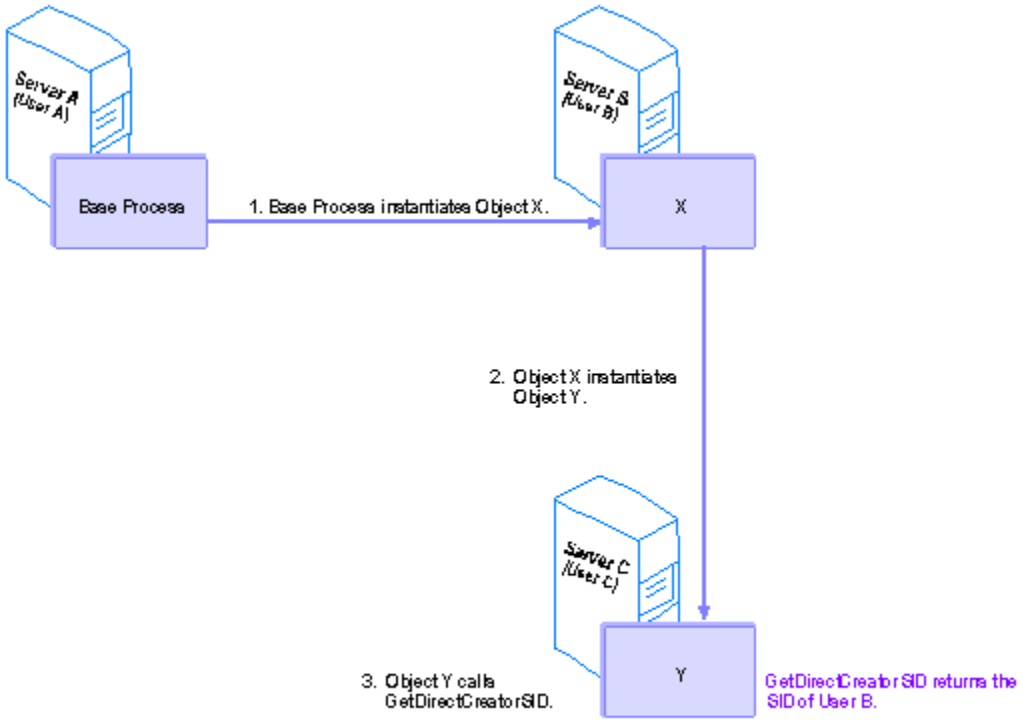
Part

username
The user name associated with the process that directly created the current object.

securityproperty
An object variable that evaluates to a **SecurityProperty** object.

Remarks

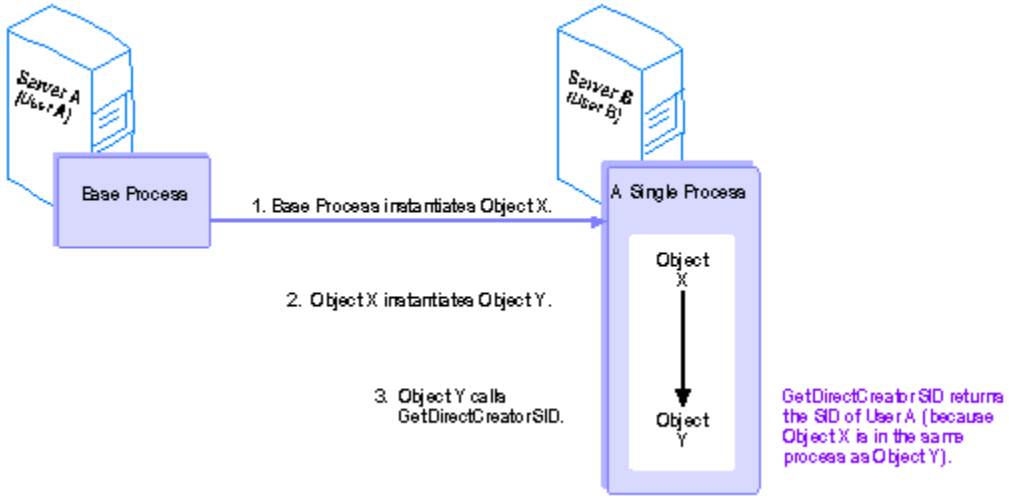
You use the **GetDirectCreatorName** method to determine the user name associated with the process that created the current object. The following scenarios illustrate the functionality of the **GetDirectCreatorName** method.



A base process running on server A, as user A, creates object X on server B, running as user B. Then object X creates object Y, running on server C. If object Y calls **GetDirectCreatorName**, the name of user B is returned.

Security can only be enforced across process boundaries. This means that if an object creates another object within the same process, when the second object calls **GetDirectCreatorName**, it will

get the name of the most immediate creator outside its own process boundary, not the user name associated with the object that actually created it.



A base client running on server A, as user A, creates object X on server B, running as user B. Then object X creates object Y, running in the same process as object X, also on server B. When object Y calls **GetDirectCreatorName**, the name of user A is returned, not the name of user B.

Example

See Also

Programmatic Security, Advanced Security Methods, **ObjectContext** Object

GetDirectCreatorName Method Example

```
Public Function ComponentDirectCreator() As String

    Dim objCtx As ObjectContext

    Set objCtx = GetObjectContext()
    ComponentDirectCreator = _
        objCtx.Security.GetDirectCreatorName()

End Function
```

GetOriginalCallerName Method

Retrieves the user name associated with the base process that initiated the sequence of calls from which the call into the current object originated.

Applies To

SecurityProperty Object

Syntax

username = *securityproperty*.**GetOriginalCallerName( )**

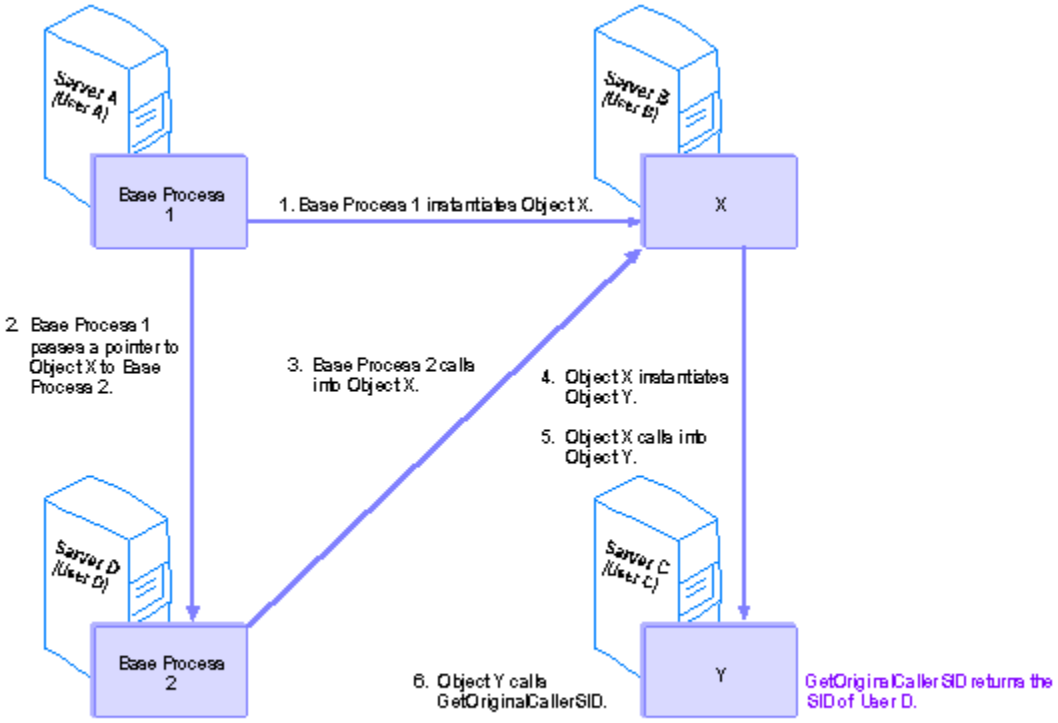
Part

username
The user name associated with the base process that initiated the call sequence from which the current method was called.

securityproperty
An object variable that evaluates to a **SecurityProperty** object.

Remarks

You use the **GetOriginalCallerName** method to determine the user name associated with the original process that initiated the call sequence from which the current method was called. The following scenario illustrates the functionality of the **GetOriginalCallerName** method.



Base process 1, running on server A as user A, creates object X on server B, running as user B. Then base process 1 passes its reference on object X to base process 2, running on server D as user D. Base process 2 uses that reference to call into object X. object X then calls into object Y, running on server C. If object Y then calls **GetOriginalCallerName**, the name of user D is returned.

Note Usually, an object's original caller is the same process as its original creator. The only situation in which the original caller and the original creator would be different is one in which the original creator passes a reference to another process, and the other process initiates the call sequence (as in the preceding example).

Note The path to the original caller is broken if any object along the chain was created by some other means than **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**. For example, if base process 1 uses **CoCreateInstance** to create X, when Y calls **GetOriginalCallerName,** the name it gets back will be the name of user B, not user D. This is because the call sequence is traced back through the objects' context and MTS can only create a context for an object that's created with either **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**.

Example

See Also

Programmatic Security, Advanced Security Methods, **ObjectContext** Object

GetOriginalCallerName Method Example

```
Public Function ComponentOriginalCaller() As String

    Dim objCtx As ObjectContext

    Set objCtx = GetObjectContext()
    ComponentOriginalCaller = _
        objCtx.Security.GetOriginalCallerName()

End Function
```

GetOriginalCreatorName Method

Retrieves the user name associated with the original base process that initiated the activity in which the current object is executing.

Applies To

SecurityProperty Object

Syntax

username = *securityproperty*.**GetOriginalCreatorName( )**

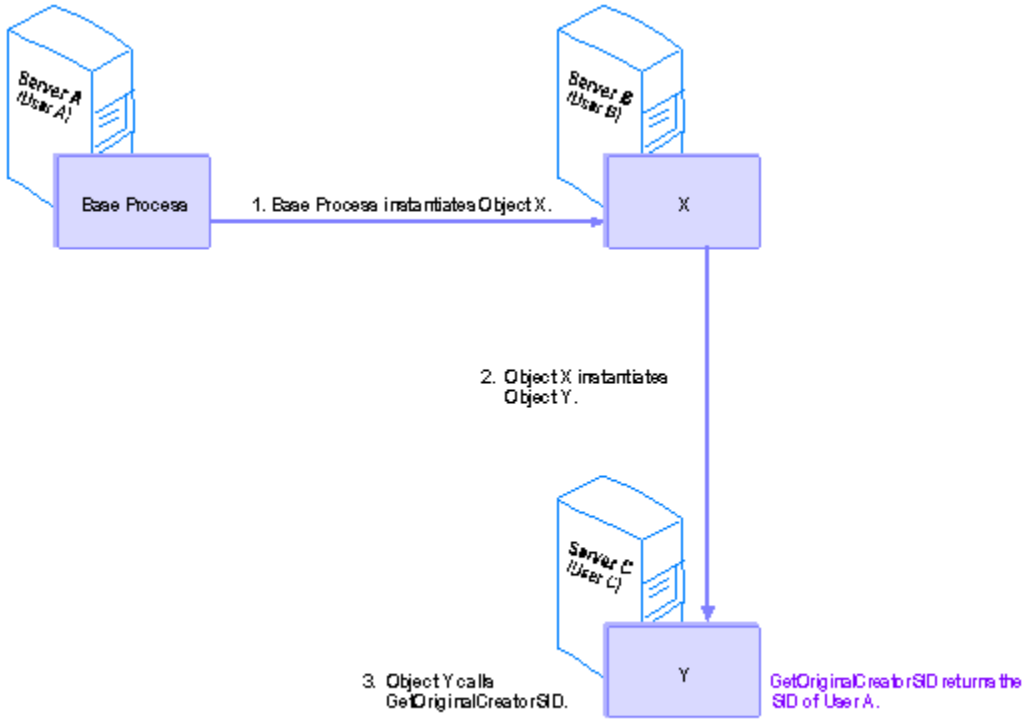
Part

username
The user name associated with the base process that initiated the activity in which the current object is executing.

securityproperty
An object variable that evaluates to a **SecurityProperty** object.

Remarks

You use the **GetOriginalCreatorName** method to determine the user name associated with the process that initiated the activity in which the current object is executing. The following scenario illustrates the functionality of the **GetOriginalCreatorName** method.



A base process running on server A, as user A, creates object X on server B, running as user B. Then object X creates object Y, running on server C. If object Y calls **GetOriginalCreatorName**, the name of user A is returned.

Note The path to the original creator is broken if an object is created by some other means than **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**. For example, if the base process on server A uses **CoCreateInstance** to create X, when Y calls **GetOriginalCreatorName,** the name it gets back will be the name of user B, not user A. This is because the creation sequence is traced back through the objects' context and MTS can only create a context for an object that's created with either **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**.

Example

See Also

Programmatic Security, Advanced Security Methods, **ObjectContext** Object

GetOriginalCreatorName Method Example

```
Public Function ComponentOriginalCreator() As String

    Dim objCtx As ObjectContext

    Set objCtx = GetObjectContext()
    ComponentOriginalCreator = _
        objCtx.Security.GetOriginalCreatorName()

End Function
```

ISecurityProperty Interface

The **ISecurityProperty** interface is used to determine the security ID of the current object's caller or creator.

Remarks

The header file for the **ISecurityProperty** interface is mtx.h. You must also link mtxguid.lib to your project to use this interface.

You obtain a reference to an object's **ISecurityProperty** interface by calling **QueryInterface** on the object's **ObjectContext**. For example:

```
m_pIObjectContext->QueryInterface (IID_ISecurityProperty,
(void**)&m_pISecurityProperty);
```

The **ISecurityProperty** interface provides the following methods.

Method	Description
GetDirectCallerSID	Retrieves the security ID of the external process that called the currently executing method.
GetDirectCreatorSID	Retrieves the security ID of the external process that directly created the current object.
GetOriginalCallerSID	Retrieves the security ID of the base process that initiated the call sequence from which the current method was called.
GetOriginalCreatorSID	Retrieves the security ID of the base process that initiated the activity in which the current object is executing.
ReleaseSID	Releases the security ID returned by one of the other **ISecurityProperty** methods.

See Also

Programmatic Security, Advanced Security Methods, **IObjectContext** Interface

ISecurityProperty::GetDirectCallerSID Method

Retrieves the security ID of the external process that called the currently executing method.

Provided By

ISecurityProperty Interface

HRESULT ISecurityProperty::GetDirectCallerSID (
  PSID* *ppSid*
);

Parameters

ppSid
  [out] A reference to the security ID of the process from which the current method was invoked.

Return Values

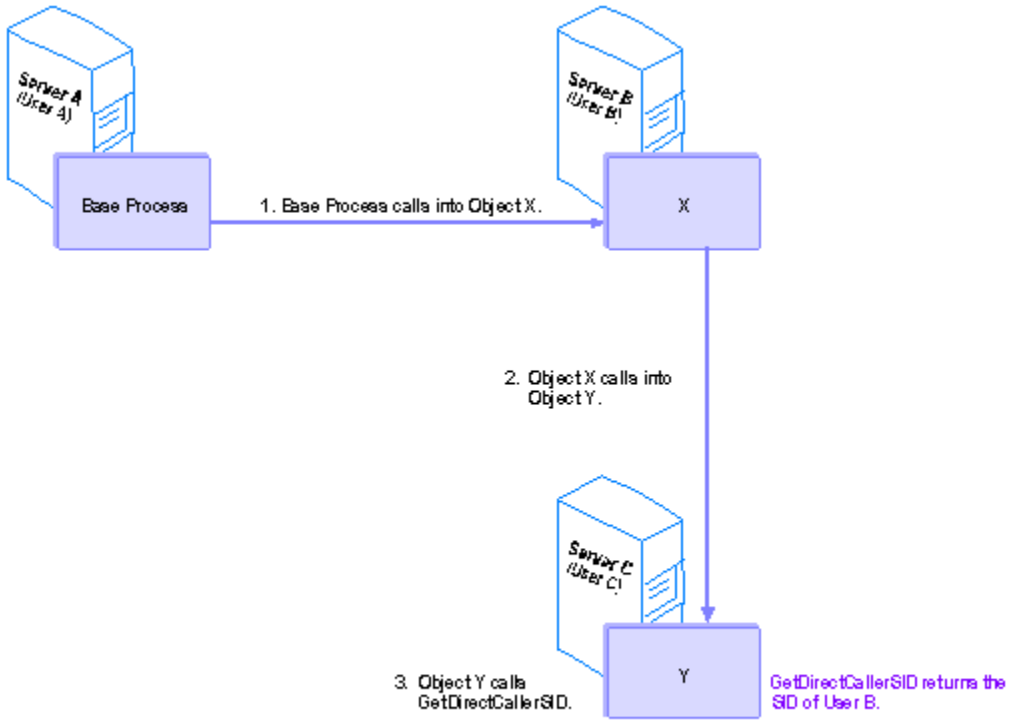
S_OK
  The security ID of the process that called the current method is returned in the parameter *ppSid*.

E_INVALIDARG
  The argument passed in the *ppSid* parameter is a NULL pointer.

E_UNEXPECTED
  An unexpected error occurred.

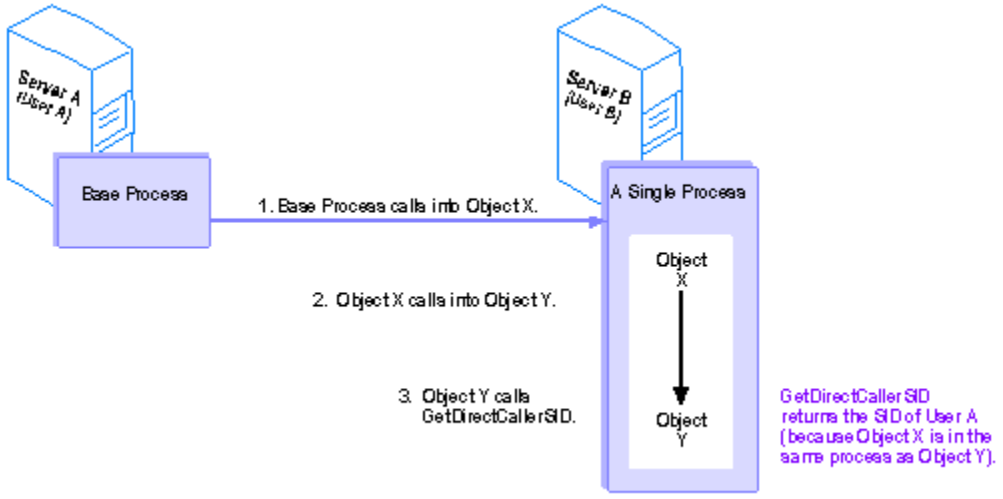
Remarks

You use the **GetDirectCallerSID** method to determine the security ID of the process that called the object's currently executing method. The following scenarios illustrate the functionality of the **GetDirectCallerSID** method.



A base process running on server A, as user A, calls into object X on server B, running as user B. Then object X calls into object Y, running on server C. If object Y calls **GetDirectCallerSID**, the the security ID of user B is returned.

Security can only be enforced across process boundaries. This means that the the security ID returned by **GetDirectCallerSID** is the the security ID associated with the process that called into the process in which the current object is running, not necessarily the immediate caller into the object itself. If an object calls into another object within the same process, when the second object calls **GetDirectCallerSID**, it will get the the security ID of the most immediate caller outside its own process boundary, not the the security ID of the object that directly called into it.



A base process, running on server A as user A, calls into object X on server B, running as user B. Then object X calls into object Y, running in the same process as object X, also on server B. When object Y calls **GetDirectCallerSID**, the the security ID of user A is returned , not the the security ID of

user B.

You must call **ReleaseSID** on a security ID when you finish using it.

Example

See Also

Programmatic Security, Advanced Security Methods, **IObjectContext** Interface

GetDirectCallerSID Method Example

```
#include <mtx.h>


IObjectContext* pIObjectContext = NULL;
ISecurityProperty* pISecurityProperty = NULL;
PSID pSid = NULL;
HRESULT hr;

// Get a reference to the ISecurityProperty interface.
pIObjectContext->QueryInterface(IID_ISecurityProperty,
   (void**)&pISecurityProperty);

// Obtain the caller's security ID.
hr = pISecurityProperty->GetDirectCallerSID(&pSid)

// Do some security checking here.

// Release the security ID.
pISecurityProperty->ReleaseSID(pSid);
```

ISecurityProperty::GetDirectCreatorSID Method

Retrieves the security ID of the current object's immediate (out-of-process) creator.

Provided By

ISecurityProperty Interface

HRESULT ISecurityProperty::GetDirectCreatorSID (
  PSID* *ppSid*
);

Parameters

ppSid
  [out] A reference to the security ID of the process that directly created the current object.

Return Values

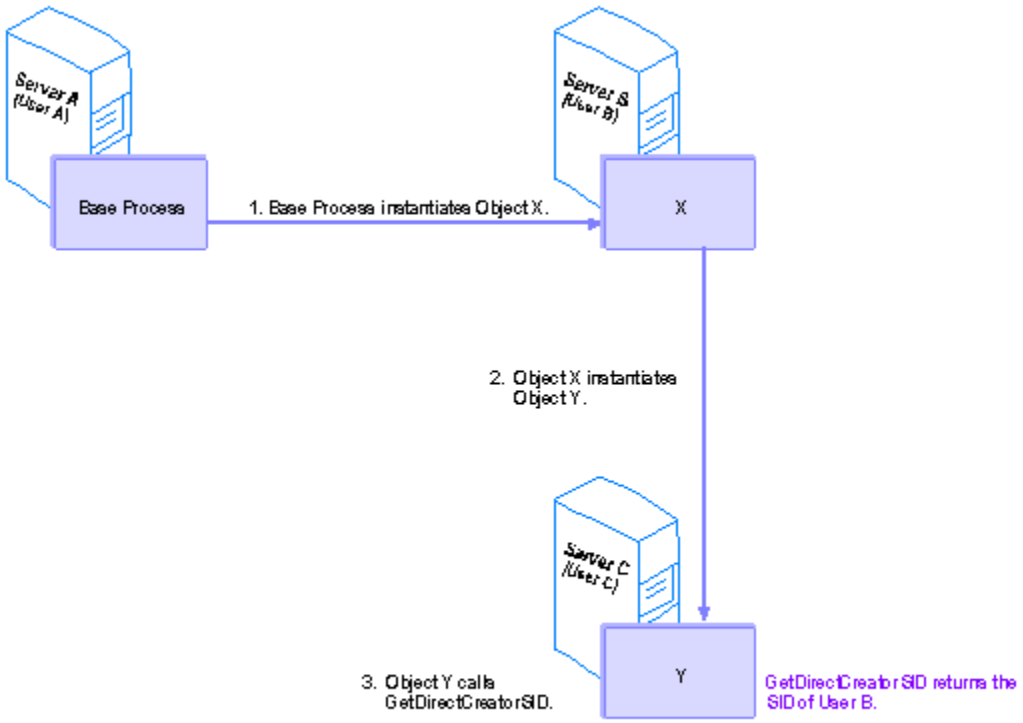
S_OK
  The security ID of the process that directly created the current object is returned in the parameter *ppSid.*

E_INVALIDARG
  The argument passed in the *ppSid* parameter is a NULL pointer.

E_FAIL
  An unexpected error occurred.

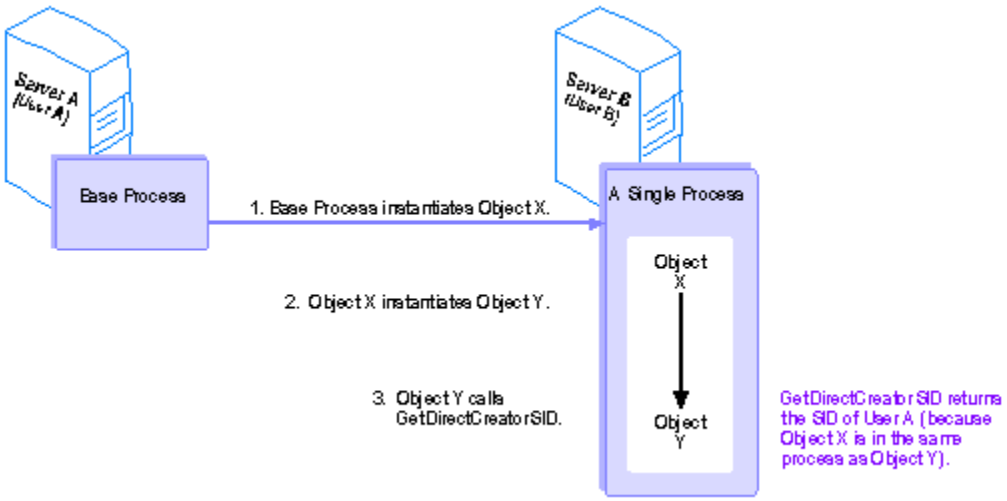
Remarks

You use the **GetDirectCreatorSID** method to determine the security ID of the process that created the current object. The following scenarios illustrate the functionality of the **GetDirectCreatorSID** method.



A base process running on server A, as user A, creates object X on server B, running as user B. Then object X creates object Y, running on server C. If object Y calls **GetDirectCreatorSID**, the the security ID of user B is returned.

Security can only be enforced across process boundaries. This means that if an object creates another object within the same process, when the second object calls **GetDirectCreatorSID**, it will get the the security ID of the most immediate creator outside its own process boundary, not the security ID of the object that actually created it.



A base client running on server A, as user A, creates object X on server B, running as user B. Then object X creates object Y, running in the same process as object X, also on server B. When object Y calls **GetDirectCreatorSID**, the the security ID of user A is returned, not the the security ID of user B.

You must call **ReleaseSID** on a security ID when you finish using it.

Example

See Also

Programmatic Security, Advanced Security Methods, **IObjectContext** Interface

GetDirectCreatorSID Method Example

```
#include <mtx.h>


IObjectContext* pIObjectContext = NULL;
ISecurityProperty* pISecurityProperty = NULL;
PSID pSid = NULL;
HRESULT hr;

// Get a reference to the ISecurityProperty interface.
pIObjectContext->QueryInterface(IID_ISecurityProperty,
   (void**)&pISecurityProperty);

// Obtain the creator's security ID.
hr = pISecurityProperty->GetDirectCreatorSID(&pSid)

// Do some security checking here.

// Release the security ID.
pISecurityProperty->ReleaseSID(pSid);
```

ISecurityProperty::GetOriginalCallerSID Method

Retrieves the security ID of the base process that initiated the sequence of calls from which the call into the current object originated.

Provided By

ISecurityProperty Interface

HRESULT ISecurityProperty::GetOriginalCallerSID (
  PSID* *ppSid*
);

Parameters

ppSid
  [out] A reference to the security ID of the base process that initiated the call sequence from which the current method was called.

Return Values

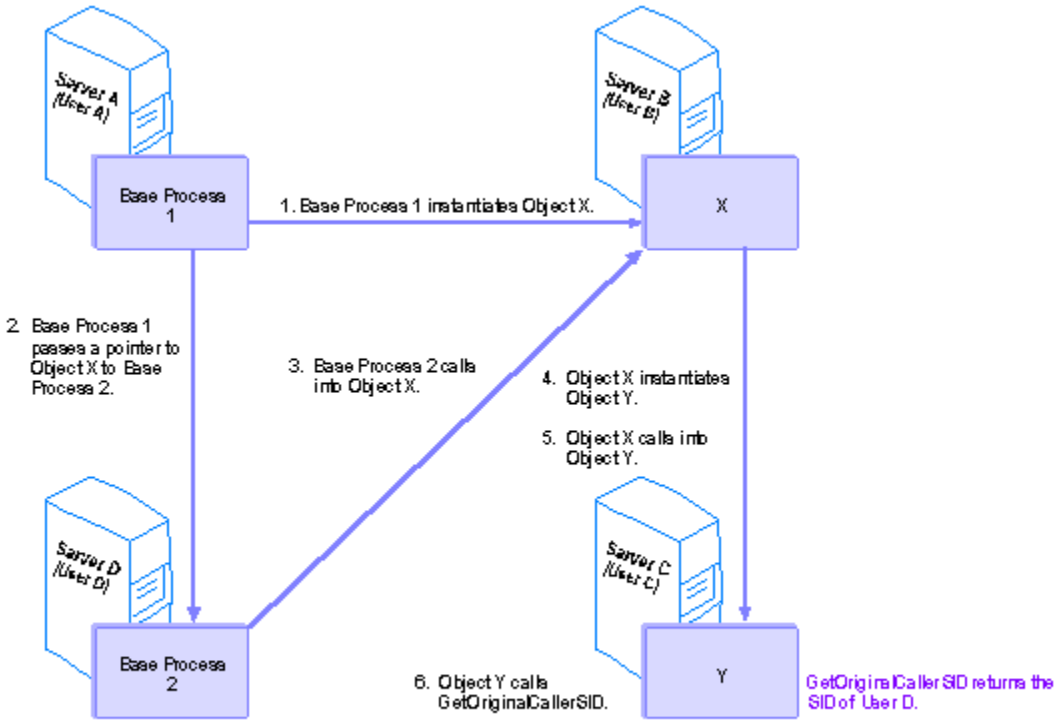
S_OK
  The security ID of the base process that originated the call into the current object is returned in the parameter *ppSid*.

E_INVALIDARG
  The argument passed in the *ppSid* parameter is a NULL pointer.

E_FAIL
  An unexpected error occurred.

Remarks

You use the **GetOriginalCallerSID** method to determine the security ID of the original process that initiated the call sequence from which the current method was called. The following scenario illustrates the functionality of the **GetOriginalCallerSID** method.



Base process 1, running on server A as user A, creates object X on server B, running as user B. Then base process 1 passes its reference on object X to base process 2, running on server D as user D. Base process 2 uses that reference to call into object X. object X then calls into object Y, running on server C. If object Y then calls **GetOriginalCallerSID**, the the security ID of user D is returned.

Note Usually, an object's original caller is the same process as its original creator. The only situation in which the original caller and the original creator would be different is one in which the original creator passes a reference to another process, and the other process initiates the call sequence (as in the preceding example).

Note The path to the original caller is broken if any object along the chain was created by some other means than **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**. For example, if base process 1 uses **CoCreateInstance** to create X, when Y calls **GetOriginalCallerSID,** the the security ID it gets back will be the the security ID of user B, not user D. This is because the call sequence is traced back through the objects' context and MTS can only create a context for an object that's created with either **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**.

You must call **ReleaseSID** on a security ID when you finish using it.

Example

See Also

Programmatic Security, Advanced Security Methods, **IObjectContext** Interface

GetOriginalCallerSID Method Example

```
#include <mtx.h>


IObjectContext* pIObjectContext = NULL;
ISecurityProperty* pISecurityProperty = NULL;
PSID pSid = NULL;
HRESULT hr;

// Get a reference to the ISecurityProperty interface.
pIObjectContext->QueryInterface(IID_ISecurityProperty,
   (void**)&pISecurityProperty);

// Obtain the original caller's security ID.
hr = pISecurityProp->GetOriginalCallerSID(&pSid)

// Do some security checking here.

// Release the security ID.
pISecurityProperty->ReleaseSID(pSid);
```

ISecurityProperty::GetOriginalCreatorSID Method

Retrieves the security ID of the original base process that initiated the activity in which the current object is executing.

Provided By

ISecurityProperty Interface

HRESULT ISecurityProperty::GetOriginalCreatorSID (
  PSID* *ppSid*
);

Parameters

ppSid
  [out] A reference to the security ID of the base process that initiated the activity in which the current object is executing.

Return Values

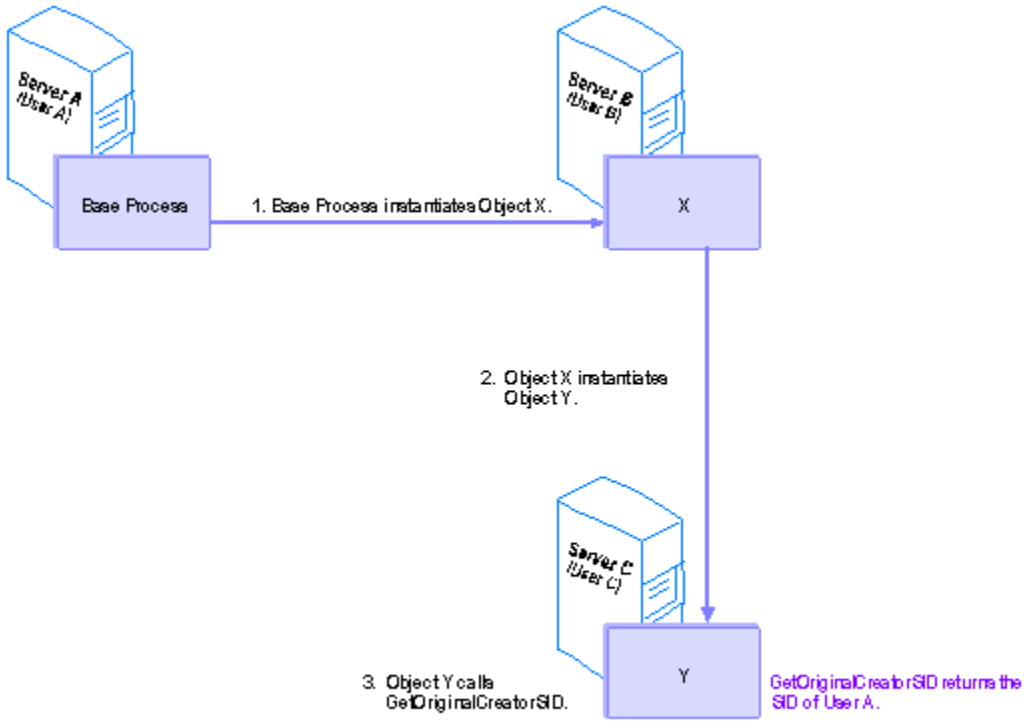
S_OK
  The security ID of the original creator is returned in the parameter *ppSid*.

E_INVALIDARG
  The argument passed in the *ppSid* parameter is a NULL pointer.

E_FAIL
  An unexpected error occurred.

Remarks

You use the **GetOriginalCreatorSID** method to determine the security ID of the process that initiated the activity in which the current object is executing. The following scenario illustrates the functionality of the **GetOriginalCreatorSID** method.



A base process running on server A, as user A, creates object X on server B, running as user B. Then object X creates object Y, running on server C. If object Y calls **GetOriginalCreatorSID**, the the security ID of user A is returned.

Note The path to the original creator is broken if an object is created by some other means than **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**. For example, if the base process on server A uses **CoCreateInstance** to create X, when Y calls **GetOriginalCreatorSID,** the the security ID it gets back will be the the security ID of user B, not user A. This is because the creation sequence is traced back through the objects' context and MTS can only create a context for an object that's created with either **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**.

You must call **ReleaseSID** on a security ID when you finish using it.

Example

See Also

Programmatic Security, Advanced Security Methods, **IObjectContext** Interface

GetOriginalCreatorSID, ReleaseSID Methods Example

```
#include <mtx.h>


IObjectContext* pIObjectContext = NULL;
ISecurityProperty* pISecurityProperty = NULL;
PSID pSid = NULL;
HRESULT hr;

// Get a reference to the ISecurityProperty interface.
pIObjectContext->QueryInterface(IID_ISecurityProperty,
   (void**)&pISecurityProperty);

// Obtain the original creator's security ID.
hr = pISecurityProp->GetOriginalCreatorSID(&pSid)

// Do some security checking here.

// Release the security ID.
pISecurityProperty->ReleaseSID(pSid);
```

ISecurityProperty::ReleaseSID Method

Releases a security ID that was obtained from the **GetDirectCallerSID**, **GetDirectCreatorSID**, **GetOriginalCallerSID**, or **GetOriginalCreatorSID** method.

Provided By

ISecurityProperty Interface

HRESULT ISecurityProperty::ReleaseSID (
  PSID *pSid*
);

Parameters

pSid
  [in] A reference to a security ID that was obtained by invoking one of the **ISecurityProperty** methods.

Return Values

S_OK
  The security ID, passed in the *pSid* parameter, was released.

E_INVALIDARG
  The argument passed in the *pSid* parameter is not a reference to a security ID.

Remarks

You should always invoke the **ReleaseSID** method to release any security ID pointers returned by the **GetDirectCallerSID, GetDirectCreatorSID, GetOriginalCallerSID,** and **GetOriginalCreatorSID** methods of the **ISecurityProperty** interface.

See Also

Programmatic Security, Advanced Security Methods, **IObjectContext** Interface

getDirectCallerName Method

Retrieves the user name associated with the external process that called the currently executing method.

String getDirectCallerName ( );

Provided By

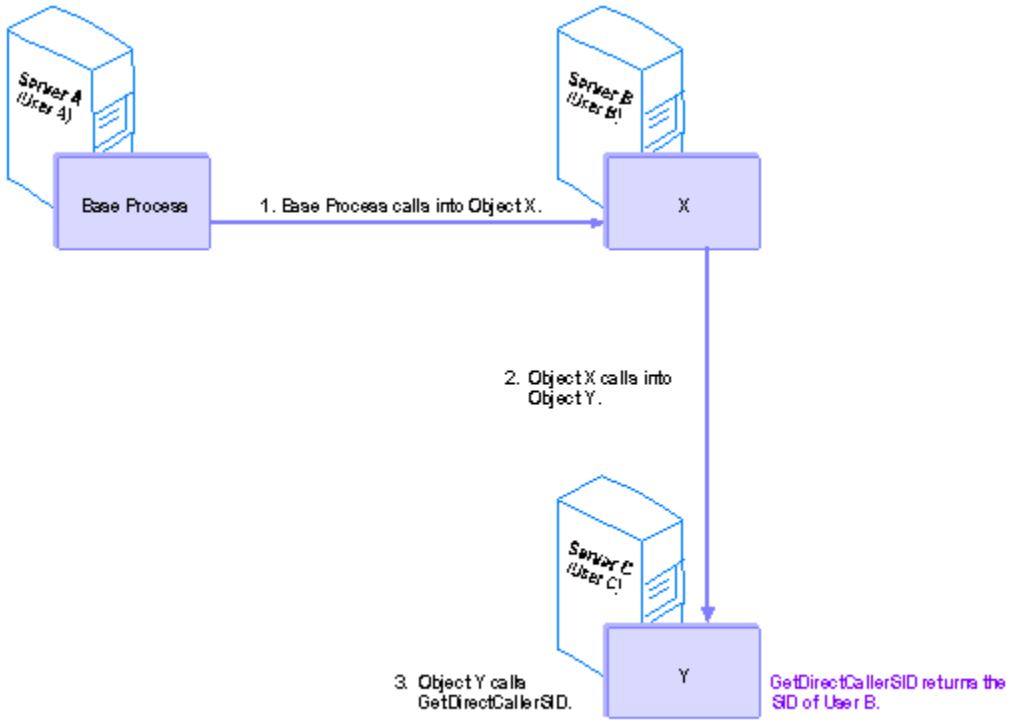
Context class

Return Values

User name associated with the process from which the current method was invoked.

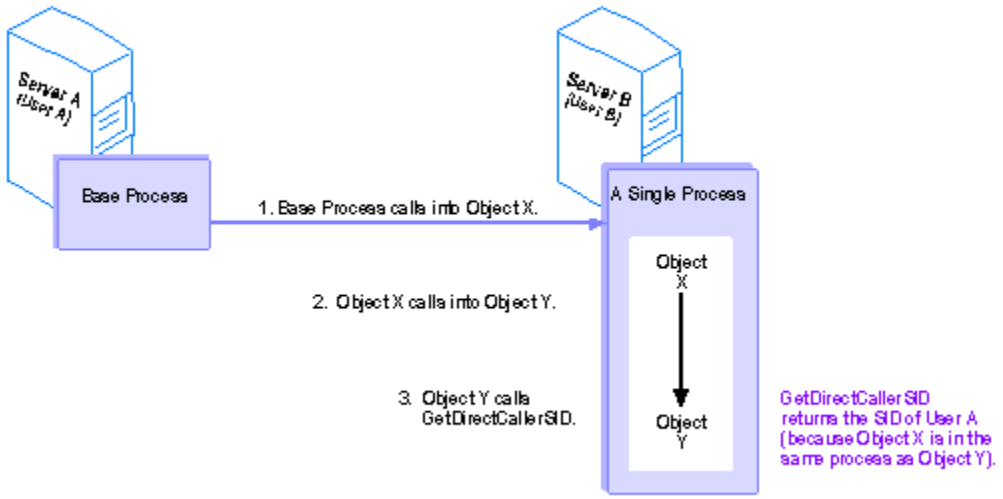
Remarks

You use the **getDirectCallerName** method to determine the user name associated with the process that called the object's currently executing method. The following scenarios illustrate the functionality of the **getDirectCallerName** method.



A base process running on server A, as user A, calls into object X on server B, running as user B. Then object X calls into object Y, running on server C. If object Y calls **getDirectCaller**, the name of user B is returned.

Security can only be enforced across process boundaries. This means that the name returned by g**etDirectCallerName** is the name associated with the process that called into the process in which the current object is running, not necessarily the immediate caller into the object itself. If an object calls into another object within the same process, when the second object calls **getDirectCallerName**, it will get the name of the most immediate caller outside its own process boundary, not the name of the object that directly called into it.



A base process, running on server A as user A, calls into object X on server B, running as user B. Then object X calls into object Y, running in the same process as object X, also on server B. When object Y calls **getDirectCallerName**, the name of user A is returned , not the name of user B.

See Also

Programmatic Security, Advanced Security Methods, **Context** class

getDirectCreatorName Method

Retrieves the user name associated with the current object's immediate (out-of-process) creator.

Provided By

Context class

String getDirectCreatorName ( );

Return Values

User name associated with the process that directly created the current object.

Remarks

You use the **getDirectCreatorName** method to determine the the process that created the current object. The following scenarios illustrate the functionality of the **getDirectCreatorName** method.

A base process running on server A, as user A, creates object X on server B, running as user B. Then object X creates object Y, running on server C. If object Y calls **getDirectCreatorName**, the name of user B is returned.

Security can only be enforced across process boundaries. This means that if an object creates another object within the same process, when the second object calls **getDirectCreatorName**, it will get the of the most immediate creator outside its own process boundary, not the user name associated with the object that actually created it.

A base client running on server A, as user A, creates object X on server B, running as user B. Then object X creates object Y, running in the same process as object X, also on server B. When object Y calls **getDirectCreatorName**, the name of user A is returned, not the name of user B.

See Also

Programmatic Security, Advanced Security Methods, **Context** class

getOriginalCallerName Method

Retrieves the user name associated with the base process that initiated the sequence of calls from which the call into the current object originated.

Provided By

Context class

String getOriginalCallerName ( );

Return Values

User name associated with the the base process that initiated the call sequence from which the current method was called.

Remarks

You use the **getOriginalCallerName** method to determine the user name associated with the original process that initiated the call sequence from which the current method was called. The following scenario illustrates the functionality of the **getOriginalCallerName** method.

Base process 1, running on server A as user A, creates object X on server B, running as user B. Then base process 1 passes its reference on object X to base process 2, running on server D as user D. Base process 2 uses that reference to call into object X. object X then calls into object Y, running on server C. If object Y then calls **getOriginalCallerName**, the name of user D is returned.

Note Usually, an object's original caller is the same process as its original creator. The only situation in which the original caller and the original creator would be different is one in which the original creator passes a reference to another process, and the other process initiates the call sequence (as in the preceding example).

Note The path to the original caller is broken if any object along the chain was created by some other means than **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**. For example, if base process 1 uses **CoCreateInstance** to create X, when Y calls **getOriginalCallerName,** the name it gets back will be the name of user B, not user D. This is because the call sequence is traced back through the objects' context and MTS can only create a context for an object that's created with either **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**.

See Also

Programmatic Security, Advanced Security Methods, **Context** class

getOriginalCreatorName Method

Retrieves the user name associated with the original base process that initiated the activity in which the current object is executing.

Provided By

Context class

String getOriginalCreatorName ( );

Return Values

User name associated with the the base process that initiated the activity in which the current object is executing.

Remarks

You use the **getOriginalCreatorName** method to determine the the process that initiated the activity in which the current object is executing. The following scenario illustrates the functionality of the **getOriginalCreatorName** method.

A base process running on server A, as user A, creates object X on server B, running as user B. Then object X creates object Y, running on server C. If object Y calls **getOriginalCreatorName**, the name of user A is returned.

Note The path to the original creator is broken if an object is created by some other means than **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**. For example, if the base process on server A uses **CoCreateInstance** to create X, when Y calls **getOriginalCreatorName,** the name it gets back will be the name of user B, not user A. This is because the creation sequence is traced back through the objects' context and MTS can only create a context for an object that's created with either **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**.

See Also

Programmatic Security, Advanced Security Methods, **Context** class

MTS Error Codes

The following errors can be returned by Microsoft Transaction Server (MTS) objects.

S_OK
: The call succeeded.

E_INVALIDARG
: One or more of the arguments passed in is invalid.

E_UNEXPECTED
: An unexpected error occurred.

CONTEXT_E_NOCONTEXT
: The current object doesn't have a context associated with it. This is probably either because its component hasn't been installed in a package or it wasn't created with one of the MTS **CreateInstance** methods.

CONTEXT_E_ROLENOTFOUND
: The role specified in the *szRole* parameter in the **IObjectContext::IsCallerInRole** method does not exist.

E_OUTOFMEMORY
: There's not enough memory available to instantiate the object. This error code can be returned by **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**.

REGDB_E_CLASSNOTREG
: The specified component is not registered as a COM component. This error code can be returned by **IObjectContext::CreateInstance** or **ITransactionContext::CreateInstance**.

DISP_E_ARRAYISLOCKED
: One or more of the arguments passed in contains an array that is locked. This error code can be returned by the **ISharedProperty::put_Value** method.

DISP_E_BADVARTYPE
: One or more of the arguments passed in isn't a valid VARIANT type. This error code can be returned by the **ISharedProperty::put_Value** method.

MTS Supported Variant Types

The following Automation types are supported by Microsoft Transaction Server.

VT_BOOL
VT_BSTR
VT_CARRAY
VT_CLSID
VT_CY
VT_DATE
VT_DECIMAL
VT_DISPATCH
VT_EMPTY
VT_ERROR
VT_HRESULT
VT_INT
VT_I1
VT_I2
VT_I4
VT_I8
VT_LPSTR
VT_LPWSTR
VT_NULL
VT_PTR
VT_R4
VT_R8
VT_SAFEARRAY
VT_UINT
VT_UI1
VT_UI2
VT_UI4
VT_UI8
VT_UNKNOWN
VT_USERDEFINED
VT_VARIANT
VT_VOID

The following types are not supported and will cause the server process to terminate with an error.

VT_FILETIME
VT_STREAM
VT_STREAMED_OBJECT
VT_BLOB_OBJECT
VT_BLOB
VT_STORAGE
VT_STORED_OBJECT
VT_CF

MTS Administrative Reference

The Microsoft Transaction Server (MTS) Administrative reference provides object and method information for Microsoft® Visual Basic or Microsoft Visual Basic Scripting Edition (VBScript) programmers, and interface and function information for Microsoft Visual C++® programmers. In addition, this technical reference provides a detailed description of the collections used by the administration objects.

This section contains the following sections:

Using MTS Administration Objects

Using MTS Collection Types

Automating MTS Administration With Visual Basic

MTS Administration Object Methods

Automating MTS Administration With Visual C++

See Also

Automating MTS Administration

Automating MTS Administration With Visual Basic

This reference topic provides guidance for Microsoft® Visual Basic or Microsoft Visual Basic Scripting Edition (VBScript) programmers who want to use the administration objects to automate tasks that an administrator performs using the Microsoft Transaction Server (MTS) Explorer. The MTS Visual Basic reference contains the following topics:

See Also

Automating MTS Administration

Using MTS Administration Objects

Use the MTS administration objects to automate administration for MTS packages.

This section describes how the following administration objects are used:

MTS Catalog Object

MTS CatalogObject Object

MTS CatalogCollection Object

MTS PackageUtil Object

MTS ComponentUtil Object

MTS RemoteComponentUtil Object

MTS RoleAssociationUtil Object

MTS Catalog Object

The Catalog object enables you to connect to an MTS catalog and access collections. This general administration object supports the following methods.

Method	Description
GetCollection	Gets a collection on the catalog without reading any objects from the catalog.
Connect	Connects to a remote catalog and returns a root collection.
MajorVersion	Returns the major version number of the catalog.
MinorVersion	Returns the minor version number of the catalog.

See Also

MTS CatalogObject Object, **MTS CatalogCollection Object**, **MTS PackageUtil Object**, **MTS ComponentUtil Object**, **MTS RemoteComponentUtil Object**, **MTS RoleAssociationUtil Object**

MTS CatalogObject Object

The **CatalogObject** object allows you to get and set object properties. This general administration interface supports the following methods.

Method	Description
Value	Gets and sets a property value
Key	Gets the value of the key property
Name	Gets the name of the object
IsPropertyReadOnly	**True** if the property cannot be set
IsPropertyWriteOnly	**True** if the property only supports set
Valid	**True** if all properties were successfully read from the catalog data store

See Also

MTS Catalog Object, **MTS CatalogCollection Object**, **MTS PackageUtil Object**, **MTS ComponentUtil Object**, **MTS RemoteComponentUtil Object**, **MTS RoleAssociationUtil Object**

MTS CatalogCollection Object

Use the **CatalogCollection** object to enumerate, add, delete, and modify catalog objects and to access related collections This general administration interface supports the following methods.

Method	Description
Item	Returns an object by index. The index is zero-based.
Count	Returns number of objects in the collection.
Remove	Removes an item according to its index position.
Add	Adds an object to the collection.
Populate	Reads all the collection objects from the catalog data store.
SaveChanges	Saves changes made to the collection into the catalog data store.
GetCollection	Gets a collection related to a specific object.
Name	Gets the name of a collection.
AddEnabled	Returns true if the **Add** method is enabled.
RemoveEnabled	Returns true if the **Remove** method is enabled.
GetUtilInterface	Gets the utility interface for the collection.
PopulateByKey	Reads the selected collection objects from the catalog data store.
DataStoreMajorVersion	Returns the major version number of the catalog data store.
DataStoreMinorVersion	Returns the minor version number of the catalog data store.

See Also

Add Method (CatalogCollection), **Remove Method (CatalogCollection)**, **MTS Catalog Object**, **MTS CatalogObject Object**, **MTS PackageUtil Object**, **MTS ComponentUtil Object**, **MTS RemoteComponentUtil Object**, **MTS RoleAssociationUtil Object**

MTS PackageUtil Object

The **PackageUtil** object enables installing and exporting a package. Instantiate this object by calling **GetUtilInterface** on a **Packages** collection.

This utility administration interface supports the following methods:

Method	Description
InstallPackage	Installs a pre-built package.
ExportPackage	Exports a package.
ShutdownPackage	Shuts down a package, thereby terminating that server process.

See Also

MTS Catalog Object, **MTS CatalogObject Object**, **MTS CatalogCollection Object**, **MTS ComponentUtil Object**, **MTS RemoteComponentUtil Object**, **MTS RoleAssociationUtil Object**

MTS ComponentUtil Object

Call the **ComponentUtil** object to install a component in a specific collection and import components registered as in-process servers. Create this object by calling **GetUtilInterface** on a **ComponentsInPackage** collection. This utility administration interface supports the following methods.

Method	Description
InstallComponent	Installs a component from a DLL.
ImportComponent	Imports a component that is already registered as an in-process server. Supply the CLSID of the component.
ImportComponentByName	Imports a component that is already registered as an in-process server. Supply the ProgID of the component.
GetCLSIDS	Returns an array of installable CLSIDs in the DLL/type library. Note that changes are not made to the data store.

See Also

MTS Catalog Object, **MTS CatalogObject Object**, **MTS CatalogCollection Object**, **MTS PackageUtil Object**, **MTS RemoteComponentUtil Object**, **MTS RoleAssociationUtil Object**

MTS RemoteComponentUtil Object

Using the **RemoteComponentUtil** object, you can program your application to pull remote components from a package on a remote server. Instantiate this object by calling **GetUtilInterface** on a **RemoteComponents** collection. This utility administration interface supports the following methods.

Method	Description
InstallRemoteComponent	Pulls remote components from a package on a remote server. Supply the package ID and CLSID.
InstallRemoteComponentByName	Pulls remote components from a package on a remote server. Supply the package name and ProgID.

See Also

GetUtilInterface Method (CatalogCollection), **MTS Catalog Object**, **MTS CatalogObject Object**, **MTS CatalogCollection Object**, **MTS PackageUtil Object**, **MTS ComponentUtil Object**, **MTS RoleAssociationUtil Object**

MTS RoleAssociationUtil Object

Call methods on the **RoleAssociationUtil** object to associate roles with a component or interface. Create this object by calling the **GetUtilInterface** method on a **RolesForPackageComponent** or **RolesForPackageComponentInterface** collection. This utility administration interface supports the following methods.

Method	Description
AssociateRole	Associates the role by role ID with the component or interface.
AssociateRoleByName	Associates the role by role name with the component or interface.

See Also

GetUtilInterface Method (CatalogCollection), **MTS Catalog Object**, **MTS CatalogObject Object**, **MTS CatalogCollection Object**, **MTS PackageUtil Object**, **MTS ComponentUtil Object**, **MTS RemoteComponentUtil Object**

Using MTS Collection Types

This topic describes the collections and collection properties supported by the MTS catalog. Additional collections and properties may be added in future versions. Use the **DataStoreMajorVersion** and **DataStoreMinorVersion** properties of a collection to distinguish between catalog versions.

The MTS catalog data store supports the following collection types:

MTS LocalComputer Collection

The **LocalComputer** contains a single object that corresponds to the computer whose catalog that you are accessing. If you call the **Connect** method on the **Catalog** object, the **LocalComputer** collection contains information about the computer whose catalog you are accessing. This collection does not support the **Add, Remove** or **GetUtilInterface** methods.

The following table provides a list of the properties supported by the **LocalComputer** collection.

Property	Description
Name	"My Computer". Data Type: String Default value: N/A Access: Read only.
Description	Description of the computer. Data Type: String Default value: Empty string Access: Read/write
Transaction Timeout	The transaction timeout setting in seconds. Data Type: Integer Default value: 60 Access: Read/write
RemoteComponent InstallPath	The path in which remote component files will be installed when remote components are configured Data Type: String Default value: the subdirectory "Remote" in the MTS install path Access: Read/write
PackageInstall Path	The default path in which component files will be installed when pre-built packages are installed. Data Type: String Default value: the subdirectory "Packages" in the MTS install path Access: Read/write
ResourcePooling Enabled	"Y" enables resource pooling. "N" disables resource pooling. Note that resource pooling is currently only supported by the ODBC resource dispenser. Data Type: String Default value: "Y" Access: Read/write
ReplicationShare	When replicating the MTS catalog, the name of a share that the target system should use to access installed packages. Data Type: String Default value: Empty string Access: Read/write
RemoteServer Name	The computer name to be generated when you use the MTS Explorer to create a client executable. If this string is blank or empty, the physical computer name of the exporting computer is used. If you put the name of the remote server as the string, the application executable generated by the MTS Explorer points to that remote server name. Data Type: String Default value: Empty string Access: Read/write

See Also

RelatedCollectionInfo, **PropertyInfo**, **ErrorInfo**

MTS ComputerList Collection

The **ComputerList** collection provides access to the list of computers shown in the MTS Explorer Computers folder. This collection supports the **Add** and **Remove** methods. This collection does not support the **GetUtilInterface** method.

The following table provides a list of the properties supported by the **ComputerList** collection.

Property	Description
Name	The name of the computer. Data Type: String Default value: "NewComputer" Access: Read/Write while using the **Add** method. After adding, read-only.

See Also

RelatedCollectionInfo, **PropertyInfo**, **ErrorInfo**, **Add Method (CatalogCollection)**, **Remove Method (CatalogCollection)**

MTS Packages Collection

As the top-level collection managed by the MTS Explorer, the **Packages** collection contains the packages installed on the local machine running MTS. Packages contain a set of components that run in the same server process, and define declarative security constructs that determine access to components at run time. The **Packages** collection supports the **Add** method and **Remove** method on the **CatalogCollection** object. In addition, the **GetUtilInterface** method of this collection returns a **PackageUtil** object which can be used to install and export packages.

The following table provides a list of the properties supported by the **CatalogObject** objects within the **Packages** collection.

Property	Description
Name	Name of the package. Data Type: String Default value: "New package" Access: Read/Write
ID	A universally unique identifier (UUID) for the package. Data Type: String Default value: A unique identifier is generated Access: Read/Write when using the **Add** method. Read-only after using the **Add** method.
Description	Describes the package. Description fields hold a maximum of 500 characters. Data Type: String Default value: None Access: Read/Write
IsSystem	Identifies an MTS system package. "N" signifies that the package is not a Transaction Server system package, and "Y" indicates that package is an MTS system package. Data Type: String Default value: "N" Access: Read only
Authentication	Sets authentication level for calls. Possible values are 0 through 6, which correspond to the Remote Procedure Call (RPC) authentication settings. Data Type: Long Default value: 4 Access: Read/Write
ShutdownAfter	Sets the delay before shutting down a server process after it becomes idle. Shutdown latency ranges from 0 to 1440 minutes. Data Type: Long Default value: 3 Access: Read/Write
RunForever	Enables a server process to continue if a package is idle. If value is set to "Y", the server process will not shut down when left idle. If set to "N", the process will shut down according the value set by the **ShutDownAfter** property. Data Type: String Default value: "N" Access: Read/Write
SecurityEnabled	Checks the security credentials of any client that calls the package if value is set to "Y." Data Type: String

	Default value: "N" Access: Read/Write
Identity	Sets the server process identity for the package. Specify a valid Windows NT user account or "Interactive User" to have the package assume the identity of the current logged-on user. Data Type: String Default value: "Interactive User" Access: Read/Write
Password	Sets the password used by the server process to log on under the identity above. Data Type: String Default value: None Access: Write only
Activation	Sets the package level activation property to either "Local" or "Inproc". The Local setting determines that objects within the package will run within a dedicated local server process. A package running under the Local activation setting is a "server package". The Inproc activation setting means objects run in their creator's process. A package running under the Inproc activation setting is a "library package" Data Type: String Default Value: "Local" Access: Read/Write
Changeable	Sets whether changes to the package settings, or those of its components, are allowed (either programmatically, or through the MTS UI). Data Type: String Default Value: Y Access: Read/Write
Deleteable	Sets whether the package or its components can be deleted (either programmatically, or through the MTS UI). Data Type: String Default Value: "Y" Access: Read/Write
CreatedBy	Informational string to describe the package creator. Data Type: String Default Value: Empty string Access: Read/Write

See Also

ComponentsInPackage, **RolesInPackage**, **RelatedCollectionInfo**, **PropertyInfo**, **ErrorInfo**

MTS ComponentsInPackage Collection

The **ComponentsInPackage** collection contains the set of components that run in the same server process and compose a package. This collection supports the **Remove** method. The **Add** method is not supported. You must use the **ComponentUtil** object to install components into the package.

The following table provides a list of the properties supported by the **CatalogObjects** within the **ComponentsInPackage** collection.

Property	Description
ProgID	The name that identifies the component. Data Type: String Default value: None Access: Read only
CLSID	The universally unique identifier (UUID) for the component. Data Type: String Default value: None Access: Read only
Transaction	Determines how a component supports transactions. Must be one of the following transaction settings: "Required" "Requires New" "Not Supported" "Supported" Data Type: String Default value: "Not supported" Access: Read/Write
Description	Describes the component. Description fields hold a maximum of 500 characters. Data Type: String Default value: None Access: Read/Write
PackageID	Defines the identity of the owning package. Data Type: String Default value: None Access: Read only
PackageName	Defines the name of the owning package. Data Type: String Default value: "New Package" Access: Read only
ThreadingModel	Determines how instances of the component are assigned to threads for method execution. Possible values are those supported by Component Object Model (COM). Data Type: String Default value: None Access: Read only
SecurityEnabled	Checks the security credentials of any client that calls the component if value is set to "Y." Data Type: String Default value: "Y" Access: Read/Write
DLL	Displays the name of the DLL containing the component implementation. Data Type: String Default value: None

	Access: Read only
IsSystem	Identifies an MTS system component. "N" signifies that the package is not a Transaction Server system component, and "Y" indicates that package is an MTS system component. Data Type: String Default value: "N" Access: Read only

See Also

InterfacesForComponent, **RolesForPackageComponent**, **RelatedCollectionInfo**, **PropertyInfo**, **ErrorInfo**

MTS RemoteComponents Collection

Use the Microsoft Transaction Server (MTS) Explorer on a client computer to add remote component entries that see components installed on a remote server. This is often described as "pulling" remote component information from a server. Configuring remote components automatically copies proxy/stub DLLs and type libraries from the server to the client. The **RemoteComponents** collection supports the **Remove** method. This collection does not support the **Add** method on the **CatalogCollection** object. Instead you must call **GetUtilInterface** to obtain the **RemoteComponentUtil** object in order to add new remote components.

The following table provides a list of the properties supported by the **CatalogObjects** within the **RemoteComponents** collection.

Property	Description
ProgID	The name that identifies the remote component. Data Type: String Default value: None Access: Read only
CLSID	The universally unique identifier (UUID) for the component. Data Type: String Default value: None Access: Read only
Description	Describes the remote component. Description fields hold a maximum of 500 characters. Data Type: String Default value: None Access: Read/Write
Server	Name of the server hosting the remote component. Data Type: String Default value: None Access: Read only

See Also

InterfacesForRemoteComponent, **RelatedCollectionInfo**, **PropertyInfo**, **ErrorInfo**

MTS InterfacesForComponent and InterfacesForRemoteComponent Collections

The **InterfacesForComponent** and **InterfacesForRemoteComponent** collections provide information about a selected interface. **InterfacesForComponent** or **InterfacesForRemoteComponent** collections are listed for each component, and can be used by an administrator to identify or manage the interface. These collections do not support the **Add**, **Remove**, or **GetUtilInterface** methods.

The following table provides a list of the properties supported by the **CatalogObjects** within the **InterfacesForComponent** and the **InterfacesForRemoteComponent** collections.

Property	Description
Name	Displays the friendly name of the interface. Data Type: String Default value: None Access: Read only
ID	Displays the unique interface identifier (IID). Data Type: String Default value: None Access: Read only
Description	Describes the interface. Description fields hold a maximum of 500 characters. Data Type: String Default value: None Access: Read/Write
ProxyCLSID	Displays the CLSID of the proxy/stub. Data Type: String Default value: None Access: Read only
ProxyDLL	Displays the file name of the proxy/stub DLL. Data Type: String Default value: None Access: Read only
ProxyThreadingModel	Displays the threading model of the selected proxy/stub. Data Type: String Default value: None Access: Read only
TypeLibID	Displays the UUID of the type library. Data Type: String Default value: None Access: Read only
TypeLibVersion	Displays the version of the type library. Data Type: String Default value: None Access: Read only
TypeLibLangID	Displays the language identification number of the type library. Data Type: String Default value: None Access: Read only
TypeLibPlatform	Displays the platform of the type library. Data Type: String

	Default value: None Access: Read only
TypeLibFile	Displays the file name of the type library. Data Type: String Default value: None Access: Read only

See Also

MethodsForInterface (InterfacesForComponent only), **RolesForPackageComponentInterface (InterfacesForComponent only)**, **RelatedCollectionInfo**, **PropertyInfo**, **ErrorInfo**

MTS RolesForPackageComponent and RolesForPackageComponentInterface Collections

The **RoleForPackageComponent** and **RolesForPackageComponentInterface** collections contain the roles associated with a component or interface. You add existing roles to these collections from a package's **RolesInPackage** collection. The **Add** method is not supported by this collection. Use the **RoleAssociationUtil** methods to add roles to this collection. The **CatalogCollection Remove** method is supported by this collection.

The following table provides a list of the properties supported by the **CatalogObject(s)** within the **RolesForPackageComponent** and the **RolesForPackageComponentInterface** collections.

Property	Description
Name	Displays the name of the role associated with a component. Data Type: String Default value: None Access: Read only
ID	Displays the universally unique identifier (UUID) of the role. Data Type: String Default value: None Access: Read only
Description	Describes the role. Description fields hold a maximum of 500 characters. Data Type: String Default value: None Access: Read only

See Also

RelatedCollectionInfo, **PropertyInfo**, **ErrorInfo**

MTS MethodsForInterface Collection

The **MethodsForInterface** collection contains the methods defined in an interface. Method properties are used to display information about the methods exposed by an interface. This collection does not support the **Add**, **Remove**, or **GetUtilInterface** methods.

The following table provides a list of the properties supported by the **CatalogObjects** within the **MethodsForInterface** collection.

Property	Description
Name	Displays the name of a method. Data Type: String Default value: None Access: Read only
Description	Describes the method. Description fields hold a maximum of 500 characters. Data Type: String Default value: None Access: Read only

See Also

RelatedCollectionInfo, **PropertyInfo**, **ErrorInfo**

MTS RolesInPackage Collection

The **RolesInPackage** collection defines a class of users for a set of components in a package. Each role defines a set of users allowed to invoke interfaces on a component. Roles can be applied to both components and component interfaces. This collection supports the **Add** and **Remove** methods. This collection does not support the **GetUtilInterface** method.

The following table provides a list of the properties supported by the **CatalogObjects** within the **RolesInPackage** collection.

Property	Description
Name	The role name. Data Type: String Default value: "New Role" Access: Read/write
ID	The universally unique identifier (UUID) for the role. Data Type: String Default value: A unique identifier is generated. Access: Read/Write while using the Add method. After adding an object, Read-only.
Description	Describes the new **Role.** Description fields hold a maximum of 500 characters. Data Type: String Default value: None Access: Read/write

See Also

UsersInRole, **RelatedCollectionInfo**, **PropertyInfo**, **ErrorInfo**

MTS UsersInRole Collection

The **UsersInRole** collection lists the members of the class of users that have been authorized to invoke methods on the component or component interface associated with the role. This collection supports the **Add** and **Remove** methods. This collection does not support the **GetUtilInterface** method.

The following table provides a list of the properties supported by the **CatalogObjects** within the **UsersInRole** collection.

Property	Description
User	The name of an NT user account or group. Data Type: String Default value: "New User" Access: Read/Write while using the Add method. After adding, read-only.

See Also

RelatedCollectionInfo, **PropertyInfo**, **ErrorInfo**

MTS ErrorInfo Collection

The **ErrorInfo** collection is used to retrieve extended error information about methods that deal with multiple objects. This collection does not support the **Add**, **Remove**, or **GetUtilInterface** methods. Use the **GetCollection** method on a collection to access the **ErrorInfo** collection associated with the original collection. The **ErrorInfo** collection is accessible from any collection except **ErrorInfo**, **RelatedCollectionInfo**, and **PropertyInfo**. When calling methods on a utility object, extended error information may be created in the **ErrorInfo** collection associated with the collection used to create the utility object.

The following table provides a list of the properties supported by the **CatalogObjects** within the **ErrorInfo** collection.

Property	Description
Name	Name of the object or file. Data Type: String Default value: None Access: Read only.
ErrorCode	Error code for the object or file. Data Type: Long Default value: None Access: Read only

See Also

MTS LocalComputer Collection, **MTS ComputerList Collection**, **MTS Packages Collection**, **MTS ComponentsInPackage Collection**, **MTS RemoteComponents Collection**, **MTS InterfacesForComponent and InterfacesForRemoteComponent Collections**, **MTS RolesForPackageComponent and RolesForPackageComponentInterface Collections**, **MTS MethodsForInterface Collection**, **MTS RolesInPackage Collection**, **MTS UsersInRole Collection**, **MTS PropertyInfo Collection**, **MTS RelatedCollectionInfo Collection**

MTS PropertyInfo Collection

The **PropertyInfo** collection is used to retrieve information about the properties that a specified collection supports. This collection does not support the **Add**, **Remove**, or **GetUtilInterface** methods. The **PropertyInfo** collection is accessible from any collection by using the **GetCollection** method.

The following table provides a list of the properties supported by the **CatalogObject(s)** within the **PropertyInfo** collection.

Property	Description
Name	Name of the property. Data Type: String Default value: None Access: Read only

See Also

MTS LocalComputer Collection, **MTS ComputerList Collection**, **MTS Packages Collection**, **MTS ComponentsInPackage Collection**, **MTS RemoteComponents Collection**, **MTS InterfacesForComponent and InterfacesForRemoteComponent Collections**, **MTS RolesForPackageComponent and RolesForPackageComponentInterface Collections**, **MTS MethodsForInterface Collection**, **MTS RolesInPackage Collection**, **MTS UsersInRole Collection**, **MTS ErrorInfo Collection**, **MTS RelatedCollectionInfo Collection**

MTS RelatedCollectionInfo Collection

The **RelatedCollectionInfo** collection is used to retrieve information about other collections related to the collection from which this collection is called. The **RelatedCollectionInfo** collection is accessible from any collection by using the **GetCollection** method. The **RelatedCollectionInfo** collection will contain one object for each collection that is accessible from the original collection. Related collections follow the MTS Explorer folder hierarchy. This collection does not support the **Add**, **Remove**, or **GetUtilInterface** methods.

The following table provides a list of the properties supported by the **CatalogObjects** within the **RelatedCollectionInfo** collection.

Property	Description
Name	Name of the related collection. Data Type: String Default value: None Access: Read only

See Also

MTS LocalComputer Collection, **MTS ComputerList Collection**, **MTS Packages Collection**, **MTS ComponentsInPackage Collection**, **MTS RemoteComponents Collection**, **MTS InterfacesForComponent and InterfacesForRemoteComponent Collections**, **MTS RolesForPackageComponent and RolesForPackageComponentInterface Collections**, **MTS MethodsForInterface Collection**, **MTS RolesInPackage Collection**, **MTS UsersInRole Collection**, **MTS ErrorInfo Collection**, **MTS PropertyInfo Collection**

MTS Visual Basic Error Codes

The following table lists the error codes returned by methods called on the MTS catalog collection and catalog utility objects.

Error code	Description
Visual Basic run-time error 5	Indicates one of the following: An invalid collection or property name was entered. An out parameter was NULL. The value is not one of the supported values or falls outside the supported range. The property is read-only. The property cannot be changed after the object is created. An invalid index was entered.
Visual Basic run-time error 445	Object has been removed from the collection or the method is not supported on this object.
mtsErrObjectErrors	Errors were encountered processing some objects or file. See the **ErrorInfo** collection for object and file-specific error codes.
mtsErrNoUser	User ID for user in role is not valid.
mtsErrUserPasswdNotValid	Package identity user ID and/or password are not valid.
mtsErrAuthenicationLevel	Required authentication level (package privacy) could not be set for package updates.
mtsErrPDFReadFail	An error occurred reading the package file.
mtsErrPDFVersion	Package file version is invalid.
mtsErrBadPath	Package file path is invalid.
mtsErrPackageExists	Package with the same ID is already installed.
mtsErrRoleExists	A role with the same ID is already installed. The role ID in the package file is likely corrupted.
mtsErrCantCopyFile	Errors occurred copying one or more files to the install directory.
mtsErrInvalidUserids	One or more user IDS for roles were invalid.
mtsErrCLSIDOrIIDMismatch	One or more component or interface identifiers in a component DLL do not match the identifiers saved in the package file. The package file is out of date.
mtsErrPackDirNotFound	Package install directory is invalid due to general registry read/write errors.
mtsErrPDFWriteFail	An error occurred writing the package file.
mtsErrNoTypeLib	Could not find the type library for one or more components.

The following table lists the object or file-specific error codes returned in **ErrorInfo** collections.

Error code	Description
mtsErrObjectInvalid	One or more object properties is corrupted or invalid.
mtsErrKeyMissing	One or more objects is not found in the catalog data store.
mtsErrAlreadyInstalled	Component is already installed.
mtsErrDownloadFailed	One or more component files could not be copied to the client.
mtsErrRemoteInterface	No interface information is available for the component. Component files could not be downloaded.
mtsErrCoReqCompInstalled	Component in the same DLL file is already installed.
mtsErrNoRegistryCLSID	Component's CLSID is corrupted.
mtsErrBadRegistryProgID	Component's ProgID is corrupted.
mtsErrDllLoadFailed	Component's DLL could not be loaded.
mtsErrDllRegisterServer	**DllRegisterServer** method failed during component self-registration.
mtsErrNoServerShare	No file share is available on the server to copy component files from the network path.
mtsErrNoAccessToUNC	Network path registered for this component could not be accessed.
mtsErrBadRegistryLibID	Component type library ID is corrupted.
mtsErrTreatAs	Component **TreatAs** key was found, but is not supported.
mtsErrBadForward	IID forward entry is corrupted.
mtsErrBadIID	IID is corrupted.
mtsErrRegistrarFailed	Component registrar method failed during component install.
mtsErrCompFileDoesNotExist	Component file does not exist.
mtsErrCompFileLoadDLLFail	DLL file could not be loaded.
mtsErrCompFileGetClassObj	**DllGetClassObject** function call failed during the DLL self-registration process.
mtsErrCompFileClassNotAvail	Class coded in the type library was not supported.
mtsErrCompFileBadTLB	Type library was corrupted.
mtsErrCompFileNotInstallable	File does not contain COM components or type library information.
mtsErrNotChangeable	Changes to this object and sub-objects have been disabled.

mtsErrNotDeletable	Delete function for this object has been disabled.
mtsErrSession	Catalog version is not supported.

The following tables lists general read and write registry errors.

Error Code	Description
mtsErrNoRegistryRead	Access control failure reading a registry key.
mtsErrNoRegistryWrite	Access control failure writing a registry key.
mtsErrNoRegistryRepair	Access control failure writing a registry key.

MTS Administration Object Methods

The following topics list the methods of the MTS administration objects:

Add Method (CatalogCollection)
AddEnabled Property (CatalogCollection)
AssociateRole Method (RoleAssociationUtil)
AssociateRoleByName Method (RoleAssociationUtil)
Connect Method (Catalog)
Count Property (CatalogCollection)
DataStoreMajorVersion Property (CatalogCollection)
DataStoreMinorVersion Property (CatalogCollection)
ExportPackage Method (PackageUtil)
GetCLSIDs Method (ComponentUtil)
GetCollection Method (Catalog)
GetCollection Method (CatalogCollection)
GetUtilInterface Method (CatalogCollection)
ImportComponent Method (ComponentUtil)
ImportComponentByName Method (ComponentUtil)
InstallComponent Method (ComponentUtil)
InstallPackage Method (PackageUtil)
InstallRemoteComponent Method (RemoteComponentUtil)
InstallRemoteComponentByName Method (RemoteComponentUtil)
IsPropertyReadOnly Property (CatalogObject)
IsPropertyWriteOnly Property (CatalogObject)
Item Property (CatalogCollection)
Key Property (CatalogObject)
MajorVersion Property (Catalog)
MinorVersion Property (Catalog)
Name Property (CatalogObject)
Name Property (CatalogCollection)
Populate Method (CatalogCollection)
PopulateByKey Method (CatalogCollection)
Remove Method (CatalogCollection)
RemoveEnabled Property (CatalogCollection)
SaveChanges Method (CatalogCollection)
ShutdownPackage Method
Valid Property (CatalogObject)
Value Property (CatalogObject)

See the Automating MTS Administration topic for sample code that demonstrates how these methods are used to program administration using Microsoft® Visual Basic or Microsoft Visual Basic Scripting Edition (VBScript).

Add Method (CatalogCollection)

Adds a member to a collection object and returns the **CatalogObject** object.

Syntax

object**.Add**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

Remarks

Call the **Add** method to create a new object in a collection. This method is supported in the following collections:

Packages
RolesInPackage
UsersInRole

To install or create objects in other collections, use the catalog utility interfaces. Note that you must call the **SaveChanges** method to write the new object to the catalog data store.

For a list of the MTS collections and their properties, see Using MTS Collections.

See Also

MTS Packages Collection, **MTS RolesInPackage Collection**, **MTS UsersInRole Collection**, **AddEnabled Property (CatalogCollection)**

AddEnabled Property (CatalogCollection)

Returns a **Boolean** value indicating whether the **Add** method is enabled on this collection.

Syntax

object.**AddEnabled**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

Remarks

If the value returned is **True**, then you can call the **Add** method to create a new object in a collection. If the value returned is **False**, you must use the catalog utility object methods to create a new object.

For a list of the MTS collections and their properties, see the Using MTS Collections topic.

See Also

Add Method (CatalogCollection)

AssociateRole Method (RoleAssociationUtil)

Associates a role with a component or component interface.

Syntax

object.**AssociateRole(***ID***)**

Parameters

object
Required. An object variable that evaluates to a **RoleAssociationUtil** object.

ID
Required. A **String** expression that specifies the role ID of the roles to associate with the object.

Remarks

The changes are applied immediately to the catalog.

For a list of properties supported by **Role** collections,see the Using MTS Collections topic.

See Also

AssociateRoleByName Method (RoleAssociationUtil)

AssociateRoleByName Method (RoleAssociationUtil)

The **AssociateRoleByName** method associates a role with a specified component or component interface.

Syntax

object.**AssociateRoleByName(***name***)**

Parameters

object
Required. An object variable that evaluates to a **RoleAssociationUtil** utility object.

name
Required; **String**. An expression providing the name of the role to associate with a component or component interface.

Remarks

The changes are applied immediately to the catalog. For a list of properties supported by **Role** collections,see the Using MTS Collections topic.

See Also

AssociateRole Method (RoleAssociationUtil)

Connect Method (Catalog)

Connects to a catalog and returns a root collection.

Syntax

set root object.**Connect(***name***)**

Parameters

root
Required. String containing the root collection that serves as a starting point to locate top-level collections.

object
Required. An object variable that evaluates to a catalog object.

name
Required; **String**. String containing the name of a remote computer. To connect to a local computer, supply an empty string as this argument.

Remarks

The **Connect** method returns a root collection, which is bound to the connected computer. A root collection serves as a starting point to locate top-level collections, and does not contain any objects or properties.

Note that you can also use the **GetCollection** method to locate a package on the local computer without first having to call the **Connect** method.

You can get the following collections from the root collection:

Packages
RemoteComponents
RelatedCollectionInfo
ComputerList
LocalComputer
PropertyInfo

For a list of the MTS collections and their properties, see the Using MTS Collections topic.

See Also

GetCollection Method (CatalogCollection)

Count Property (CatalogCollection)

Returns an integer value indicating the number of objects in a collection.

Syntax

object.**Count**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

Remarks

Upon instantiation, a **CatalogCollection** object returns a count of zero. Call the **Populate** method to read from the **CatalogCollection** object, and then use the **Count** method to return the number of objects in the collection.

for a list of the MTS collections and their properties, see the Using MTS Collections topic.

See Also

Populate Method (CatalogCollection)

DataStoreMajorVersion Property (CatalogCollection)

Returns an integer value indicating the major version number of the catalog.

Syntax

object.**DataStoreMajorVersion**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

See Also

DataStoreMinorVersion Property (CatalogCollection),

DataStoreMinorVersion Property (CatalogCollection)

Returns an integer value indicating the minor version number of the catalog.

Syntax

object.**DataStoreMajorVersion**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

See Also

DataStoreMajorVersion Property (CatalogCollection)

ExportPackage Method (PackageUtil)

Exports a package.

Syntax

object.**ExportPackage(***PackID, FileName, Options***)**

Parameters

object
Required. An object variable that evaluates to a **PackageUtil** object.

PackID
Required; **String**. An object variable that specifies the package ID of the package to export.

FileName
Required; **String**. An object variable that provides the name of the package file to export.

Options
Required; **Long**. An integer value specifying export options. This parameter can be 0 or MtsExportUsers, which includes users in roles in the package file.

GetCLSIDs Method(ComponentUtil)

Returns an array of installable class identifiers (CLSIDs) in the component DLL and/or type library.

Syntax

object.**GetCLSIDs(***bstrDLLFile, bstrTypeLibFile, aCLSIDs***)**

Parameters

BstrDLLFile
Required; **String**. A string variable that evaluates to the path of the DLL that you want checked.

BstrTypeLibFile
Required; **String**. A string variable that evaluates to the path of the type library that you want checked. If the type library is embedded with the DLL (as is the case with DLLs generated by Microsoft Visual Basic), this parameter should be an empty string).

aCLSIDS
Required; **Variant**. An output array of CLSIDs (VARIANTS) that can be installed from the supplied DLL and/or type library.

GetCollection Method (Catalog)

Instantiates a **CatalogCollection** object.

Syntax

set x object.**GetCollection(***Name***)**

Parameters

x
Required. An object variable (a variant, or object variable, or a **CatalogCollection** variable) for the returned collection.

object
Required. An object variable that evaluates to a catalog object.

Name
Required; **String**. A string expression containing the name of the collection to instantiate.

Remarks

You can use this method to get the following collections:

Packages
ComputerList
LocalComputer
RemoteComponents
RelatedCollectionInfo

After using the **GetCollection** method, you must fill the object by calling the **Populate** method. See the **Populate** method topic for further detail.

For a list of the MTS collections and their properties, see the Using MTS Collections topic..

GetCollection Method (CatalogCollection)

Retrieves a collection from the catalog.

Syntax

set x object.**GetCollection(***name, key***)**

Parameters

x
Required. An object variable (a variant, or object variable, or a **CatalogCollection** variable) for the returned collection.

object
Required. An object variable that evaluates to a **CatalogCollection** object.

name
Required. A **String** expression containing the name of the collection to instantiate.

key
Required. A **Variant** expression containing the key of the object from which to navigate.

Remarks

Note that the **GetCollection** method gets an empty collection; you must call the **Populate** method to fill the collection.

For a list of the MTS collections and their properties, see the Using MTS Collections topic.

See Also

Populate Method (CatalogCollection),

GetUtilInterface Method (CatalogCollection)

Instantiates a utility object for the collection.

Syntax

set util object.**GetUtilInterface**

Parameters

util
Required. An object variable that evaluates to a catalog utility object.

object
Required. An object variable that evaluates to a **CatalogCollection** object.

Remarks

Call the **GetUtilInterface** method to instantiate any one of the **PackageUtil**, **ComponentUtil**, **RemoteComponentUtil**, and **RoleAssociationUtil** utility objects. This method is only supported on **Packages**, **ComponentsInPackage**, **RemoteComponents**, **RolesForPackageComponent**, and **RolesForPackageComponentInterface** collections.

For a list of the MTS collections and their properties, see the Using MTS Collections topic.

ImportComponent Method (ComponentUtil)

Imports a component that is already registered as an in-process (in-proc) server.

Syntax

object.**ImportComponent(***CLSID***)**

Parameters

object
Required. An object variable that evaluates to a **ComponentUtil** object.

CLSID
Required; **String**. An expression containing the CLSID of the component to be installed.

Remarks

The changes are applied immediately to the catalog.

For a description of the **Component** collection and its associated properties, see the Using MTS Collections topic.

See Also

ImportComponentByName Method (ComponentUtil)

ImportComponentByName Method (ComponentUtil)

Imports a component that is already registered as an in-process server by the component's programmatic identifier (ProgID).

Syntax

object.**ImportComponentByName(***ProgID***)**

Parameters

object
Required. An object variable that evaluates to a **ComponentUtil** object.

ProgID
Required. A **String** expression identifying the ProgID of the component to be installed.

Remarks

The changes are applied immediately to the catalog. For a description of the **Component** collection and its associated properties, see the Using MTS Collections topic.

See Also

ImportComponent Method (ComponentUtil)

InstallComponent Method (ComponentUtil)

Installs a component into a package.

Syntax

object.**InstallComponent(***filepath, typelibrary, proxy-stub***)**

Parameters

object
Required. An object variable that evaluates to a **ComponentUtil** object.

filepath
Required. A **String** expression that provides the file path of the DLL containing the components to install.

typelibrary
Required. A **String** expression that provides the file path of the type library to use. Pass an empty string as this argument if the type library is embedded in the DLL.

proxy-stub
Required. A **String** expression that provides the file path of a custom proxy-stub DLL to use. Pass an empty string as this argument if there is no custom proxy-stub DLL.

Remarks

The changes are applied immediately to the catalog.

For a description of the **Components** collection and its associated properties, see the Using MTS Collections topic.

See Also

InstallRemoteComponent Method (RemoteComponentUtil), **InstallRemoteComponentByName Method (RemoteComponentUtil)**

InstallPackage Method (PackageUtil)

Installs a component or components that are valid within a package's collection.

Syntax

object.**InstallPackage(***FileName, InstallPath, options***)**

Parameters

object
Required. An object variable that evaluates to a **Package** utility object.

FileName
Required; **String**. String expression evaluating to the name of the package to install.

InstallPath
Required; **String**. String expression evaluating to the install path for component files.

options
Required; **Long**. A long value specifying install options. This parameter can be 0 or mtsInstallUsers, which adds users saved in the package file. If this option is not specified, users saved in the package file are not installed.

Remarks

All component files must be in the same directory of the package file. Component files arecopied to the install path specified as an argument of the **InstallPackage** method.

The changes are applied immediately to the catalog.

For a description of the **Components** collection and its associated properties, see the Using MTS Collections topic..

InstallRemoteComponent Method (RemoteComponentUtil)

Pulls remote components from a package on a remote server.

Syntax

object.**InstallRemoteComponent(***computer, PackID, CLSID***)**

Parameters

object
Required. An object variable that evaluates to **a RemoteComponentUtil** object.

computer
Required; **String**. A string expression providing the name of the remote computer.

PackID
Required; **String**. A string expression providing the package identification of the package containing the remote component.

CLSID
Required; **String**. A string expression containing the CLSID of the remote component.

Remarks

The changes are applied immediately to the catalog.

See the Working with Remote MTS Computers topic for a complete description of how to pull components from a remote server.

See Also

InstallRemoteComponentByName Method (RemoteComponentUtil)

InstallRemoteComponentByName Method (RemoteComponentUtil)

Pulls remote components by name from the package on a remote server.

Syntax

object.**InstallRemoteComponentByName(***computer, PackName, ProgID***)**

Parameters

object
Required. An object variable that evaluates to a **RemoteComponentUtil** object.

computer
Required; **String**. A string expression providing the name of the remote computer.

PackName
Required; **String**. A string expression providing the name of the package containing the remote component.

ProgID
Required; **String**. A string value containing the ProgID of the remote component.

Remarks

The changes are applied immediately to the catalog. See the Working with Remote MTS Computers topic for a complete description of how to pull components from a remote server.

See Also

InstallRemoteComponent Method (RemoteComponentUtil)

IsPropertyReadOnly Property (CatalogObject)

Returns a **Boolean** value that indicates if the property for an object is set to read-only.

Syntax

object.**IsPropertyReadOnly(***value***)**

Parameters

object
Required. An object variable that evaluates to a catalog object property.

value
Required. The name of the value for which to check the read-only property.

Remarks

If the value returned by the **IsPropertyReadOnly** method is **True**, then you cannot modify the property. If the value returned is **False**, you can modify the property using the **Value** property.

IsPropertyWriteOnly Property (CatalogObject)

Returns a **Boolean** value that indicates if the property for an object is set to write-only.

Syntax

object.**IsPropertyWriteOnly(***propertyname***)**

Parameters

object
Required. An object variable that evaluates to a catalog object property.

propertyname
Required. The name of the property for which to check the write-only status.

Remarks

If the value returned by the **IsPropertyWriteOnly** method is **True**, then you can write but not read the property value. If the value returned is **False**, you can read the property value.

See Also

IsPropertyReadOnly Property (CatalogObject)

Item Property (CatalogCollection)

Gets a specific object in a collection.

Syntax

object.**Item(***index***)**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

index
Required; **Long**. A zero-based index that specifies the position of a member of the collection. Must be a number from 0 through the value of the collection's **Count** property -1.

Key Property (CatalogObject)

Gets the value of the key of the object.

Syntax

object.**Key**

Parameters

object
Required. An object variable that evaluates to a **CatalogObject** object.

Remarks

All catalog objects have a key. The object key is a single property that uniquely identifies the object. To access a related collection using the **GetCollection** method, provide the key of the object from which you want to navigate (such as the package identifier). The following table provides the key property for each collection supported:

Collection	Key Property
Packages	ID
ComponentsInPackage	CLSID
RolesInPackage	ID
RolesForPackageComponent	ID
RolesForPackageComponentInterface	ID
UsersInRole	User
InterfacesForComponent	IID
InterfacesForRemoteComponent	IID
RemoteComponents	CLSID
MethodsForInterface	Name

MajorVersion Property (Catalog)

Returns an integer value indicating the major version number of the catalog.

Syntax

object.**MajorVersion**

Parameters

object
Required. An object variable that evaluates to a catalog object.

See Also

MinorVersion Property (Catalog)

MinorVersion Property (Catalog)

Returns an integer value indicating the minor version number of the catalog.

Syntax

object.**MinorVersion**

Parameters

object
Required. An object variable that evaluates to a catalog object.

See Also

MajorVersion Property (Catalog)

Name Property (CatalogObject)

Gets the name of an object.

Syntax

object.**Name**

Parameters

object
Required. An object variable that evaluates to a catalog object.

Remarks

All catalog objects have a name property. The following table provides the name property for each collection supported:

Collection	Name Property
Packages	Name
ComponentsInPackage	ProgID
RolesInPackage	Name
RolesForPackageComponent	Name
RolesForPackageComponentInterface	Name
UsersInRole	User
InterfacesForComponent	Name
InterfacesForRemoteComponent	Name
RemoteComponents	ProgID
MethodsForInterface	Name

Name Property (CatalogCollection)

Gets the name of the collection.

Syntax

object.**Name**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

Remarks

See the Using MTS Collections topic for a list of properties supported by each MTS collection.

Populate Method (CatalogCollection)

Fills a collection with objects from the catalog.

Syntax

object.**Populate**

Parameters

object
Required. An object variable that evaluates to the **CatalogCollection** object that you would like to fill.

Remarks

The **Populate** method reads the contents of the **CatalogCollection** object. Any changes that are still pending (such as property changes, objects added, or objects removed) are lost. See the **SaveChanges** method topic for instruction on how to preserve changes made to a **CatalogCollection** object.

PopulateByKey Method (CatalogCollection)

Populates the collection with information for the specified objects.

Syntax

object.**PopulateByKey(***aCLSIDs***)**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

aCLSIDS
Required; **Variant**. An array of object keys (VARIANTS) denoting which objects should have their information read from the catalog.

Remove Method (CatalogCollection)

Removes an item from an object given the index position.

Syntax

object.**Remove(***index***)**

Parameters

object
Required; **String**. An object variable that evaluates to a **CatalogCollection** object.

index
Required; **Long**. A zero-based index indicating the position of the object to remove.

Remarks

The object is removed from the collection and all objects with higher indices are shifted up. Note that the **Count** property of a collection changes after the **Remove** method has been called.

Call the **SaveChanges** method to save the changes made to the collection using the **Remove** method.

RemoveEnabled Property (CatalogCollection)

Returns a Boolean value indicating that you can use the **Remove** method to delete an object from the collection.

Syntax

object.**RemoveEnabled**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

Remarks

If the value returned is **True**, then you can call the **Remove** method to remove an object from the collection. If the value returned is **False**, objects cannot be removed from the collection.

SaveChanges Method (CatalogCollection)

Saves changes to a collection in the catalog, and returns an integer indicating the number of changes applied to the collection.

Syntax

object.**SaveChanges**

Parameters

object
Required. An object variable that evaluates to a **CatalogCollection** object.

Remarks

The **SaveChanges** method works exclusively on the collection on which you call it, and applies all pending changes to the catalog. If no changes are pending, then the method returns zero.

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ShutdownPackage Method (PackageUtil)

Initiates the shutting down of a package. Shutting down a package terminates that application's process.

Syntax

object.**ShutdownPackage(***bstrPackageID***)**

Parameters

object
Required. An object variable that evaluates to a **PackageUtil** object.

bstrPackageID
Required. A string variable that evaluates to the **PackageID** of a **Package CatalogObject**.

Remarks

The **ShutdownPackage** method shuts down a single package process.

See the Using MTS Collections topic for a list of the MTS collections and their properties.

Valid Property (CatalogObject)

Returns a Boolean value indicating whether all the object properties were successfully read from the catalog.

Syntax

object.**Valid**

Parameters

object
Required. An object variable that evaluates to a **CatalogObject**.

Remarks

If this property is **False** it indicates that one or more object properties could not be read from the catalog during a call to **Populate**. This property will be **True** for objects that have been added to the collection using the **Add** method.

Value Property (CatalogObject)

Gets or sets a value for an object property.

Syntax

object.**Value(***property***)** *value*

Parameters

object
Required. An object variable that evaluates to a **CatalogObject**.

property
Required. A **String** expression of any type that specifies the name of the property whose value to get or set.

value
Required. A **String** expression that specifies the value of the property to get or set.

Remarks

See the Using MTS Collections topic for a description the properties supported by the MTS administration objects.

Automating MTS Administration With Visual C++

The topics in this section describe administration interfaces supported by Microsoft Transaction Server (MTS). This reference describes the following topics:

MTS Visual C++ Error Codes
ICatalog
ICatalogObject
ICatalogCollection
IPackageUtil
IComponentUtil
IRemoteComponentUtil
IRoleAssociationUtil

The **ICatalog**, **ICatalogObject**, and **ICatalogCollection** interfaces provide top-level functionality such as creating and modifying objects. The **ICatalog** interface enables you to connect to specific servers and access collections. Call the **ICatalogCollection** interface to enumerate, create, delete, and modify objects, as well as to access related collections. The **ICatalogObject** interface is used to get and set properties on an object.

The utility interfaces (**IRemoteComponentUtil** and **IRoleAssociationUtil**) allows you to program very specific tasks for collection types, such as associating a role with a user or class of users.

MTS Visual C++ Error Codes

The following is a list of the error codes returned by methods called on the catalog collection and catalog utility interfaces.

E_INVALIDARG
Indicates one of the following:

An invalid collection or property name was entered.

An out parameter was NULL.

The value is not one of the supported values or falls outside the supported range.

The property is read-only.

The property cannot be changed after the object is created.

An invalid index was entered.

E_NOTIMPL
Object has been removed from the collection and is not supported on this collection.

E_MTS_OBJECTERRORS
Errors were encountered processing some objects or file. See the **ErrorInfo** collection for object/file-specific error codes.

E_MTS_NOUSER
User ID for user in role is not valid.

E_MTS_USERPASSWDNOTVALID
Package identity user ID and/or password are not valid

E_MTS_AUTHENICATIONLEVEL
Required authentication level (package privacy) could not be set for package updates.

E_MTS_PDFREADFAIL
An error occurred reading the package file.

E_MTS_PDFVERSION
Package file version is invalid.

E_MTS_BADPATH
Package file path is invalid.

E_MTS_PACKAGEEXISTS
Package with the same ID is already installed.

E_MTS_ROLEEXISTS
A role with the same ID is already installed. The role ID in the package file is likely corrupted.

E_MTS_CANTCOPYFILE
Errors occurred copying one or more files to the install directory.

E_MTS_INVALIDUSERIDS
One or more user IDS for roles were invalid.

E_MTS_CLSIDORIIDMISMATCH
One or more component/interface identifiers in a component DLL does not match the identifiers saved in the package file. The package file is out of date.

E_MTS_PACKDIRNOTFOUND
Package install directory is invalid due to general registry read/write errors.

E_MTS_PDFWRITEFAIL
An error occurred writing the package file.

E_MTS_NOTYPELIB
Could not find the type library for one or more components.

The following is a list of the object or file-specific error codes returned in **ErrorInfo** collections:

E_OBJECTINVALID

One or more object properties is corrupted or invalid.

E_KEYMISSING
One or more objects is not found in the catalog.

E_ALREADYINSTALLED
Component is already installed.

E_DOWNLOADFAILLED
One or more component files could not be copied to the client.

E_REMOTEINTERFACE
No interface information is available for the component. Component files could not be downloaded.

E_COREQCOMPINSTALLED
Component in the same DLL file is already installed.

E_NOREGISTRYCLSID
Component's CLSID is corrupted.

E_BADREGISTRYPROGID
Component's ProgID is corrupted.

E_DLLLOADFAILED
Component's DLL could not be loaded.

E_DLLREGISTERSERVER
DllRegisterServer method failed during component self-registration.

E_NOSERVERSHARE
No file share is available on the server to copy component files from the network path.

E_NOACCESSTOUNC
Network path registered for this component could not be accessed.

E_BADREGISTRYLIBID
Component type library ID is corrupted.

E_TREATAS
Component **TreatAs** key was found, but is not supported.

E_BADFORWARD
IID forward entry is corrupted.

E_BADIID
IID is corrupted.

E_REGISTRARFAILED
Component registrarmethod failed during component install.

E_COMFILE_DOESNOTEXIST
Component file does not exist.

E_COMFILE_LOADDLLFAIL
DLL file could not be loaded.

E_COMFILE_GETCLASSOBJ
DllGetClassObject method call failed during the DLL self-registration process.

E_COMFILE_CLASSNOTAVAIL
Class coded in the type library was not supported.

E_COMFILE_BADTLB
Type library was corrupted.

E_COMFILE_NOTINSTALLABLE
File does not contain COM components or type library information.

The following is a list of general read and write registry errors:

E_MTS_NOREGISTRYREAD
Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE

Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
Access control failure writing a registry key.

E_MTS_NOTCHANGEABLE
Changes to this object and sub-objects have been disabled.

E_MTS_NOTDELETABLE
Delete function for this object has been disabled.

E_MTS_SESSION
Server catalog version is not supported.

MTS ICatalog Interface

The **Catalog** object enables you to connect to specific servers and access collections. The **ICatalog** interface contains the following methods:

ICatalog::GetCollection
ICatalog::Connect
ICatalog::get_MajorVersion
ICatalog::get_MinorVersion

ICatalog::GetCollection

The **GetCollection** method retrieves a local collection without reading any objects from the catalog.

Syntax

HRESULT ICatalog::GetCollection(
BSTR *bstrCollName*
IDispatch ** *ppCatalogCollection***);**

Parameters

bstrCollName [in]
BSTR containing the name of the collection to retrieve from the catalog.

ppCatalogCollection [out]
Pointer to a pointer to the **CatalogCollection** object.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Invalid collection name passed as a parameter. Can also indicate that an *Out* parameter is NULL.

ICatalog::Connect

The **Connect** method connects to a remote computer and returns a *root collection*, which is bound to a remote computer.

HRESULT ICatalog::Connect(
BSTR *bstrConnectString*
IDispatch ** *ppCatalogCollection***);**

Parameters

bstrConnectString [in]
BSTR expression containing the name of a remote computer.

ppCatalogCollection [out]
Pointer to a pointer to the **CatalogCollection** object.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

Comments

A root collection serves as a starting point to locate packages, and contains neither objects nor properties. Note that you can also use the **GetCollection** method to get a top-level collection on a local server without using the **Connect** method.

ICatalog::get_MajorVersion

The **get_MajorVersion** method returns the major version number of an administration object.

HRESULT ICatalog::get_MajorVersion(
long* *retval***);**

Parameters

retval [out]
Pointer to the major version number of the MTS object.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

Comments

Call the **get_MajorVersion** method and the **get_MinorVersion** method to determine if you are using the most current version of MTS

ICatalog::get_MinorVersion

The **get_MinorVersion** method retrieves the minor version number of an MTS administration object.

HRESULT ICatalog::get_MinorVersion(
long * *retval*
);

Parameters

retval [out]
Pointer to the minor version number of the MTS object.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

Comments

Call the **get_MajorVersion** method and the **get_MinorVersion** method to determine if you are using the most current version of MTS.

MTS ICatalogCollection Interface

The **CatalogCollection** object can be used to enumerate objects, create, delete, and modify objects, and access related collections. The **ICatalogCollection** interface contains the following methods:

ICatalogCollection::get_NewEnum
ICatalogCollection::get_Item
ICatalogCollection::get_Count
ICatalogCollection::Remove
ICatalogCollection::Add
ICatalogCollection::Populate
ICatalogCollection::SaveChanges
ICatalogCollection::GetCollection
ICatalogCollection::get_Name
ICatalogCollection::get_AddEnabled
ICatalogCollection::get_RemoveEnabled
ICatalogCollection::GetUtilInterface
ICatalogCollection::get_DataStoreMajorVersion
ICatalogCollection:: get_DataStoreMinorVersion
ICatalogCollection::PopulateByKey

ICatalogCollection::get_NewEnum

The **get_NewEnum** method returns the **IEnumVariant** enumerator interface.

HRESULT ICatalogCollection::get_NewEnum(
IUnknown ** *ppEnumVariant***);**

Parameters

ppEnumVariant [out]
Pointer to a pointer to the **IEnumVariant** interface.

Return Values

S_OK
Method completed successfully.

INVALIDARG
Out parameter is NULL.

ICatalogCollection::get_Item

The **get_Item** method returns an object from the collection represented by the index.

HRESULT ICatalogCollection::get_Item(
long *1Index*
IDispatch** *ppCatalogObject***);**

Parameters

1Index [in]
Index to the object in the collection.

ppCatalogObject [out]
Pointer to a pointer to the **Catalog** object.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out of range of index.

Comments

A collection object contains zero or more items (all MTS collections are zero-based).

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::get_Count

The **get_Count** method returns the number of objects in the collection.

HRESULT ICatalogCollection::get_Count(
   long * *retval***);**

Parameters

retval [out]
   Pointer to the number of elements in the object collection.

Return Values

S_OK
   Method completed successfully.

E_INVALIDARG
   Out parameter is NULL.

Remarks

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::Add

The **Add** method adds a default object to the collection and returns a pointer to the new object.

HRESULT ICatalogCollection::Add(
IDispatch ** *ppCatalogObject***);**

Parameters

ppCatalogObject [out]
Pointer to a pointer to the new **Catalog** object.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

E_NOTIMPL
Not supported on this collection.

Remarks

To update the objects in a collection, call the **Add** method to create a new object either before or after populating a collection.

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::Populate

The **Populate** method reads the collection objects from the catalog.

HRESULT ICatalogCollection::Populate( );

Return Values

S_OK
Method completed successfully.

REGDB_E_CLASSNOTREG
The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

E_MTS_OBJECTERRORS
Collection was read but some objects were invalid. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOREGISTRYREAD
Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
Access control failure writing a registry key.

Remarks

You call the **Populate** method to fill a package collection with objects from the catalog. This method uses the **CoCreateInstance** function internally, so **CoCreateInstance** error codes are included in the **Populate** method's return values.

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::SaveChanges

The **SaveChanges** method saves changes to the collection into the catalog.

HRESULT ICatalogCollection::SaveChanges(
   long * *retval***);**

Parameters

retval [out]
   Number of changes applied to the collection.

Return Values

S_OK
   Method completed successfully.

E_INVALIDARG
   Out parameter is NULL.

E_MTS_OBJECTERRORS
   Errors were encountered processing some objects. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOUSER
   User ID is invalid.

E_MTS_USERPASSWDNOTVALID
   Package identity user ID and/or password are invalid.

E_MTS_AUTHENTICATIONLEVEL
   Required authentication level (package privacy) could not be set for package updates.

E_MTS_NOREGISTRYREAD
   Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
   Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
   Access control failure writing a registry key.

CLASS_E_NOAGGREGATION
   Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
   The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

Remarks

You must call this method after modifying any object in the collection. The **SaveChanges** method works exclusively on the collection on which it is called. This method also applies all pending changes to objects within a given collection. This method uses the **CoCreateInstance** function internally, so **CoCreateInstance** error codes are included in the **SaveChanges** method's return values.

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::GetCollection

The **GetCollection** method retrieves a collection related to a specific object. Data is not read from the catalog. See the **Populate** method topic for more information.

HRESULT ICatalogCollection::GetCollection(
   BSTR *bstrCollName*
   VARIANT *varObjectKey*
   IDispatch ** *ppCatalogCollection***);**

Parameters

bstrCollName [in]
   BSTR containing the name of the collection.

VarObjectKey [in]
   Value of the object key.

retval [out]
   Pointer to a pointer to the **CatalogCollection** object.

Return Values

S_OK
   Method completed successfully.

E_INVALIDARG
   Invalid collection name passed as a parameter. Can also indicate that an *Out* parameter is NULL.

Remarks

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::get_Name

The **get_Name** method gets the name of a collection.

HRESULT ICatalogCollection::get_Name(
VARIANT*retval***);**

Parameters

retval [out]
Pointer to the name of the collection.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

Remarks

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::get_AddEnabled

The **get_AddEnabled** method returns a value that indicates if the **Add** method is supported in this collection.

HRESULT ICatalogCollection::get_AddEnabled(
VARIANT_BOOL* *varAddEnabled*);

Parameters

VarObjectKey [out]
Boolean value indicating if the **Add** method is supported in this collection.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

Remarks

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::Remove

The **Remove** method removes an item from a collection, given the index of the item.

HRESULT ICatalogCollection::Remove(
  long *1Index***);**

Parameters

1Index [in]
  Index position of the object to remove.

Return Values

S_OK
  Method completed successfully.

E_INVALIDARG
  Invalid index was entered.

E_NOTIMPL
  Collection does not support removing objects.

Remarks

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::get_RemoveEnabled

The **get_RemoveEnabled** method returns a value that indicates if the **Remove** method is supported in this collection.

HRESULT ICatalogCollection:: get_RemoveEnabled(
VARIANT_BOOL * *boolRemoveEnabled*);

Parameters

boolRemoveEnabled [out]
Boolean value indicating if the **Remove** method is supported in this collection.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

Remarks

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::GetUtilInterface

The **GetUtilInterface** method returns a pointer to the interface of the utility object for a package, component, remote component, or role collection.

HRESULT ICatalogCollection::GetUtilInterface(
IDispatch** *ppUtil***);**

Parameters

ppUtil [out]
Pointer to a pointer to the interface on a utility object.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

E_NOTIMPL
Not supported on this collection.

Remarks

Call the **GetUtilInterface** method to program your application for specific administration tasks, such as creating a package or installing a component.

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogCollection::get_DataStoreMajorVersion

The **get_DataStoreMajorVersion** method returns the major version number of the catalog from which you get the collection.

HRESULT ICatalogCollection::get_DataStoreMajorVersion(
long * *retval***);**

Parameters

retval [out]
Pointer to a pointer to the MTS major version number.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

See Also

get_DataStoreMinorVersion

ICatalogCollection::get_DataStoreMinorVersion

The **get_DataStoreMinorVersion** method returns the minor version number of the catalog from which you get a collection.

HRESULT ICatalogCollection:: get_DataStoreMinorVersion(
long * *retval***);**

Parameters

retval [out]
Pointer to a pointer to the MTS minor version number.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

See Also

get_DataStoreMajorVersion

ICatalogCollection::PopulateByKey

The **PopulateByKey** method populates the collection with information for the specified objects.

HRESULT ICatalogCollection::PopulateByKey(
SAFEARRAY * *saKeys*);

Parameters

saKeys [in]
Pointer to a safearray of **VARIANT**s containing the CLSIDs of components for which the collection object should refresh its information.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
Out parameter is NULL.

Remarks

See the Using MTS Collections topic for a list of the MTS collections and their properties.

MTS ICatalogObject Interface

The **CatalogObject** object provides methods to get and set properties on an object. The **ICatalogObject** interface contains the following methods:

- **ICatalogObject::get_Value**
- **ICatalogObject::put_Value**
- **ICatalogObject::get_Key**
- **ICatalogObject::get_Name**
- **ICatalogObject::IsPropertyReadOnly**
- **ICatalogObject::IsPropertyWriteOnly**
- **ICatalogObject::get_Valid**

ICatalogObject::get_Value

The **get_Value** method gets a property value of an object in a collection.

HRESULT ICatalogObject::get_Value(
   BSTR *bstrPropName*
   VARIANT * *retval***);**

Parameters

bstrPropName [in]
   BSTR expression containing the name of the property.

retval [out]
   Pointer to the value of the property.

Return Values

S_OK
Method completed successfully.

E_INVALIDARG
   Invalid property name passed as a parameter. Can also indicate that an *Out* parameter is NULL.

E_NOTIMPL
   Object has been removed from the collection.

Remarks

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogObject::put_Value

The **put_Value** method sets the property value of an object in a collection.

HRESULT ICatalogObject::put_Value(
   BSTR *bstrPropName*
   VARIANT *val***);**

Parameters

bstrPropName [in]
   BSTR containing the name of the property to set.

val [in]
   Variant containing the new value for the property.

Return Values

S_OK
   Method completed successfully.

E_INVALIDARG
   Invalid property name entered. Can also indicate either the property value is not one of the supported values or falls outside the supported range, the property is read-only, or the property cannot be changed after the object is created.

E_NOTIMPL
   Object has been removed from the collection.

Remarks

See the Using MTS Collections topic for a list of the MTS collections and their properties.

ICatalogObject::get_Key

The **get_Key** method gets the value of the **Key** property.

HRESULT ICatalogObject::get_Key(
  VARIANT* *retval*);

Parameters

retval [out]
  Pointer to the value of the **Key** property.

Return Values

S_OK
  Method completed successfully.

E_INVALIDARG
  Out parameter is NULL.

E_NOTIMPL
  Object has been removed from the collection.

Comments

All MTS objects have a key. The object key is a single property that uniquely identifies the object. To create a related collection in your Package collection object, provide the key of the object from which you want to navigate (such as the package identifier). The following table provides the key property for each collection supported by the MTS Explorer.

Collection	Key Property
Packages	ID
ComponentsInPackage	CLSID
RolesInPackage	ID
RolesForPackageComponents	ID
UsersInRole	User
InterfacesForComponent	IID
InterfacesForRemoteComponent	IID

ICatalogObject::get_Name

The **get_Name** method provides the name of an object in the catalog.

HRESULT ICatalogObject::get_Name(
  VARIANT * *retval* **);**

Parameters

retval [out]
  Pointer to the name of the object.

Return Values

S_OK
  Method completed successfully.

E_INVALIDARG
  Out parameter is NULL.

E_NOTIMPL
  Object has been removed from the collection.

All MTS objects have a name property. The following table provides the name property for each collection supported by the MTS Explorer:

Collection	Name Property
Packages	Name
ComponentsInPackage	ProgID
RolesInPackage	Name
RolesForPackageComponents	Name
UsersInRole	User
InterfacesForComponent	Name
InterfacesForRemoteComponent	Name
RemoteComponents	ProgID
MethodsForInterface	Name

ICatalogObject::IsPropertyReadOnly

The **IsPropertyReadOnly** method determines if a property is read-only.

HRESULT ICatalogObject::IsPropertyReadOnly(
   BSTR *bstrPropName*
   VARIANT_BOOL * *retval***);**

Parameters

bstrPropName [in]
   BSTR containing the name of the property.

retval [out]
   Boolean indicating if the property is read-only.

Return Values

S_OK
   Method completed successfully.

E_INVALIDARG
   Invalid property name passed as a parameter. Can also indicate that an *Out* parameter is NULL.

Remarks

For more information about read-only property values and collections, see the Using MTS Collections topic.

See Also

IsPropertyWriteOnly

ICatalogObject::IsPropertyWriteOnly

The **IsPropertyWriteOnly** method indicates if a property can be written but not read.

HRESULT ICatalogObject::IsPropertyWriteOnly(
   BSTR *bstrPropName*
   VARIANT_BOOL * *retval***);**

Parameters

bstrPropName [in]
   BSTR containing the name of the property that may or may not be write-only.

retval [out]
   Boolean indicating if the property is write-only.

Return Values

S_OK
   Method completed successfully.

E_INVALIDARG
   Invalid property name passed as a parameter. Can also indicate that an *Out* parameter is NULL.

Remarks

For more information about read-only property values and collections, see the Using MTS Collections topic.

See Also

IsPropertyReadOnly

ICatalogObject::get_Valid

The **get_Valid** method determines if properties on an object were successfully read from the catalog.

HRESULT ICatalogObject::get_Valid(
  VARIANT_BOOL * *retval***);**

Parameters

retval [out]
  Boolean indicating if properties were successfully read. If this method returns **True**, all properties on an object were read from the catalog.

Return Values

S_OK
  Method completed successfully.

E_INVALIDARG
  Out parameter is NULL.

E_NOTIMPL
  Object removed from the collection.

MTS IPackageUtil Interface

The **IPackageUtil** object enables a package to be installed and exported within the **Packages** collection. The **IPackageUtil** interface contains the following methods:

IPackageUtil::InstallPackage

IPackageUtil::ExportPackage

IPackageUtil::ShutdownPackage

See the Using MTS Collections topic for a list of the MTS collections and their properties.

IPackageUtil::InstallPackage

The **InstallPackages** method installs a pre-built package.

HRESULT IPackageUtil::InstallPackage(
BSTR *bstrPackageFile*
BSTR *bstrInstallPath*
long *1Options*);

Parameters

bstrPackageFile [in]
BSTR containing the name of the package file to install.

bstrInstallPath [in]
BSTR containing component install path.

1Options [in]
Integer specifying export options. This method supports MtsExportUsers, which includes users in roles in the package file.

Return Values

S_OK
Method completed successfully.

E_MTS_OBJECTERRORS
Errors were encountered processing objects and/or files. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_PDFREADFAIL
Error occurred reading the package file.

E_MTS_PDFVERSION
Package file version is invalid.

E_MTS_BADPATH
Package file path is invalid.

E_MTS_PACKAGEEXISTS
Package with the same ID is already installed.

E_MTS_ROLEEXISTS
Role with the same ID is already installed. The role ID in the package file is likely corrupted.

E_MTS_CANTCOPYFILE
Errors occurred copying one or more files to the install directory.

E_MTS_INVALIDUSERIDS
One or more user IDs for roles were invalid.

E_MTS_CLSIDORIIDMISMATCH
One or more component/interface identifiers in a component DLL do not match the identifiers saved in the package file. The package file is likely out of date.

E_MTS_PACKDIRNOTFOUND
The package install directory is invalid.

E_MTS_NOREGISTRYREAD
Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not

installed properly on the target computer.

Remarks

Because this method uses the **CoCreateInstance** function internally, **CoCreateInstance** error codes are included in the **InstallPackage** method's return values.

IPackageUtil::ExportPackage

The **ExportPackage** method exports a package according to its package identifier.

HRESULT IPackageUtil::ExportPackage(
BSTR *bstrPackageID*
BSTR *bstrPackageFile*
long *1Options*);

Parameters

bstrPackageID [in]
BSTR containing the unique identifier of the package to export.

bstrPackageFile [in]
BSTR containing the name of the package file to export.

1Options [in]
Either zero (for no option selected) or MtsExportUsers, which includes users in roles in the package file.

Return Values

S_OK
Method completed successfully.

E_MTS_OBJECTERRORS
Errors were encountered processing objects and/or files. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_PDFWRITEFAIL
Error occurred writing the package file.

E_MTS_NOTYPELIB
Could not find the type library for one or more components.

E_MTS_NOREGISTRYREAD
Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

Remarks

Because this method uses the **CoCreateInstance** function internally, **CoCreateInstance** error codes are included in the **ExportPackage** method's return values.

IPackageUtil::ShutdownPackage

The **ShutdownPackage** method shuts down a single package, thereby terminating that application process.

HRESULT IPackageUtil::ShutdownPackage(
BSTR *bstrPackageID*

Parameters

bstrPackageID [in]
BSTR containing the unique identifier of the package to shut down.

Return Values

S_OK
Method completed successfully.

E_MTS_OBJECTERRORS
Errors were encountered processing objects and/or files. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_PDFWRITEFAIL
Error occurred writing the package file.

E_MTS_NOTYPELIB
Could not find the type library for one or more components.

E_MTS_NOREGISTRYREAD
Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

MTS IComponentUtil Interface

The **IComponentUtil** object provides methods to install a component in a specific collection and to import components registered as an in-proc server. The **IComponentUtil** interface contains the following methods:

- **IComponentUtil::InstallComponent**
- **IComponentUtil::ImportComponent**
- **IComponentUtil::ImportComponentByName**
- **IComponentUtil::GetCLSIDs**

IComponentUtil::InstallComponent

The **InstallComponent** method installs a component.

HRESULT IComponentUtil::InstallComponent(
BSTR *bstrDLLFile*
BSTR *bstrTypelibFile*
BSTR *bstrProxyStubDLL***);**

Parameters

bstrDLLFile [in]
BSTR containing the name of the DLL file providing the components to install.

bstrTypelibFile [in]
BSTR containing the name of the external type library file. If the type library file is embedded in the DLL, pass in an empty string for this parameter.

bstrProxyStubDLL [in]
BSTR containing the name of the proxy-stub DLL file. If there is no proxy-stub DLL associated with the component, pass in an empty string for this parameter.

Return Values

S_OK
Method completed successfully.

E_MTS_OBJECTERRORS
Errors were encountered processing components and/or files. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOREGISTRYREAD
Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

Remarks

Because this method uses the **CoCreateInstance** function internally, **CoCreateInstance** error codes are included in the **InstallComponent** method's return values.

IComponentUtil::ImportComponent

The **ImportComponent** method imports a component that is already registered as an in-process (in-proc) server.

HRESULT IComponentUtil::ImportComponent(
  BSTR *bstrCLSID***);**

Parameters

bstrCLSID [in]
  BSTR containing the CLSID of the component to import.

Return Values

S_OK
  Method completed successfully.

E_MTS_OBJECTERRORS
  Errors were encountered processing components and/or files. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOREGISTRYREAD
  Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
  Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
  Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
  The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

Remarks

Because this method uses the **CoCreateInstance** function internally, **CoCreateInstance** error codes are included in the **ImportComponent** method's return values.

See Also

ImportComponentByName

IComponentUtil::ImportComponentByName

The **ImportComponentByName** method imports a component that is already registered as an in-process (in-proc) server. This method uses the component's programmatic identifier (ProgID) for the import procedure.

HRESULT IComponentUtil::ImportComponentByName(
  BSTR *bstrProgID***);**

Parameters

bstrProgID [in]
  BSTR containing the **ProgID** of the component to import.

Return Values

S_OK
  Method completed successfully.

E_MTS_OBJECTERRORS
  Errors were encountered processing components and/or files. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOREGISTRYREAD
  Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
  Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
  Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
  The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

Remarks

Because this method uses the **CoCreateInstance** function internally, **CoCreateInstance** error codes are included in the **ImportComponentByName** method's return values.

See Also

ImportComponent

IComponentUtil::GetCLSIDs

The **GetCLSIDs** method fills an array with the installable component CLSIDs from a DLL and/or type library.

HRESULT IComponentUtil::GetCLSIDs(
BSTR *bstrDLLFile*
BSTR *bstrTypeLibFile*
SAFEARRAY** *ppsaCLSIDs)*

Parameters

bstrDLLFile [in]
BSTR containing the name of the DLL file providing the components to check for allowable installation.

bstrTypeLibFile [in]
BSTR containing the name of the external type library file to check for installable components.

ppsaCLSIDs [out]
Pointer to a pointer to a SAFEARRAY containing VARIANTS which contain the CLSIDs of installable components in the given DLL and/or type library.

Return Values

S_OK
Method completed successfully.

E_MTS_OBJECTERRORS
Errors were encountered processing components and/or files. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOREGISTRYREAD
Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

MTS IRemoteComponentUtil Interface

You can use the **IRemoteComponentUtil** object to program your application to pull remote components from a package on a remote server. The **IRemoteComponentUtil** interface contains the following methods:

IRemoteComponentUtil::InstallRemoteComponent
IRemoteComponentUtil::InstallRemoteComponentByName

IRemoteComponentUtil::InstallRemoteComponent

The **InstallRemoteComponent** method pulls a component to install from a package on a remote server.

HRESULT IRemoteComponentUtil::InstallRemoteComponent(
BSTR *bstrServer*
BSTR *bstrPackageID*
BSTR *bstrCLSID***);**

Parameters

bstrServer [in]
BSTR containing the name of the remote server from which to pull the component to install.

PackageID [in]
BSTR containing the identifier of the package containing the remote component.

bstrCLSID [in]
BSTR containing the class identifier (CLSID) of the remote component.

Return Values

S_OK
Method returned successfully.

E_MTS_OBJECTERRORS
Errors were encountered processing components and/or files. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOREGISTRYREAD
Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

Remarks

Because this method uses the **CoCreateInstance** function internally, **CoCreateInstance** error codes are included in the **InstallRemoteComponent** method's return values.

See Also

InstallRemoteComponentByName

IRemoteComponentUtil::InstallRemoteComponentByName

The **InstallRemoteComponentByName** method pulls remote components from the package on a remote server and installs the component by package name and programmatic ID (ProgID).

HRESULT IRemoteComponentUtil::InstallRemoteComponentByName(
BSTR *bstrSever*
BSTR *PackageName*
BSTR *bstrProgID***);**

Parameters

bstrSever [in]
BSTR containing the name of the remote server from which to pull the component to install.

PackageName [in]
BSTR containing the name of the package containing the remote component.

bstrProgID [in]
BSTR containing the ProgID of the component to install.

Return Values

S_OK
Method completed successfully.

E_MTS_OBJECTERRORS
Errors were encountered processing objects. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOREGISTRYREAD
Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

Remarks

Because this method uses the **CoCreateInstance** function internally, **CoCreateInstance** error codes are included in the **InstallRemoteComponentByName** method's return values.

See Also

InstallRemoteComponent

MTS IRoleAssociationUtil Interface

Call methods on the **IRoleAssociationUtil** object to associate roles with a component or component interface. The **IRoleAssociationUtil** interface contains the following methods:

IRoleAssociationUtil::AssociateRole

IRoleAssociationUtil::AssociateRoleByName

IRoleAssociationUtil::AssociateRole

The **AssociateRole** method associates a role with a component or component interface.

HRESULT IRoleAssociationUtil::AssociateRole(
  BSTR *bstrRoleID*
  );

Parameters

bstrRoleID [in]
  BSTR containing the ID of the role to associate with a component or component interface.

Return Values

S_OK
  Method returned successfully.

E_MTS_OBJECTERRORS
  Errors were encountered processing objects. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOREGISTRYREAD
  Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
  Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
  Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
  The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

Remarks

Because this method uses the **CoCreateInstance** function internally, **CoCreateInstance** error codes are included in the **AssociateRole** method's return values.

See Also

AssociateRoleByName

IRoleAssociationUtil::AssociateRoleByName

The **AssociateRoleByName** method associates a role by its name with a specified component or component interface.

HRESULT IRoleAssociationUtil::AssociateRoleByName(
  BSTR *bstrRoleName***);**

Parameters

bstrRoleName [in]
  BSTR containing the name of the role to associate with a component or component interface.

Return Values

S_OK
  Method returned successfully.

E_MTS_OBJECTERRORS
  Errors were encountered processing components and/or files. See the **ErrorInfo** collection for object-specific error codes.

E_MTS_NOREGISTRYREAD
  Access control failure reading a registry key.

E_MTS_NOREGISTRYWRITE
  Access control failure writing a registry key.

E_MTS_NOREGISTRYREPAIR
  Access control failure writing a registry key.

REGDB_E_CLASSNOTREG
  The **MTXCatEx.CatalogServer.1** component is not registered on the target computer. MTS is not installed properly on the target computer.

Remarks

Because this method uses the **CoCreateInstance** function internally, **CoCreateInstance** error codes are included in the **AssociateRoleByName** method's return values.

See Also

AssociateRole

GetObjectContext

Visual Basic **GetObjectContext** Function

Visual C++ **GetObjectContext** Function

Visual J++ **MTx.GetObjectContext** Method

Context.getObjectContext Method

SafeRef

Visual Basic	**SafeRef** Function
Visual C++	**SafeRef** Function
Visual J++	**Context.safeRef** Method
	MTx.SafeRef Method

IObjectContext Interface, ObjectContext Object, Context Class

Visual Basic	**ObjectContext** Object
Visual C++	**IObjectContext** Interface
Visual J++	**Context** Class
	IObjectContext Interface

SetAbort Method

Visual Basic **SetAbort** Method

Visual C++ **IObjectContext::SetAbort** Method

Visual Basic **Context.setAbort** Method

IObjectContext.SetAbort Method

SetComplete Method

Visual Basic	**SetComplete** Method
Visual C++	**IObjectContext::SetComplete** Method
Visual J++	**Context.setComplete** Method
	IObjectContext.SetComplete Method

EnableCommit Method

Visual Basic	**EnableCommit** Method
Visual C++	**IObjectContext::EnableCommit** Method
Visual J++	**Context.enableCommit** Method
	IObjectContext.EnableCommit Method

DisableCommit Method

Visual Basic	**DisableCommit** Method
Visual C++	**IObjectContext::DisableCommit** Method
Visual J++	**Context.disableCommit** Method
	IObjectContext.DisableCommit Method

CreateInstance Method

Visual Basic	**CreateInstance** Method
Visual C++	**IObjectContext::CreateInstance** Method
Visual J++	**Context.createObject** Method
	IObjectContext.CreateInstance Method

IsInTransaction Method

Visual Basic	**IsInTransaction** Method
Visual C++	**IObjectContext::IsInTransaction** Method
Visual J++	**Context.isInTransaction** Method
	IObjectContext.IsInTransaction Method

IsCallerInRole Method

Visual Basic	**IsCallerInRole** Method
Visual C++	**IObjectContext::IsCallerInRole** Method
Visual J++	**Context.isCallerInRole** Method
	IObjectContext.IsCallerInRole Method

IsSecurityEnabled Method

Visual Basic	**IsSecurityEnabled** Method
Visual C++	**IObjectContext::IsSecurityEnabled** Method
Visual J++	**Context.isSecurityEnabled** Method
	IObjectContext.IsSecurityEnabled Method

ITransactionContextEx Interface, TransactionContext Object

Visual Basic **TransactionContext** Object

Visual C++ **ITransactionContextEx** Interface

Visual J++ **ITransactionContextEx** Interface

Abort Method

Visual Basic	**Abort** Method
Visual C++	**ITransactionContextEx::Abort** Method
Visual J++	**ITransactionContextEx.Abort** Method

Commit Method

Visual Basic	**Commit** Method
Visual C++	**ITransactionContextEx::Commit** Method
Visual J++	**ITransactionContextEx.Commit** Method

IObjectControl Interface

Visual Basic **ObjectControl** Interface

Visual C++ **IObjectControl** Interface

Visual J++ **IObjectControl** Interface

Activate Method

Visual Basic	**Activate** Method
Visual C++	**IObjectControl::Activate** Method
Visual J++	**IObjectControl.Activate** Method

CanBePooled Method

Visual Basic **CanBePooled** Method

Visual C++ **IObjectControl::CanBePooled** Method

Visual J++ **IObjectControl.CanBePooled** Method

Deactivate Method

Visual Basic	**Deactivate** Method
Visual C++	**IObjectControl::Deactivate** Method
Visual J++	**IObjectControl.Deactivate** Method

ISharedProperty Interface, SharedProperty Object

Visual Basic **SharedProperty** Object

Visual C++ **ISharedProperty** Interface

Visual J++ **ISharedProperty** Interface

Value

Visual Basic	**Value** Property
Visual C++	**ISharedProperty::get_Value** Method
Visual C++	**ISharedProperty::put_Value** Method
Visual J++	**ISharedProperty.getValue** Method
Visual J++	**ISharedProperty.putValue** Method

ISharedPropertyGroup Interface, SharedPropertyGroup Object

Visual Basic **SharedPropertyGroup** Object

Visual C++ **ISharedPropertyGroup** Interface

Visual J++ **ISharedPropertyGroup** Interface

CreateProperty Method

Visual Basic **CreateProperty** Method

Visual C++ **ISharedPropertyGroup::CreateProperty** Method

Visual J++ **ISharedPropertyGroup.CreateProperty** Method

CreatePropertyByPosition Method

Visual Basic **CreatePropertyByPosition** Method

Visual C++ **ISharedPropertyGroup::CreatePropertyByPosition** Method

Visual J++ **ISharedPropertyGroup.CreatePropertyByPosition** Method

Property

Visual Basic **Property** Property

Visual C++ **ISharedPropertyGroup::get_Property** Method

Visual J++ **ISharedPropertyGroup.getProperty** Method

Context.getProperty Method

Context.getPropertyNames Method

PropertyByPosition

Visual Basic	**PropertyByPosition** Property
Visual C++	**ISharedPropertyGroup::get_PropertyByPosition** Method
Visual J++	**ISharedPropertyGroup.getPropertyByPosition** Method

ISharedPropertyGroupManager Interface, SharedPropertyGroupManager Object

Visual Basic **SharedPropertyGroupManager** Object

Visual C++ **ISharedPropertyGroupManager** Interface

Visual J++ **ISharedPropertyGroupManager** Interface

CreatePropertyGroup Method

Visual Basic **CreatePropertyGroup** Method

Visual C++ **ISharedPropertyGroupManager::CreatePropertyGroup** Method

Visual J++ **ISharedPropertyGroupManager.CreatePropertyGroup** Method

Group

Visual Basic	**Group** Property
Visual C++	**ISharedPropertyGroupManager::get_Group** Method
Visual J++	**ISharedPropertyGroupManager.getGroup** Method

get_NewEnum, get__NewEnum Methods

Visual C++ **ISharedPropertyGroupManager::get__NewEnum** Method

Visual J++ **ISharedPropertyGroupManager.get_NewEnum** Method

IGetContextProperties Interface

Visual C++ **IGetContextProperties** Interface

Visual J++ **IGetContextProperties** Interface

Count Method

Visual Basic **Count** Method

Visual C++ **Count** Method

Visual J++ **Count** Method

EnumNames Method

Visual C++ **EnumNames** Method

Visual J++ **EnumNames** Method

GetProperty Method

Visual C++ **GetProperty** Method

Visual J++ **GetProperty** Method

ISecurityProperty Interface

Visual Basic **SecurityProperty** Object

Visual C++ **ISecurityProperty** Interface

GetDirectCallerName Method

Visual Basic **GetDirectCallerName** Method

Visual C++ **GetDirectCallerSID** Method

Visual J++ **Context.getDirectCallerName** Method

GetDirectCreatorName Method

Visual Basic **GetDirectCreatorName** Method

Visual C++ **GetDirectCreatorSID** Method

Visual J++ **Context.getDirectCreatorName** Method

GetOriginalCallerName Method

Visual Basic **GetOriginalCallerName** Method

Visual C++ **GetOriginalCallerSID** Method

Visual J++ **Context.getOriginalCallerName** Method

GetOriginalCreatorName Method

Visual Basic **GetOriginalCreatorName** Method

Visual C++ **GetOriginalCreatorSID** Method

Visual J++ **Context.getOriginalCreatorName** Method

Post Method, Step1 (Visual Basic)

```
Public Function Post(ByRef lngAccount As Long, _
    ByRef lngAmount As Long) As String

    On Error GoTo ErrorHandler
    Post = "Hello from Account!!!"
    Exit Function
' Return the error message indicating that
' an error occurred.
ErrorHandler:
    Err.Raise Err.Number, "Bank.Account.Post", _
        Err.Description
End Function
```

Post Method, Step2 (Visual Basic)

```
Public Function Post(ByVal lngAccountNo As Long, _
    ByVal lngAmount As Long) As String

    Dim strResult As String

    On Error GoTo ErrorHandler

    ' obtain the ADO environment and connection
    Dim adoConn As New ADODB.Connection
    Dim varRows As Variant

    adoConn.Open strConnect

    On Error GoTo ErrorCreateTable

    ' update the balance
    Dim strSQL As String
    strSQL = "UPDATE Account SET Balance = Balance + "_
        + Str$(lngAmount) + " WHERE AccountNo = "
        + Str$(lngAccountNo)

TryAgain:
    adoConn.Execute strSQL, varRows

    ' if anything else happens
    On Error GoTo ErrorHandler

    ' get resulting balance which may have been
    ' further updated via triggers
    strSQL = "SELECT Balance FROM Account " _
        + "WHERE AccountNo = " + Str$(lngAccountNo)

    Dim adoRS As ADODB.Recordset
    Set adoRS = adoConn.Execute(strSQL)
    If adoRS.EOF Then
        Err.Raise Number:=APP_ERROR, _
            Description:="Error. Account " _
            + Str$(lngAccountNo) + " not on file."
    End If

    Dim lngBalance As Long
    lngBalance = adoRS.Fields("Balance").Value

    ' check if account is overdrawn
    If (lngBalance) < 0 Then
        Err.Raise Number:=APP_ERROR, _
            Description:="Error. Account " _
            + Str$(lngAccountNo) _
            + " would be overdrawn by " _
            + Str$(lngBalance) + ". Balance is still "
            + Str$(lngBalance - lngAmount) + "."
    Else
        If lngAmount < 0 Then
            strResult = strResult _
                & "Debit from account "
```

```
                & lngAccountNo & ", "
        Else
            strResult = strResult _
                & "Credit to account "
                & lngAccountNo & ", "
        End If
        strResult = strResult + "balance is $"
            & Str$(lngBalance) & ". (VB)"
    End If

    ' cleanup
    Set adoRS = Nothing
    Set adoConn = Nothing

    Post = strResult

Exit Function

ErrorCreateTable:
    On Error GoTo ErrorHandler

    ' create the account table
    Dim objCreateTable As CreateTable
    Set objCreateTable = _
    GetObjectContext.CreateInstance("Bank.CreateTable")
    objCreateTable.CreateAccount

    GoTo TryAgain

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

    Post = ""         ' indicate that an error occurred
    Err.Raise Err.Number, "Bank.Accout.Post", _
        Err.Description

End Function
```

Post Method, Step3 (Visual Basic)

```
Public Function Post(ByVal lngAccountNo As Long, _
    ByVal lngAmount As Long) As String

    Dim strResult As String

    On Error GoTo ErrorHandler

    ' obtain the ADO environment and connection
    Dim adoConn As New ADODB.Connection
    Dim varRows As Variant

    adoConn.Open strConnect

    On Error GoTo ErrorCreateTable

    ' update the balance
    Dim strSQL As String
    strSQL = "UPDATE Account SET Balance = Balance + "_
        + Str$(lngAmount) + " WHERE AccountNo = "
        + Str$(lngAccountNo)

TryAgain:
    adoConn.Execute strSQL, varRows

    ' if anything else happens
    On Error GoTo ErrorHandler

    ' get resulting balance which may have been
    ' further updated via triggers
    strSQL = "SELECT Balance FROM Account " _
        + "WHERE AccountNo = " + Str$(lngAccountNo)

    Dim adoRS As ADODB.Recordset
    Set adoRS = adoConn.Execute(strSQL)
    If adoRS.EOF Then
        Err.Raise Number:=APP_ERROR, _
            Description:="Error. Account " _
            + Str$(lngAccountNo) + " not on file."
    End If

    Dim lngBalance As Long
    lngBalance = adoRS.Fields("Balance").Value

    ' check if account is overdrawn
    If (lngBalance) < 0 Then
        Err.Raise Number:=APP_ERROR, _
            Description:="Error. Account " _
            + Str$(lngAccountNo) _
            + " would be overdrawn by " _
            + Str$(lngBalance) + ". Balance is still "
            + Str$(lngBalance - lngAmount) + "."
    Else
        If lngAmount < 0 Then
            strResult = strResult _
                & "Debit from account "
```

```
                & lngAccountNo & ", "
        Else
            strResult = strResult _
                & "Credit to account "
                & lngAccountNo & ", "
        End If
        strResult = strResult + "balance is $"
            & Str$(lngBalance) & ". (VB)"
    End If

    ' cleanup
    Set adoRS = Nothing
    Set adoConn = Nothing

    ' we are finished and happy
    GetObjectContext.SetComplete

    Post = strResult

Exit Function

ErrorCreateTable:
    On Error GoTo ErrorHandler

    ' create the account table
    Dim objCreateTable As CreateTable
    Set objCreateTable = _
    GetObjectContext.CreateInstance("Bank.CreateTable")
    objCreateTable.CreateAccount

    GoTo TryAgain

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

    GetObjectContext.SetAbort          ' we are unhappy

    Post = ""         ' indicate that an error occurred
    Err.Raise Err.Number, "Bank.Accout.Post", _
        Err.Description

End Function
```

Perform Method, Step4 (Visual Basic)

```
Public Function Perform(ByVal lngPrimeAccount As Long,_
    ByVal lngSecondAccount As Long, ByVal lngAmount _
    As Long, ByVal lngTranType As Long) As String

    Dim strResult As String

    On Error GoTo ErrorHandler

    ' create the account object using our context
    Dim objAccount As Bank.Account
    Set objAccount = _
        GetObjectContext.CreateInstance("Bank.Account")

    If objAccount Is Nothing Then
        Err.Raise ERROR_NUMBER, _
        Description:="Could not create account object"
    End If

    ' call the post function based on the
    ' transaction type
    Select Case lngTranType

        Case 1
            strResult = objAccount.Post(lngPrimeAccount, 0 - lngAmount)
            If strResult = "" Then
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult
            End If

        Case 2
            strResult = objAccount.Post(lngPrimeAccount, lngAmount)
            If strResult = "" Then
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult
            End If

        Case 3
            Dim strResult1 As String, strResult2 As String
            ' do the credit
            strResult1 = objAccount.Post(lngSecondAccount, lngAmount)
            If strResult1 = "" Then
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult1
            Else
                ' then do the debit
                strResult2 = objAccount.Post(lngPrimeAccount, 0 -
lngAmount)
                If strResult2 = "" Then
                ' debit failed
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult2
                Else
                    strResult = strResult1 + "  " + strResult2
                End If
            End If
```

```
        Case Else
            Err.Raise ERROR_NUMBER, _
                Description:="Invalid Transaction Type"

    End Select

    ' we are finished and happy
    GetObjectContext.SetComplete

    Perform = strResult

    Exit Function

ErrorHandler:

    GetObjectContext.SetAbort          ' we are unhappy

    Perform = ""       ' indicate that an error occured

    Err.Raise Err.Number, "Bank.MoveMoney.Perform", _
        Err.Description

End Function
```

GetNextReceipt Method, Step5 (Visual Basic)

```
Public Function GetNextReceipt() As Long

    On Error GoTo ErrorHandler

    ' If Shared property does not already exist
    ' it will be initialized
    Dim spmMgr As SharedPropertyGroupManager
    Set spmMgr = CreateObject("MTxSpm.SharedPropertyGroupManager.1")

    Dim spmGroup As SharedPropertyGroup
    Dim bResult As Boolean
    Set spmGroup = _
        spmMgr.CreatePropertyGroup("Receipt", _
        LockMethod, Process, bResult)

    Dim spmPropNextReceipt As SharedProperty
    Set spmPropNextReceipt = _
        spmGroup.CreateProperty("Next", bResult)

    ' Set the initial value of the Shared Property to
    ' 0 if the Shared Property didn't already exist.
    ' This is not entirely necessary but demonstrates
    ' how to initialize a value.
    If bResult = False Then
        spmPropNextReceipt.Value = 0
    End If

    ' Get the next receipt number and update property
    spmPropNextReceipt.Value = spmPropNextReceipt.Value + 1

    ' we are finished and happy
    GetObjectContext.SetComplete

    GetNextReceipt = spmPropNextReceipt.Value

    Exit Function

ErrorHandler:
    GetObjectContext.SetAbort          ' we are unhappy

    ' indicate that an error occured
    GetNextReceipt = -1

    Err.Raise Err.Number, _
        "Bank.GetReceipt.GetNextReceipt", _
        Err.Description

End Function
```

Perform Method, Step5 (Visual Basic)

```
Public Function Perform(ByVal lngPrimeAccount As Long,_
    ByVal lngSecondAccount As Long, ByVal lngAmount _
    As Long, ByVal lngTranType As Long) As String

    Dim strResult As String

    On Error GoTo ErrorHandler

    ' create the account object using our context
    Dim objAccount As Bank.Account
    Set objAccount = _
        GetObjectContext.CreateInstance("Bank.Account")

    If objAccount Is Nothing Then
        Err.Raise ERROR_NUMBER, _
        Description:="Could not create account object"
    End If

    ' call the post function based on the
    ' transaction type
    Select Case lngTranType

        Case 1
            strResult = objAccount.Post(lngPrimeAccount, 0 - lngAmount)
            If strResult = "" Then
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult
            End If

        Case 2
            strResult = objAccount.Post(lngPrimeAccount, lngAmount)
            If strResult = "" Then
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult
            End If

        Case 3
            Dim strResult1 As String, strResult2 As String
            ' do the credit
            strResult1 = objAccount.Post(lngSecondAccount, lngAmount)
            If strResult1 = "" Then
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult1
            Else
                ' then do the debit
                strResult2 = objAccount.Post(lngPrimeAccount, 0 -
lngAmount)
                If strResult2 = "" Then
                ' debit failed
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult2
                Else
                    strResult = strResult1 + "  " + strResult2
                End If
            End If
```

```
        Case Else
            Err.Raise ERROR_NUMBER, _
                Description:="Invalid Transaction Type"

    End Select

    ' Get Receipt Number for the transaction
    Dim objReceiptNo As Bank.GetReceipt
    Dim lngReceiptNo As Long

    Set objReceiptNo = GetObjectContext.CreateInstance("Bank.GetReceipt")
    lngReceiptNo = objReceiptNo.GetNextReceipt
    If lngReceiptNo > 0 Then
        strResult = strResult & "; Receipt No: " _
            & Str$(lngReceiptNo)
    End If

    ' we are finished and happy
    GetObjectContext.SetComplete

    Perform = strResult

    Exit Function

ErrorHandler:

    GetObjectContext.SetAbort          ' we are unhappy

    Perform = ""       ' indicate that an error occured

    Err.Raise Err.Number, "Bank.MoveMoney.Perform", _
        Err.Description

End Function
```

StatefulPerform Method, Step6 (Visual Basic)

```
Public PrimeAccount As Long
Public SecondAccount As Long

Public Function StatefulPerform(ByVal lngAmount _
    As Long, ByVal lngTranType As Long) As String
    StatefulPerform = Perform(PrimeAccount, _
        SecondAccount, lngAmount, lngTranType)
End Function
```

Update Method, Step7 (Visual Basic)

```
Public Function Update() As Long

    On Error GoTo ErrorHandler

    ' get result set and then update table
    ' with new receipt number
    Dim adoConn As New ADODB.Connection
    Dim adoRsReceipt As ADODB.Recordset
    Dim lngNextReceipt As Long
    Dim strSQL As String

    strSQL = "Update Receipt set NextReceipt = NextReceipt + 100"

    adoConn.Open strConnect

    ' Assume that if there is an ado error then
    ' the receipt table does not exist
    On Error GoTo ErrorCreateTable

TryAgain:

    adoConn.Execute strSQL

    strSQL = "Select NextReceipt from Receipt"
    Set adoRsReceipt = adoConn.Execute(strSQL)
    lngNextReceipt = adoRsReceipt!NextReceipt

    Set adoConn = Nothing
    Set adoRsReceipt = Nothing

    ' we are finished and happy
    GetObjectContext.SetComplete

    Update = lngNextReceipt

    Exit Function

ErrorCreateTable:

    On Error GoTo ErrorHandler

    ' create the receipt table
    Dim objCreateTable As CreateTable
    Set objCreateTable = CreateObject("Bank.CreateTable")
    objCreateTable.CreateReceipt

    GoTo TryAgain

ErrorHandler:

    If Not adoConn Is Nothing Then
        Set adoConn = Nothing
    End If
    If Not adoRsReceipt Is Nothing Then
        Set adoRsReceipt = Nothing
```

```
    End If
    GetObjectContext.SetAbort      ' we are unhappy
    Update = -1        ' indicate that an error occured
    Err.Raise Err.Number, "Bank.UpdateReceipt.Update", Err.Description
End Function
```

GetNextReceipt Method, Step7 (Visual Basic)

```
Public Function GetNextReceipt() As Long

    On Error GoTo ErrorHandler

    ' If Shared property does not already exist
    ' it will be initialized
    Dim spmMgr As SharedPropertyGroupManager
    Set spmMgr = CreateObject("MTxSpm.SharedPropertyGroupManager.1")

    Dim spmGroup As SharedPropertyGroup
    Dim bResult As Boolean
    Set spmGroup = _
        spmMgr.CreatePropertyGroup("Receipt", _
        LockMethod, Process, bResult)

    Dim spmPropNextReceipt As SharedProperty
    Set spmPropNextReceipt = _
        spmGroup.CreateProperty("Next", bResult)

    ' Set the initial value of the Shared Property to
    ' 0 if the Shared Property didn't already exist.
    ' This is not entirely necessary but demonstrates
    ' how to initialize a value.
    If bResult = False Then
        spmPropNextReceipt.Value = 0
    End If

    Dim spmPropMaxNum As SharedProperty
    Set spmPropMaxNum = spmGroup.CreateProperty("MaxNum", bResult)

    Dim objReceiptUpdate As Bank.UpdateReceipt
    If spmPropNextReceipt.Value >= spmPropMaxNum.Value Then
        Set objReceiptUpdate =
GetObjectContext.CreateInstance("Bank.UpdateReceipt")
        spmPropNextReceipt.Value = objReceiptUpdate.Update
        spmPropMaxNum.Value = spmPropNextReceipt.Value + 100
    End If

    ' Get the next receipt number and update property
    spmPropNextReceipt.Value = spmPropNextReceipt.Value + 1

    ' we are finished and happy
    GetObjectContext.SetComplete

    GetNextReceipt = spmPropNextReceipt.Value

    Exit Function

ErrorHandler:
    GetObjectContext.SetAbort          ' we are unhappy

    ' indicate that an error occured
    GetNextReceipt = -1

    Err.Raise Err.Number, "Bank.GetReceipt.GetNextReceipt", Err.Description
```

```
End Function
```

Perform Method, Step8 (Visual Basic)

```
Public Function Perform(ByVal lngPrimeAccount As Long,_
    ByVal lngSecondAccount As Long, ByVal lngAmount _
    As Long, ByVal lngTranType As Long) As String

    Dim strResult As String

    On Error GoTo ErrorHandler

    ' check for security
    If (lngAmount > 500 Or lngAmount < -500) Then
        If Not GetObjectContext.IsCallerInRole("Managers") Then
            Err.Raise Number:=APP_ERROR, _
            Description:="Need 'Managers' role for amounts over $500"
        End If
    End If

    ' create the account object using our context
    Dim objAccount As Bank.Account
    Set objAccount = _
        GetObjectContext.CreateInstance("Bank.Account")

    If objAccount Is Nothing Then
        Err.Raise ERROR_NUMBER, _
        Description:="Could not create account object"
    End If

    ' call the post function based on the
    ' transaction type
    Select Case lngTranType

        Case 1
            strResult = objAccount.Post(lngPrimeAccount, 0 - lngAmount)
            If strResult = "" Then
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult
            End If

        Case 2
            strResult = objAccount.Post(lngPrimeAccount, lngAmount)
            If strResult = "" Then
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult
            End If

        Case 3
            Dim strResult1 As String, strResult2 As String
            ' do the credit
            strResult1 = objAccount.Post(lngSecondAccount, lngAmount)
            If strResult1 = "" Then
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult1
            Else
                ' then do the debit
                strResult2 = objAccount.Post(lngPrimeAccount, 0 -
lngAmount)
```

```
                If strResult2 = "" Then
                ' debit failed
                Err.Raise ERROR_NUMBER, _
                    Description:=strResult2
                Else
                    strResult = strResult1 + "  " + strResult2
                End If
            End If

        Case Else
            Err.Raise ERROR_NUMBER, _
                Description:="Invalid Transaction Type"

    End Select

    ' Get Receipt Number for the transaction
    Dim objReceiptNo As Bank.GetReceipt
    Dim lngReceiptNo As Long

    Set objReceiptNo = GetObjectContext.CreateInstance("Bank.GetReceipt")
    lngReceiptNo = objReceiptNo.GetNextReceipt
    If lngReceiptNo > 0 Then
        strResult = strResult & "; Receipt No: " _
            & Str$(lngReceiptNo)
    End If

    ' we are finished and happy
    GetObjectContext.SetComplete

    Perform = strResult

    Exit Function

ErrorHandler:

    GetObjectContext.SetAbort          ' we are unhappy

    Perform = ""       ' indicate that an error occured

    Err.Raise Err.Number, "Bank.MoveMoney.Perform", _
        Err.Description

End Function
```

Post Method, Step8 (Visual Basic)

```
Public Function Post(ByVal lngAccountNo As Long, _
    ByVal lngAmount As Long) As String

    Dim strResult As String

    On Error GoTo ErrorHandler

    ' check for security
    If (lngAmount > 500 Or lngAmount < -500) Then
        If Not GetObjectContext.IsCallerInRole("Managers") Then
            Err.Raise Number:=APP_ERROR, _
            Description:="Need 'Managers' role for amounts over $500"
        End If
    End If

    ' obtain the ADO environment and connection
    Dim adoConn As New ADODB.Connection
    Dim varRows As Variant

    adoConn.Open strConnect

    On Error GoTo ErrorCreateTable

    ' update the balance
    Dim strSQL As String
    strSQL = "UPDATE Account SET Balance = Balance + "_
        + Str$(lngAmount) + " WHERE AccountNo = "
        + Str$(lngAccountNo)

TryAgain:
    adoConn.Execute strSQL, varRows

    ' if anything else happens
    On Error GoTo ErrorHandler

    ' get resulting balance which may have been
    ' further updated via triggers
    strSQL = "SELECT Balance FROM Account " _
        + "WHERE AccountNo = " + Str$(lngAccountNo)

    Dim adoRS As ADODB.Recordset
    Set adoRS = adoConn.Execute(strSQL)
    If adoRS.EOF Then
        Err.Raise Number:=APP_ERROR, _
            Description:="Error. Account " _
            + Str$(lngAccountNo) + " not on file."
    End If

    Dim lngBalance As Long
    lngBalance = adoRS.Fields("Balance").Value

    ' check if account is overdrawn
    If (lngBalance) < 0 Then
        Err.Raise Number:=APP_ERROR, _
            Description:="Error. Account " _
```

```
            + Str$(lngAccountNo) _
            + " would be overdrawn by " _
            + Str$(lngBalance) + ". Balance is still "
            + Str$(lngBalance - lngAmount) + "."
    Else
        If lngAmount < 0 Then
            strResult = strResult _
                & "Debit from account "
                & lngAccountNo & ", "
        Else
            strResult = strResult _
                & "Credit to account "
                & lngAccountNo & ", "
        End If
        strResult = strResult + "balance is $"
            & Str$(lngBalance) & ". (VB)"
    End If

    ' cleanup
    Set adoRS = Nothing
    Set adoConn = Nothing

    ' we are finished and happy
    GetObjectContext.SetComplete

    Post = strResult

Exit Function

ErrorCreateTable:
    On Error GoTo ErrorHandler

    ' create the account table
    Dim objCreateTable As CreateTable
    Set objCreateTable = _
    GetObjectContext.CreateInstance("Bank.CreateTable")
    objCreateTable.CreateAccount

    GoTo TryAgain

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

    GetObjectContext.SetAbort          ' we are unhappy

    Post = ""         ' indicate that an error occurred
    Err.Raise Err.Number, "Bank.Accout.Post", _
        Err.Description

End Function
```

□V,µ,¢ƒIƒuƒWƒFƒNƒg,Ì□ì□¬

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,ÉƒAƒCƒeƒ€,ð'Ç‰Á,µ,Ü,·□B[□V‹K□ì□¬]
ƒRƒ}ƒ“ƒh,Å'Ç‰Á,³,ê,éƒAƒCƒeƒ€,Í□AƒEƒBƒ“ƒhƒE,Ì‰E'¤,É•\Ž¦,³,ê,Ä,¢
,éƒtƒHƒ‹ƒ_,É,æ,Á,Äˆ Ù,È,è,Ü,·□B,½,Æ,¦,Î□A[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ðŠJ,¢,Ä,¢,é,Æ,«,É [□V‹K□ì□¬]
,ðƒNƒŠƒbƒN,·,é,Æ□A□V,µ,¢ƒRƒ“ƒsƒ…□[ƒ^,ª'Ç‰Á,³,ê,Ü,·□B

ƒc□[ƒ‹ ƒo□[,ÌƒVƒ‡□[ƒgƒJƒbƒg□F 

[ƒRƒ“ƒsƒ…□[ƒ^] ƒtƒHƒ‹ƒ_,ÉƒRƒ“ƒsƒ…□[ƒ^,ð'Ç‰Á,·,é•û–@,É,Â,¢,Ä,Í□A□uMTS ƒT□[ƒo□[,Ì□\□¬□v,ðŽQ□Æ,µ,Ä,-
,¾,³,¢□B
□V,µ,¢ƒpƒbƒP□[ƒW,ð□ì□¬,·,é•û–@,É,Â,¢,Ä,Í□A□u‹ó,Ì MTS ƒpƒbƒP□[ƒW,ð□ì□¬,·,é□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B
□V,µ,¢ƒ□□[ƒ‹,ð□ì□¬,·,é•û–@,É,Â,¢,Ä,Í□A□u□V,µ,¢ MTS ƒ□□[ƒ‹,Ì'Ç‰Á□v,ðŽQ□Æ,µ,Ä,,¾,³,¢□B

‘å‚«‚¢ƒAƒJƒRƒ“

ƒEƒBƒ“ƒhƒE‚Ì‰¤‘‚ÌƒAƒJƒeƒ€‚ð‘å‚«‚¢ƒAƒJƒRƒ“‚Å•\Ž¦‚µ‚Ü‚·□B

ƒc□[ƒ‹ ƒo□[‚ÌƒVƒ‡□[ƒgƒJƒbƒg□F 

□¬,³,¢ƒAƒCƒRƒ“

ƒEƒBƒ“ƒhƒE‚Ì‰¤‚ÌƒAƒCƒeƒ€‚ð□¬,³,¢ƒAƒCƒRƒ“‚Å•\Ž¦,µ,Ü,·□B

ƒc□[ƒ‹ ƒo□[‚ÌƒVƒ‡□[ƒgƒJƒbƒg□F 

ˆê——•\Ž¦

ƒEƒBƒ“ƒhƒE‚Ì‰‰E‘¤‚ÌƒAƒCƒeƒ€‚ðˆê——‚Å•\Ž¦‚µ‚Ü‚·□B

ƒc□[ƒ‹ ƒo□[‚ÌƒVƒ‡□[ƒgƒJƒbƒg□F 

ƒvƒ□ƒpƒeƒB•\Ž¦

Œ»□Ý'I'ð,³,ê,Ä,¢,éƒtƒHƒ‹ƒ_"à,ÌƒAƒCƒeƒ€,Ìƒvƒ□ƒpƒeƒB,Ì□Ý'è,ð•\Ž¦,µ,Ü,·□Bƒvƒ□ƒpƒeƒB,Í□AMicrosoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ"ƒhƒE,Ì‰E'¤,É—ñŒ`Ž®,Å•\Ž¦,³,ê,Ü,·□B

ƒc□[ƒ‹ ƒo□[,ÌƒVƒ‡□[ƒgƒJƒbƒg□F 

•\Ž¦,³,ê,éƒvƒ□ƒpƒeƒB,Í□A'I'ð,µ,Ä,¢,éƒIƒuƒWƒFƒNƒg,ÌŽí—Þ,É,æ,Á,ÄˆÙ,È,è,Ü,·□BŽŸ,Ì•\ ,Í□A,»,ê,¼,ê,ÌƒtƒHƒ‹ƒ_,²,Æ,É•\Ž¦,³,ê,éƒvƒ□ƒpƒeƒB,ð,Ü,Æ,ß,½,à,Ì,Å,·□B

ƒtƒHƒ‹ƒ_	ƒvƒ□ƒpƒeƒB
fRƒ"ƒsƒ…□[ƒ^	–¼'O □\ ƒRƒ"ƒsƒ…□[ƒ^,ÉŠ„,è"–,Ä,ç,ê□AWindows NT ,Ìƒhƒ□ƒCƒ""à,Å"FŽ¯,³,ê,é–¼'O,Å,·□B,½,¾,µ□ATransaction Server ,ðŽÀ□s,µ,Ä,¢ ,éƒRƒ"ƒsƒ…□[ƒ^,Í□Aƒ}ƒC ƒRƒ"ƒsƒ…□[ƒ^,ÆŒÄ,Î,ê,Ü,·□B
	ƒ^ƒCƒ€ƒAƒEƒg
fCƒ"ƒXƒg□[ƒ‹,³,ê,½ƒpƒbƒP□[ƒW	–¼'O □\ ƒpƒbƒP□[ƒW,ÉŠ„,è"–,Ä,½–¼'O,Å,·□B
	ƒZƒLƒ…ƒŠƒeƒB □\ ƒpƒbƒP□[ƒW,ÌƒZƒLƒ…ƒŠƒeƒB,ª—LŒø,É,È,Á,Ä,¢,é,©,Ç,¤,©,ðŽ¦,µ,Ü,·□B
	"F□Ø □\ "F□Øƒ`ƒFƒbƒN,ÌƒŒƒxƒ‹,Å,·□B
	ƒVƒƒƒbƒgƒ_ƒEƒ" □\ ƒAƒNƒeƒBƒrƒeƒB,ª,È,¢ ,Æ,«,É□AƒpƒbƒP□[ƒW,ªƒVƒƒƒbƒgƒ_ƒEƒ",·,é,Ü,Å,ÌŽžŠÔ,Å,·□B
	□í,ÉŽÀ□s □\ "ñƒAƒNƒeƒBƒu,Ì,Æ,«,Å,àƒpƒbƒP□[ƒW,ªƒVƒƒƒbƒgƒ_ƒEƒ",µ,È,¢ ,±,Æ,ðŽ¦,µ,Ü,·□B
	ƒAƒJƒEƒ"ƒg □\ ƒpƒbƒP□[ƒW ID ,É□Ý'è,³,ê,½ Windows NT ƒAƒJƒEƒ"ƒg,Å,·□B
	ƒpƒbƒP□[ƒW ID □\ ƒpƒbƒP□[ƒW,ÉŠ„,è"–,Ä,ç,ê,½"Ä—p^ê^ÓŽ¯•ÊŽq (UUID) ,Å,·□B
fRƒ"ƒ|□[ƒlƒ"ƒg	Prog ID □\ ƒRƒ"ƒ|□[ƒlƒ"ƒg,ÉŠ„,è"–,Ä,ç,ê,½–¼'O,Å,·□B
	ƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ" □\ ƒRƒ"ƒ|□[ƒlƒ"ƒg,ªƒgƒ‰ƒ"ƒUƒNƒVƒ‡ƒ",ðƒTƒ|□[ƒg,µ,Ä,¢,é,©,Ç,¤,©,ðŽ¦,µ,Ü,·□B
	DLL
	CLSID
	MTX Žg—p □\ ƒRƒ"ƒ|□[ƒlƒ"ƒg,ª Transaction Server ŠÂ‹«,ÅŽÀ□s,³,ê,Ä,¢,é,©,Ç,¤,©,ðŽ¦,µ,Ü,·□B
	ƒCƒ" ƒvƒ□ƒZƒX □\ ƒRƒ"ƒ|□[ƒlƒ"ƒg,ª•Ê,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,ÅŽÀ□s,³,ê,Ä,¢,é,±,Æ,ðŽ¦,µ,Ü,·□B
	ƒ□□[ƒJƒ‹ □\ ƒRƒ"ƒ|□[ƒlƒ"ƒg,ªƒ□□[ƒJƒ‹ ƒRƒ"ƒsƒ…□[ƒ^,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,ÅŽÀ□s,³,ê,Ä,¢,é,±,Æ,ðŽ¦,µ,Ü,·
	ƒŠƒ,□[ƒg □\ ƒRƒ"ƒ|□[ƒlƒ"ƒg,ªƒŠƒ,□[ƒg ƒRƒ"ƒsƒ…□[ƒ^,ÌƒT□[ƒo□[ƒvƒ□ƒZƒX,ÅŽÀ□s,³,ê,Ä,¢,é,±,Æ,ðŽ¦,µ,Ü,·□B
	ƒT□[ƒo□[□\ ƒRƒ"ƒ|□[ƒlƒ"ƒg,ªŽÀ□s,³,ê,Ä,¢,éƒŠƒ,□[ƒg ƒRƒ"ƒsƒ…□[ƒ^,Ì–¼'O,Å,·□B

ƒXƒfŒƒbƒh □\ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒXƒfŒƒbƒh ƒ‚ƒƒ‹,Å,·□B

ƒZƒLƒ…ƒŠƒeƒB □\ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒZƒLƒ…ƒŠƒeƒB,ª—
LŒø,É,È,Á,Ä,¢,é,©,Ç,¤,©,ðŽ¦,µ,Ü,·□B

f□□[f‹ –¼‘O □\ f□□[f‹,ÉŠ„,è“–,Ä,ç,ê,½–¼‘O,Å,·□B

ƒ□□[ƒ‹ ID □\ ƒ□□[ƒ‹,ÉŠ„,è“–,Ä,ç,ê,½^ê^Ó,ÌŽ¯•ÊŽq,Å,·□B

fCƒ“ƒ^□[ƒtƒFƒCƒX –¼‘O □\ fCƒ“ƒ^□[ƒtƒFƒCƒX,ÉŠ„,è“–,Ä,ç,ê,½–¼‘O,Å,·□B

fCƒ“ƒ^□[ƒtƒFƒCƒX ID □\ ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ÉŠ„,è“–
,Ä,ç,ê,½^ê^Ó,ÌŽ¯•ÊŽq,Å,·□B

ƒvƒ□ƒLƒV DLL

TypeLib ƒtƒ@ƒCƒ‹ □\ ƒ^ƒCƒv ƒ‰ƒCƒuƒ
‰ƒŠ,ðŠÜ,Þƒtƒ@ƒCƒ‹,Ì–¼‘O,Å,·□B

f□ƒ\ƒbƒh –¼‘O □\ f□ƒ\ƒbƒh,ÉŠ„,è“–,Ä,ç,ê,½–¼‘O,Å,·□B

f□□[f‹ –¼‘O □\ ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ü,½,ÍƒCƒ“ƒ^□[ƒtƒFƒCƒX,É’Ç
f□ƒ“ƒo□[ƒVƒbƒv ‰Á,³,ê,½ƒ□□[ƒ‹,É□AŠ„,è“–,Ä,ç,ê,½–¼‘O,Å,·□B

ƒ□□[ƒ‹ ID □\ ƒ□□[ƒ‹,ÉŠ„,è“–,Ä,ç,ê,½^ê^Ó,ÌŽ¯•ÊŽq,Å,·□B

□ó‘Ô•\Ž¦

ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ü,½,ÍƒRƒ“ƒsƒ…□[ƒ^,ÌŒ»□Ý,Ì□ó‘Ô,ð•\Ž¦,µ,Ü,·□B

ƒc□[ƒ‹ ƒo□[,ÌƒVƒ‡□[ƒgƒJƒbƒg□F 

ƒRƒ“ƒsƒ…□[ƒ^,Ì□ó‘Ô

- [–¼‘O] □\ ƒRƒ“ƒsƒ…□[ƒ^,ÉŠ„,è“–,Ä,ç,ê,½–¼‘O,Å,·□B
- [DTC] □\ MS DTC ,ª‹N“®,³,ê,Ä,¢,é,©,Ç,¤,©,ðŽ¦,µ,Ü,·□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì□ó‘Ô

- [Prog ID] □\ ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉŠ„,è“–,Ä,ç,ê,½–¼‘O,Å,·□B
- [ƒlƒuƒWƒFƒNƒg] □\ ƒT□[ƒo□[ƒvƒ□ƒZƒX“à,ÉŠ„,è“–,Ä,ç,ê,½ƒlƒuƒWƒFƒNƒg,Ì‘□□”,Å,·□B
- [ƒAƒNƒeƒBƒu‰»] □\ ƒNƒ‰ƒCƒAƒ“ƒg,ªŽg—p,µ,Ä,¢,éƒlƒuƒWƒFƒNƒg,Ì‘□□”,Å,·□B
- [ŒÄ,Ñ□o,µ’†] □\ Œ»□ÝƒNƒ‰ƒCƒAƒ“ƒgŒÄ,Ñ□o,µ,ðŽÀ□s,µ,Ä,¢,éƒlƒuƒWƒFƒNƒg,Ì‘□□”,Å,·□B

□Å□V,Ì□î•ñ,É□X□V

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒEƒBƒ“ƒhƒE,Ì‰ºE‘¤,É•\Ž¦,³,ê,é□î•ñ,ðŽè□ì‹Æ,Å□X□V,µ,Ü,·□B

ƒc□[ƒ‹ ƒo□[,ÌƒVƒ‡□[ƒgƒJƒbƒg□F 

—pŒê□W

2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg

ACID

ActiveX

COM (ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒIƒuƒWƒFƒNƒg ƒ‚ƒƒ‹)

DCOM

ID

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰

Microsoft Transaction Server ƒIƒuƒWƒFƒNƒg

Microsoft Transaction Server ƒRƒ“ƒ|□[ƒlƒ“ƒg

Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒI□[ƒ^ (MS DTC)

Null

ODBC ƒŠƒ\□[ƒX ƒfƒBƒXƒyƒ“ƒT

OLE ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“

XA ƒvƒ□ƒgƒRƒ‹

ƒAƒNƒeƒBƒrƒeƒB

ƒAƒp□[ƒgƒ□ƒ“ƒg ƒXƒŒƒbƒh

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹ ƒt□[ƒeƒBƒŠƒeƒB

^À‘S‚ÈŽQ□Æ

^êŠÑ□« (consistency)

ƒCƒ“ ƒ_ƒEƒg‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“

ƒCƒ“ƒXƒ^ƒ“ƒX

ƒCƒ“ƒ^□[ƒtƒFƒCƒX

ƒCƒ“ƒvƒ□ƒZƒX ƒRƒ“ƒ|□[ƒlƒ“ƒg

ƒI□[ƒvƒ“ ƒf□[ƒ^ƒx□[ƒX ƒRƒIƒNƒeƒBƒrƒeƒB (ODBC)

ƒIƒuƒWƒFƒNƒg

ƒIƒuƒWƒFƒNƒg•Ï□”

ƒIƒ^ƒ□ƒO

ŠÇ—□ŽÒ

Šù□¬‚Ìƒpƒbƒ□[ƒW

‹¤—Lƒvƒ□ƒpƒeƒB

ƒNƒ‰ƒCƒAƒ“ƒg

ƒNƒ‰ƒCƒAƒ“ƒg/ƒT□[ƒo□[

ƒNƒ‰ƒX

ƒNƒ‰ƒX ID (CLSID)

ƒNƒ‰ƒX ƒtƒ@ƒNƒgƒŠ

ƒNƒ‰ƒXƒ^

ƒOƒ‹□[ƒv

ƒOƒ□□[ƒoƒ‹ ƒAƒJƒEƒ“ƒg

Œ´Žq□« (atomicity)

ƒRƒ“ƒXƒgƒ‰ƒNƒ^

ƒRƒ“ƒeƒLƒXƒg

ƒRƒ“ƒeƒLƒXƒg ƒIƒuƒWƒFƒNƒg ƒvƒ□ƒpƒeƒB

ƒRƒ“ƒ|□[ƒlƒ“ƒg

ƒT□[ƒo□[ƒpƒbƒP□[ƒW

ƒT□[ƒo□[ƒvƒ□ƒZƒX

□Å□‰,Ì□ì□¬Œ³

□Å□‰,ÌŒÄ,Ñ□o,µŒ³

□ì□¬Œ³

Ž□‘±□« (durability)

Ž©“®ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“

ƒWƒƒXƒgƒCƒ“ƒ^ƒCƒ€ ƒAƒNƒeƒBƒx□[ƒVƒ‡ƒ“

□W□¬

□áŠQ,Ì•ª—£

ƒXƒ^ƒu

ƒXƒe□[ƒgƒtƒ‹ ƒIƒuƒWƒFƒNƒg

ƒXƒe□[ƒgŒƒX ƒIƒuƒWƒFƒNƒg

ƒXƒiƒbƒvƒCƒ“

ƒXƒŒƒbƒh

ƒZƒLƒ…ƒŠƒeƒB ID (SID)

ƒZƒ}ƒtƒH

□éŒ¾,É,æ,éƒZƒLƒ…ƒŠƒeƒB

ƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL)

ƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ

‘Î˜bŒ^ƒ□ƒOƒIƒ“ ƒt□[ƒU□[

’¼□Ú,Ì□ì□¬Œ³

’¼□Ú,ÌŒÄ,Ñ□o,µŒ³

ƒf□[ƒ^ ƒ\□[ƒX–¼ (DSN)

ƒfƒbƒhƒ□ƒbƒN

“¯ŽžŽÀ□s

ƒhƒ□ƒCƒ“

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒeƒLƒXƒg

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ^ƒCƒ€ƒAƒEƒg

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ}ƒl□[ƒWƒƒ

ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW

”F□Ø

ƒpƒbƒP□[ƒW

ƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹

ƒrƒWƒlƒX ƒ‹□[ƒ‹

ƒv□[ƒŠƒ“ƒO

ƒu□[ƒ‹’l (Boolean)

ƒtƒFƒCƒ‹ƒtƒ@□[ƒXƒg

ƒtƒH□[ƒ‹ƒg ƒgƒŒƒ‰ƒ“ƒX

•‰‰×,ÌƒoƒŠƒ“ƒX

ƒvƒ□ƒLƒV

ƒvƒ□ƒOƒ‰ƒ€ ID (progID)

ƒvƒ□ƒOƒ‰ƒ€,É,æ,éƒZƒLƒ…ƒŠƒeƒB

ƒvƒ□ƒZƒX,Ì•ª—£

ƒvƒ□ƒZƒXŠOƒRƒ“ƒ|□[ƒlƒ“ƒg

•ª—£□« (isolation)

ƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg

ƒx□[ƒX ƒvƒ□ƒZƒX

ƒ}□[ƒVƒƒŠƒ“ƒO

ƒ□ƒCƒ“ ƒXƒŒƒbƒh

ƒ□ƒ\ƒbƒh

•¶Žš—ñŽ®

ƒt□[ƒU□[–¼

ŒÄ,Ñ□o,µŒ³

ƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW

ƒŠƒ\□[ƒX ƒƒBƒXƒyƒ“ƒT ƒ}ƒl□[ƒWƒƒ

ƒŠƒ\□[ƒX ƒƒBƒXƒyƒ“ƒT

ƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ

ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg

fŠƒ,□[ƒg ƒvƒ□ƒV□[ƒWƒƒ ƒR□[ƒ‹ (RPC)

—áŠO

fŒƒvƒŠƒP□[ƒVƒ‡ƒ“

ƒ□□[ƒJƒ‹ ƒAƒJƒEƒ“ƒg

ƒ□□[ƒ‹

ACID

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŠî–{“I,Èƒvƒ□ƒpƒeƒB□BŒ´Žq□« (atomicity)□Aˆêš Ñ□« (consistency)□A•ª—£□« (isolation) ,¨,æ,ÑŽ□'±□« (durability) ,ð,¢,¢,Ü,·□B

ActiveX

ƒ\ƒtƒgƒEƒFƒA ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ç,¤,µ,ª□A,»,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌŠJ”,ÉŽg,í,ê,½Œ¾Œê,Æ,Í–³ŠÖŒW,É□Aƒlƒbƒgƒ□□[ƒNŠÂ‹«,Å,â,è,Æ,è,Å,«,é,æ,¤,É,·,é,½,ß,Ì^ê˜A,ÌƒeƒNƒmƒ□ƒW□BActiveX ,ÍƒRƒ“ƒ|□[ƒlƒ“ƒg ƒlƒuƒWƒFƒNƒg ƒ‚ƒƒ‹ (COM) ,ðŠî–{,É□ì,ç,ê,Ä,¢,Ü,·□B

ƒAƒNƒeƒBƒrƒeƒB

’P“Æ‚Ì•ªŽU˜_—□ŽÀ□sƒXƒŒƒbƒh‚ðŽ□‚Â Microsoft Transaction Server ƒIƒuƒWƒFƒNƒg‚Ì□W□‡□B‚Ç‚Ì Microsoft Transaction Server ƒIƒuƒWƒFƒNƒg‚à□A‚ ‚éƒAƒNƒeƒBƒrƒeƒB‚É‘®‚µ‚Ä‚¢‚Ü‚·□B

ŠÇ—□ŽÒ

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä Microsoft Transaction Server ,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,ÆƒpƒbƒP□[ƒW,ÌƒCƒ“ƒXƒg□[ƒ‹□A□\□¬□A,¨,æ,ÑŠÇ—□,ð□s,¤ƒt□[ƒU□[□B

□W□¬

ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒIƒuƒWƒFƒNƒg,ðŽÀ‘•,·,é,½,ß,ÌƒRƒ“ƒ|ƒWƒVƒ‡ƒ“ ƒeƒNƒjƒbƒN□B,±
,ÌƒeƒNƒjƒbƒN,É,æ,è□A□V,µ,¢ƒIƒuƒWƒFƒNƒg,ð□ì□¬,·,é,Æ,«,É□A,»,ÌƒIƒuƒWƒFƒNƒg,É•K—
v,ÈƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðƒTƒ|□[ƒg,·,éŠù‘¶,ÌƒIƒuƒWƒFƒNƒg,ðŽg,¤,±,Æ,ª,Å,«,Ü,·□B

ƒAƒp□[ƒgƒ□ƒ“ƒg ƒXƒŒƒbƒh

"ƒAƒp□[ƒgƒ□ƒ“ƒg ƒXƒŒƒbƒh" ‚Æ‚µ‚Ä□\□¬‚³‚ê‚½ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ìƒlƒuƒWƒFƒNƒg‚Ö‚ÌŒÄ‚Ñ□o‚µ‚ÉŽg‚¤ƒXƒŒƒbƒh□BŠeƒlƒuƒWƒFƒNƒg‚Í□A‚»‚ìƒlƒuƒWƒFƒNƒg‚Ì—LŒøŠúŠÔ’†□AƒAƒp□[ƒgƒ□ƒ“ƒg (ƒXƒŒƒbƒh) ‚ì’†‚Å“®□ì‚µ‚Ü‚·□B‚±‚ìƒlƒuƒWƒFƒNƒg‚Ö‚ÌŒÄ‚Ñ□o‚µ‚Í□A‚·‚×‚ÄƒAƒp□[ƒgƒ□ƒ“ƒgƒXƒŒƒbƒh‚ÅŽÀ□s‚³‚ê‚Ü‚·□B½‚Æ¦‚Î□AƒlƒuƒWƒFƒNƒg‚Ì□ó‘Ô‚ðƒXƒŒƒbƒh ƒ□□[ƒlƒ‹Ši”[—Ìˆæ (TLS) ‚É•ÛŽ□‚·‚éƒRƒ“ƒ|□[ƒlƒ“ƒg‚ðŽÀ‘•‚·‚é□ê□‡□A‚±‚ÌƒXƒŒƒbƒh ƒ‚ƒƒ‹‚ªŽg‚í‚ê‚Ü‚·□BƒRƒ“ƒ|□[ƒlƒ“ƒg‚ìƒlƒuƒWƒFƒNƒg‚Í□A1 ‚Â‚Ü‚½‚Í•¡□”‚ìƒAƒp□[ƒgƒ□ƒ“ƒg‚É•ªŽU‚Å‚«‚Ü‚·□Bƒ□ƒCƒ“ ƒXƒŒƒbƒh‚àŽQ□Æ‚µ‚Ä‚,‚¾‚³‚¢□B

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹ ƒt□[ƒeƒBƒŠƒeƒB

ƒpƒbƒP□[ƒW,ÌƒGƒNƒXƒ|□[ƒg,É,æ,Á,ÄƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŽÀ□s‰Â”\ƒtƒ@ƒCƒ‹,ð□ì□¬,Å,«,é,æ,¤,É,·,é MTS ƒGƒNƒXƒvƒ□□[ƒ‰,Ì‹@”\□B

Œ´Žq□« (atomicity)

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì,·,×,Ä,Ì□^—□,ªŽÀ□s,³,ê,é,©□A,Ü,½,Í,Ç,Ì□^—□,àŽÀ□s,³,ê,È,¢,©,Ì,Ç,¿,ç,©,ðŽ¦,·ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Ì‹@”\□B

”F□Ø

ƒVƒXƒeƒ€‚ÉƒAƒNƒZƒX‚µ‚æ‚¤‚Æ‚·‚éƒt□[ƒU□[‚Ì ID ‚ðŠm”F‚·‚é□ˆ—
□□B‚½‚Æ‚¦‚Î□Aƒt□[ƒU□[‚Ì”F□Ø‚É‚Íˆê”Ê‚ÉƒpƒXƒ□□[ƒh‚ªŽg‚í‚ê‚Ü‚·□B

Ž©“®ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“

ƒRƒ“ƒ|□[ƒlƒ“ƒg‚ÌƒIƒuƒWƒFƒNƒg—p‚É□AMicrosoft Transaction Server ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«‚ª□A‚»‚ÌƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‘®□«‚ÉŠî‚Ã‚¢‚Ä□ì□¬‚·‚éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□B

ƒu□[ƒ‹'l (Boolean)

True (□^) ,Ü,½,Í False (‹U) ,Ì'l□A, ,é,¢,Í yes ,Ü,½,Í no ,Ì'l□B

ƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg

Microsoft Transaction Server ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÌŠO•”,Å“®□ì,µ□AMicrosoft Transaction Server ƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,·,éƒNƒ‰ƒCƒAƒ“ƒg□B

ƒx□[ƒX ƒvƒ□ƒZƒX

ƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg,ªŽÀ□s,³,ê,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒvƒ□ƒZƒX□Bƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg,Í Microsoft Transaction Server ƒ‰ƒ^ƒCƒ€ŠÂ‹«,ìŠO•”,Å“®□ì,µ□AMicrosoft Transaction Server ƒIƒuƒWƒFƒNƒg,ìƒCƒ“ƒXƒ^ƒ“ƒX,ð□¶□¬,µ,Ü,·□B

ƒrƒWƒlƒX ƒ‹□[ƒ‹

Šé‹Æ“à,Ì‹Æ–±□^—□,ð’è,ß,½Œ´‘¥□B“ü—Í‹K‘¥,ðŽg—
p,µ,½•Ò□W□Aƒ□ƒOƒlƒ“,ÌŠm”F□Aff□[ƒ^ƒx□[ƒX,Ì□Æ□‡□A•û□j□AƒAƒ‹ƒSƒŠƒYƒ€•ÏŠ·,È,Ç,Ì‘g,Ý□‡,í,¹,Å,·□B"ƒrƒWƒlƒ
X ƒ□ƒWƒbƒN" ,Æ,àŒÄ,Î,ê,Ü,·□B

ŒÄ‚Ñ□o‚µŒ³

ƒIƒuƒWƒFƒNƒg‚Ìƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚·ƒNƒ
‰ƒCƒAƒ“ƒg□BƒIƒuƒWƒFƒNƒg‚ÌŒÄ‚Ñ□o‚µŒ³‚Í□A•K‚¸‚µ‚à‚ƒIƒuƒWƒFƒNƒg‚Ì□ì□¬Œ³‚Å‚Í‚ ‚è‚Ü‚¹‚ñ□B‚½‚Æ‚¦‚Î□AƒNƒ
‰ƒCƒAƒ“ƒg A ‚ªƒIƒuƒWƒFƒNƒg X ‚ð□ì□¬‚µ□AƒIƒuƒWƒFƒNƒgŽQ□Æ‚ðƒNƒ‰ƒCƒAƒ“ƒg B ‚É“n‚·‚Æ□AƒNƒ
‰ƒCƒAƒ“ƒg B ‚Í‚»‚ÌŽQ□Æ‚ðŽg‚Á‚ÄƒIƒuƒWƒFƒNƒg X ‚Ìƒ□ƒ\ƒbƒh‚ðŒÄ‚Ñ□o‚·‚±‚Æ‚ª‚Å‚«‚Ü‚·□B‚±‚Ì□ê□‡□AƒNƒ
‰ƒCƒAƒ“ƒg A ‚ª□ì□¬Œ³‚Å□AƒNƒ‰ƒCƒAƒ“ƒg B ‚ªŒÄ‚Ñ□o‚µŒ³‚É‚È‚è‚Ü‚·□B□ì□¬Œ³‚àŽQ□Æ‚µ‚Ä‚¾‚³‚¢□B

ƒJƒ^ƒ□ƒO

ƒRƒ“ƒ|□[ƒIƒ“ƒg□AƒpƒbƒP□[ƒW□A,¨,æ,Ñƒ□□[ƒ‹,Ì□\□¬□î•ñ,ð•ÛŽ□,·,é Microsoft Transaction Server ƒf□[ƒ^,ÌŠi”[□ê□Š□BƒJƒ^ƒ□ƒO,ðŠÇ—□,·,é,É,Í□AMicrosoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,¢,Ü,·□B

ƒNƒ‰ƒX

“Á’è,ÌŽí—Þ,ÌƒIƒuƒWƒFƒNƒg,ÌƒCƒ“ƒ^□[ƒtƒFƒCƒX,ð’è‹`,·,éŒ^□BƒNƒ
‰ƒX,ÍƒIƒuƒWƒFƒNƒg,ÌƒvƒfƒpƒeƒB,Æ□AƒIƒuƒWƒFƒNƒg,Ì“®□ì,ð□§Œä,·,é,½,ß,Ìƒ□ƒ\ƒbƒh,ð’è‹`,µ,Ü,·□B

ƒNƒ‰ƒX ƒtƒ@ƒNƒgƒŠ

IClassFactory ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŽÀ‘•,·,éƒIƒuƒWƒFƒNƒg□B,±,ÌƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŽg,Á,Ä“Á’è,ÌƒNƒ‰ƒX,ÌƒIƒuƒWƒFƒNƒg,ð□ì□¬,Å,«,Ü,·□B

ƒNƒ‰ƒX ID (CLSID)

COM ƒRƒ“ƒ|[ƒlƒ“ƒg‚ðŽ¯•Ê‚·‚é‚½‚ß‚Ì”—p‚ê‚ÓŽ¯•ÊŽq (UUID)□BŠe COM ƒRƒ“ƒ|□[ƒlƒ“ƒg‚Ì CLSID ‚Í Windows ƒŒƒWƒXƒgƒŠ‚É“o˜^‚³‚ê□A‚Ù‚©‚ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚©□□[ƒh‚Å‚«‚é‚æ‚¤‚É‚È‚Á‚Ä‚¢‚Ü‚·□B

ƒNƒ‰ƒCƒAƒ“ƒg

,Ù,©,Ìƒvƒ□ƒZƒX,Ü,½,ÍƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉƒT□[ƒrƒX,ð—v‹□,·,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,âƒvƒ□ƒZƒX□B

ƒNƒ‰ƒCƒAƒ“ƒg/ƒT□[ƒo□[

ƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ªƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚ÉƒT□[ƒrƒX‚ð—v‹□‚·‚é‚æ‚¤
‚È•ªŽUƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒ‚ƒf‹□BƒT□[ƒo□[‚Í“¯Žž‚É‘½‚ƒNƒ‰ƒCƒAƒ“ƒg‚ðŽ□‚Â‚±‚Æ‚ª‚Å‚«‚□AƒNƒ
‰ƒCƒAƒ“ƒg‚Í•¡□”‚ÌƒT□[ƒo□[‚Éƒf□[ƒ^‚ð—v‹□‚Å‚«‚Ü‚·□BƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚Í□AƒNƒ
‰ƒCƒAƒ“ƒg‚É‚àƒT□[ƒo□[‚É‚à‚È‚è‚¦‚Ü‚·□B

ƒNƒ‰ƒXƒ^

Microsoft Cluster Server ‚ðŽg‚Á‚Ä’Pˆê‚ìƒVƒXƒef€‚Æ‚µ‚ÄƒAƒhƒŒƒXŽw’è‚³‚ê□AŠÇ—□‚³‚ê‚é 2 ‚Âˆ È□ã‚Ì“Æ—§‚µ‚½ƒRƒ“ƒsƒ…□[ƒ^ ƒVƒXƒef€□B

COM (ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒIƒuƒWƒFƒNƒg ƒ‚ƒƒ‹)

ƒIƒuƒWƒFƒNƒgŽwŒüƒeƒNƒmƒ□ƒW,ÉŠî,Ã,¢,½ƒNƒ‰ƒCƒAƒ“ƒg/ƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ð□A^Ù,È,éƒvƒ‰ƒbƒgƒtƒH□[ƒ€□ã,ÅŠJ”,·,é,½,ß,ÌƒI□[ƒvƒ“ ƒA□[ƒLƒeƒNƒ`ƒƒ□BƒNƒ‰ƒCƒAƒ“ƒg,Í□AƒIƒuƒWƒFƒNƒg,ÉŽÀ‘•,³,ê,½ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ð’Ê,¶,ÄƒIƒuƒWƒFƒNƒg,ÉƒAƒNƒZƒX,µ,Ü,·□BCOM ,ÍŒ¾Œê,É^Ë‘¶,µ,È,¢,½,ß□AActiveX ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð□ì□¬,Å,«,éŒ¾Œê,Å, ,ê,Î□ACOM ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,à□ì□¬,Å,«,Ü,·□B

ƒRƒ“ƒ|□[ƒlƒ“ƒg

ActiveX ƒeƒNƒmƒ□[ƒW‚ÉŠî‚Ã‚¢‚Ä□ì□¬‚³‚ê‚½ƒR□[ƒh‚Ì’P‰Ê□B–¾Šm‚ÈŽd—l‚ðŽ□‚Â‰ê˜A‚ÌƒT□[ƒrƒX‚ð□A–¾Šm‚É‹K’è‚³‚ê‚½ƒCƒ“ƒ^□[ƒtƒFƒCƒX‚ð’Ê‚¶‚Ä’ñ‹Ÿ‚µ‚Ü‚·□BƒRƒ“ƒ|□[ƒlƒ“ƒg‚í□AŽÀ□sŽž‚ÉƒNƒ‰ƒCƒAƒ“ƒg‚ª—v‹□‚·‚éƒIƒuƒWƒFƒNƒg‚ð’ñ‹Ÿ‚µ‚Ü‚·□B

“¯ŽžŽÀs

•¡”,Ì^—,ðŒðŒÝ,ÉŽÀs,·,é,±,Æ,É,æ,èAƒvƒƒZƒX,âƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª“¯Žž,ÉŽÀs,³,ê,é,æ,¤,ÉŒ©,¦,é,±,ÆB

ˆêŠÑ« (consistency)

Ž‘±“I,Èƒf[ƒ^,ª‚A,»,Ìƒf[ƒ^,ð•Ï‚X,µ,½ƒrƒWƒlƒX ƒ‹[ƒ‹,ª‹‚,ß,é‚ó‘Ô,Éˆê’v,µ,Ä,¢,é‚ó‘Ô‚B

ƒRƒ“ƒXƒgƒ‰ƒNƒ^

C++ ,¨,æ,Ñ Java ,Å□AƒNƒ‰ƒX,ÌƒCƒ“ƒXƒ^ƒ“ƒX,ª□éŒ¾,³,ê,é,½,Ñ,ÉŽ©“®“I,ÉŒÄ,Ñ□o,³,ê,é“ÁŽê,È□‰Šú‰»ŠÖ□”□B,±,ÌŠÖ□”,Í□A□‰Šú‰»,³,ê,Ä,¢,È,¢ƒIƒuƒWƒFƒNƒg,ðŽg,Á,½,½,ß,É□¶,¶,éfGƒ‰□[,ð–hŽ~,µ,Ü,·□BƒRƒ“ƒXƒgƒ‰ƒNƒ^,ÍƒNƒ‰ƒXŽ©‘Ì,Æ“¯,¶–¼‘O,ðŽ□,¿□A‘I,ð•Ô,·,±,Æ,Í,Å,«,Ü,¹,ñ□B

ƒRƒ“ƒefLƒXƒg

□Š’è‚Ì Microsoft Transaction Server ƒIƒuƒWƒFƒNƒg‚ÉˆÃ–Ù“I‚ÉŠÖ˜A•t‚¯‚ç‚ê‚Ä‚¢
‚é□ó‘Ô□BƒRƒ“ƒefLƒXƒg‚É‚Í□AƒIƒuƒWƒFƒNƒg‚Ì□ì□¬Œ³‚Ì ID
‚È‚Ç□AƒIƒuƒWƒFƒNƒg‚ÌŽžÀ□sŠÂ‹«‚ÉŠÖ˜A‚µ‚é□î•ñ‚ªŠÜ‚Ü‚ê‚Ü‚·□B‚Ü‚½□AƒIƒuƒWƒFƒNƒg‚Ì□ì‹Æ‚ð□‹□s‚·‚éƒgƒ
‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ªƒRƒ“ƒefLƒXƒg‚ÉŠÜ‚Ü‚ê‚é‚±‚Æ‚à‚ ‚è‚Ü‚·□BƒIƒuƒWƒFƒNƒg
ƒRƒ“ƒefLƒXƒg‚Í□AŠT”O“I‚É‚ÍƒyƒŒ□[ƒefBƒ“ƒO ƒVƒXƒef€‚ªƒvƒ□ƒOƒ‰ƒ€‚ÌŽžÀ□s‚Ì‚½‚É•ÛŽ□‚·‚éƒvƒ□ƒZƒX
ƒRƒ“ƒefLƒXƒg‚ÉŽ—‚Ä‚¢‚Ü‚·□BMicrosoft Transaction Server ƒ
‰ƒ“ƒ^ƒCƒ€ŠÂ‹«‚ª□AŠeƒIƒuƒWƒFƒNƒg‚ÌƒRƒ“ƒefLƒXƒg‚ðŠÇ—□‚µ‚Ü‚·□B

ƒRƒ“ƒeƒLƒXƒg ƒIƒuƒWƒFƒNƒg ƒvƒ□ƒpƒeƒB

Internet Information Server ŒÅ—L,ÌƒIƒuƒWƒFƒNƒg,È,Ç,ÌƒRƒ“ƒeƒLƒXƒg
ƒIƒuƒWƒFƒNƒg,©,çŽæ“¾,Å,«,éƒvƒ□ƒpƒeƒB□B

□ì□¬Œ³

ƒRƒ“ƒ|□[ƒlƒ“ƒg,©,ç'ñ‹Ÿ,³,ê,éƒIƒuƒWƒFƒNƒg,ð (CreateObject□ACoCreateInstance□A,Ü,½,Í CreateInstance ƒ□ƒ\ƒbƒh,ðŽg,Á,Ä) □ì□¬,·,éƒNƒ‰ƒCƒAƒ“ƒg□BƒNƒ ‰ƒCƒAƒ“ƒg,ª□ì□¬,µ,½ƒIƒuƒWƒFƒNƒg,É,Í□A,»,ìƒIƒuƒWƒFƒNƒg,ìƒ□ƒ\ƒbƒh,ðŒÄ,Ñ□o,·,½,ß,ìƒIƒuƒWƒFƒNƒgŽQ□Æ,ª —^,¦,ç,ê,Ü,·□BŒÄ,Ñ□o,µŒ³,àŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒf□[ƒ^ ƒ\□[ƒX–¼ (DSN)

ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ª ODBC ƒf□[ƒ^ ƒ\□[ƒX,Ö,Ì□Ú‘±,ð—v‹□,·,é,Æ,«,ÉŽg,¤–¼‘O□B

ƒfƒbƒhƒ□ƒbƒN

2 ,Âˆè□ã,ìƒXƒCƒbƒh,ª‰i‘±“I,Éƒuƒ□ƒbƒN,³,ê,Ä,¢,é (‘Ò‹@,µ,Ä,¢,é)
□ó‘Ô□BŠeƒXƒCƒbƒh,Í□Aƒuƒ□ƒbƒN,³,ê,½,Ù,©,ìƒXƒCƒbƒh,Ì 1 ,Â,ª”r‘¼“I,É•ÛŽ□,µ,Ä,¢,éƒŠƒ\□[ƒX,ð‘Ò,Á,Ä,¢
,Ü,·□B,½,Æ,¦,Î□AƒXƒCƒbƒh A ,ªƒCƒfR□[ƒh 1 ,ðƒ□ƒbƒN,µ,È,ª,çƒCƒfR□[ƒh 2 ,ðƒ□ƒbƒN,·,é,½,ß,É‘Ò‹@,µ,Ä,¢
,é,Æ,«,É□AƒXƒCƒbƒh B ,ªƒCƒfR□[ƒh 2 ,ðƒ□ƒbƒN,µ,È,ª,çƒCƒfR□[ƒh 1 ,ðƒ□ƒbƒN,·,é,½,ß,É‘Ò‹@,µ,Ä,¢
,é□ê□‡□A2 ,Â,ìƒXƒCƒbƒh,Íƒfƒbƒhƒ□ƒbƒN□ó‘Ô,É,È,è,Ü,·□B

□éŒ¾,É,æ,éƒZƒLƒ…ƒŠƒefB

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,Á,Ä□\□¬,·,éƒZƒLƒ…ƒŠƒefB□Bƒ□□[ƒ‹,ð'è‹`,·,é,±,Æ,É,æ,Á,Ä□Aƒpƒbƒ□[ƒW□AƒRƒ"ƒ|□[ƒlƒ"ƒg□A,¨,æ,ÑƒCƒ"ƒ^□[ƒtƒFƒCƒX,Ö,ÌƒAƒNƒZƒX,ð□§Œä,µ,Ü,·□Bƒ□□[ƒ‹,É,æ,è□A,Ç,Ìƒt□[ƒU□[,ªƒRƒ"ƒ|□[ƒlƒ"ƒg,ÌƒCƒ"ƒ^□[ƒtƒFƒCƒX,ðŒÄ,Ñ□o,·,±,Æ,ª,Å,«,é,©,ð□Ý'è,µ,Ü,·□Bƒvƒ□ƒOƒ‰ƒ€,É,æ,éƒZƒLƒ…ƒŠƒefB,àŽQ□Æ,µ,Ä,,¾,³,¢□B

’¼□Ú‚ÌŒÄ‚Ñ□o‚µŒ³

Œ»□Ý‚ÌƒT□[ƒo□[ƒvƒ□ƒZƒX‚ðŒÄ‚Ñ□o‚µ‚¢‚éƒvƒ□ƒZƒX (ƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg‚Ü‚½‚ÍƒT□[ƒo□[ƒvƒ□ƒZƒX) ‚ÌID□B

’¼□Ú‚Ì□ì□¬Œ³

Œ»□Ý‚ÌƒfƒuƒWƒFƒNƒg‚ð’¼□Ú□ì□¬‚µ‚½ƒvƒ□ƒZƒX (ƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg‚Ü‚½‚ÍƒT□[ƒo□[ƒvƒ□ƒZƒX) ‚Ì ID□B

DCOM

ActiveX ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ç,¤,µ,ªƒlƒbƒgƒ□□[ƒN,ð‰î,µ,Ä’¼□Ú’Ê□M,Å,«,é,æ,¤,É,·,é,½,ß,ÌƒlƒuƒWƒFƒNƒg ƒvƒ□ƒgƒRƒ‹□BDCOM ,ÍŒ¾Œê,É^Ë‘¶,µ,È,¢,Ì,Å□AActiveX ƒRƒ“ƒ| □[ƒlƒ“ƒg,ð□ì□¬,Å,«,éŒ¾Œê,Å, ,ê,Î□ADCOM ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,à□ì□¬,Å,«,Ü,·□B

ƒhƒ□ƒCƒ“

Windows NT ,É,¨,¢,Ä□AWindows NT ƒT□[ƒo□[ƒlƒbƒgƒ□□[ƒN,ÌŠÇ—□ŽÒ,ª’è‹`,µ□A‹¤’Ê,ÌƒfƒBƒŒƒNƒgƒŠ ƒf□[ƒ^ƒx□[ƒX,ð‹¤—L,·,éƒRƒ“ƒsƒ…□[ƒ^,Ì□W□‡□Bƒhƒ□ƒCƒ“,Å,Í□Aƒhƒ□ƒCƒ“ŠÇ—□ŽÒ,ª□W’†ŠÇ— □,·,éƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,ÆƒOƒ‹□[ƒv ƒAƒJƒEƒ“ƒg,ÉƒAƒNƒZƒX,Å,«,Ü,·□BŠeƒhƒ□ƒCƒ“,É,Íˆêˆ Ó,Ì–¼‘O,ª•t,¢,Ä,¢ ,Ü,·□B

Ž□‘±□« (durability)

□áŠQ,ª”□¶,µ,Ä,à•ÛŽ□,³,ê,é□ó‘Ô□B

ƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL)

ŠÖ□”,ðŽg—p,·,éƒvƒ□ƒZƒX,©,ç“Æ—§,µ,ÄƒRƒ“ƒpƒCƒ‹□AƒŠƒ“ƒN□A,¨,æ,ÑŠi”[,³,ê,é 1
,Â,Ü,½,Í•¡□”,ÌŠÖ□”,ðŠÜ,Þƒtƒ@ƒCƒ‹□BƒlƒyƒŒ□[ƒeƒBƒ“ƒO ƒVƒXƒeƒ€
,Í□AŒÄ,Ñ□o,µƒvƒ□ƒZƒX,Ì‹N“®Žž,Ü,½,ÍŽÀ□sŽž,É□ADLL ,ð,»,Ìƒvƒ□ƒZƒX,ÌƒAƒhƒŒƒX‹óŠÔ,ÉŠ„,è“–,Ä,Ü,·□B

—áŠO

ƒvƒ□ƒOƒ‰ƒ€‚ÌŽÀ□s’†‚É”□¶‚µ□A’Ê□í‚Ì□§Œäƒtƒ□□[‚ÌŠO‘¤‚ÅƒVƒtƒgƒEƒFƒA‚ðŽÀ□s‚µ‚æ‚¤
‚Æ‚·‚éˆÙ□í‚È□ó‘Ô‚Ü‚½‚ÍƒGƒ‰ƒ□[□B

ƒtƒFƒCƒ‹ƒtƒ@□[ƒXƒg

□áŠQ,Ì••,¶□ž,ß,ð—e^Õ,É,·,é,½,ß,Ì Microsoft Transaction Server ,ÌŒ´‘¥□BTransaction Server ,Å—\Šú,µ,È,¢“à•”ƒGƒ‰□[□ó‘Ô,ª”¶,·,é,Æ□A’¼,¿,ÉƒvƒfƒZƒX,ª‹§□I—¹,³,ê□A□áŠQ,ÉŠÖ,·,é□Ú□×,ª Windows NT ,ÌƒCƒxƒ“ƒg ƒ□ƒO,É‹L˜^,³,ê,Ü,·□B

□áŠQ‚Ì•ª—£

□áŠQ‚ðƒVƒXƒefƒ€‚Ì‚Ù‚©‚ÌƒRƒ“ƒ|□[ƒlƒ“ƒg‚ÉŠgŽU‚¹‚¸□A□áŠQ‚Ì‰ºe‹¿‚ðƒRƒ“ƒ|□[ƒlƒ“ƒg“à‚É—}‚¦‚é‚±‚Æ□B

ƒtƒH□[ƒ‹ƒg ƒgƒŒƒ‰ƒ“ƒX

ƒGƒ‰□[□A□áŠQ□A,Ü,½,ÍŠÂ‹«□ó‘Ô,Ì•Ï‰» (’â“d,È,Ç) ,©,ç•œ‹Œ,·,é,½,ß,ÌƒVƒXƒeƒ€‹@”\□B–{—ˆ□AƒtƒH□[ƒ‹ƒg ƒgƒŒƒ‰ƒ“ƒX,Å,Í□Aƒt□[ƒU□[,Ì□ì‹Æ,âƒtƒ@ƒCƒ‹,É‰e‹¿,ð‹y,Ú,·,±,Æ,È,-□AŠ®‘S,ÉŽ©“®,Å•œ‹Œ,ª□s,í,ê,Ü,·□B,±,ê,É‘Î,µ,ÄŽè□ì‹Æ,É,æ,é•û–@,Å,Í□AƒofbƒNƒAƒbƒv ƒtƒ@ƒCƒ‹,©,çƒf□[ƒ^,ð•œŒ³,·,é,È,Ç,Ì□ì‹Æ,ª•K—v,É,È,è,Ü,·□B

ƒOƒ□□[ƒoƒ‹ ƒAƒJƒEƒ“ƒg

ƒt□[ƒU□[,ìƒz□[ƒ€ ƒhƒ□ƒCƒ“,Ì’Ê□í,ìƒt□[ƒU□[ƒAƒJƒEƒ“ƒg□BŠù’è,Ì□Ý’è,Å,Í□A,Ù,Æ,ñ,Ç,ìƒAƒJƒEƒ“ƒg,ª□AƒOƒ□□[ƒoƒ‹ ƒAƒJƒEƒ“ƒg,É□Ý’è,³,ê,Ä,¢,Ü,·□B•¡□”,ìƒhƒ□ƒCƒ“,ª—˜—p,Å,«,é□ê□‡,Í□Aƒlƒbƒgƒ□□[ƒN□ã,ÌŠeƒt□[ƒU□[,ª 1 ,Â,ìƒhƒ□ƒCƒ“,¾,¯,ÉƒOƒ□□[ƒoƒ‹ ƒAƒJƒEƒ“ƒg,ð 1 ,Â,¾,¯Ž□,Â,æ,¤,É,·,é,Ì,ª—□‘z“I,Å,·□B

ƒOƒ‹□[ƒv

1 ,Âˆ È□ã,Ì Windows NT ƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,Ì□W,Ü,è,ðŽ¯•Ê,·,é–¼‘O□B

ID

ƒpƒbƒP□[ƒW,ÉƒAƒNƒZƒX,Å,«,éƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,ðŽw’è,·,éƒpƒbƒP□[ƒW ƒvƒ□ƒpƒeƒB□BWindows NT ƒhƒ□ƒCƒ“,Ì“Á’è,Ìƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,Ü,½,Íƒt□[ƒU□[,ÌƒOƒ‹□[ƒv,ðŽw’è,Å,«,Ü,·□B

ƒCƒ“ ƒ_ƒEƒg‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ð’²□®‚·‚éƒT□[ƒo□[‚ª—˜—p‚Å‚«‚È‚¢‚½‚ß‚É□A□€”õ‚□®‚Á‚Ä‚¢
‚Ä‚à□AƒRƒ~ƒbƒg‚·‚é‚©ƒAƒ{□[ƒg‚·‚é‚©‚Ì”»’f‚ª‰º‚ª‚ê‚Ä‚¢‚È‚¢ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“□B

ƒCƒ“ƒvƒ□ƒZƒX ƒRƒ“ƒ|□[ƒlƒ“ƒg

ƒNƒ‰ƒCƒAƒ“ƒg,Ìƒvƒ□ƒZƒX‹óŠÔ,Å“®□ì,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg□B’Ê□í,Í□Aƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL) ,É,È,è,Ü,·□B

ƒCƒ“ƒXƒ^ƒ“ƒX

“Á’è,ÌƒRƒ“ƒ|[ƒlƒ“ƒg ƒNƒ‰ƒX,ÌƒIƒuƒWƒFƒNƒg□BŠeƒCƒ“ƒXƒ^ƒ“ƒX,É,Í□A“ÆŽ©,Ìƒvƒ‰ƒCƒx□[ƒg,Èƒf□[ƒ^—v‘f,âƒ□ƒ“ƒo•Ï□”,ª, ,è,Ü,·□BƒRƒ“ƒ|□[ƒlƒ“ƒg ƒCƒ“ƒXƒ^ƒ“ƒX,Í□AƒIƒuƒWƒFƒNƒg,Æ“¯,¶,à,Ì,ðˆÓ–¡,µ,Ü,·□B

‘Î˜bŒ^ƒ□ƒOƒIƒ“ ƒt□[ƒU□[

Œ»□Ý□AMicrosoft Transaction Server ƒRƒ“ƒsƒ…□[ƒ^,Éƒ□ƒOƒIƒ“,µ,Ä,¢,éƒt□[ƒU□[□B

ƒCƒ“ƒ^□[ƒtƒFƒCƒX

˜_—□“I‚ÉŠÖ˜A‚µ‚Ä‚¢‚é‘€□ì‚Ü‚½‚Íƒ□ƒ\ƒbƒh‚ìƒOƒ‹□[ƒv‚Å□AƒRƒ“ƒ|□[ƒlƒ“ƒg
ƒIƒuƒWƒFƒNƒg‚Ö‚ìƒAƒNƒZƒX‚ð’ñ‹Ÿ‚µ‚Ü‚·□B

•ª—£□« (isolation)

•À□s,µ,ÄŽÀ□s,³,ê,Ä,¢,é 2 ,Â,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª□A“¯ŽžŽÀ□s,³,ê,Ä,¢,È,¢,æ,¤,ÉŒ©,¹,©,¯,é“Á□«□B,±,Ì“Á□«,É,æ,Á,Ä□AƒVƒXƒeƒ€,ª^ê“x,É 1 ,Â,Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,¾,¯,ðŽÀ□s,µ,Ä,¢,é,æ,¤,ÉŒ©,¦,Ü,·□B

ƒWƒƒXƒgƒCƒ“ƒ^ƒCƒ€ ƒAƒNƒeƒBƒx□[ƒVƒ‡ƒ“

ƒNƒ‰ƒCƒAƒ“ƒg,©,ç,Ì—v‹□,ðŽÀ□s,·,é,Ì,É•K—v,È,Æ,«,¾,¯ Microsoft Transaction Server ƒIƒuƒWƒFƒNƒg,ðƒAƒNƒeƒBƒu‰»,Å,«,é‹@”\□BƒNƒ‰ƒCƒAƒ“ƒg,ªƒIƒuƒWƒFƒNƒg,Ö,ÌŽQ□Æ,ð•ÛŽ□,µ,Ä,¢ ,éŠÔ,à,»,ÌƒIƒuƒWƒFƒNƒg,ð”ñƒAƒNƒeƒBƒu‰»,Å,«,é,½,ß□AƒAƒCƒhƒ‹□ó‘Ô,ÌƒT□[ƒo□[ƒŠƒ\□[ƒX,ð,æ,èŒø—¦“I,É—˜—p,Å,«,Ü,·□B

ƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW

ƒpƒbƒP□[ƒW,ð□ì□¬,·,éƒNƒ‰ƒCƒAƒ“ƒg,Ìƒvƒ□ƒZƒX“à,ÅŽÀ□s,³,ê,éƒpƒbƒP□[ƒW□Bƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì’Ç□Õ□Aƒ□□[ƒ‹,Ìƒ`ƒFƒbƒN□Aƒvƒ□ƒZƒX,Ì•ª—£,ðƒTƒ|□[ƒg,µ,Ü,¹,ñ□BMTS ,Í□Aƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,ÆƒT□[ƒo□[ƒpƒbƒP□[ƒW,Ì 2 Ží—Þ,Ìƒpƒbƒ P□[ƒW,ðƒTƒ|□[ƒg,µ,Ü,·□B

•‰‰×,ìƒoƒ‰ƒ“ƒX

ƒlƒbƒgƒ□□[ƒN□ì‹Æ,É,æ,é□^—□,Ì•‰
‰×,ð•¡□”,ìƒT□[ƒo□[,É•ªŽU,µ□Aƒlƒbƒgƒ□□[ƒN‘S‘Ì,ìƒpƒtƒH□[ƒ}ƒ“ƒX,ðŒü□ã,³,¹,é,±,Æ□B

ƒ□□[ƒJƒ‹ ƒAƒJƒEƒ“ƒg

□M—Š‚³‚ê‚é‘¤‚ÌƒhƒfƒCƒ“‚É’Ê□íŽg‚¤ƒAƒJƒEƒ“ƒg‚ðŽ□‚½‚È‚¢ƒt□[ƒU□[‚Ì‚½‚ß‚É□Aƒ□□[ƒJƒ‹ ƒhƒ□ƒfƒCƒ“‚Å—^‚¦‚ç‚ê‚éƒAƒJƒEƒ“ƒg□Bƒ□□[ƒJƒ‹ ƒAƒJƒEƒ“ƒg‚Í‘Î˜bŒ^ƒ□ƒOƒIƒ“‚ÉŽg‚¤‚±‚Æ‚Í‚Å‚«‚Ü‚¹‚ñ□B1 ‚Â‚Ìƒhƒ□ƒfƒCƒ“‚Å□ì□¬‚³‚ê‚½ƒ□□[ƒJƒ‹ ƒAƒJƒEƒ“ƒg‚Í□A□M—Š‚³‚ê‚é‘¤‚Ìƒhƒ□ƒfƒCƒ“‚Å‚ÍŽg—p‚Å‚«‚Ü‚¹‚ñ□B

ƒ□ƒCƒ“ ƒXƒŒƒbƒh

"ƒVƒ“ƒOƒ‹ ƒXƒŒƒbƒh" ,Æ,µ,ÄŽw’è,³,ê,½ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ì,·,×,Ä,ÌƒIƒuƒWƒFƒNƒg,ðŽÀ□s,·,é,Æ,«,ÉŽg—p,³,ê,é’P‘ê,ÌƒXƒŒƒbƒh□BƒAƒp□[ƒgƒ□ƒ“ƒg ƒXƒŒƒbƒh,àŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒ}□[ƒVƒƒƒŠƒ“ƒO

ƒCƒ“ƒ^□[ƒtƒFƒCƒX ƒ□ƒ\ƒbƒh,Ìƒpƒ‰ƒ□□[ƒ^,ð□AƒXƒCŒƒbƒh,âƒvƒ□ƒZƒX,Ì‹«ŠE,ð‰z,¦,ÄƒpƒbƒP□[ƒW‰»,µ,Ä‘—,é□^—□□B

ƒ□ƒ\ƒbƒh

ƒlƒuƒWƒFƒNƒg‚É‘Î‚µ‚Ä□ì—p‚·‚éƒvƒ□ƒV□[ƒWƒƒ (ŠÖ□”)□B

Microsoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒl□[ƒ^ (MS DTC)

•¡□”,ÌƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,É,Ü,½,ª,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ð'²□®,·,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ}ƒl□[ƒWƒƒ□B•¡□”,ÌƒRƒ“ƒsƒ…□[ƒ^□ã,É'¶□Ý,·,é‰Â"\□«,Ì, ,é•¡□”,ÌƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,É□^—□,ª,Ü,½,ª,é□ê□‡,Å,à□A,»,Ì□^—□,ð 1 ,Â,ÌƒAƒgƒ~ƒbƒN ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,Æ,µ,ÄƒRƒ~ƒbƒg,Å,«,Ü,·□B

Microsoft Transaction Server ƒRƒ“ƒ|□[ƒlƒ“ƒg

Microsoft Transaction Server ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÅŽÀ□s,³,ê,é COM ƒRƒ“ƒ|□[ƒlƒ“ƒg□BTransaction Server ƒRƒ“ƒ|□[ƒlƒ“ƒg,Í□Aƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL) ,Å□AƒlƒuƒWƒFƒNƒg,ð□ì□¬,·,é,½,ß,ÌƒNƒ‰ƒX ƒtƒ@ƒNƒgƒŠ,ðŽÀ‘•,µ□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒCƒ“ƒ^□[ƒtƒFƒCƒX,Ì,·,×,Ä,ª•W□€ƒ}□[ƒVƒƒŠƒ“ƒO,Ì,½,ß,Éƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ,É‹L□q,³,ê,Ä,¢,é•K—v,ª, ,è,Ü,·□B

Microsoft Transaction Server ƒGƒNƒXƒvƒ□□[ƒ‰

Microsoft Transaction Server ƒRƒ“ƒ|□[ƒlƒ“ƒg,ð•ªŽUƒRƒ“ƒsƒ…□[ƒ^ ƒlƒbƒgƒ□□[ƒN,Å□\□¬,µ□AŠÇ—□,·,é,½,ß,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“□B

Microsoft Transaction Server ƒIƒuƒWƒFƒNƒg

Microsoft Transaction Server ƒ‰ƒ“ƒ^ƒCƒ€ŠÂ‹«,ÅŽÀ□s,³,ê,é COM ƒIƒuƒWƒFƒNƒg,Å□ATransaction Server ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒ‚ƒƒ‹,Æ”z’uƒ‚ƒƒ‹,É□€‹’,·,é,à,Ì,Å,·□B

Null

ƒf□[ƒ^‚ª‚È‚¢‚±‚Æ□A‚Ü‚½‚Í•s–¾‚Å‚ ‚é‚±‚Æ‚ðŽ¦‚·’l□B

ƒIƒuƒWƒFƒNƒg

COM ƒRƒ“ƒ|□[ƒIƒ“ƒg,Ìƒ‰ƒ“ƒ^ƒCƒ€ ƒCƒ“ƒXƒ^ƒ“ƒX□BƒIƒuƒWƒFƒNƒg,ÍƒRƒ“ƒ|□[ƒIƒ“ƒg,ÌƒNƒ‰ƒX ƒtƒ@ƒNƒgƒŠ,É,æ,Á,Ä□ì□¬,³,ê,Ü,·□BƒIƒuƒWƒFƒNƒg,ÍƒCƒ“ƒXƒ^ƒ“ƒX,Æ“¯,¶,à,Ì,ðˆÓ–¡,µ,Ü,·□B

ƒIƒuƒWƒFƒNƒg•Ï□”

ƒIƒuƒWƒFƒNƒg,Ö,ÌŽQ□Æ,ðŠÜ,Þ•Ï□”□B

ODBC ƒŠƒ\□[ƒX ƒfƒBƒXƒyƒ“ƒT

•W□€ ODBC ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒO ƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŽg—p,·,é Microsoft Transaction Server ƒRƒ“ƒ|□[ƒlƒ“ƒg,Ìƒf□[ƒ^ƒx□[ƒX□Ú‘±,Ìƒv□[ƒ‹,ðŠÇ—□,·,éƒŠƒ\□[ƒX ƒfƒBƒXƒyƒ“ƒT□B

OLE ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“

ƒRƒ“ƒ|□[ƒlƒ“ƒg ƒIƒuƒWƒFƒNƒg ƒ‚ƒƒ‹ (COM) ‚ÉŠî‚Ã‚ƒIƒuƒWƒFƒNƒgŽwŒü‚Ì 2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg ƒvƒ□ƒgƒRƒ‹□BMicrosoft •ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒR□[ƒfƒBƒ□[ƒ^ (DTC) ‚É‚æ‚Á‚Ä²□®‚³‚ê‚é•ªŽUƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚ÉŽQ‰Á‚·‚é‚½‚ß‚ÉƒSƒ\□[ƒX ƒ}ƒ□[ƒWƒƒ‚ªŽg—p‚µ‚Ü‚·□B

ƒl□[ƒvƒ“ ƒf□[ƒ^ƒx□[ƒX ƒRƒlƒNƒeƒBƒrƒeƒB (ODBC)

,³,Ü,´,Ü,Èƒf□[ƒ^ ƒ\□[ƒX,É□Ú‘±,·,é,Æ,«,ÉŽg,¤•W□€ƒvƒ□ƒOƒ‰ƒ~ƒ“ƒOŒ¾ŒêƒCƒ“ƒ^□[ƒtƒFƒCƒX□B

□Å□‰‚ÌŒÄ‚Ñ□o‚µŒ³

□^—□‚ðŠJŽn‚µ‚½ƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg‚Ì ID□B

□Å□‰,Ì□ì□¬Œ³

Œ»□Ý,ìƒIƒuƒWƒFƒNƒg,ð□ì□¬,µ,½ƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg,Ì ID□B□Å□‰,Ì□ì□¬Œ³,ªƒIƒuƒWƒFƒNƒg,ð,Ù,©,ìƒx□[ƒX ƒNƒ‰ƒCƒAƒ“ƒg,É"n,µ,½□ê□‡,¾,¯□A□Å□‰,ÌŒÄ,Ñ□o,µŒ³,Æ□Å□‰,Ì□ì□¬Œ³,ª•Ê,É,È,è,Ü,·□B□Å□‰ ,ÌŒÄ,Ñ□o,µŒ³,àŽQ□Æ,µ,Ä,,¾,³,¢□B

ƒvƒ□ƒZƒXŠOƒRƒ“ƒ|□[ƒlƒ“ƒg

ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒNƒ‰ƒCƒAƒ“ƒg,Æ,Í•Ê,Ìƒvƒ□ƒZƒX‹óŠÔ,Å“®□ì,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg□BMicrosoft Transaction Server ,Å,Í□AƒRƒ“ƒ|□[ƒlƒ“ƒg,ð‘ã—□ƒT□[ƒo□[ƒvƒ□ƒZƒX,Éƒ□□[ƒh,·,é,±,Æ,É,æ,è□ADLL ,Æ,µ,ÄŽÀ‘•,³,ê,Ä,¢ ,éƒRƒ“ƒ|□[ƒlƒ“ƒg,ðƒNƒ‰ƒCƒAƒ“ƒg,Ìƒvƒ□ƒZƒX,ÌŠO•”,ÅŽg—p,Å,«,Ü,·□B

ƒpƒbƒP□[ƒW

ŠÖ˜A,·,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‹@”\,ðŽÀ□s,·,é 1 ‘g,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg□BƒpƒbƒP□[ƒW“à,Ì,·,×,Ä,ÌƒRƒ“ƒ|□[ƒlƒ“ƒg,Í□A“¯,¶ Microsoft Transaction Server ƒT□[ƒo□[ƒvƒ□ƒZƒX,Å“®□ì,µ,Ü,·□BƒpƒbƒP□[ƒW,Í□AƒZƒLƒ…ƒŠƒeƒB□î•ñ,ª,¢ ,ÂŠm”F,³,ê,é,©,ð’è‹`,·,é□M—Š‹«ŠE,Å, ,è□A1 ‘g,ÌƒRƒ“ƒ| □[ƒlƒ“ƒg,Ì”z’u’PˆÊ,Å,à, ,è,Ü,·□BƒpƒbƒP□[ƒW,ð□ì□¬,·,é,É,Í□ATransaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ðŽg,¢ ,Ü,·□BƒpƒbƒP□[ƒW,Í□Aƒ‰ƒCƒuƒ‰ƒŠ ƒpƒbƒP□[ƒW,Ü,½,ÍƒT□[ƒo□[ƒpƒbƒP□[ƒW,Ì,¢,¸,ê,©,Å,·□B

ƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹

ƒpƒbƒP□[ƒW,Ìƒ□□[ƒ‹,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉŠÖ,·,é□î•ñ,ªŠi”[,³,ê,Ä,¢,éƒtƒ@ƒCƒ‹□BƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,ð□ì□¬,·,é,É,Í□ATransaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ÌƒpƒbƒP□[ƒW ƒGƒNƒXƒ|□[ƒg‹@”\,ðŽg,¢,Ü,·□BŠù□¬,ÌƒpƒbƒP□[ƒW,ð□ì□¬,·,é,Æ□AŠÖ˜A,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹ (DLL□Aƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ□A,¨,æ,ÑŽÀ‘•,³,ê,Ä,¢,ê,Îƒvƒ□ƒLƒV/ƒXƒ^ƒu DLL) ,ª□A□ì□¬,³,ê,½ƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,Æ“¯,¶ƒffƒBƒŒƒNƒgƒŠ,ÉƒRƒs□[,³,ê,Ü,·□B

ƒv□[ƒŠƒ“ƒO

ƒIƒuƒWƒFƒNƒg□Aƒf□[ƒ^ƒx□[ƒX□Ú‘±,È,Ç□AŠù,ÉŠ„,è“–,Ä,ç,ê,½ƒŠƒ\□[ƒX,Ì□W□‡,ð—˜—p,µ,ÄƒpƒtƒH□[ƒ}ƒ“ƒX,ð□Å“K‰»,·,é,±,Æ□Bƒv□[ƒŠƒ“ƒO,É,æ,è□AƒŠƒ\□[ƒXŠ„,è“–,Ä,ð,æ,èŒø—¦“I,É□s,¤,±,Æ,ª,Å,«,Ü,·□B

Šù□¬,ìƒpƒbƒP□[ƒW

ƒpƒbƒP□[ƒW,ìƒ□□[ƒ‹,ÆƒRƒ“ƒ|□[ƒlƒ“ƒg,ÉŠÖ,·,é□î•ñ,ªŠi”[,³,ê,Ä,¢,éƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹□BƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,ð□ì□¬,·,é,É,Í□ATransaction Server ƒGƒNƒXƒvƒ□□[ƒ‰,ìƒpƒbƒP□[ƒW ƒGƒNƒXƒ|□[ƒg‹@”\,ðŽg,¢,Ü,·□BŠù□¬,ìƒpƒbƒP□[ƒW,ð□ì□¬,·,é,Æ□AŠÖ˜A,·,éƒRƒ“ƒ|□[ƒlƒ“ƒg ƒtƒ@ƒCƒ‹ (DLL□Aƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ□A,¨,æ,ÑŽÀ‘•,³,ê,Ä,¢,ê,Îƒvƒ□ƒLƒV/ƒXƒ^ƒu DLL) ,ª□A□ì□¬,³,ê,½ƒpƒbƒP□[ƒW ƒtƒ@ƒCƒ‹,Æ“¯,¶ƒfƒBƒŒƒNƒgƒŠ,ÉƒRƒs□[,³,ê,Ü,·□B

ƒvƒ□ƒZƒX‚Ì•ª—£

‚ ‚éƒT□[ƒo□[ƒvƒ□ƒZƒX‚ð‚Ù‚©‚ÌƒT□[ƒo□[ƒvƒ□ƒZƒX‚©‚ç•ª—£‚·‚é‚½‚ß‚É□A•Ê‚Ìƒ□ƒ‚ƒŠ—Ìˆæ‚ÅŽÀ□s‚·‚é•û–@□Bƒvƒ□ƒZƒX‚Ì•ª—£‚É‚æ‚Á‚Ä□AƒT□[ƒo□[ƒvƒ□ƒZƒX‚Í‚Ù‚©‚ÌƒvƒŠƒP□[ƒVƒ‡ƒ“‚Ì’v–½“I‚ÈƒGƒ‰□[‚©‚ç•ÛŒì‚³‚ê‚Ü‚·□B‚Ü‚½□AƒT□[ƒo□[ƒvƒ□ƒZƒX‚ð•ª—£‚·‚é‚±‚Æ‚É‚æ‚è□A•ª—£‚³‚ê‚½ƒvƒ□ƒZƒX‚ªƒAƒvƒŠƒP□[ƒVƒ‡ƒ“‚Ì’v–½“I‚ÈƒGƒ‰□[‚É‚æ‚Á‚Ä‚¹‚¹‚¹‚é‚±‚Æ‚à–hŽ~‚Å‚«‚Ü‚·□Bƒvƒ□ƒZƒX‚Ì•ª—£‚ðƒTƒ|□[ƒg‚·‚é MTS ƒpƒbƒP□[ƒW‚Í□AƒT□[ƒo□[ƒpƒbƒP□[ƒW‚ÆŒÄ‚Î‚ê‚Ü‚·□B

ƒvƒ□ƒOƒ‰ƒ€ ID (progID)

COM ƒRƒ“ƒ|□[ƒlƒ“ƒg,ðŽ¯•Ê,·,é–¼‘O□B,½,Æ,¦,Î□ABank.MoveMoney ,ðƒvƒ□ƒOƒ‰ƒ€ ID ,É,·,é,± ,Æ,ª,Å,«,Ü,·□B

ƒvƒ□ƒOƒ‰ƒ€‚É‚æ‚éƒZƒLƒ…ƒŠƒeƒB

ƒRƒ“ƒ|□[ƒlƒ“ƒg‚É‚æ‚è‘ñ‹Ÿ‚³‚ê‚éŽè‘±‚«‚É‚æ‚éƒ□ƒWƒbƒN‚Å□A—v‹□‚³‚ê‚½‘€□ì‚ðŽÀ□s‚·‚éŒ ŒÀ‚ªƒNƒ‰ƒCƒAƒ“ƒg‚É‚ ‚é‚©‚Ç‚¤‚©‚ð”»’f‚µ‚Ü‚·□B□éŒ¾‚É‚æ‚éƒZƒLƒ…ƒŠƒeƒB‚àŽQ□Æ‚µ‚Ä‚,,¾‚³‚¢□B

ƒvƒ□ƒLƒV

ˆÙ,È,éƒXƒŒƒbƒh,âˆÙ,È,éƒvƒ□ƒZƒX,È,Ç□AˆÙ,È,éŽÀ□sŠÂ‹«,ÅŽÀ□s,³,ê,Ä,¢,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg,ðŒÄ,Ñ□o,·,½,ß,ÉƒNƒ‰ƒCƒAƒ“ƒg,ª•K—v,Æ,·,éƒpƒ‰ƒ□□[ƒ^ ƒ}□[ƒVƒƒŠƒ“ƒO,â'Ê□M,ð□s,¤ƒCƒ“ƒ^□[ƒtƒFƒCƒXŒÅ—L,ÌƒIƒuƒWƒFƒNƒg□Bƒvƒ□ƒLƒV,Í□AƒNƒ ‰ƒCƒAƒ“ƒg,Æ"¯,¶□ê□Š,É, ,è□A'Î‰ž,·,éƒXƒ^ƒu,Æ'Ê□M,µ,Ü,·□B'Î‰ž,·,éƒXƒ^ƒu,Í□AŒÄ,Ñ□o,³,ê,Ä,¢ ,éƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg,Æ"¯,¶□ê□Š,É, ,è,Ü,·□B

ƒŠƒ‚□[ƒg ƒRƒ“ƒ|□[ƒlƒ“ƒg

ˆÙ‚È‚éƒRƒ“ƒsƒ…□[ƒ^□ã‚ÌƒNƒ‰ƒCƒAƒ“ƒg‚ª—˜—p‚·‚éƒRƒ“ƒ|□[ƒlƒ“ƒg□B

ƒŠƒ‚□[ƒg ƒvƒ□ƒV□[ƒWƒƒ ƒR□[ƒ‹ (RPC)

, ,éƒvƒ□ƒZƒX‚©‚ç•Ê‚Ìƒvƒ□ƒZƒX‚ÅŽÀ□s‚³‚ê‚Ä‚¢‚é‹@”\‚ðŒÄ‚Ñ□o‚·‚½‚ß‚Ì‹KŠi□Bƒvƒ□ƒZƒX‚Í“¯‚¶ƒRƒ“ƒsƒ…□[ƒ^□ã‚É‚ ‚Á‚Ä‚à□Aƒlƒbƒgƒ□□[ƒN□ã‚Ì‘Ù‚È‚éƒRƒ“ƒsƒ…□[ƒ^□ã‚É‚ ‚Á‚Ä‚à‚©‚Ü‚¢‚ñ□B

ƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“

, ,éƒRƒ“ƒsƒ…□[ƒ^,©,ç•Ê,ÌƒRƒ“ƒsƒ…□[ƒ^,ÉƒJƒ^ƒ□ƒO,ðƒRƒs□[,·,é‘€□ì□BƒŒƒvƒŠƒP□[ƒVƒ‡ƒ“,Í□AƒNƒ‰ƒXƒ^‰»,³,ê,½•¡□”,Ì MTS ƒT□[ƒo□[,ð“¯Šú‰»,·,é,½,ß,ÉŽg,¢,Ü,·□B

ƒŠƒ\□[ƒX ƒfƒBƒXƒyƒ“ƒT

ƒvƒ□ƒZƒX“à‚ÌŽ□‘±□«‚Ì‚È‚¢ƒŠƒ\□[ƒX‚Ì“¯Šú‚ÆŠÇ—□‚ð□s‚¢□AMicrosoft Transaction Server ƒIƒuƒWƒFƒNƒg‚ªŠÈ’P‚É□A‚µ‚©‚àŒø—¦‚æ‚ƒŠƒ\□[ƒX‚ð‹¤—L‚Å‚«‚é‚æ‚¤‚É‚·‚é‚½‚ßƒT□[ƒrƒX□B‚½‚Æ‚¦‚Î□AODBC ƒŠƒ\□[ƒX ƒfƒBƒXƒyƒ“ƒT‚Íƒf□[ƒ^ƒx□[ƒX□Ú‘±‚Ìƒv□[ƒ‹‚ðŠÇ—□‚µ‚Ü‚·□B

ƒŠƒ\□[ƒX ƒffBƒXƒyƒ“ƒT ƒ}ƒl□[ƒWƒƒ

ƒŠƒ\□[ƒX ƒffBƒXƒyƒ“ƒT,Ì□W□‡,Ì“à•”,Å□ˆ—□,ð’²□®,·,éƒ_ƒCƒiƒ~ƒbƒN ƒŠƒ“ƒN ƒ‰ƒCƒuƒ‰ƒŠ (DLL)□B

ƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ

Ž□‘±“I,Èƒf□[ƒ^,ðŠÇ—□,·,éƒVƒXƒeƒ€ ƒT□[ƒrƒX□BƒT□[ƒo□[ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,Í□AƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,ðŽg,Á,Ä□AŽèŽ□,¿,Ì□ÝŒÉ□AŽó’□Žc,è□A”„Š|‹à,ÌƒŒƒR□[ƒh,È,Ç,ÌƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌŽ□‘±“I,È□ó‘Ô, ð•ÛŽ□,µ,Ü,·□BƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,Íƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ}ƒl□[ƒWƒƒ,Æ‹¦²,µ,Ä“®□ì,µ□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,É‘Î,µ,Ä (2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg ƒvƒ□ƒgƒRƒ‹,ðŽg,Á,Ä) Œ´Žq□« (atomicity) ,Æ•ª—£□« (isolation) ,ð•Û□Ø,µ,Ü,·□BMicrosoft SQL Server ,ÍƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,Ì^ê— á,Å,·□B

ƒ□□[ƒ‹

ˆê˜A,ìƒRƒ“ƒ|□[ƒlƒ“ƒg,ìƒt□[ƒU□[,Ì•ª—Þ,ð’è‹`,·,éƒVƒ“ƒ{ƒ‹–¼□BŠeƒ□□[ƒ‹,É,Í□A,Ç,ìƒt□[ƒU□[,ªƒRƒ“ƒ|□[ƒlƒ“ƒg,ìƒCƒ“ƒ^□[ƒtƒFƒCƒX,ðŒÄ,Ñ□o,·,±,Æ,ª,Å,«,é,©,ª’è‹`,³,ê,Ä,¢,Ü,·□B

ˆÀ‘S‚ÈŽQ‚Æ

Œ»‚Ý‚ÌƒuƒWƒFƒNƒg‚ðRƒ“ƒeƒLƒXƒg‚ÌŠO•”‚É“n‚µ‚Ä‚àˆÀ‘S‚È‚ACŒ»‚Ý‚ÌƒuƒWƒFƒNƒg‚Ö‚ÌŽQ‚Æ‚B

ƒZƒLƒ…ƒŠƒefB ID (SID)

ƒZƒLƒ…ƒŠƒefB ƒVƒXƒef€‚Éƒ□ƒOƒlƒ“‚µ‚½ƒt□[ƒU□[‚ðŽ¯•Ê‚·‚é‚½‚ß‚Ìˆê‚Ó‚Ì–¼‘O□BƒZƒLƒ…ƒŠƒefB ID (SID) ‚Å 1 □l‚Ìƒt□[ƒU□[‚ðŽ¯•Ê‚·‚é□ê□‡‚à□Aƒt□[ƒU□[‚ÌƒOƒ‹□[ƒv‚ðŽ¯•Ê‚·‚é□ê□‡‚à‚ ‚è‚Ü‚·□B

ƒZƒ}ƒtƒH

ƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ‚âƒŠƒ\□[ƒX ƒffƒBƒXƒyƒ“ƒT‚Ì“à•”‚ÅŽg—p‚³‚ê‚éƒ□ƒbƒN‹@□\□BƒZƒ}ƒtƒH‚É‚ÍVƒ“ƒ{ƒ‹–¼‚ª‚È‚-□A‹¤—Lƒ‚□[ƒh‚Æ”r‘¼ƒ‚□[ƒh‚Å‚¾‚¯ƒAƒNƒZƒX‚³‚ê‚Ü‚·□B‚Ü‚½□Affƒbƒhƒ□ƒbƒN‚ÍŒŸ□o‚³‚ê‚¸□AŽ©“®“I‚È‰ð•ú‚âƒRƒ~ƒbƒg‚à‚è‚Ü‚¹‚ñ□B

ƒT□[ƒo□[ƒpƒbƒP□[ƒW

ƒ□□[ƒJƒ‹ ƒRƒ“ƒsƒ…□[ƒ^□ã‚Ìƒpƒbƒ‰P□[ƒW“ÆŽ©‚Ìƒvƒ□ƒZƒX“à‚Å•ª—£
‚³‚ê‚ÄŽÀ□s‚³‚ê‚éƒpƒbƒP□[ƒW□BƒT□[ƒo□[ƒpƒbƒP□[ƒW‚Í□Aƒ□□[ƒ‹‚ÉŠî‚Ã‚ƒZƒLƒ…ƒŠƒeƒB□AƒŠƒ\□[ƒX‚Ì‹¤—
L□Aƒvƒ□ƒZƒX‚Ì•ª—£□A‚¨‚æ‚Ñ (ƒpƒbƒP□[ƒW‚Ì’Ç□Õ‚È‚Ç‚Ì) ƒvƒ□ƒZƒXŠÇ—□‚ðƒTƒ|□[ƒg‚µ‚Ü‚·□BMTS ‚Í□Aƒ‰ƒCƒuƒ
‰ƒŠ ƒpƒbƒP□[ƒW‚ÆƒT□[ƒo□[ƒpƒbƒP□[ƒW‚Ì 2 Ží—Þ‚Ìƒpƒbƒ‰P□[ƒW‚ðƒTƒ|□[ƒg‚µ‚Ü‚·□B

ƒT□[ƒo□[ƒvƒ□ƒZƒX

Microsoft Transaction Server ƒRƒ“ƒ|□[ƒlƒ“ƒg,ÌƒzƒXƒg,Æ,È,éƒvƒ□ƒZƒX□BMicrosoft Transaction Server ƒRƒ“ƒ|□[ƒlƒ“ƒg,í□AƒNƒ‰ƒCƒAƒ“ƒg,ÌƒRƒ“ƒsƒ…□[ƒ^,Ì‘ã—□ƒT□[ƒo□[ƒvƒ□ƒZƒX□A,Ü,½,ÍƒNƒ‰ƒCƒAƒ“ƒg ƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒvƒ□ƒZƒX,Ì,¢,¸,ê,©,Éƒ□□[ƒh,·,é,±,Æ,ª,Å,«,Ü,·□B

‹¤—Lƒvƒ□ƒpƒeƒB

“¯ˆêƒT□[ƒo□[ƒvƒ□ƒZƒX“à‚Ì‚·‚×‚Ä‚ÌƒIƒuƒWƒFƒNƒg‚©‚ç‹¤—Lƒvƒ□ƒpƒeƒB ƒ}ƒl□[ƒWƒƒ‚ð‰î‚µ‚Ä—˜—p‚Å‚«‚é•Ï□”□B‹¤—Lƒvƒ□ƒpƒeƒB‚É‚Í□AƒoƒŠƒAƒ“ƒgŒˆ‚Å•\‚·‚±‚Æ‚Ì‚Å‚«‚é‚ ‚ç‚ä‚éŒˆ‚Ì‚Ì‚ð•ÛŽ□‚·‚é‚±‚Æ‚ª‚Å‚«‚Ü‚·□B

ƒXƒiƒbƒvƒCƒ“

Microsoft Management Console (MMC) ,ªƒzƒXƒg‚Æ‚È‚éŠÇ—□ƒvƒ□ƒOƒ‰ƒ€□BWindows NT □ã‚Ì MTS ƒGƒNƒXƒvƒ□□[ƒ‰‚ÍƒXƒiƒbƒvƒCƒ“‚Å‚·□B

ƒXƒe□[ƒgƒtƒ‹ ƒIƒuƒWƒFƒNƒg

1 ,Âˆ È□ã,ÌƒNƒ‰ƒCƒAƒ“ƒg,©,ç,ÌŒÄ,Ñ□o,µ,É,æ,è’~□Ï,³,ê,½ƒvƒ‰ƒCƒx□[ƒg,È□ó‘Ô,ð•ÛŽ□,·,éƒIƒuƒWƒFƒNƒg□B

ƒXƒe□[ƒgƒŒƒX ƒIƒuƒWƒFƒNƒg

1 ,Âˆè□ã,ÌƒNƒ‰ƒCƒAƒ“ƒg,©,ç,ÌŒÄ,Ñ□o,µ,É,æ,è’~□Ï,³,ê,½ƒvƒ‰ƒCƒx□[ƒg,È□ó‘Ô,ð•ÛŽ□,µ,È,¢ƒIƒuƒWƒFƒNƒg□B

•¶Žš—ñŽ®

•¶Žš‚ª˜A‘±‚µ‚Ä•À‚Ñ□A•¶Žš—ñ‚Æ‚µ‚Ä]‰¿‚Å‚«‚éŽ®□B

ƒXƒ^ƒu

ˆÙ,È,éƒXƒŒƒbƒh,âˆÙ,È,éƒvƒ□ƒZƒX,È,Ç□AˆÙ,È,éŽÀ□sŠÂ‹«,ÅŽÀ□s,³,ê,Ä,¢,éƒNƒ
‰ƒCƒAƒ“ƒg,©,ç,ÌŒÄ,Ñ□o,µ,ðŽó,¯,é,½,ß,ÉƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg,ª•K—v,Æ,·,éƒpƒ‰ƒ□□[ƒ^
ƒ}□[ƒVƒƒŠƒ“ƒO,â'Ê□M,ð□s,¤ƒCƒ“ƒ^□[ƒtƒFƒCƒXŒÅ—L,ÌƒIƒuƒWƒFƒNƒg□BƒXƒ^ƒu,ÍƒAƒvƒŠƒP□[ƒVƒ‡ƒ“
ƒIƒuƒWƒFƒNƒg,Æ"¯,¶□ê□Š,É, ,è□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“ ƒIƒuƒWƒFƒNƒg,ðŒÄ,Ñ□o,µ,Ä,¢,éƒNƒ
‰ƒCƒAƒ“ƒg,Æ"¯,¶□ê□Š,É, ,é□A'Î‰ž,·,éƒvƒ□ƒLƒV,Æ'Ê□M,µ,Ü,·□B

ƒXƒfŒƒbƒh

ƒIƒyƒfŒ□[ƒeƒBƒ“ƒO ƒVƒXƒeƒ€,ª CPU ŽžŠÔ,ðŠ„,è“–,Ä,éŠî–
{'P^Ê□BƒXƒfŒƒbƒh,Í□AŒ»□Ý•Ê,ÌƒXƒfŒƒbƒh,ªŽÀ□s,µ,Ä,¢
,é•”•ª,ðŠÜ,ß□AƒAƒvƒŠƒP□[ƒVƒ‡ƒ“,ÌƒR□[ƒh,Ì,Ç,Ì•”•ª,Å,àŽÀ□s,Å,«,Ü,·□B1
,Â,ÌƒvƒZƒX,Ì,·,×,Ä,ÌƒXƒfŒƒbƒh,Í□A,»,ÌƒvƒZƒX,Ì
‰¼‘zƒAƒhƒfŒƒX‹óŠÔ□AƒOƒ□□[ƒof‹•Ï□”□A,¨,æ,ÑƒIƒyƒfŒ□[ƒeƒBƒ“ƒO ƒVƒXƒef€,ÌƒŠƒ\□[ƒX,ð‹¤—L,µ,Ä,¢,Ü,·□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“

ƒAƒgƒ~ƒbƒN,È□ˆ—□,Æ,µ,ÄŽÀ□s,³,ê,é□ì‹Æ’Pˆ Ê□B,Â,Ü,è□A□ˆ—□,Í‘S‘Ì,Æ,µ,Ä□¬Œ÷,·,é,©□A,Ü,½,ÍŽ¸”s,·,é,©,Ì,Ç,¿,ç,©,Å,·□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒRƒ“ƒeƒLƒXƒg

ƒNƒ‰ƒCƒAƒ“ƒg‚ª 1 ‚Â‚Ìƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“‚É 1 ‚Â‚ˆÈ□ã‚ÌƒlƒIƒuƒWƒFƒNƒg‚ð“®“I‚É‘g‚Ý□ž‚Þ‚±‚Æ‚ª‚Å‚«‚é‚æ‚¤‚É‚·‚é‚½‚ß‚ÌƒIƒuƒWƒFƒNƒg□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ}ƒl□[ƒWƒƒ

Œ´Žq□« (atomicity) ,ðŽÀŒ»,·,é,½,ß,É□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ÌŒ‹‰Ê,ð'²□®,·,é–ðŠ„,ð‰Ê,½,·ƒVƒXƒeƒ€ƒT□[ƒrƒX□Bƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ}ƒl□[ƒWƒƒ,É,æ,è□AƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,Íƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ðƒRƒ~ƒbƒg,·,é,©ƒAƒ{□[ƒg,·,é,©,É,Â,¢,Ä□Aˆêš Ñ,µ,½”»'f,ð‰º,·,±,Æ,ª,Å,«,Ü,·□B

ƒgƒŒ□[ƒX ƒ□ƒbƒZ□[ƒW

ƒXƒ^□[ƒgƒAƒbƒv□AƒVƒƒbƒgƒ_ƒEƒ“,È,Ç□AMicrosoft Transaction Server
,Ì,³,Ü,´,Ü,ÈƒAƒNƒefBƒrƒefB,ÌŒ»□Ý,Ì□ó‘Ô,ª‹L□q,³,ê,½ƒ□ƒbƒZ□[ƒW□B

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ^ƒCƒ€ƒAƒEƒg

ƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ}ƒl□[ƒWƒƒ,É,æ,Á,ÄŽ©“®“I,ÉƒAƒ{□[ƒg,³,ê,é,Ü,Å□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ªƒAƒNƒeƒBƒu,È□ó‘Ô,ðˆÛŽ□,Å,«,éŽžŠÔ,Ì□ãŒÀ□B

ƒ^ƒCƒv ƒ‰ƒCƒuƒ‰ƒŠ

ActiveX ƒeƒNƒmƒ□ƒW,ðŽg,Á,ÄƒIƒuƒWƒFƒNƒg,ðŒöŠJ,·,é,½,ß,Ìƒf□[ƒ^Œ^□Aƒ‚ƒWƒ…
□[ƒ‹□A,¨,æ,ÑƒCƒ“ƒ^□[ƒfƒFƒCƒX,Ì•W□€“I,È‹L□q,ðŠÜ,Þƒtƒ@ƒCƒ‹□B

2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg

•¡□”,ÌƒT□[ƒo□[,É“K—p,³,ê,éƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“,ª□AŠmŽÀ,É,·,×,Ä,ÌƒT□[ƒo□[,ÅŠ®—¹,·,é,©□A,Ü,½,Í,Ü,Á,½,-□ˆ—□,³,ê,È,¢,æ,¤,É,·,é,½,ß,Ìƒvƒ□ƒgƒRƒ‹□B2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg,Í□Aƒgƒ‰ƒ“ƒUƒNƒVƒ‡ƒ“ ƒ}ƒl□[ƒWƒƒ,É,æ,Á,Ä’²□®,³,ê□AƒŠƒ\□[ƒX ƒ}ƒl□[ƒWƒƒ,É,æ,Á,ÄƒTƒ|□[ƒg,³,ê,Ü,·□B

ƒt□[ƒU□[–¼

Windows NT ,Ìƒt□[ƒU□[ƒAƒJƒEƒ“ƒg,ðŽ¯•Ê,·,é–¼‘O□B

XA ƒvƒ□ƒgƒRƒ‹

X/Open DTP ƒOƒ‹□[ƒv,É,æ,Á,Ä’è‹`,³,ê,½ 2 ƒtƒF□[ƒY ƒRƒ~ƒbƒg ƒvƒ□ƒgƒRƒ‹□BInformix□AOracle□A,¨,æ,Ñ DB2 ,È,Ç□A‘½,,Ì UNIX ƒf□[ƒ^ƒx□[ƒX,ÍƒlƒCƒeƒBƒu,Å XA ,ðƒTƒ|□[ƒg,µ,Ü,·□B